DISSERTATION

GPS-Linked High Accuracy NTP Time Processor for Distributed Fault-Tolerant Real-Time Systems

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik

von

Dipl.-Ing. Dietmar Loy
Jörgerstraße 3/51
A-1170 Wien
Matr.-Nr. 8625389
geboren am 31. März 1967

loy@ict.tuwien.ac.at
http://www.ict.tuwien.ac.at/staff/loy

durchgeführt am Institut für Computertechnik
der Technischen Universität Wien

Begutachter: O. Univ. Prof. Dipl.-Ing. Dr. techn. Dietmar Dietrich
Univ. Dozent Dipl.-Ing. Dr. techn. Ulrich Schmid

Wien, ...

To my parents Helga and Karl Loy and my sister Elisabeth, who were always supporting me and my work. Thank you for giving me the strength, the ability and possibility to carry out this thesis.

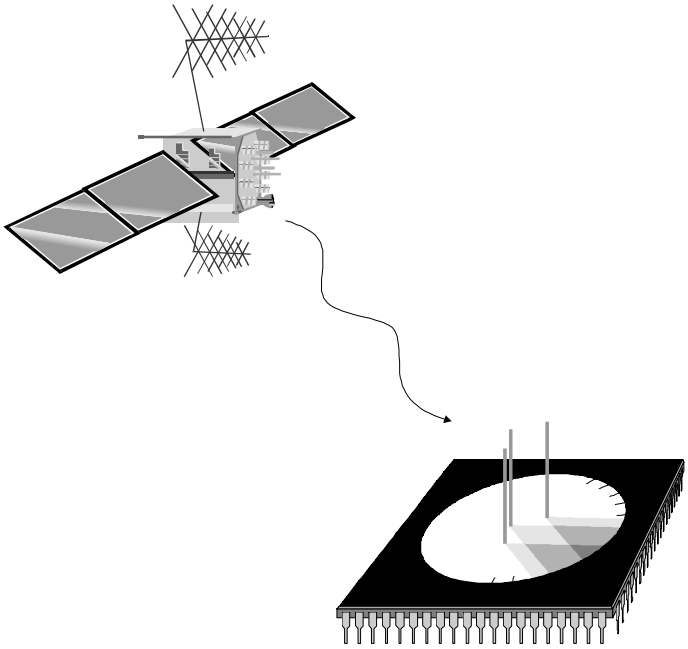
I would like to thank my doctor father, Prof. Dietmar Dietrich. I really appreciate your confidence in my work and my ideas. Let me conclude this period of life with a quotation.

> *Vater werden ist nicht schwer,*
> *Vater sein dagegen sehr.*
> *Wilhelm Busch, Julchen*

Finally I would like to thank the SYNUTC-team, the fabulous head of the project Ulrich Schmid, "Mr. Synopsys" Martin Horauer and Klaus Schossmaier. Thank you Martin for shaping and bending all the nuts and bolts in the UTCSU during long hours.

Universal Time Coordinated Clock Synchronization Unit

UTCSU[1]



Eins, zwei, drei, im Sauseschritt
läuft die Zeit; wir laufen mit.
Wilhelm Busch, Julchen

[1] This work is supported by the Austrian Science Foundation (FWF) under contract no. P10244-ÖMA.

Quis leget haec?

Flaccus (34-62 AD)

KURZFASSUNG

Dreifach ist der Schritt der Zeit;
zögernd kommt die Zukunft hergezogen,
pfeilschnell ist das Jetzt entflogen,
ewig still steht die Vergangenheit.
Schiller, Sprüche des Konfuzius

Schiller läßt Konfuzius sagen: "… pfeilschnell ist das Jetzt entflogen, …". Um das "Jetzt" präziser in den Griff zu bekommen, machten sich vor drei Jahren zwei Informatiker und zwei Elektrotechniker an die Arbeit. Die ersten Ergebnisse werden hier vorgestellt.

Zeit spielt sowohl im technischen als auch im kaufmännischen Leben eine wichtige Rolle. Solange man mit zentralen Rechensystemen arbeitete, war die Problematik einer gemeinsamen Zeitbasis nicht gegeben. Mit der Dezentralisierung der Rechensysteme mittels Netzwerken entstand sowohl im technischen Bereich, z. B. in der Automatisierungstechnik, als auch im kaufmännischen Bereich, z. B. bei Transaktionssystemen, das Problem einer fehlenden gemeinsamen Uhrzeit. Eine Vielzahl von Lösungen wird angeboten, doch der Bedarf an immer genaueren und hochauflösenderen Uhren wächst ständig.

Diese Arbeit zielt auf Synchronisationsgenauigkeiten im Bereich von 1 µs bei verteilten Uhren in fehlertoleranten Echtzeitsystemen ab. Dabei genügt es längst nicht mehr, einen Quarz-Oszillator mit einem Zähler zu verbinden und dies als Uhr zu bezeichnen. Zeitprozessoren sind eigenständige Einheiten wie Numerikprozessoren oder Netzwerkprozessoren.

Zu Beginn dieser Ausarbeitung werden verschiedene Uhrensysteme vorgestellt und Strategien zur Uhrensynchronisation diskutiert. Kenngrößen zu ihrer Qualifizierung werden bestimmt und gängige Zeitformate behandelt. Das im Internet verwendete NTP-Zeitformat wird dabei in einer erweiterten Form (ENTP) als Zeitbasis in dem hier vorgestellten Zeitprozessor mit dem Namen UTCSU (Universal Time Coordinated Clock Synchronization Unit) dienen. Eine Definition der Komponenten des zugrundegelegten Übertragungsnetzwerkes rundet die Einleitung ab.

Das Herzstück der UTCSU bildet eine neuartige, auf einem Addierer basierende Uhr, deren Frequenz und Phase programmierbar ist. Sie wird mit einem kostengünstigen Quarz-Oszillator betrieben und mittels errechneter Korrekturwerte werden Drift und Alterung des Oszillators kompensiert. Eine weitere Novität ist die zu jedem Zeitpunkt berechnete Worst-Case-Genauigkeit der Uhrzeit, die als unsymmetrisches Intervall um die lokale Uhrzeit angegeben wird. Das integrierte GPS (Global Positioning System) Interface erlaubt den direkten Anschluß handelsüblicher GPS-Zeitempfänger und ermöglicht der UTCSU somit den Zugang zu einer hochgenauen, globalen Zeitinformation, nämlich UTC, der weltweit einzigen offiziellen Uhrzeit.

Eintreffende und abgesandte Datenpakete werden mit einem Zeitstempel versehen, der sowohl die lokale Uhrzeit als auch die aktuelle Genauigkeit enthält. Im Anwendungsprogramm oder in der Physik der Anwendung auftretende Ereignisse werden über neun Kanäle aufgezeichnet und jedes mit einem Zeitstempel belegt. Ein zeitgesteuertes Ausgangssignal aktiviert externe Prozesse. 96-Bit Zeit- und Genauigkeitsinformation stehen an einem externen Port zur Weiterverarbeitung bereit.

Umfassende integrierte Test- und Selbsttesteinheiten geben Aufschluß über den Zustand der UTCSU und die Gültigkeit der Zeitstempel. Dazu werden unter anderem alle generierten Zeitstempel mit Prüfbits versehen und sowohl Signaturen als auch Blocksummen über längere Sequenzen erstellt.

Die UTCSU wird in einem ASIC implementiert. Eine VHDL-Beschreibung wird mittels Logiksynthese in eine Gatternetzliste übergeführt und nach abgeschlossenem Back-End-Design in einem 0.7 µm CMOS Prozeß als Standardzellenentwurf gefertigt.

ABSTRACT

"If it wasn't hard, they wouldn't call it HARDWARE!"

Keywords: Real-time systems, external clock synchronization, Universal Time Coordinated (UTC), Network Time Protocol (NTP), Application Specific Integrated Circuit (ASIC), Very high speed integrated circuit Hardware Description Language (VHDL IEEE 1076.1), Global Positioning System (GPS), Adder-Based Clock (ABC), accuracy intervals, continuous amortization, snapshots, Built-In Test Equipment (BITE), Built-In Self-Test (BIST), Boundary-Scan IEEE 1149.1

Designing a time service for distributed fault-tolerant real-time systems based on packet networks requires to decide on the tradeoff between hardware and software support. Targeting a 1 µs precision and similar accuracy measures in case of external UTC (Universal Time Coordinated) supply, this thesis concentrates on the requirements of the proper hardware support and presents an implementation of an ASIC with the acronym UTCSU (Universal Time Coordinated Clock Synchronization Unit). All pertinent features of the UTCSU are addressed.

A novel adder-based clock, adjustable in phase and frequency, maintains local time inside the UTCSU. Drift and aging of the driving low-cost quartz crystal oscillator is compensated by correction values obtained from an clock synchronization algorithm. In addition to local time, current clock accuracies are provided as asymmetric intervals around the local clock. The time format used internally is an extended version of the Internet Network Time Protocol (NTP) format with 32 bits in the integer part and 59 bits in the fractional part. Hardware support is provided for the programmed insertion and deletion of leap seconds in the local clock. A NTP-interface exports local time together with accuracy information on primary output pins.

A Synchronization Subnet Unit (SSU) can timestamp each received and transmitted data packet on one communication channel. Up to six SSUs are hosted in the UTCSU which allows the design of a triple-redundant gateway node with six independent network media connected to it. Three GPS Units interface the one-pulse-per-second (1PPS) output of GPS timing-receivers. UTCSU external application events are timestamped through a 9-channel external input event recorder.

Comprehensive test and self-test logic enables the UTCSU to be used in fault-tolerant real-time systems. Therefore every single generated timestamp is protected by an 8-bit checksum. Furthermore a sequence of timestamps can be compressed by either a blocksum adder or a signature analysis register (SAR), and the results compared with reference data, which are obtained from the synchronous operation mode, where multiple UTCSUs run in parallel. External events can be simulated internally during power-up testing and during the system integration phase, which makes the debugging phase easier. Snapshots of the current state of the UTCSU can be taken at any given time.

The design followed a strict top-down design flow with a VHDL RTL-description synthesized into a gate-level netlist followed by back-end design for fabrication in a 0.7 µm CMOS standard cell process. Scan-path as well as boundary-scan logic according IEEE 1149.1 were inserted automatically.

CONTENTS

KURZFASSUNG 9
ABSTRACT 11
ABBREVIATIONS 19

1 INTRODUCTION 21

1.1 SYSTEM WIDE UNIQUE TIME 21
1.2 STATE OF THE ART 23
1.2.1 Dedicated Time Channels 23
1.2.2 No Dedicated Time Channels 24
1.2.3 AMI S65C60 (CSU) 26
1.3 MOTIVATION FOR THIS THESIS 27
1.3.1 Technical Reasons 27
1.3.2 FWF-Founding 28

2 BASIC ISSUES 29

2.1 SYNCHRONIZATION SUBNETS 29
2.2 SYSTEM ENVIRONMENT 30
2.2.1 Fieldbus Environment 30
2.2.2 LAN Environment 30
2.3 BASIC SYNCHRONIZATION STRATEGY 31
2.4 BASIC CLOCK SYNCHRONIZATION PRINCIPLES 32
2.5 PARAMETERS AND DEFINITIONS 35
2.6 REQUIRED HARDWARE SUPPORT 35
2.6.1 Novel Ideas 35
2.6.2 Basic Services 36
2.6.3 Advanced Services 38
2.6.4 Standard Network Node Architecture 41

3 MAINTAINING LOCAL TIME 43

3.1 TIME NOTIONS 43
3.2 TIME REPRESENTATION 46
3.2.1 NTP Format 46
3.2.2 Extended NTP Format 46
3.3 CRYSTAL OSCILLATORS 47
3.3.1 Definitions and Characteristics 48
3.3.2 No Compensation (XO) 48
3.3.3 Temperature Compensation (TCXO, DTCXO) 48
3.3.4 Voltage Controlled (VCO) 49
3.3.5 Oven Controlled 49
3.4 GPS OSCILLATORS 49
3.5 ATOMIC OSCILLATORS 49

3.6 PRINCIPLES OF RATE AND STATE CORRECTION 50
3.6.1 Analog Technique 52
3.6.2 Single Counter Approach 52
3.6.3 Adder-Based Approach 52
3.6.4 Interleaving Adder and Counter 54

4 GPS SUPPORT 55

4.1 HOW GPS WORKS 55
4.2 ANALYSIS ON GPS INTERFACES 56
4.3 TIME EXCHANGE PROTOCOLS 57
4.3.1 Proprietary Solutions 57
4.3.2 NMEA-0183 57
4.3.3 LORAN 57
4.3.4 RTCM SC-104 57
4.4 RECEIVER DESIGNS 57
4.5 INTERFACE TO UTCSU 58
4.5.1 10 MHz-Interface 58
4.5.2 1PPS-Interface 59
4.6 GPS TIME RECEIVER EVALUATION 60

5 BITE & FAULT-TOLERANCE 61

5.1 TEST CONCEPT 61
5.2 PRE-FABRICATION PHASE 63
5.3 POST-FABRICATION PHASE 63
5.3.1 Scan-Path 63
5.3.2 Test Multiplexer 64
5.4 DEVICE INTEGRATION PHASE 64
5.4.1 Five Pin Test Access Port 65
5.4.2 Mandatory Instructions 65
5.4.3 Optional Instructions 65
5.4.4 Instruction Codes 65
5.5 OPERATIONAL PHASE 66
5.5.1 Timestamp Validation 66
5.5.2 Timestamp Compression 67
5.5.3 Blocksum Signature 68
5.5.4 Redundant Operation 69
5.5.5 On-Line Event Testing 69
5.5.6 Reliable Mode 70
5.5.7 Version Control 72
5.5.8 Hardware-Snapshot 72
5.5.9 Software-Snapshot 72

6 UTCSU BLOCK DIAGRAM 73

6.1 BLOCK DIAGRAM 73
6.2 REGISTER MAP 73

7 EXTERNAL INTERFACES 77

7.1 CPU-PORT 77
7.1.1 Dynamic Bus Sizing 78
7.1.2 Multiple CPU Access 80

7.2 CSUTIME-PORT 81
7.2.1 Normal Operation 81
7.2.2 Test Mode 81
7.3 SYSTEM-PORT 81
7.3.1 System Reset 81
7.3.2 Interrupt Handling 82
7.3.3 System Clock 82
7.4 EVENT-PORT 82
7.4.1 External Event Inputs 82
7.4.2 Event Outputs 82
7.5 TEST-PORT 82
7.5.1 Scan-Path Port 82
7.5.2 Boundary-Scan Port 82

8 UNIT SPECIFICATION 83

8.1 LOCAL TIME UNIT 83
8.1.1 Validating Register Layout 84
8.1.2 Initialization and Verification 86
8.1.3 Amortization Phase 87
8.1.4 Pure Phase 88
8.1.5 NTP Time vs. GPS Time 89
8.1.6 Debug and Test 89
8.1.7 Error Detection and Correction Unit 91
8.1.8 Atomic Sampling Across Units 94
8.1.9 Internal NTP-Bus 94
8.1.10 Configuration LTU 94
8.1.11 Status LTU 95
8.1.12 Interrupt Sources LTU 95
8.1.13 Register Map 95
8.1.14 Programmers Model 96
8.1.15 Initialization Phase 99
8.1.16 LTU Test Strategy 100
8.2 ACCURACY UNIT 101
8.2.1 Validating Register Layout 102
8.2.2 Initialization after Reset 103
8.2.3 Amortization Phase 103
8.2.4 Pure Phase 105
8.2.5 Debug and Test 106
8.2.6 Atomic Sampling Across Units 107
8.2.7 Configuration ACU 107
8.2.8 Status ACU 107
8.2.9 Interrupt Sources ACU 108
8.2.10 Register Map 108
8.2.11 Programmers Model 109
8.2.12 Initialization Phase 112
8.2.13 ACU Test Strategy 113
8.3 SYNCHRONIZATION SUBNET UNIT 115
8.3.1 Receive Action 116
8.3.2 Transmit Action 116
8.3.3 Duty Timer Operation 117
8.3.4 Configuration SSU 118
8.3.5 Status SSU 118
8.3.6 Interrupt Sources SSU 119
8.3.7 Register Map 119

8.3.8 Programmers Model 120
8.3.9 SSU Test Strategy 122
8.3.10 Self-Test Option 123
8.4 GPS UNIT 124
8.4.1 One Pulse Per Second Event 124
8.4.2 Configuration GPU 125
8.4.3 Status GPU 125
8.4.4 Interrupt Sources GPU 126
8.4.5 Register Map 126
8.4.6 Programmers Model 126
8.4.7 GPU Test Strategy 127
8.4.8 Self-Test Option 128
8.5 APPLICATION UNIT 129
8.5.1 Event Recording 130
8.5.2 Event Generation 131
8.5.3 Configuration APU 131
8.5.4 Status APU 131
8.5.5 Interrupt Sources APU 132
8.5.6 Register Map 132
8.5.7 Programmers Model 133
8.5.8 APU Test Strategy 134
8.5.9 Self-Test Option 135
8.6 SNAPSHOT UNIT 136
8.6.1 Hardware-Snapshot 136
8.6.2 Software-Snapshot 137
8.6.3 Synchronous Operation 138
8.6.4 Configuration SNU 138
8.6.5 Status SNU 138
8.6.6 Interrupt Sources SNU 139
8.6.7 Register Map 139
8.6.8 Programmers Model 139
8.6.9 SNU Test Strategy 140
8.6.10 Self-Test Option 141
8.7 BUILT-IN TEST UNIT 142
8.7.1 Test at the Unit-Level 142
8.7.2 Test at the Chip-Level 143
8.7.3 Signature and Blocksum Computation Timing 144
8.7.4 Pure Phase Counter 145
8.7.5 Configuration BTU 146
8.7.6 Interrupt Sources BTU 146
8.7.7 Register Map 147
8.7.8 Programmers Model 147
8.7.9 BTU Test Strategy 149
8.8 INTERRUPT UNIT 150
8.8.1 Chip Version Identification 150
8.8.2 Status Information 151
8.8.3 Configuration Data 151
8.8.4 Interrupt Processing 152
8.8.5 Register Map 154
8.8.6 Programmers Model 154
8.8.7 ITU Test Strategy 157
8.9 NTP-INTERFACE UNIT 158
8.10 BUS INTERFACE UNIT 160
8.10.1 Write Timing 160
8.10.2 Read Timing 162

8.10.3 Module Select 163
8.10.4 Data Port Routing 163
8.10.5 Data Port Control 164
8.11 CLOCK UNIT 165
8.12 SYNCHRONIZATION UNIT 165
8.12.1 Metastability Firmness 165
8.12.2 Single/Dual Stage Synchronizer 166
8.12.3 Uncertainty Assessment 166

9 DESIGN FLOW 167

9.1 FRONT END DESIGN 167
9.1.1 System Simulation 168
9.1.2 RTL Simulation 168
9.1.3 Logic Synthesis 169
9.2 TEST INTEGRATION 171
9.2.1 Scan-Path Insertion 172
9.2.2 Boundary-Scan Insertion 172
9.2.3 Fault Simulation 172
9.3 BACK END DESIGN 173

10 FUTURE ASPECTS 175

10.1 ACCURACY AND PRECISION 175
10.2 SECURITY 175
10.3 TIMESTAMPING DATA PACKETS 176
10.4 UTCSU FAMILY 176
10.5 APPLICATION SUPPORT 176
10.6 REDUCE DIE SIZE 176
10.7 UTCSU MACROCELL 177
10.8 CONCLUSION 177

LIST OF FIGURES 179
LIST OF TABLES 183
LITERATURE 185

ABBREVIATIONS

⊕ ..... Boolean XOR Operator
AC ..... Alternating Current
ACU ..... Accuracy Unit
A-DUTY ..... Acknowledgement D-packets Duty
AMS ..... Austria Micro Systems
APU ..... Application Unit
ASIC ..... Application Specific Integrated Circuit
BIH ..... Bureau International de l'Heure
BIST ..... Built-In Self-Test
BITE ..... Built-In Test Equipment
BIU ..... Bus Interface Unit
BS ..... Boundary Scan
BTU ..... Built-In Test Unit
CAN ..... Control Area Network
C/A-Code ..... Coarse Acquisition Code
CDDI ..... Copper Distributed Data Interface
CLU ..... Clock Unit
COMCO ..... Communication Coprocessor
CPU ..... Central Processing Unit
CS ..... Checksum
CSP ..... Clock Synchronization Packet
EDCU ..... Error Detection and Correction Unit
ENTP ..... Extended NTP
EXR ..... External Reset
D-DUTY ..... Transmission Delay Measurement Duty
FAN ..... Field Area Network
FDDI ..... Fiber Distributed Data Interface
GPS ..... Global Positioning System
GPU ..... GPS Unit
IC ..... Integrated Circuit
ITU ..... Interrupt Unit
JD ..... Julian Day
JTAG ..... Joint Test Action Group
LAN ..... Local Area Network

LON ..... Local Operating Network
LT ..... Local Time
LTU..... Local Time Unit
MC68030 ..... Motorola CPU 68030
MISR..... Multiple Input Linear Feedback Shift Register
MJD ..... Modified Julian Date
MTBF..... Mean Time Between Failures
NAVSTAR ..... NAVigation System with Time And Ranging
NBS..... National Bureau of Standards
NCSC..... Network Clock Synchronization Coprocessor
NIST ..... US National Institute of Standards and Technology
NMEA..... National Maritime Electronics Association
NTU ..... Network Time Interface Unit
NTP..... Network Time Protocol
P-Code..... Precise-Code
PLL ..... Phase-Locked-Loop
POR..... Power-On Reset
PPS..... Pulse per Second
RFC..... Request for Comments
R-DUTY..... Clock Rate Algorithm Duty
RTCM ..... Radio Technical Commission for Maritime Services
SA ..... Selective Availability
S-DUTY ..... Clock State Algorithm Duty
SNU ..... Snapshot Unit
SSN..... Synchronization Subnet
SSU ..... Synchronization Subnet Unit
SWR..... Software Reset
SYU..... Synchronization Unit
TAI..... International Atomic Time
TBC..... To Be Characterized
TCXO..... Temperature Compensated XO
T-DUTY..... CSP Reception Termination Duty
TDP..... Time Distribution Protocol
TS ..... Timestamp
UTC ..... Universal Time Coordinated
UTCSU..... Universal Time Coordinated Clock Synchronization Unit
UUT ..... Unit Under Test
VCO..... Voltage Controlled Oscillator
VHDL..... VHS IC Hardware Description Language
VHS IC..... Very High Speed Integrated Circuit
VLSI..... Very Large Scale Integration
VME..... Versa Module Europe
WAN..... Wide Area Network
XMT ..... Transmit
XO..... Quartz Crystal Oscillator

CHAPTER 1

1 INTRODUCTION

Wer weiß, was in der Zeiten Hintergründe schlummert.
Schiller, Don Carlos

The technique that is used to coordinate the notion of time is known as clock synchronization [SIM90]. A distributed controller systems consists of a set of processing units that communicate via message passing and they do not have access to a central time source ie a clock. But it is frequently necessary for the processors to obtain some common notion of time, where 'time' can mean either an approximation to real-time, like UTC (Universal Time Coordinated), or simply a number obtained from a counter (logical time).

1.1 SYSTEM WIDE UNIQUE TIME

Providing a system wide unique time source was no big deal in the era of central computers. Upcoming events E_1, E_2 were timestamped $t(E_1)$, $t(E_2)$ with a central computer clock to guarantee the timed order of these events, cf. Figure 1-1 for a typical centralized control application. The conveyor belt moves the workpiece B between tools which are controlled by a centralized computer. A central clock records events (eg E1 from a capacitive switch or E3 from a light sensor) and triggers events (eg E2 activates the drill). Database transactions aim at a single central database.

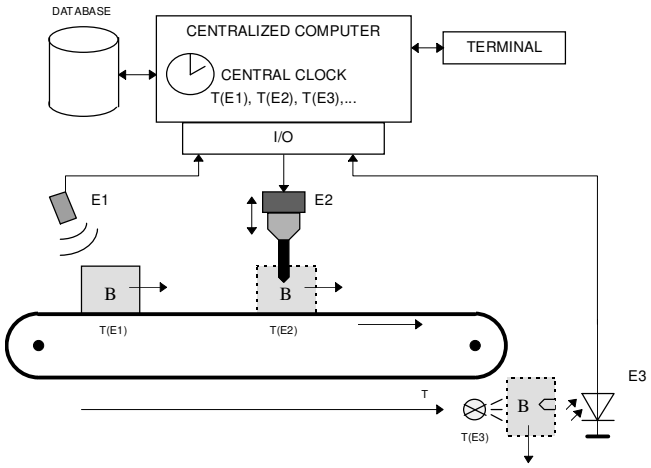


Figure 1-1 A Centralized Clock Triggers and Records Events E_1, E_2, E_3, ...

Modern real-time systems are more and more implemented as heterogeneous hierarchical distributed systems. A European trend in industrial communication is illustrated in Figure 1-2 as a vendor

independent network hierarchy with anticipating standards [SCHN94]. Multitasking, multiprocessing, multithreading, and distributed intelligence are the keywords for today's and future control systems.

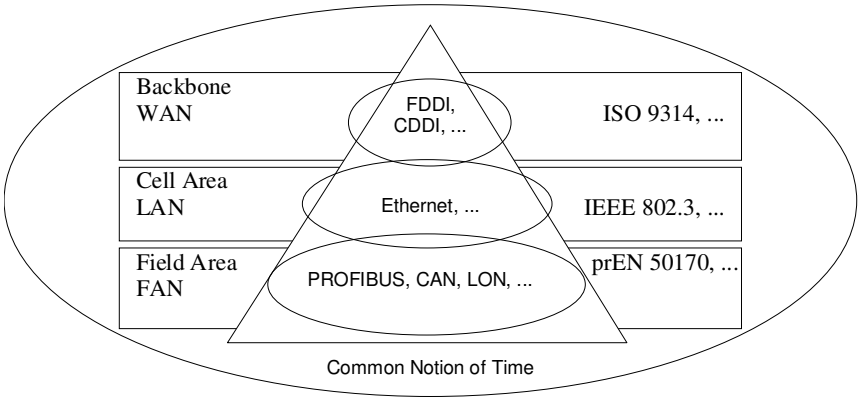


Figure 1-2 Computer Integrated Manufacturing (CIM) Pyramid [SCHN94]

The process control is broken up into small portions running on a big variety of different microcontrollers (aka. nodes), cf. Figure 1-3. This evolution positively influences the reliability and expandability of distributed systems but introduces new problems which were unknown in centralized control applications (eg common notion of time).

To provide a system wide common notion of time offers the advantage to replace asynchronous system models by synchronous models in which synchronous algorithms can be employed. Some problems are solvable only with synchronous algorithms and algorithms for synchronous system models are easier to solve and show better performance. Such aspects are presented in [LIS93] and [LAM84].

Practical relevance is confirmed by the development and introduction of NTP, the Internet network time protocol standard which is widely accepted and adopted by thousands of Internet-Hosts [MILLS91]. In database systems, concurrency control and version management rely on accurate and chronoscopic timestamps. Factories, aircraft, space vehicles and more mundane applications like power distribution in utility companies depend on real-time processing and therefore on a common time source.

Taking into considerations that distributed real-time systems, when interacting with the outside world, get in touch with real earthly clocks, there is no other way as to provide a system wide unique time. For embedded systems, an internal consistent time is often adequate. Complex applications more often require a system time synchronized to an external time reference eg UTC (universal time coordinated). In this context we should mention that UTC is the only official time, hence basis for compulsory specifications [PSYN93].

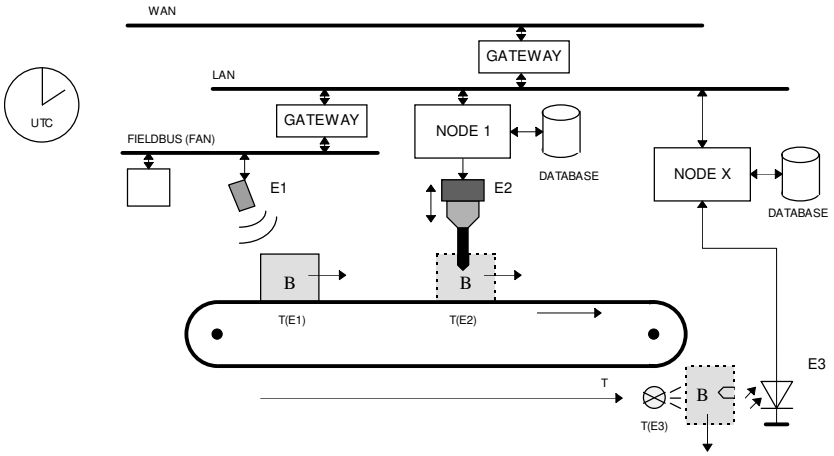


Figure 1-3 Decentralized Process Control based on the CIM-Pyramid

Centralized techniques to distribute time information are well established in other fields. Navigation on planes and ships is supported by radio frequency time transmitters like the German DCF-77 system [SCHI77] or navigation aids as LORAN-C, OMEGA and nowadays GPS (global positioning

system). These systems are based on one or several RF transmitters with a modulated carrier transmitting digital time code information to receivers [EK73]. For a large number of network nodes there are obvious questions of reliability and cost of this scheme. A radio transmitter is a single point of failure, hence not best qualified for fault-tolerant processing. In the subsequent sections we shall concentrate on methods which are principally adequate to provide a network wide common notion of time in distributed fault-tolerant computer systems.

The resulting application requirements are as follows:

- Clocks should never be set back and forth (monotonicity). Monotonicity is obligatory in order to guarantee that if $E_i \rightarrow E_j$ then $C_i(E_i) < C_j(E_j)$ is always true. Some time standards are not chronoscopic in the sense that it is not (always) possible to measure short intervals of time. In fact UTC contains leap seconds in order to cope with the small disagreements between atomic time and astronomic time.
- Clock adjustments should be small, clock accuracy targets 1 µs.
- Easy joining and reintegration of faulty nodes. New network nodes and nodes going back into service should be able to join the synchronized network within a bounded time interval.
- Graceful degradation in case of lost synchronization. Nodes falling out of synchronization should behave in a way that there are well defined limits on how badly performance is affected.
- The synchronization protocol must operate continuously and provide update information at rates sufficient to compensate the drift of the local clock oscillator.
- Minimize the number of messages exchanged for clock synchronization. For example, broadcast networks by their nature reduce the number of clock synchronization packets from $O(n^2)$ in a point to point network to O(n) where n is the number of joined nodes. Broadcast networks help keep the synchronization traffic low.
- Clock synchronization should not disrupt other activities. CPU computing power and network traffic consumed by the clock synchronization algorithm must be limited to a small percentage of total system performance.
- The primary reference sources must deliver time related to international standards (UTC). Systems providing external clock synchronization require a link to international time references either via wire or radio frequency or through calibrated atomic clocks.
- Full time and short timestamps. Short and compact timestamps are needed to efficiently use the network media and to limit the packet length on the network. This is a serious problem in fieldbus networks with very short and efficient data packets.

1.2 STATE OF THE ART

An alternative approach to the centralized time distribution is to use software and some kind of hardware support in a network node and to design clock synchronization algorithms. A classification for deterministic algorithms is given in [SIM90]. But before we are heading towards distributed clock synchronization, a quick summary on dedicated time channels is given in the following section.

1.2.1 Dedicated Time Channels

In principle, it is possible to set up special network facilities designed for time distribution, such as timecode rebroadcasts on a dedicated TV or FM subcarrier or cable, eg telephone, or satellite links. Common to all these methods is a dedicated channel reserved for time distribution. The following paragraphs will show some examples starting will the satellite based NAVSTAR GPS.

Thanks new technology NAVSTAR GPS (Global Positioning System) time receivers are available at reasonable cost already. GPS is based on 24 satellites in the orbit, providing full time and position information covering most territories 24 hours a day. The satellites are controlled by a number of monitoring stations around the world. GPS time is derived from clock sources in the monitoring station

and atomic clocks in the space vehicles (SVs). Currently the block II SVs contain two caesium and two rubidium clocks. Time and position information from the satellites is transmitted on two frequencies L1 (1575.42 MHz) and L2 (1227.6 MHz) using spread spectrum modulation. Two codes are in service, the P-code also known as precise code available only to military services and the C/A-code known as coarse acquisition code available to the public at no charge [WEL87]. The accuracy of the C/A-code is deliberately deteriorated through selective availability (SA). Direct view from the antenna to the satellite is compulsory, see Figure 1-4. Four satellite views are necessary to receive position and time information. If the position is known, as is the case in clock synchronization applications, only a single satellite is required to track accurate time information. Time accuracy is in the range of 300 ns with SA on. NAVSTAR GPS is an excellent source for UTC, however to allow GPS to become the single source for time keeping is to put too much faith in a complex process involving many levels of procedures, personnel, hardware and software systems and user interfaces. After all, GPS must be seen as a centralized system because it is operated by the US Air Force and therefore controlled by one institution. Even if technical single points of failures are tolerated, a single point of failure in the operation can shut down the whole system immediately.

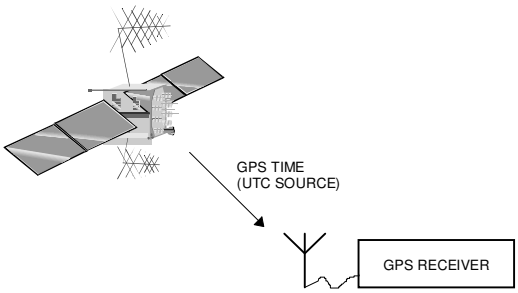


Figure 1-4 GPS-Based Time Distribution requires Direct Satellite View

Wired clock distribution spends an extra cable dedicated to time services, cf. Figure 1-5. Commercial available IRIG (Inter Range Instrumentation Group) based systems are not fault-tolerant. The IRIG-H timecode format established in 1970 and used by NIST (US National Institute of Standards and Technology) radio broadcast does not include year information and advance notice of leap second insertion. In fact, the NBS (National Bureau of Standards) telephone time service [NBS88] does include both, year and advance leap second information.

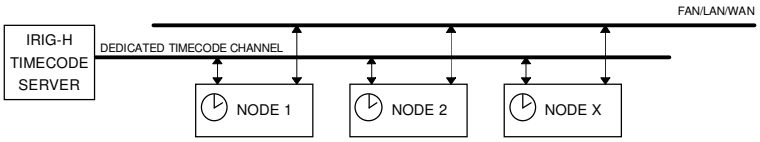


Figure 1-5 Some Timecodes like the IRIG-H are Based on Dedicated Time-Channels

The scientific literature lists numerous papers dealing with so called "hardware-methods" (see [RSB90] for an overview). Stimulated by fault-tolerant clocking systems [SMI81, KES84], combinations of clock voting with PLLs are presented in [SR88, VM88].

1.2.2 No Dedicated Time Channels

Here the time information is distributed across the network together with user data. Only a single channel is required. This technique is the more classical method to synchronize clocks. Two philosophies are distinguished, logical clocks and real-time synchronization.

Logical clocks are based on the relation $\rightarrow$ (happened before) which was introduced in [LAM78]. Although logical clocks have no relationship to real (Newtonian) time, their quality is consistency of causalities induced by the relation $\rightarrow$. For two consecutive events E_i, E_j on different nodes i, j and $E_i \rightarrow E_j$ is always true that $C_i(E_i) < C_j(E_j)$ in which C_i, C_j represent the logical clocks on nodes i, j at events E_i, E_j. Many authors dedicated their work-force to logical clocks eg [JEF85] or [FID91]. An inherent advantage is that logical clocks can be implemented without hardware support but as soon as

interaction with real world problems occur, the connection to an external is missing. Hence this method is not a candidate for real-time systems.

Real-time synchronization is the general term for what will be discussed in the following subsections. One can say that this topic attracts most of the papers dealing with clock synchronization. A classification based on the characteristics listed in Table 1-1 is given in [SIM90].

System Characteristic	Explanation
asynchronous	Relative processor speeds and message delivery times are unbounded
partially synchronous	Processors may have real-time clocks that are approximately the same The message delivery time is always within some bounds
reliable	All components are assumed to work correctly
unreliable	Communication faults may occur Processors may exhibit a range of faulty behavior

Table 1-1 System Model Characteristics [SIM90]

These and following papers are based on the (partially synchronous, unreliable) system model characterized by

- a distributed system consisting of n processors;
- fully connected, error free point to point network with guaranteed upper and lower bounded transmission delays $\delta \in [\delta_l, \delta_u]$ for non-faulty transmissions;
- a hardware clock for each node, in which the drift referring to the real (Newtonian) time is bounded by constant ρ ($\approx 10^{-6}$) for non-faulty nodes [SL93].

A theoretical lower bound for the synchronization precision π is explored in [LL84, MA92, HS91]. Precision π is defined as follows. Every non-faulty node p, q must satisfy $\left|C_p(t) - C_q(t)\right| \leq \pi$ for some fixed π and any t with $C_p(t)$ being the correct clock of node p. The clock synchronization algorithm has to guarantee this behavior.

A considerable amount of papers is dedicated to various algorithms for clock synchronization and their analysis. Virtually all of the algorithms rely on successive rounds, called resynchronization intervals R. Within a single interval, node clocks are running without interaction from outside at their given speed determined by the node specific hardware clock. At the end of each resynchronization interval each node determines a correction value from others for its own clock that ensures precision of non-faulty nodes despite of the fact that some nodes might be faulty. The next resynchronization interval is started after correcting its own clock with the previously calculated correction value.

Following a classification in [RSB90] leads to three types of clock synchronization algorithms as listed below.

- *Convergence-averaging algorithms (*[LM85, LL88, MS85])
  The basic idea is, that every node determines its clock difference to the other nodes in the network at the end of a resynchronization interval. These differences are the input of a fault-tolerant averaging function.
- *Convergence non-averaging algorithms (*[HSSD84, CAS86, VR92, DB93]*)*
  These algorithms are based on the idea that the end of the resynchronization interval for all nodes is determined by the end of the resynchronization interval of a single node, called the synchronizer. Fault-tolerance is accomplished, because each node tries to become the synchronizer but only one non-faulty node can win.
- *Consistency algorithms (*[LM85])
  At the end of every resynchronization interval any node using this type of algorithm applies the interactive consistency algorithm [LSP82] to distribute its own clock to all other nodes. At the end of this phase, every node has a set of clock differences to the clocks of all other

nodes. This set is consistent with respect to the set on the other nodes, hence a consistent correction value for the own clock can be computed.

There are of course other algorithms used in fault-tolerant clock synchronization. Interval based algorithms as proposed in [MAR83] propagate the nodes clock accuracy information in addition to the timestamp and also [MIL91] for the Internet NTP services. Clock accuracy α is defined as $\left|C_p(t) - t\right| \leq \alpha$ for every non-faulty node p and all t. Of course t must be available from a reference source.

Very important in our context are solutions based on broadcast networks instead of point to point connections [KO87, DB93, VER92].

The basic elements required for a dedicated clock synchronization hardware which supports most of the algorithms listed above can be summarized as follows

- A local hardware clock with programmable state (phase) and rate (frequency)
- Timer features for administering resynchronization intervals
- Timestamping data packets
- Management of accuracy information

A new approach satisfying these requirements along with a highly sophisticated accuracy management can be found in Chapter 3.

1.2.3 AMI S65C60 (CSU)

The Clock Synchronization Unit (CSU) of [KO87] offers hardware support for some of the basic elements listed above. Figure 1-6 illustrates the block diagram of the CSU, which is manufactured by AMS, the former AMI silicon foundry in Styria near Graz [AMI86]. The concept behind clock rate and clock state correction in the CSU is to omit or to insert single clock pulses at the clock register input. Of course, this is only possible if the oscillator runs at a higher frequency, in the CSU at least four times higher than the desired clock resolution. Rate corrections must be programmed in µs/s and state corrections in µs. As drawbacks of this method must be mentioned the high frequency system clock, the fixed amortization interval of 0.5 s in the CSU and the problem of not equally distributed additional or omitted clock pulses within the synchronization interval.

The clock register is limited to 20 bits and cannot be set, the clock resolution is given with 1 µs. No means are provided for synchronization to a global time source. The AUX-register can record a single external event (eg application event) and the TIME-OUT-block generates an external event after a programmed time-out. System speed is reduced in debug mode by factor 10, 100, 1000 or 10000. A 16-bit bus interface connects the CSU with the system bus.

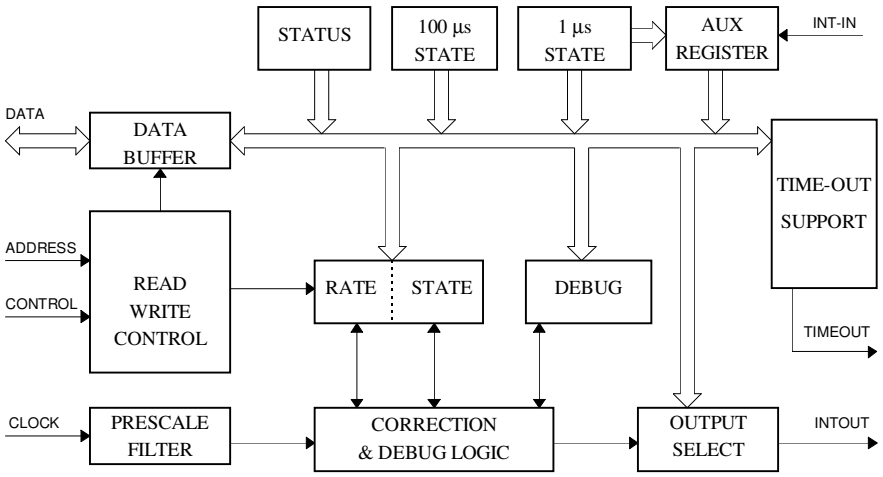


Figure 1-6 Clock Synchronization Unit of [KO87], AMI S65C60

1.3 MOTIVATION FOR THIS THESIS

Everything started out 1993 when Ulrich Schmid from the Department of Automation was looking for partners to design a new Network Clock Synchronization Coprocessor (NCSC) for a MC68030 based VME-Bus CPU module, cf. Section 2.3. He found Martin Horauer and Dietmar Loy at the Institute of Computer Technology (ICT) what turned out to develop into a very fruitful long lasting cooperation.

1.3.1 Technical Reasons

Given the requirements on clock synchronization quality and application support aimed at the NCSC-project, we quickly realized that we do not have the components around to implement the clock hardware according Section 1.2.2. Even the CSU from Section 1.2.3 could not meet our additional requirements shown below:

- high precision and accuracy (< 200 ns),
- freely programmable resynchronization interval length,
- maintenance of clock accuracy information,
- at least 32-bit timestamps,
- time representation in a common notion like NTP-format,
- real-time clock (UTC time, GPS time,...) vs. logical clock (simple counter),
- interface to a global time source (eg GPS receivers),
- support for gateway nodes and redundant subnets,
- provisions for application support,
- self-test and error detection capability in fault-tolerant applications.

Nobody was really concerned about precision and accuracy in the range of a couple 100 ns in the beginning of clock synchronization research. Typical applications - if any - target at a precision in the range of ms. To achieve our synchronization quality, a new clock synchronization unit implemented as an Application Specific Integrated Circuit (ASIC) has to be designed from scratch. Its name is UTCSU which is the acronym for Universal Time Coordinated Clock Synchronization Unit.

Network topologies can vary, network nodes are different, the characteristics of the clock oscillators for the local clock hardware can change and clock synchronization algorithms are manifold, hence the length of the clock resynchronization interval should be programmable as it influences the time-constant of the whole system. The AMI chip relies on a fixed amortization period of 0.5 s. This could result in reduced synchronization quality due to infrequent oscillator-drift adaptation. The newly designed UTCSU must be prepared to overcome this shortcoming.

One can deduce from the previous sections that some clock synchronization algorithm (eg interval based clock validation ICV) insist on clock accuracy information. No hardware components, which support clock accuracy, are known to the author up to now. The reader will be introduced in the UTCSU asymmetric accuracy concept in Chapter 2.

A 20-bit timestamp, what is used in the AMI S65C60 chip, would overrun every 100 ms under the assumptions of a 100 ns time granularity. Of course, external means can be assigned to extend the timestamp but this raises other problems if one thinks of atomic[1] sampling or atomic setting. In this project, we agreed on the timestamp formats presented in Section 3.2.

Some applications strictly rely on a global time like UTC, above all data base transactions, process control and many more. Most hardware clocks, including the CSU, simply run a logical clock without any reference to the real-world time. The UTCSU is specially designed to provide global time in two flavors, with and without leap second correction, cf. Section 3.2.2. Consequently an interface to a global time source must be at hand. Although the UTCSU is prepared to interface with GPS receivers

[1] Atomic sampling in this context means sampling several registers at the same moment.

in the first place, the GPS-interface is general enough to interface to any global time source as long as it is equipped with a 1PPS output.

Almost every computer network is made up of subnets, interconnected with bridges, routers or gateways. Hence simultaneous timestamping of data packets on at least two different subnets is a strong requirement for any clock hardware. If one thinks of double or triple redundant subnets, this requirement is even stronger and must be extended to the number of connected subnets. To wash out all future requirements, the UTCSU can simultaneously timestamp six subnets (eg a gateway node in a triple-redundant environment).

One should never lose sight of the primary goal. We make all these clock synchronization efforts in order to provide accurate timestamps to the application layer. Indeed, our primary focus must be on the support of the application layer. Section 8.4 will reveal the services dedicated to application timestamping.

Self-test capability and error detection in the clock hardware layer will be addressed in this thesis for the first time. Multiple UTCSUs can be organized in a multiple-redundant configuration. Any system design like a two out of three decision is allowed and promoted. Timestamps are validated with a checksum, comprehensive snapshots assist during system installation and debug phase.

1.3.2 FWF[2]-Founding

The promising character and the scientific quality of the UTCSU-project are underpinned by founding through the Austrian Science Foundation (FWF) under contract no. P10244-ÖMA. The economic usability is confirmed in the support of the Austrian National Bank and in concrete inquiries from industry.

[2] FWF = Austrian Science Foundation

CHAPTER 2

2 BASIC ISSUES

Davon wird man noch reden in den spätesten Zeiten.
Schiller, Wilhelm Tell

Chapter 2 talks about novel ideas underlying the design of the Universal Time Coordinated Clock Synchronization Unit (UTCSU), outlines the behaviour of basic and advanced UTCSU services and shows how to embed the UTCSU in different type network nodes. Examples are given for a Profibus fieldbus node and an Ethernet node.

2.1 SYNCHRONIZATION SUBNETS

Systems under consideration in this thesis are made up of a set of interconnected synchronization subnets (SSNs) [RFC1119, SS95]. The synchronization subnet is a collection of nodes connected by a network of point-to-point or, preferably, broadcast/multicast type. Nodes in the SSN can be classified as follows:

- A *C-node (client)* participates only in a passive way and exhausts the existing synchronization activities on the network to establish a synchronized local clock in a cheap manner.
- An *S-node (secondary)* participates actively in the clock synchronization process and constitutes the core of the synchronization network.
- A *P-node (primary)* has its S-node capabilities extended by access to a primary time source like UTC (eg via a GPS receiver) and donors this most accurate time information to its SSN.
- A *G-node (gateway)* connects two or more SSNs. Special functionality is expected from a G-node to allow time dissemination and interconnection to other SSNs.

A sample system with three synchronization subnets is shown in Figure 2-1. SSN 1 comprises all four types of nodes (primary, secondary, client and gateway). For fault-tolerant reasons, both primary nodes could inject first class UTC time into the subnet. Two gateway-nodes inter-network SSN 1 and SSN 2 and disseminate SSN 1 time into SSN 2, hence redundancy is supported.

SSN 2 appears as a time consumer subnet with nodes of type secondary, client and of course gateway. Observe that SSN 2 interconnects to two neighbouring subnets in which SSN 3 is designed in a triple-redundant fault-tolerant fashion. Note that more general topologies - beyond the three - like our below - are supported.

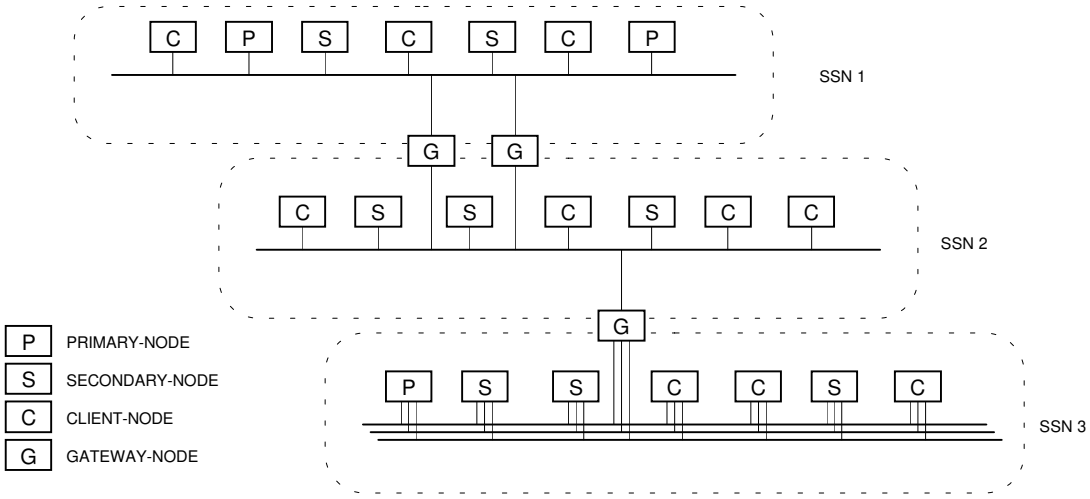


Figure 2-1 System Components of a Synchronization Subnet (SSN)

2.2 SYSTEM ENVIRONMENT

The applicability of the UTCSU is not limited to a particular network type. Timestamping can deal with short data packets (eg fieldbus systems) and long data packets (eg local and wide area networks). One representative for each field shows up in the next subsections.

2.2.1 Fieldbus Environment

Profibus [BEN92] is a widely accepted fieldbus system standardized in prEN 50170. Master stations (M) form a logical ring and slave stations (S) are polled by the masters. The physical communication medium is a bus, terminated on both ends. Bus controllers are available from different companies. One possible configuration is shown in Figure 2-2 with three master and two slave stations. Master 1 could be a primary-node with a link to a GPS receiver and the other two masters are actively involved in the clock synchronization process as secondary-nodes. Slave stations S1 and S2 are just consumers of time and run as client-nodes, to speak in the UTCSU-jargon.

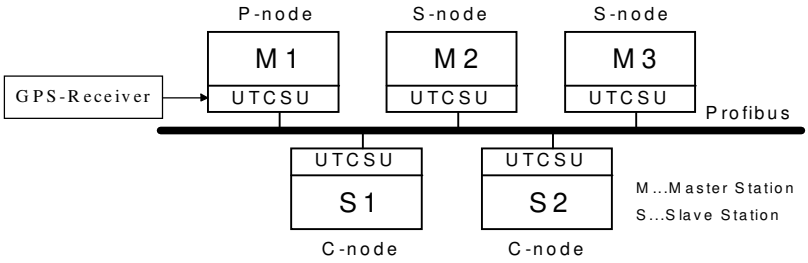


Figure 2-2 UTCSU in a Profibus Environment

2.2.2 LAN Environment

Another example is an Ethernet based distributed control system with microprocessor equipped network nodes. One specific implementation was done in [HOR94] and is displayed in Figure 2-3. Network nodes use standard VME-Bus 68030 CPU boards with a piggy-back Ethernet interface. One objective of the project was to replace this piggy-back and to design a new network interface, ie a NCSC based on the Intel 82596 network controller and the UTCSU-chip. This system is employed in a distributed monitoring system (the Versatile Timing Analyzer VTA, [SCHM943]) at the Department of Automation.

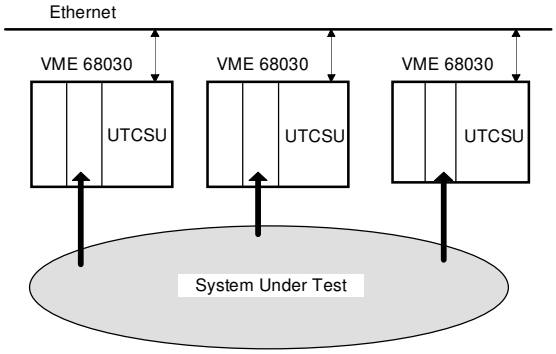


Figure 2-3 UTCSUs in an Ethernet Based Versatile Timing Analyzer

2.3 BASIC SYNCHRONIZATION STRATEGY

In order to implement a precise and accurate local clock, the network node must be equipped with a hardware clock that satisfies the required resolution and stability. A logical clock is then constructed under expenditure of these components plus software components that adjust the apparent time and frequency in response to *periodic corrections* computed by the clock synchronization algorithm. Structured into three layers, viz application layer, clock synchronization layer and clock hardware layer, Figure 2-4 depicts the entities required to do clock synchronization at a single network node [SS95].

The *application layer* runs the application software, which transmits/receives application related data packets and makes use of the lower layer timing features.

All clock related software components reside in the *clock synchronization layer*. There are entities to transmit and receive clock synchronization packets (CSPs) and entities to measure transmission delays, local clock rates as well as local clock states. The clock synchronization algorithm (ie interval based clock validation technique ICV, [SCHM95]) is hosted in the entity named clock algorithm. Basically this entity consists of a clock state and a clock rate algorithm.

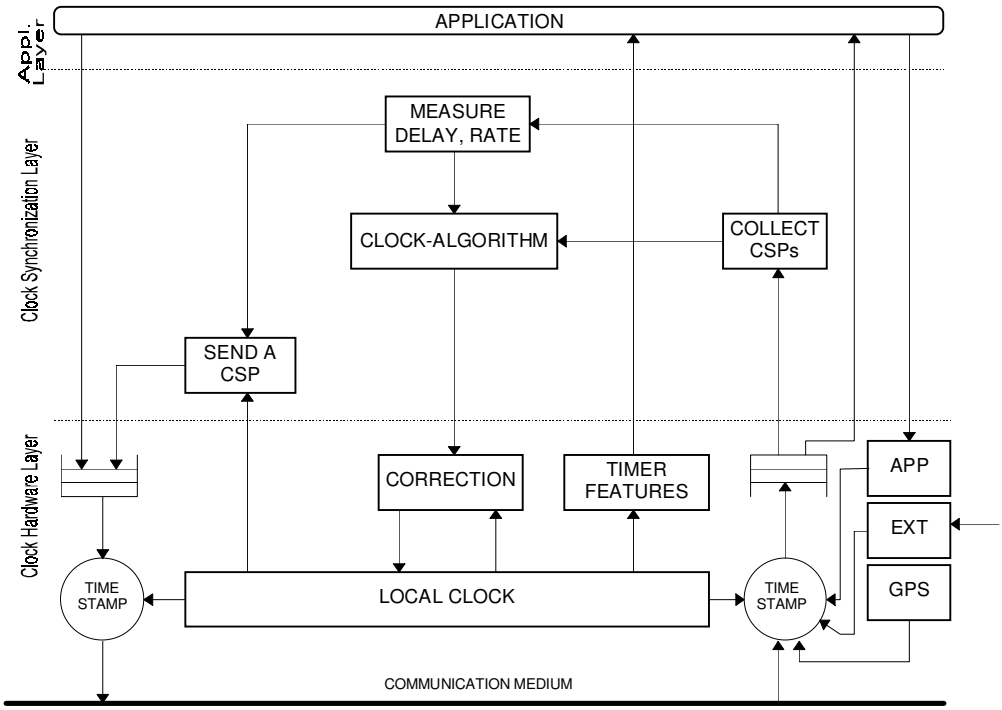


Figure 2-4 Entities in the Triple Layer Clock Synchronization Approach

In order to produce a clock precision and accuracy in the range of a couple hundred nanoseconds, massive support from the *clock hardware layer* is decisive. All hardware components involved in clock synchronization are situated on a Network Clock Synchronization Coprocessor (NCSC), which

embraces coprocessors, network controllers, memory, interface logic and the novel Universal Time Coordinated Clock Synchronization Unit (UTCSU) implemented in an application specific integrated circuit (ASIC). Martin Horauer developed an Ethernet based prototype NCSC in his diploma thesis under the supervision of Dietmar Loy and Ulrich Schmid, for details see [HOR94].

The UTCSU, as introduced in Chapter 2 later on, meets the clock hardware layer functionality and is responsible for

- maintaining a state and rate controllable local hardware clock,
- maintaining local clock accuracy information,
- assisting the communication protocol to exchange CSPs (timestamping at OSI-layer 2),
- dissemination of time through gateway-nodes,
- providing links to GPS receivers,
- supporting application oriented timer and timestamping features,
- providing debug information during system installation.

One salient challenge in the design of the UTCSU is to handle several asynchronous interfaces between the application layer, the clock synchronization layer, a monitoring station, network controller(s) and GPS receiver(s).

2.4 BASIC CLOCK SYNCHRONIZATION PRINCIPLES

Clock oscillators underlie drift and aging, hence local clock values have to be corrected after some period. The clock correction round concept is illustrated in Figure 2-5. Clocks $C_p(t)$ resp. $C_q(t)$ on nodes *p* resp. *q* are adjusted instantaneously if $C_p = C_q = k \cdot R$. Due to the asynchronous operation of nodes *p* resp. *q*, instantaneous clock correction on nodes *p* resp. *q* happens at different real-times $t_p^{R,k}$ resp. $t_q^{R,k}$ defined by $C_p(t_p^{R,k}) = C_q(t_q^{R,k}) = k \cdot R$ where k denotes the round number and R the length of the round. Instantaneous correction in rounds guarantees $\left|C_p(t) - C_q(t)\right| \leq \pi$ with precision π.

Correction values for the local clock are computed with one of the clock synchronization algorithms given in Section 1.2.2, which rely on the exchange of timestamped data packets called clock synchronization packets (CSPs). Timestamping data packets is described in Section 2.6.2.4.

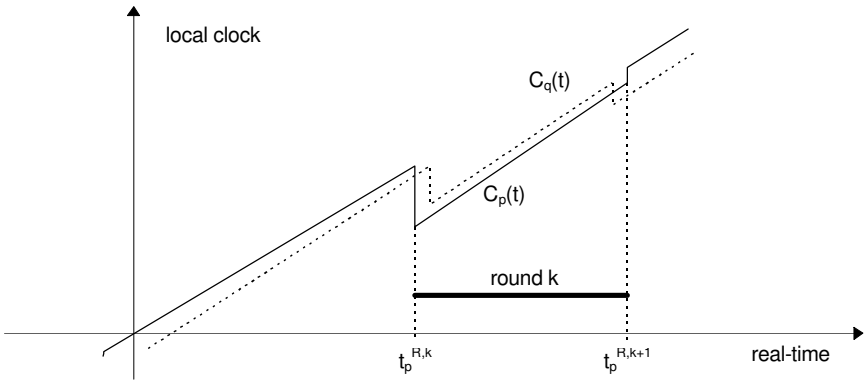


Figure 2-5 Clock Correction happens in Rounds

The progress of local time provided by clock synchronization at each node can be modeled by a sequence of overlapping virtual clocks C_0, C_1, C_2, … [SS95]. Each virtual clock C_i has a birth time t^0_i, where $t^0_i < t^0_{i+1}\ \forall\ i \geq 0$. Conceptually, the lifetime of a virtual clock is infinite, but in practice it gets redeemed by the generation of the next virtual clock, cf. Figure 2-6. R_i denotes the resynchronization period. Due to a computational overhead, the virtual clock C_i goes into effect at time t^1_i.

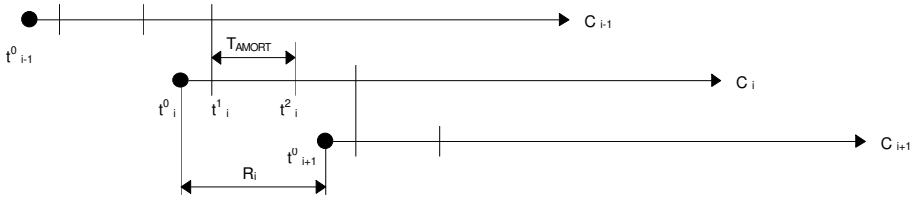


Figure 2-6 Sequence of Virtual Clocks

The problem with instantaneous clock correction is the jump of the local clock $C_p(t)$ at times $t_p^{R,k}$. But most applications require a composite time over all correction rounds. Therefore the instantaneous clock correction is replaced by continuous amortization as shown in Figure 2-7.

Clock values have to be monotonic and continuous, which excludes discrete jumps forward or backwards. A technique called continuous amortization (see [SC90]) smears out the corrections over a specific amount of time T_{AMORT} by adjusting the clock rate. One can divide each round into two periods named *pure phase* and *amortization phase*. Amortization phase starts at time t_i^1 and ends at time t_i^2, when pure phase supersedes amortization phase. Pure phase continues until the next amortization phase is started. Observe that state correction is primarily done during amortization phase while rate correction takes place all the time.

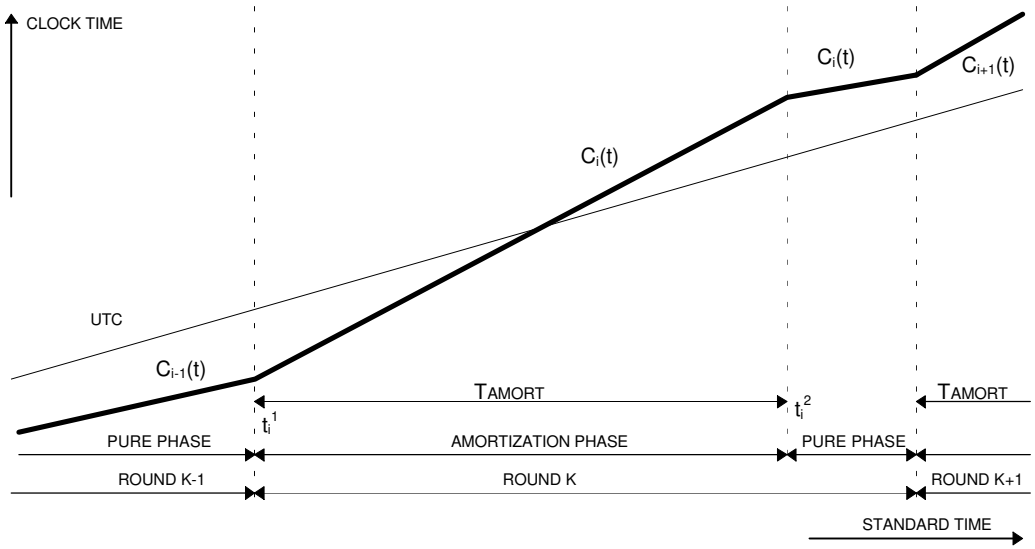


Figure 2-7 Local Clock with Continuous Amortization

Some clock synchronization algorithms (ie [SS96]) require clock accuracy information specified as an asymmetric interval around the local clock that encloses UTC. Extending the concept of virtual clocks in Section 2.6.2.1 with clock accuracy information leads to the following definition of a virtual clock C_i.

- $C_i(t)$ is the clock value
- $E_i(t)$ represents the upper envelope
- $e_i(t)$ represent the lower envelope
- $A_i(t) = E_i(t) - C_i(t)$ is the upper accuracy and
- $a_i(t) = C_i(t) - e_i(t)$ is the lower accuracy

These quantities are defined $\forall t \geq t^0_i$ and satisfy the condition $e_i(t) \leq c_i(t) \leq E_i(t)$. For notational convenience all quantities can be packed into a clock vector

$$\mathbf{S}_i(t) = (C_i(t), E_i(t), e_i(t), A_i(t), a_i(t))$$

An initial clock vector $\mathbf{S}_i(t^0_i)$ can be calculated at birth time t^0_i. In a simplified model, the components of the clock vector change linearly over time. Clock C_i changes with rate ω_i, hence $C_i(t) = C_i(t^0_i) + \omega_i \cdot (t - t^0_i)$ for $t \geq t^0_i$. The upper envelope $E_i(t)$ grows with rate $\omega_i + \Lambda_i$, where Λ_i expresses the deterioration in order to account for clock drifts. The lower envelope $e_i(t)$ grows with rate $\omega_i - \lambda_i$. Finally the clock vector evolves to

$$\mathbf{S}_i(t) = \begin{pmatrix} C_i(t_i^0) \\ E_i(t_i^0) \\ e_i(t_i^0) \\ A_i(t_i^0) \\ a_i(t_i^0) \end{pmatrix} + \begin{pmatrix} \omega_i \\ \omega_i + \Lambda_i \\ \omega_i - \lambda_i \\ \Lambda_i \\ \lambda_i \end{pmatrix} \left(t - t_i^0\right) \qquad \forall t \geq t_i^0$$

All five components of the clock vector are illustrated in Figure 2-8.

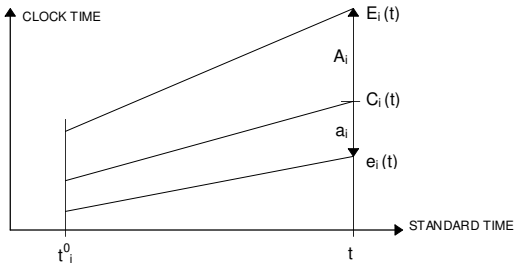


Figure 2-8 State Quantities of a Virtual Clock C_i

The following Figure 2-9 shows a correction round with the virtual clock $C_i(t)$, the upper envelope $E_i(t)$ resp. the lower envelope $e_i(t)$ and the upper accuracy $A_i(t)$ resp. the lower accuracy $a_i(t)$.

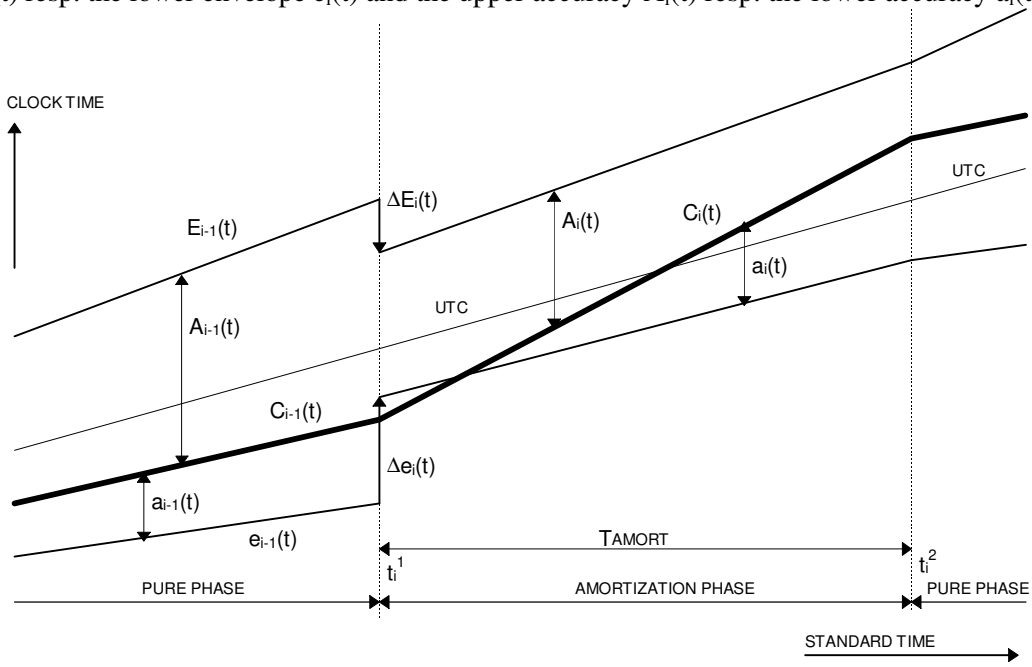


Figure 2-9 Upper and Lower Envelopes Surround UTC

Both envelopes $E_i(t)$ and $e_i(t)$ are realigned at the beginning of a resynchronization round (t_i^1), when the new virtual clock takes over. Two rules must be followed at time t_i^1.

- Clock values $C_i(t)$ are not allowed to jump. They must be corrected by a temporarily rate change during the amortization interval T_{AMORT}.

- Envelopes may be changed on the spot. From the equation $A_i(t) = E_i(t) - C_i(t)$ follows, that either the envelopes or the accuracies could be maintained in the UTCSU and the missing quantity is computed with the given relation. Of course, an ASIC-designer is always concerned about die size, hence the smaller number, which is the accuracy, should be implemented. It stands to reason that accuracies may jump as well. More specifically, the differences $\Delta E_i(t^1)$ and $\Delta e_i(t^1)$ need to be added to $A_i(t^1)$ and $a_i(t^1)$ respectively. During amortization phase (T_{AMORT}), accuracies experience a modified deterioration value as well.

2.5 PARAMETERS AND DEFINITIONS

In order to validate different clock synchronization algorithms and even complete systems utilizing clock synchronization, some characteristics are defined in [MILL91, SS96] and summarized below.

- *Granularity*
  can be seen as clock resolution. It can be defined as the smallest possible difference between two adjacent timestamps or the time step in the adjustable clock whatever returns the smaller number.
- *Precision*
  is the maximum absolute difference between two different clocks in the system, cf. page 25
- *Accuracy*
  is the maximum deviation of any non-faulty clock from real-time (eg UTC), cf. page 26.
- δ_{ij}
  is the transmission delay from node i to node j satisfying $\delta_{ij} \in \left[d_{ij} - \varepsilon_{ij}, d_{ij} + \varepsilon_{ij}\right]$, d_{ij} represents the deterministic part and ε_{ij} a bound on the random part of δ_{ij}.
- ρ
  is the upper bound on the rate at which non-faulty clocks drift from real-time. ρ ensures that $(t - t_0)(1 - \rho) \le C(t) - C(t_0) \le (t - t_0)(1 + \rho)$ is true for all clock ticks $t \ge t_0$. C(t) is the clock value at time t.
- *Stability*
  defines how well a clock can maintain a constant frequency.
- *Offset*
  between two clocks is their actual time difference, while
- *Skew*
  is the actual frequency difference between the two clocks.
- *Synchronize frequency (rate)*
  means to adjust the clocks to run at the same frequency.
- *Synchronize time (phase, state)*
  means to set the clocks to agree at a particular epoch with a global time reference like UTC.
- *Synchronize clocks*
  means to synchronize in rate and state.

2.6 REQUIRED HARDWARE SUPPORT

2.6.1 Novel Ideas

As mentioned earlier, the UTCSU was designed from scratch relative to [KO87]. Only a catchword-list of the novel ideas is given in this section, details on the analysis and implementation are presented in the subsequent sections.

- The *adder-based-clock* is a non-standard approach for a rate and state adjustable clock. It is brought into action to maintain local time and local clock accuracy. Different approaches are discussed in Chapter 3.
- A *timestamp-checksum* protects timestamps against falsification.
- *Asymmetric accuracy intervals*, implemented in hardware, are a UTCSU feature, which was never found before. It is classified as an extended basic service in my nomenclature.
- A redundant *GPS-interface* establishes a link to GPS time receivers and enables the access to a global time source, cf. Chapter 4.
- *Built-In Test and Self-Test* entities guarantee fault-free operation. Analysis and discussions are found in Chapter 5.
- *UTCSU-monitoring* is supported, cf. Section 5.5.
- A *9-channel application interface* records events and outputs a preprogrammed event, cf. Section 8.5.
- The *NTP-interface* passes a 56-bit timestamp and a 32-bit accuracystamp to the outside world. Hence even the most unimaginable applications should be satisfied.

2.6.2 Basic Services

Four basic services are listed in Section 1.2.2.

- The adjustable local clock.
- Manage clock accuracies.
- Timer features to administer resynchronization intervals.
- Timestamp data packets.

2.6.2.1 Adjustable Local Clock

Clock Synchronization happens in rounds, also called correction periods or resynchronization intervals. At the beginning of each round new clock correction values are computed and applied by the clock synchronization algorithm, in order to compensate for clock deviations due to frequency drift and other reasons.

Several methods for implementing this behaviour are discussed in Chapter 3. Among them is the novel adder-based approach, as introduced in Section 3.6.3. Observe that if the local clock should truly follow UTC, leap seconds must be inserted according Table 3-2. This requires write or set or add capability at the local clock register. Furthermore easy joining of a node which is out of synchronization and the adaptation to a global time should be considered in the design.

2.6.2.2 Maintain Clock Accuracy Information

Clock accuracies represent an interval around local time in which UTC lies. The actual implementation is described in Section 8.2. A careful reader will observe the negative lower accuracy in Figure 2-9, which starts at time t_i^1 and lasts for some time. If the application program could read a negative accuracy as zero, this translates to a valid accuracy interval enlargement, one can save the sign bit in the accuracystamp passed to the application layer.

2.6.2.3 Administration of Correction Periods

Every round starts with an amortization phase that lasts for time T_{AMORT}. In order to administer the rounds, an AMORTDUTY, which starts the amortization phase, and an AMORTTIMER, which fixes the length of the amortization phase, are required. The AMORTDUTY is implemented in a duty-timer, cf. Section 8.3.3, while the AMORTTIMER is realized in the AMORTTIMER in Section 8.1.3.

2.6.2.4 Timestamping Data Packets

Common to all clock synchronization algorithms is, to timestamp incoming and outgoing data packets. Of course, there are various different networks around and many ways how to timestamp packets in these networks. If one considers different strategies, it is important to have in mind that timestamps added to the network data packets shouldn't be a big burden in terms of excessive packet length. A reasonable network qualification is the classification into

(1) Fieldbus Area Networks (FANs), with very compact data packets to exchange a small number of control information in a short time, and

(2) Local Area Networks (LANs) and Wide Area Networks (WANs), with rather complex long data packets.

Whereas "wasting" communication bandwidth by transmitting long timestamps poses no problems in (2), one should be very careful with each additional byte in (1). The UTCSU must support both types, possibly intermixed for different network types connected by a gateway node. Consequently one should consider the two principal packet formats in Figure 2-10 and Figure 2-11.

LAN and WAN packets are more complex. Adding extra space for the transmit timestamp and the receive timestamp does not very much increase the total packet length. The scenario shown in Figure 2-10 transparently maps the sampled transmit timestamp (TIMESTAMP XMT) on the fly into the outgoing data stream (DATA PACKET). On the other hand, the timestamp (TIMESTAMP RCV), sampled at packet reception, is transparently mapped into the communication coprocessor (COMCO) receive buffer.

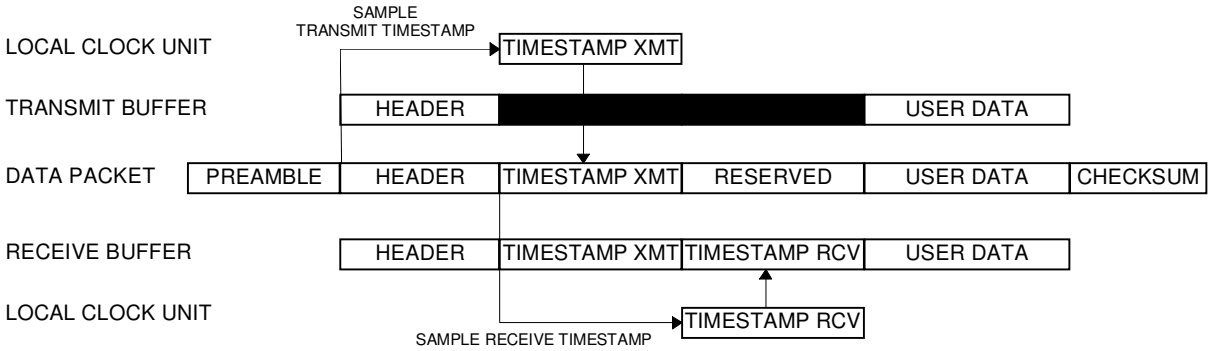


Figure 2-10 Packet Format and Timestamping in LAN and WAN

FAN packets are very compact and every additional byte counts. Here it's not possible to reserve space in the data packed to hold two timestamps. Hence only the transmit timestamp (TIMESTAMP XMT) is packet into the data packet and the receive timestamp (TIMESTAMP RCV) is stored into an extra timestamp memory, cf. Figure 2-11.

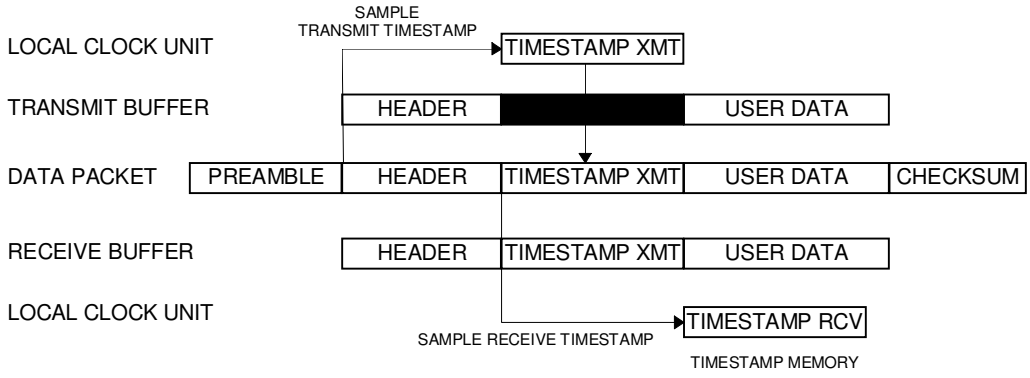


Figure 2-11 Packet Format and Timestamping in FAN

Both cases require, at least at the transmitter side, transparent mapping of the sampled timestamp into the transmit resp. receive buffer. Observe that timestamp alignment at word or longword boundaries in the buffers could result in spare bytes which must clearly be seen as an unwanted and unnecessary overhead. Precautions are taken in Section 8.3 to avoid these padding bytes.

Whenever a data packet is transmitted, the network controller has to raise a signal in the UTCSU and as a result a timestamp plus an accuracystamp are sampled into the transmit register set. The network controller reads the transmit register and inserts the timestamp/accuracystamp into the outgoing data packet. Any details on the actual implementation are revealed in Section 8.3 and in particular in Subsection 8.3.2.

Almost the same happens whenever a data packet is received, except that only the timestamp is sampled into the receive register and is read by the clock synchronization CPU, see Section 8.3.1. Figure 2-12 illustrates the transmit timestamp action with the shaded circle on the left and the receive timestamp action with the shaded circle on the right. Timestamping data packets is a service of the clock hardware layer.

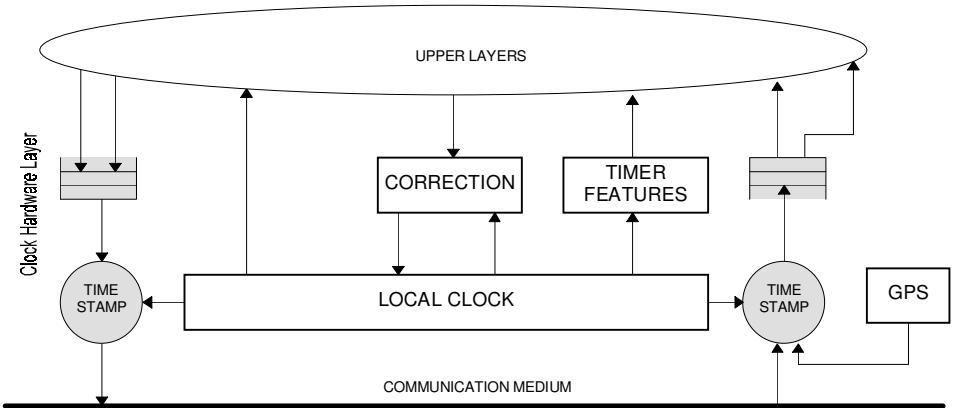


Figure 2-12 Entities for Timestamping Data Packets in the Clock Hardware Layer

To process the receive timestamp leaves several options. One possibility is to raise an interrupt in the CPU which pulls out the timestamp from the UTCSU, however, a short receive FIFO is required in the UTCSU as new packets may arrive before the first receive timestamp is processed. A better solution is to directly map the receive timestamp into the receive buffer. No FIFO is needed but it requires to reserve space in the data packet and additional logic is necessary to funnel the timestamp into the receive buffer memory. Finally, reception timestamps can be collected in a dedicated receive timestamp memory using DMA-transfer to move the timestamp from the UTCSU into the memory. Care must be taken to keep the order between received timestamps and received data packets.

2.6.3 Advanced Services

The previous section listed four basic services, which are mandatory to make advanced high performance clock synchronization feasible. This section concentrates on more advanced strategies and UTCSU-services to further improve the clock synchronization process and the support of various applications. Debug and test features are introduced and their effectiveness is proven. By the way, the list below is just a quick summary of the many UTCSU advanced services known. A classification into three categories, viz local clock, application support and test can be defined.

2.6.3.1 Category I Services

Category I services (local clock) improve the local clock behavior and extend the data packet timestamping capability of the clock hardware layer to a number of synchronization subnets.

- The local hardware clock can be *synchronized to a global time standard (UTC).* This requires write capability to the local clock register to set the global time at the moment of joining the global time source. Clocks out of synchronization can easily join clocks in synchronization by cloning their clock registers.

- Some time notions insert leap seconds every now and then. The local clock in the UTCSU supports hardwired *leap second correction*, viz leap second insertion and omission.
- A *timestamp protection* mechanisms detects and corrects corrupted timestamps. Single bit errors can be corrected and up to triple bit errors are detectable.
- *Bounds* can be defined on accuracies. An upper bound monitors the upper accuracy $A_i(t)$ and a lower bound the lower accuracy $a_i(t)$. If bounds are violated, an interrupt is raised.
- *Gateway nodes* interconnect synchronization subnets. They forward user data and time information from one port to the other. Observe that the clock hardware layer must provide a sufficient number of data packet timestamping entities.
- A *GPS receiver* interconnection provides a global time source for the UTCSU.

2.6.3.2 Category II Services

Category II services (application support) allocate services above all accessed by the application layer. They are listed below.

- A *9-channel event recorder* timestamps events on external input pins. Services in the application layer can reference the records.
- A *single channel event generator* creates an external event on a primary output pin, if an application programmable time reference exceeds.
- Six software programmable *duty-timer* raise an interrupt, if an application dependent time reference exceeds.
- The UTCSU internal local time plus accuracy information is brought to the outside world through a 96-bit wide *NTP-interface*.

2.6.3.3 Category III Services

Category III services (test) comprise means for debugging during system integration and for test while the system is up and running.

- *Snapshots* give an overview of the current state of the UTCSU. While software snapshots are triggered by the application or the clock synchronization layer, hardware snapshots are captured by an event on an external input pin.
- *Test and self-test* services allow self-checking clock operation. Timestamps and accuracy stamps underlie a signature and a blocksum calculation.
- In a fault-tolerant environment, two or more UTCSUs can run in parallel. This is called *synchronous operation*.

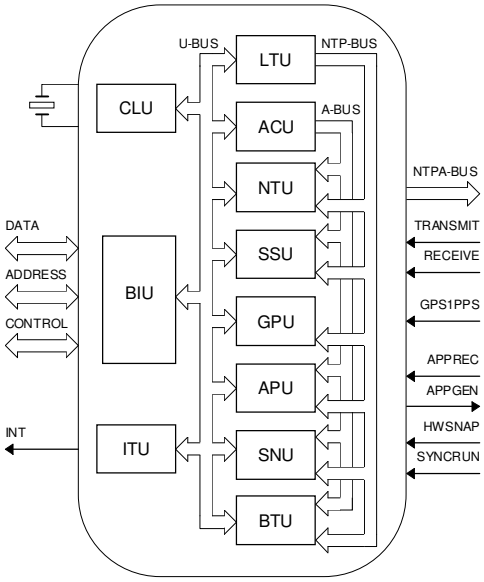
2.6.3.4 Auxiliary Services

There are some more support services underlying the basic and advanced services, which are hard to classify. A loose arrangement is given below.

- An advanced *interrupt handler* filters and processes up to 64 different interrupt sources.
- All external input pins are *polarity programmable*, hence easily adapted to the surrounding hardware.
- The *bus interface* to a HOST-CPU offers dynamic data bus sizing and can interface to big-endian as well as little-endian type CPUs.

2.6.3.5 Simplified Block Diagram

With all those basic and advanced services in mind, one can think of a simplified UTCSU block diagram. The one in Figure 2-13 comprises all the building blocks that result if the four basic services plus the three categories with the advanced services plus the auxiliary services are modeled in hardware.



unit	full name
ACU	Accuracy Unit
APU	Application Unit
BIU	Bus Interface Unit
BTU	Built-In Test Unit
CLU	Clock Unit
GPU	GPS Unit
ITU	Interrupt Unit
LTU	Local Time Unit
NTU	NTP-Interface Unit
SNU	Snapshot Unit
SSU	Synchronization Subnet Unit

Figure 2-13 Simplified UTCSU Block Diagram

A building block is called unit and the UTCSU consists of 10 different units in the first place. One will find replicated units later on. The adjustable local clock is modeled in the local time unit (LTU), accuracy information is maintained in the accuracy unit (ACU), data packets are timestamped in the synchronization subnet unit (SSU) and the application unit (APU) holds the external event recorder plus event generator.

The global positioning unit (GPU) interfaces to a GPS global time receiver, the snapshot unit (SNU) captures software and hardware snapshots, the built-in test unit (BTU) monitors the operation of the UTCSU and the network time interface unit (NTU) offers a local time interface to the outside world. A Unit-Bus (UBUS) connects all units through the bus interface unit (BIU) to a CPU while the NTP-Bus and the A-Bus supply the units with timestamps and accuracystamps. The interrupt unit (ITU) services all pending interrupts. Of course, a clock unit (CLU) must be added to generate the system clock and the reset conditions.

Chapter 3 is devoted to an analysis of different implementation strategies for the local time unit and the accuracy unit. Chapter 4 is devoted to the GPU and Chapter 5 to the built-in test unit (BTU) as well as the snapshot unit (SNU).

2.6.3.6 Gateway-Node

Gateway-nodes deserve special attention in this chapter. The basic building block to timestamp data packets, according Section 2.6.2.4, in a synchronization subnet is the synchronization subnet unit (SSU). A single SSU can serve a single channel. Replicating the SSU increases the number of channels to be served. In order to act as a gateway-node in a triple redundant network architecture, the number of SSUs must be increased to six (SSU1 - SSU6) as shown in Figure 2-14. The communication coprocessor (COMCO) blocks represent similar or different network controllers.

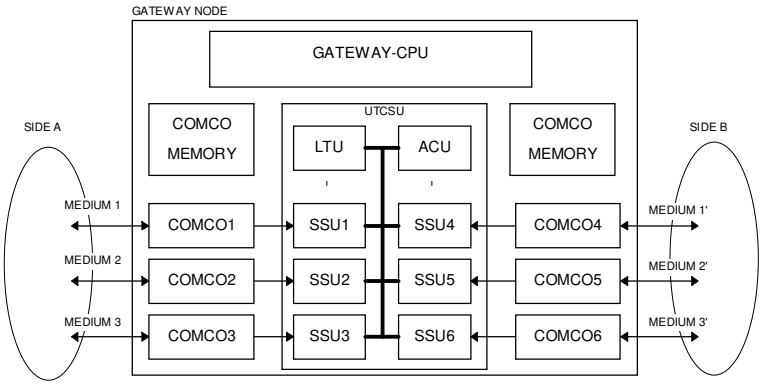


Figure 2-14 The UTCSU is Prepared to Serve as a Gateway-Node in a Triple Redundant Network

2.6.4 Standard Network Node Architecture

Any P-, S-, C- and G-node hardware follows the standard network node architecture as proposed in Figure 2-15. A network controller interconnects the CPU with the network medium. Transmitted data packets (XMT) and received data packets (RCV) as well as CPU events (EVT) along with application events (APPEVTs) are timestamped by the UTCSU. The conglomerate of Network Controller (including local memory) and UTCSU is called Network Clock Synchronization Coprocessor (NCSC).

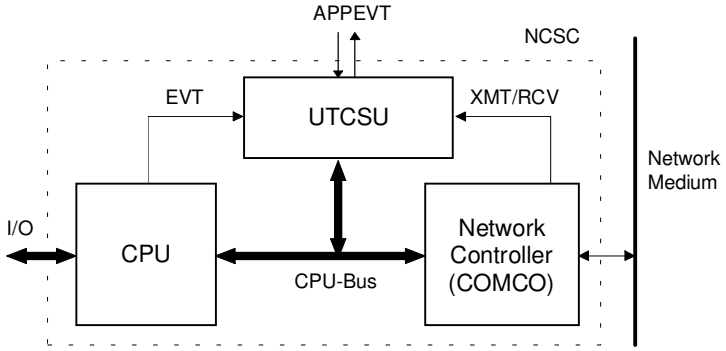


Figure 2-15 Network Clock Synchronization Coprocessor (NCSC) based on the UTCSU

CHAPTER 3

3 MAINTAINING LOCAL TIME

Die Zeit ist aus den Fugen.
Shakespeare, Hamlet

Before 1967, the second was defined by the 1/86400 part of the mean solar day. The first caesium clock came out in 1955 providing a stability of 10^{-9}. It took ten years for the technology to mature to a stability performance of 10^{-12}, which is the stability of a commercial caesium oscillator today. Since 1967, the second has been defined by the duration of 9192631770 periods of the emitted radiation in a transition between two hyperfine levels of the fundamental state in a Cs133 atom. To protect the time standard against a malfunction in a single atomic clock, a mean atomic time is derived from several atomic clocks [RG90].

Frequency sources range from high priced, high performance type oscillators (atomic clocks) used in national time standards, to low price, low performance uncompensated quartz crystal oscillators used in electronic equipment. We give an overview of oscillators appropriate for the UTCSU.

All types of reasonable priced oscillators underlie drift, noise and aging. One subsection defines the terms stability, jitter and wander. The chapter ends up with proposals for rate and state adjustable clock designs.

3.1 TIME NOTIONS

A large variety of different notions of time is know in literature. To filter out what is best applicable in terms of the characteristics proposed in Section 1.1, we step back in history to see where our notion of time and date originates. A condensed summary is given in [RFC1119, p.7] and a very detailed analysis can be found in [ZEM90].

For many years, time information was most important for worldwide navigation and space science which depend on observations of Sun, Moon and stars. Sidereal time is based on transitions of stars across the celestial meridian of an observer. The mean sidereal day is 23 hours, 56 minutes and 4.09 seconds.

One tropical year is one complete revolution of the Earth around the Sun. In 1958 one second was defined as 1/31556925.9747 of the tropical year that began this century. Using this scale the tropical year has 365.2421987 days and is increasing by about 5.3 ms per year.

The International Astronomical Union has adopted the use of the standard second and Joseph Justus Scaligers[1] Julian Day (JD) to date cosmological events. 86400 standard seconds are called a standard day and 365.25 standard days are called a standard year. Scaliger recognized that there is no better order in history then the day. Therefore he suggested to count the days - not as a number but as a measure in days. JD 0.0 corresponds to 0^h (midnight) on the first day of the Julian Era, this was Janu-

[1] Joseph Justus Scaliger was born on Aug 5th, 1540 in Agen, South France. Scaliger developed the concept of the Julian Era and the Julian Day, cf. [ZEM90, p. 61-70].

ary 1, 4713 BC. JD 1721424.0 corresponds to 0^h on the first day of the Christian Era, this is January 1^{st}, 1 AD. In 1973 the International Astronomic Union in Sydney passed a decision for the Modified Julian Date (MJD) which is defined as MJD = JD - 2400001. MJD = 0 corresponds to Wednesday Nov. 17^{th}, 1858 0^h.

Table 3-1 shows the relationship between the Gregorian Date and the Julian Day. For example: March 31^{st}, 1967 converts to JD = 2439581.

1900		2415079
60	+	21915
7	+	2556
MAR	+	0
Day	+	31
	JD =	2439581

a AD	**JD**	**a**			**m**
0	1721117	1	365	MAR	0
100	1757642	2	730	APR	31
200	1794167	3	1095	MAY	61
300	1830692	4	1461	JUN	92
400	1867117	5	1826	JUL	122
500	1903742	6	2191	AUG	153
600	1940267	7	2556	SEP	184
700	1976792	8	2922	OCT	214
800	2013317	9	3287	NOV	245
900	2049842	10	3652	DEC	275
1000	2086367	11	4017	JAN	306
1100	2122892	12	4383	FEB	337
1200	2159417	13	4748		
1300	2195942	14	5113		
1400	2232467	15	5478		
1500	2268992	16	5844		
1600	2305507	17	6209		
1700	2342031	18	6574		
1800	2378555	19	6939		
1900	2415079	20	7305		
2000	2451604	40	14610		
2100	2488128	60	21915		
2200	2524652	80	29220		
2300	2561176				
2400	2597701				

Table 3-1 Conversion Aid between Gregorian Calendar and JD [ZEM90, p. 138]

In 1967 the standard second was redefined as 9192631770 periods of the radiation of a specific transition in a caesium-133 atom. Since 1972 the time and frequency standards of the world have been based on International Atomic Time (TAI). TAI is defined in terms of standard seconds established in 1967. The Bureau International de l'Heure (BIH) uses astronomical observations to determine Coordinated Universal Time (UTC). Conventional civil time is based on UT1, a modification of UTC with corrections for Earth orbit and inclination and polar migration. UT1 is currently slowing down relative to TAI by a fraction of a second per year. A leap second is inserted or deleted in the UTC timescale at the last day of June or December when the magnitude of corrections reaches 0.7 s.

June 1972	Dec. 1972	Dec. 1973	Dec. 1974
Dec. 1975	Dec. 1976	Dec. 1977	Dec. 1978
Dec. 1979	June 1981	June 1982	June 1983
June 1985	Dec. 1987	Dec. 1989	Dec. 1990
June 1992	June 1993	June 1994	Dec. 1995

Table 3-2 Dates of Leap Second Insertion as of Jan 30^{th}, 1996

For most accurate timestamping of events since 1972, it is necessary to know when exactly the leap seconds were inserted in the UTC timescale. Leap second insertion is done at 23:59:59 resulting in a second 23:59:60 on that day. A leap second is deleted by omitting second 23:59:59. Leap seconds inserted prior to January 30th, 1996 are listed in Table 3-2. It is worthwhile to mention that the difference between TAI and UT1 was set to 10 s when TAI was established in 1972.

The NTP (network time protocol) timescale is based on UTC. Network time protocol is a time service in the Internet widely accepted and used by Internet Hosts. At 0^h January 1st, 1972 (MJD 41317.0) the NTP timescale was set to 2272060800 standard seconds since 0^h January 1st, 1900 (MJD 15020.0). In other words, the NTP timescale was set to 0 on 0^h January 1st, 1900 (MJD 15021.0). When a leap second is inserted in UTC and subsequently in NTP, the knowledge of all preceding leap seconds is lost. Establishing timestamps without any corrections prior to the latest leap second insertion would result in late timestamps. NTP is solely based on JDN.

GPS (Global Positioning System) time is a continuous measure of time [DANA95]. It started at midnight (00:00) January 6th, 1980 (MJD 44244). GPS time does not introduce leap seconds and is ahead of UTC by an integer number of seconds (11 seconds on March 1st, 1996). GPS time is controlled by the master control station to be within one microsecond (less leap seconds) of UTC. The GPS downlink message stream allows the receiver to calculate an even more accurate UTC time.

Notion	First Started	Leap Seconds	Chronoscopic	Time Source
TAI	1972	no	yes	NIST RF (CCIR band 7) NIST RF (CCIR band 5) Atomic clock
UTC	-	yes	no	GPS, DCF-77, NIST
NTP	0^h, January 1st, 1900	yes	no	Internet
GPS	0^h, January 6th, 1980	no	yes	GPS

Table 3-3 Time Notions Overview

A quick summary on various time notions is given in Table 3-3. For control applications GPS time would perform best. It is always guaranteed that the difference between two timestamps returns the proper time interval between the two events in standard seconds even without the knowledge of leap second insertion dates.

NTP time performs best in NTP time servers in the Internet. Leap seconds are inserted supplying the Internet user with UTC time (official time). Subtracting one NTP timestamp from another NTP timestamp without knowledge of the days when leap seconds were inserted can result in wrong time differences.

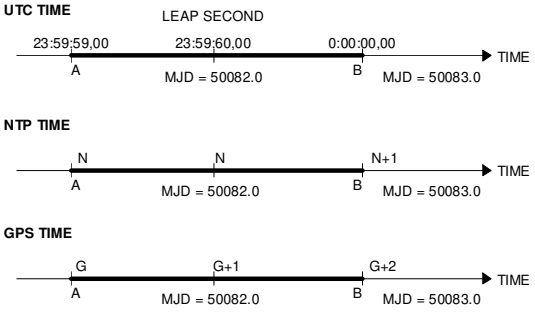


Figure 3-1 Leap Second Insertion in UTC and NTP on Dec. 31st, 1995

A short exercise, demonstrated in Figure 3-1, can clarify the problem with leap second insertion. Consider the example with a leap second inserted on MJD 50082.0 = JD 2450083 = Dec. 31st, 1995 at 23:59:59 as standard second 23:59:60 in UTC notion. If we pull out two timestamps, one at time A and one at time B and then subtract A from B we achieve the results from Table 3-4 in NTP time and in GPS time. Remember that the real-time between the 2 events A and B is 2 standard seconds.

Time Notion	Timestamp A	Timestamp B	(B - A)	Correct?
NTP	N	N+1	+1	no
GPS	G	G+2	+2	yes

Table 3-4 Influence of Leap Seconds in NTP and GPS Timestamps

3.2 TIME REPRESENTATION

The previous section introduced a wide variety of different time notions, all having their pros and cons in certain applications. Remember that the aim of this project is to establish a highly accurate common notion of time in a computer network. Without any doubt, the Internet can be called the most popular worldwide computer network known nowadays and the time notion used in the Internet is NTP-time, cf. [MILLS91, RFC1129]. Aside from the popularity of the Internet, the strength of NTP-time is the ease of doing computations and its extendibility (ENTP). Drawbacks can be seen in the process control world as NTP-time runs along with UTC, hence leap seconds are inserted and deleted according Table 3-2. But again ENTP-time format allows time representation with and without leap second correction.

3.2.1 NTP Format

All mathematical operations assumed in the network time protocol time format are two's-complement arithmetic with integer or fixed-point operands. An NTP timestamp is a 64-bit unsigned fixed-point number, with the integer part in the first 32 bits and the fraction part in the last 32 bits, cf. Figure 3-2. It is interpreted in standard seconds relative to UTC. When UTC began at 0^h January 1^{st}, 1972 (MJD 41318.0) the NTP clock was set to 2272060800.0 representing the number of standard seconds since 0^h January 1^{st}, 1900 (MJD 15021.0), cf. [RFC1129].

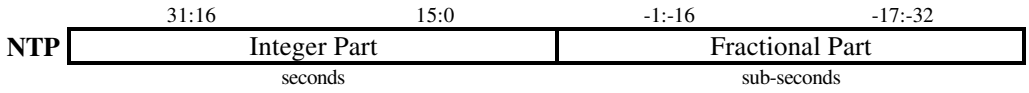


Figure 3-2 NTP Data Format [+31, -32]

Observe a statement from RFC1129, page 8 'The precision of this representation is about 232 picoseconds, which should be adequate for even the most exotic requirements.'. It seems like this thesis is even more exotic than David L. Mills could imagine in 1989, because of its required precision of 1.73 attoseconds. Look up Section 8.1 to verify the required chip internal resolution of 1.73 as.

Note that the most significant bit in the 64-bit timestamp was set some time in 1968 and that the timestamp field will overflow in 2036. Should NTP still be around in 2036, some external means have to be defined to qualify subsequent 136-year cycles.

In some particular cases, eg the NTP time server is rebooted or the clock synchronization algorithm first starts up, NTP timestamps may not be available. These invalid or undefined timestamps are indicated by setting all 64 bits in the field to zero. This all zero word accidentally appears also in the year 2036 for an interval of 232 ps, when the 64-bit field overruns and thus be considered invalid.

As a conclusion, NTP data format has shortcomings which are redressed in the Extended NTP and Modified Extended NTP data format discussed in the next section.

3.2.2 Extended NTP Format

Extended Network Time Protocol, short ENTP format, offers enhancements over the original NTP format which are inevitable for proper clock maintenance. Basically all characteristics from NTP are adopted in ENTP.

An ENTP timestamp is a 91-bit integer or fixed-point number, with the integer part in the first 32 bits and the fractional plus ultrafractional part in the remaining 59 bits, cf. Figure 3-3. Hence the enhanced resolution is 1.73 as. Table 3-5 lists all 91 ENTP-bit positions [+31, -59] plus another 5 sub-fractional bits [-60, -64] with their corresponding time resolution. From now on the notation [x, y] is used to reference all bits which lie between bit positions x and y.

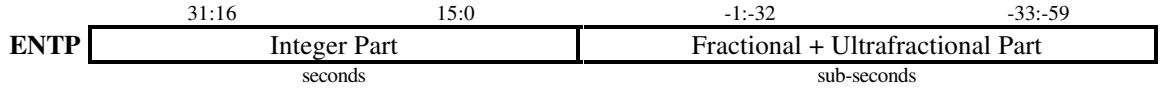


Figure 3-3 Extended NTP Data Format [+31, -59]

Another NTP drawback was mentioned earlier. The NTP timescale runs along with UTC, hence leap seconds are inserted and maybe deleted in future according Table 3-2. The effects when subtracting timestamps afflicted with leap seconds are shown in an example at the end of Section 3.1. Uncertainties of 1 s can result which are not tolerable in process control applications. One solution to that problem is to neglect leap second corrections in the integer part of the ENTP timestamp. This leads to the Modified Extended NTP (MENTP) time format. Of course, this MENTP timestamp is no longer direct compatible with an ordinary NTP timestamp, although a simple leap second addition in the fractional part of the ENTP timestamp redresses this incompatibility.

Integer Part				Fractional Part				Ultrafractional Part			
Bit	**Time**	**Bit**	**Time**	**Bit**	**Time**	**Bit**	**Time**	**Bit**	**Time**	**Bit**	**Time**
+31	68 years	+15	9.10 h	-01	500 ms	-17	7.62 µs	-33	116 ps	-49	1.77 fs
+30	34 years	+14	4.55 h	-02	250 ms	-18	3.81 µs	-34	58.2 ps	-50	888 as
+29	17 years	+13	2.27 h	-03	125 ms	-19	1.90 µs	-35	29.1 ps	-51	444 as
+28	8.5 years	+12	1.13 h	-04	62.5 ms	-20	953 ns	-36	14.6 ps	-52	222 as
+27	4.25 years	+11	34.1 min	-05	31.3 ms	-21	476 ns	-37	7.28 ps	-53	110 as
+26	2.13 years	+10	17.1 min	-06	15.6 ms	-22	238 ns	-38	3.64 ps	-54	55.5 as
+25	1.08 years	+09	8.53 min	-07	7.81 ms	-23	119 ns	-39	1.82 ps	-55	27.8 as
+24	194 days	+08	4.27 min	-08	3.81 ms	-24	59.6 ns	-40	909 fs	-56	13.9 as
+23	97.1 days	+07	2.13 min	-09	1.90 ms	-25	29.8 ns	-41	454 fs	-57	6.94 as
+22	48.5 days	+06	1.07 min	-10	976 µs	-26	14.9 ns	-42	227 fs	-58	3.47 as
+21	24.3 days	+05	32 s	-11	488 µs	-27	7.45 ns	-43	114 fs	-59	1.73 as
+20	12.2 days	+04	16 s	-12	244 µs	-28	3.72 ns	-44	56.8 fs	-60	0.87 as
+19	6.07 days	+03	8 s	-13	122 µs	-29	1.86 ns	-45	28.4 fs	-61	0.44 as
+18	3.03 days	+02	4 s	-14	61.0 µs	-30	931 ps	-46	14.2 fs	-62	0.22 as
+17	1.52 days	+01	2 s	-15	30.5 µs	-31	466 ps	-47	7.11 fs	-63	0.11 as
+16	18.0 h	00	1 s	-16	15.3 µs	-32	233 ps	-48	3.55 fs	-64	0.05 as

Table 3-5 ENTP Format

Depending on the deployment, both data formats ENTP and MENTP are used within the UTCSU project to represent local time. External events (eg data packet reception and transmission, application events, ...) are recorded in a reduced ENTP or MENTP data format, called Recorded ENTP data format (RENTP), with bounds set to [+31, -24] and displayed in Figure 3-4.

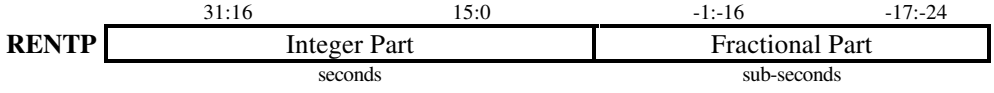


Figure 3-4 Recorded ENTP Format [+31, -24]

3.3 CRYSTAL OSCILLATORS

The family of quartz crystal oscillators is composed of a wide variety of frequency sources. Frequencies range between a few 10 kHz up to 200 MHz with regular bulk wave resonators and 1 GHz with resonators made by ionic or chemical etching. Simple quartz oscillators (XO) have a mean frequency stability of order 10^{-4} to 10^{-5}. Some of the most sophisticated oven controlled oscillators can reach short term stability in the range of 10^{-13}. Hence oscillator performance can differ by eight orders of magnitude. Volume space, power consumption and price of quartz oscillators also span several decades, cf. Figure 3-5.

3.3.1 Definitions and Characteristics

Over the last decade, many kinds of quartz oscillators were developed to satisfy new application needs. A classification involves clock oscillators (XOs), temperature compensated oscillators (TCXOs), digital temperature compensated oscillators (DTCXOs) or microcomputer compensated oscillators (MCXOs), voltage controlled oscillators (VCXOs), tactical miniature oscillators (TMXOs), miniature oven controlled oscillators (MOCXOs) and oven controlled oscillators (OCXOs). Hence the choice for a quartz crystal oscillator is always a compromise between short term stability, spectral purity, aging, temperature stability, supply voltage sensitivity, power consumption, volume space, price, etc. In addition environmental influences such as accelerations, vibrations, pressure, irradiation, acoustic noise, magnetic fields and humidity must be taken into account [GAG90].

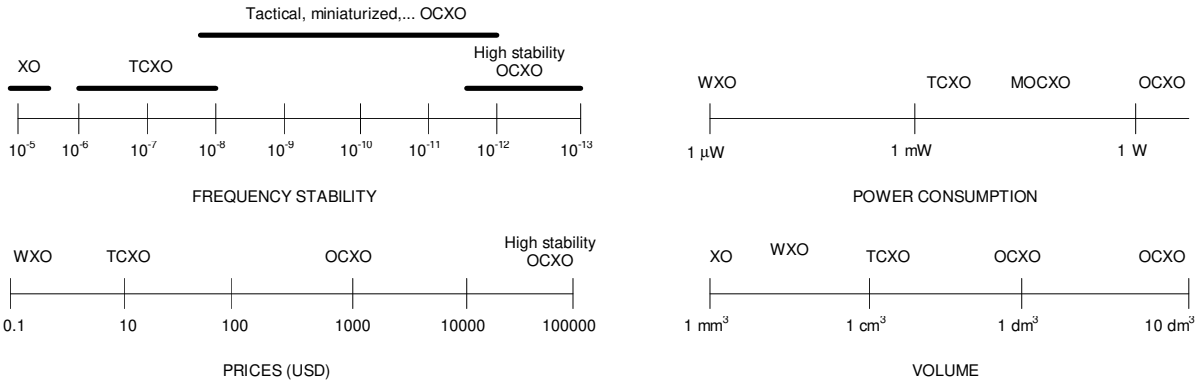


Figure 3-5 Quartz Crystal Oscillator Characteristics

Later on, one will experience a classification of clock oscillators as a function of stability, jitter and wander. Their definitions are given in [BELL86] and they are summarized below.

- *Stability* refers to the systematic variation in frequency with time.
- *Jitter* or timing jitter refers to short-term variations in frequency with components greater than 10 Hz.
- *Wander* refers to long-term variations in frequency with components less than 10 Hz.

3.3.2 No Compensation (XO)

Uncompensated quartz crystal oscillators reach a stability per day of approximately 1×10^{-6}/°C. Don't ask about aging and drift. Their low power consumption, small volume space and very low price make them popular and suitable for many applications. Extending their applicability to oscillators for distributed real-time clocks sets a high standard on the oscillator design and requires time-processors with the fitness of the UTCSU.

3.3.3 Temperature Compensation (TCXO, DTCXO)

Analog TCXOs compensate temperature influences with a network composed of three thermistors and a few resistors. The oscillator voltage, to keep the output frequency constant, versus temperature characteristic is recorded and the thermistor network designed to reproduce the same characteristic. Compensation over a wide temperature range is possible only for resonators with a high mechanical coupling factor. Today the high price of analog TCXOs is due to the calibration of the thermistor network. Stability between $\pm 1 \times 10^{-6}$ and $\pm 5 \times 10^{-7}$ is achieved.

Another solution is to replicate the compensation curve with linear segments of an operational amplifier transfer function. But whatever the compensation is, external resistors must be adjusted.

Digital temperature compensated oscillators (DTCXOs) are based on dual-mode oscillators. The resonator is at the same time frequency reference and temperature probe. Temperature calibration is still necessary but the weighting values can be kept in memory and do not need external adjustments. Frequency stability over a temperature range from - 30 °C to + 80 °C is in the order of 10^{-7}.

	Compensation	Stability	Temperature Range
analog	thermistors	$\pm 2x10^{-6}$	- 40 °C to + 80 °C
	polynomial	$\pm 5x10^{-7}$ to $\pm 10^{-6}$	- 40 °C to + 85 °C
digital	DTCXO	$\pm 2x10^{-8}$ to $\pm 2x10^{-7}$	- 30 °C to + 80 °C
	MCXO	$\pm 2x10^{-8}$ to $5x10^{-8}$	- 55 °C to + 85 °C

Table 3-6 Performance of Analog and Digital TCXOs [GAG90]

3.3.4 Voltage Controlled (VCO)

Voltage controlled oscillators (VCOs) pull the frequency of a crystal oscillator with a voltage controlled capacitor in series with the resonator. The frequency dependency is given by the relation

$$f = f_0 \left[1 + \frac{C_1}{2\left(C_0 + \Gamma\right)} \right] \qquad \text{Equ. 3-1}$$

where C_1, C_0 and Γ represent the motional, parallel and variable capacitors. Quartz oscillators have a low temperature sensitivity and allow frequency pulling between 10^{-4} and 10^{-3}. Using lithium tantalate resonators increases the pulling rate to 1 % due to their higher electromechanical coupling factor. New designs are expected with berlinite crystals ($AlPO_4$), which reconcile good electromechanical coupling and a low temperature dependency.

3.3.5 Oven Controlled

In high performance applications, the influence of temperature must be reduced by putting the resonator in a temperature controlled oven. Advantageous is the high temperature stability, the disadvantages are the increased power consumption of several watts, the large volume space and the higher price. A compromise is the tactical miniature crystal oscillator (TMXO), designed for fast warm up and low power consumption of 100 mW. It reaches short-term stability near 10^{-11} over 1 s.

High stability oven controlled oscillators (OCXOs) are designed without to many restrictions on volume and power consumption. Their main goals are short- and long-term stability and spectral purity. The resonator temperature dependence is given by

$$\frac{f - f_0}{f_0} = a_1\left(T - T_0\right) + a_2\left(T - T_0\right)^2 + a_3\left(T - T_0\right)^3 + \tilde{a}\,\frac{dT}{dt} \qquad \text{Equ. 3-2}$$

For SC-cut crystals with $a_1 = 0$, $a_2 = 10^{-9}/°C^2$ and $\tilde{a} = 10^{-7}$ s/°C, the static behaviour depends mainly on the second term and the dynamic behaviour on the last term. In order to keep the frequency variation below 10^{-13}, the temperature fluctuations in the oven must not exceed 0,1 °C for the static part and 1 µ°C/s for the dynamic part.

3.4 GPS OSCILLATORS

Some GPS time receivers [NAV95, STEL95] provide a 10 MHz GPS output signal as well as a 1PPS output signal. The 10 MHz signal is derived from the frequency information received from the space vehicle (SV) atomic clocks. As long as GPS is up and running, this 10 MHz signal is very accurate and best suited to source a local hardware clock. In times of GPS outages, performance degrades to numbers inherent to the local GPS receiver oscillator design.

3.5 ATOMIC OSCILLATORS

Presently available atomic oscillators are based on the transitions of the hydrogen, caesium and rubidium atoms. Table 3-7 lists the characteristics of these oscillators in comparison to various types of quartz crystal oscillators found in electronic equipment. Caesium oscillators are used worldwide for national primary frequency standards while local clocks used in computing equipment almost always rely on uncompensated crystal oscillators.

Oscillator Type	Stability (per day)		Drift / Aging
Hydrogen maser	$2x10^{-14}$		$1x10^{-12}$ / year
Caesium beam	$3x10^{-13}$		$3x10^{-12}$ / year
Rubidium gas cell	$5x10^{-12}$		$3x10^{-11}$ / month
Oven controlled crystal	$1x10^{-9}$	(0-50 °C)	$1x10^{-10}$ /day
Digital compensated crystal	$5x10^{-8}$	(0-60 °C)	$1x10^{-9}$ /day
Temperature compensated crystal	$5x10^{-7}$	(0-60 °C)	$3x10^{-9}$ /day
Uncompensated crystal	~$1x10^{-6}$	(per °C)	don't ask

Table 3-7 Characteristics of Standard Oscillators

The drift/aging column in Table 3-7 shows the maximum offset from nominal standard frequency due to systematic mechanical and electrical characteristics. This offset is not constant for crystal oscillators, which results in a gradual change in frequency over time, called aging. Even temperature compensated crystals must be periodically compared to a primary standard to maintain the highest accuracy. The stability column shows the maximum variation in frequency per day due to circuit noise and environmental influences.

The industry has agreed on a classification of clock oscillators as a function of minimum accuracy, minimum stability and other factors (viz stability, jitter and wander, cf. Section 3.3.1) [BELL86]. This classification defines the oscillator stratum with the more accurate oscillators assigned the lower strata and the less accurate oscillators the higher strata as shown in Table 3-8.

Stratum	Min. Stability (per day)	Min. Accuracy (per day)
1	not specified	$1x10^{-11}$
2	$1x10^{-10}$	$1.6x10^{-8}$
3	$3.7x10^{-7}$	$4.6x10^{-6}$
4	not specified	$3.2x10^{-5}$

Table 3-8 Oscillator Stratum Classification

The design, operation and maintenance of stratum-one oscillators is assumed to be consistent with national standards. Often caesium oscillators or precise crystal oscillators synchronized via LORAN-C to NIST standards make up the stratum-one frequency source.

3.6 PRINCIPLES OF RATE AND STATE CORRECTION

From the previous sections about clock oscillators one can experience that every oscillator design is affected with drift and aging. Hence any hardware clock design based on these type oscillators must be capable of adjusting phase and frequency. Observe that state correction refers to phase adjustment and rate correction refers to frequency adjustment.

Digital clocks consist of a frequency source, the clock oscillator, and a counter which counts time steps. The mathematical relation between the clock c(t) at time t and infinitesimal time steps s(τ) is given by

$$c(t) = \int_0^t s(\tau) d\tau \qquad \text{Equ. 3-3}$$

Equation 3-3 can be approximated by the following discrete sum

$$c[n] = \sum_{k=1}^{n} T \cdot s[k]. \qquad \text{Equ. 3-4}$$

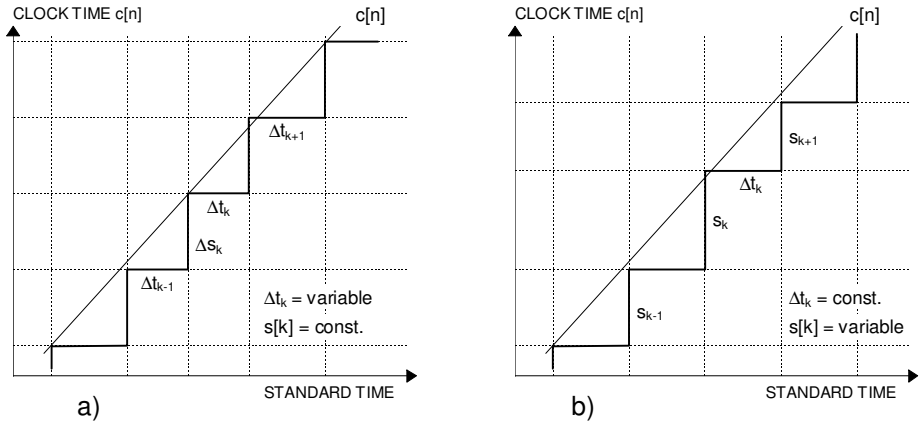


Figure 3-6 a) Clock Adjusted by Step Width Δt b) Clock Adjusted by Step Height s[k]

If c[n] follows standard time, either the number of time steps n and therefore the step width as shown in Figure 3-6 a) or the step height s[k] as shown in Figure 3-6 b) can be adjusted.

Varying the step width Δt refers to analog frequency and phase adjustment techniques whereas varying the step height results in discrete time designs found in the single counter approach and the adder-based approach later on.

Consider the integral from Equation 3-3. Approximating this integral by a sum, one obtains

$$c(nT) \approx T\,s(T) + T\,s(2T) + \cdots + T\,s(nT - T) + T\,s(nT) \qquad \text{Equ. 3-5}$$

The sum of the first n - 1 terms on the right side equals c(nT - T). Hence

$$c(nT) - c(nT - T) \approx T\,s(nT) \qquad \text{Equ. 3-6}$$

To determine c(nT), it suffices to solve the recursion equation

$$c[n] - c[n-1] = T\,s[n] \qquad n > 0 \qquad \text{Equ. 3-7}$$

with s[n] = s(nT) and the initial condition c[0] = 0. To solve Equation 3-7, one can use the digital system of Figure 3-7. The reader should keep this structure in mind, it is the basic architecture used in Section 3.6.3.

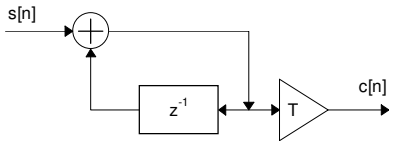


Figure 3-7 This Digital System Solves Equation 4-7

We wish to solve Equation 3-7 for n ≥ 0. Hence we do a *z*-Transform on both sides and solve for C(z). For simplicity, we can neglect the constant factor T.

$$c[n] - c[n-1] = s[n] \xleftrightarrow{z} C(z) = \frac{1}{1 - z^{-1}} S(z) + \frac{c_{-1}}{1 - z^{-1}} \qquad \text{Equ. 3-8}$$

Thus,

$$C(z) = C_{\alpha}(z) + C_{\beta}(z) \qquad \text{Equ. 3-9}$$

where

$$C_{\alpha}(z) = \frac{1}{1 - z^{-1}} S(z) \qquad \text{Equ. 3-10}$$

is the zero-state response and

$$C_{\beta}(z) = \frac{c_{-1}}{1 - z^{-1}} \qquad \text{Equ. 3-11}$$

is the zero-input response of Equation 3-7. The inverse $c_\beta[n]$ of $C_\beta(z)$ is the solution of the homogeneous equation c[n] - c[n-1] = 0

$$c_\beta[n] = c_{-1} z_1^{\,n} = c_{-1} \qquad \text{Equ. 3-12}$$

and the inverse *z*-Transform of $C_\alpha(z)$ results in a sum

$$c_\alpha[n] = \sum_{k=0}^{n} s[k] \qquad \text{Equ. 3-13}$$

The following subsections summarize different techniques how to design a phase and frequency adjustable local clock.

3.6.1 Analog Technique

A voltage controlled oscillator (VCO) allows to vary the frequency of the oscillator itself. Applied in a phase-locked-loop (PLL), the VCO can track an external clock reference in frequency and phase. VCOs are hard to implement in a pure digital CMOS process. Note that the ENTP format forces the driving frequency to relate to one of the reciprocals of the ENTP-bit time equivalents, cf. Section 3.2.2. For example, to comply to ENTP-bit -23, a frequency of 119.21 ns^{-1} = 8.388608 MHz is required.

3.6.2 Single Counter Approach

Assume the input sequence s[n] in Figure 3-7, or in other words, the right hand side in Equation 3-7 is a simple sequence of zeros and ones.

$$s[n]= 1,0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,0,1,1,0,1,0,1,0,1,0,1,\ldots \quad n \geq 0 \qquad \text{Equ. 3-14}$$

The adder from Figure 3-7 degrades to a gated counter and the final clock architecture consists of a clock oscillator connected to the gated counter, cf. Figure 3-8. Several limitations and problems occur.

- Higher input frequency required for insertion.
- To support the ENTP data format from Figure 3-3, the oscillator frequency must comply with a multiple of the inverse of an ENTP-bit time equivalent.
- Another drawback is not only a higher bandwidth, but also an unsatisfactory clock correction granularity. A small rate change means a long s[n] until a one or a zero gets modified, which entails very long amortization intervals.
- Uniformly scattering the zeros and ones in s[n] over the amortization/resynchronization interval to hold linearity must be done in hardware. This is not easy to accomplish, particularly with programmable resynchronization interval lengths.
- Leap second correction requires a one-second jump in both directions. A simple counter is far from performing this action.

An implementation that suffers from all the limitations listed above can be found in [KO87].

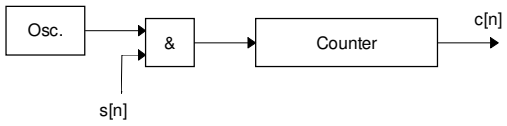


Figure 3-8 Single Counter Approach

3.6.3 Adder-Based Approach

A novel adder-based digital clock is proposed in this subsection. The clock oscillator output drives an adder input instead of the simple counter from the previous subsection. Employing an adder

gives the freedom to add a particular amount (clock step) to the clock register (NTPTIME) at every oscillator tick. The adder input signal s[n] is a sequence of clock steps s[k] in the appearance of

$$s[n]= s[0], s[1], s[2], s[3], \ldots \quad n \geq 0 \qquad \text{Equ. 3-15}$$

as shown in Figure 3-9. Observe the close relationship with Figure 3-7.

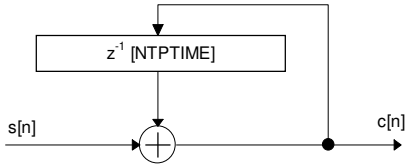


Figure 3-9 Adder Based Clock

A rate change can be achieved by slightly modifying the clock steps. An increase advances the clock speed while a decrease slows it down. This goes in effect almost instantly and holds up linearity.

Changing the phase of the clock is achieved by changing the clock rate over a given period. A positive phase shift is compensated by decreasing the clock step and vice versa. Observe that all rate and state changes proceed monotone as long as only positive numbers are added and linear as long as clock steps are not altered. But still, even non monotone changes for leap second insertion are possible by adding a negative clock step.

For better understanding, the notion of granularity for this clock has to be expanded in the following way:

- *State granularity* G_s refers to the smallest time unit represented by s[n] or c[n].
- *Time granularity* G_t refers to the time period between two updates of the local time register.

With a counter approach, the values for state granularity and time granularity are the same. However, there is nothing magic about having a much better state granularity than time granularity. The extra cost for this approach are a die size consuming full adder instead of the simple counter and the state granularity must be enhanced to deal with small rate changes. Enhancing G_s puts extra cost on the extended internal time register (NTPTIME) c[n] and the high resolution clock step register(s) s[n].

With the adder-based approach, it is possible to break the 2^k frequency constraint enforced by the ENTP format. Since every clock step value can be used, even a decimal step of 100 ns for a 10 MHz source can be selected. Note, however, to avoid that truncation errors mess up the intrinsic rate behaviour of the physical clock, ultrafractional ENTP bits must be added to the time and step registers.

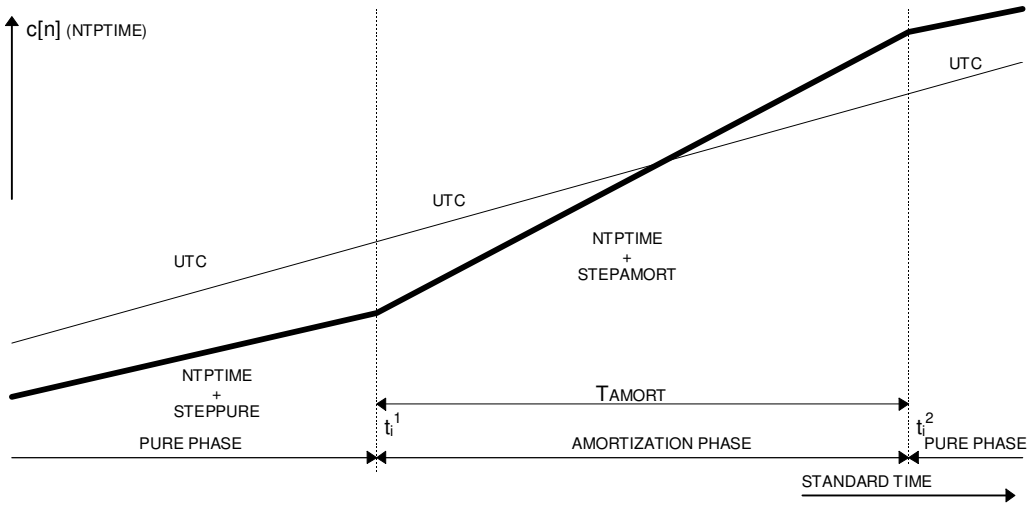


Figure 3-10 Hardware Clock (NTPTIME) with Rate and State Correction

Figure 3-10 specializes the general picture of Figure 2-7 by introducing a NTPTIME-register to hold the local clock, a STEPAMORT-register for the clock steps during amortization phase and a STEPPURE-register for the clock steps during pure phase.

The reader will find the adder-based approach implemented in the local time unit in Section 8.1 and in the accuracy unit in Section 8.2.

3.6.4 Interleaving Adder and Counter

Note that in the adder-based approach, a new clock step must be added to the NTPTIME-register on every clock tick. Lack of performance or limited die size could lead to timing problems in the adder, especially with a low G_t, cf. [HL95]. To relax the timing constraint for the adder, one can think about interleaving a simple counter and a multiple-cycle adder. Multiple cycles are granted to the adder to carry out the sum, during which a simultaneously running counter delivers local time. On completion of the addition, the counter is updated with the newly computed clock value. During the update, one must be careful not to violate clock monotonicity. Of course, the frequency of the oscillator must comply to the ENTP-bit equivalencies.

CHAPTER 4

4 GPS SUPPORT

Die Zeit befiehlt's, ihr sind wir untertan.
Shakespeare, König Heinrich IV

NAVSTAR Global Positioning System (GPS) is a constellation of satellites that orbit the earth twice a day, transmitting precise timing information. These transmissions can be collected by any GPS receiver at no charge and at any hour, and used to produce latitude, longitude, altitude and accurate time. NAVSTAR is an acronym formed from NAVigation System with Time And Ranging.

The GPS space segment constellation consists of 24 satellites [ME96] orbiting the Earth in six fixed planes that are inclined 55° from the Equator. Each space vehicle (SV) orbits at 10898 nautical miles altitude. A three-dimensional positioning, 24-hour, worlwide coverage is achieved with this constellation.

The system was developed and deployed by the US Department of Defense (DoD) to provide continuous, worldwide positioning and navigation data to US military forces around the globe. GPS has also broad civilian and commercial applications.

4.1 HOW GPS WORKS

The user segment is the collection of all GPS receivers. Navigation with the receiver is accomplished by passive multilateration. More simply, the GPS receiver's position is determined by the geometric intersection of several simultaneously observed ranges from satellites with known coordinates in space. A mimimum of four range observations are required to mathematically solve for four unknown receiver parameters (ie latitude, longitude, altitude and time). If the position is known, as is the case in a stationary GPS time receiver, by theory only a single range observation can track time, hence only a single satellite view is required.

The GPS navigation message is the data supplied to the user from a satellite. Signals are transmitted at two L-band frequencies, L1 at 1575.42 MHz and L2 at 1227.6 MHz to permit corrections of ionospheric delays. The L1 carrier is modulated with a 10.23 MHz precision ranging (P-code) signal and a 1.023 MHz coarse aquisition (C/A code) ranging signal. The L2 carrier is modulated with the P-code only. Both the L1 and L2 signals are continuously modulated with a data bit stream (navigation message) at 50 bits/s. The P-code is a pseudo random (PR) sequence with a period of 38(+) weeks and the C/A code is a shorter PR sequence of 1023 bits resulting in a period of 1 ms. Miltitary users have access to the accurate P-code while civilian users are provided with the C/A code [LH81].

Selective Availability (SA) inserts random errors into the C/A code and limits the system accuracy from 25 m with SA off to 100 m with SA on. SA degrades timing accuracy from 60 ns without SA to 330 ns with SA on. A description of different failure modes can be found in [DANA95].

4.2 ANALYSIS ON GPS INTERFACES

Off-the-shelf GPS receivers set forth a pretty similar interface to elicit position and time information. Positions (P) are pulled out of the GPS receiver via a standard RS-232C interface. Accessing accurate time is a two step process. Date and timestamp (T) with a resolution of 1 s is obtained through the serial interface while the exact start of each second is marked with either the rising or the falling edge of a one pulse per second (1PPS) signal. Some receivers use the rising edge others the falling edge to indicate the second and others use a time-mark. Time-marks don't align their rising edge with the start of the standard second but provide the offset (O) between the standard second and the time-mark signal, cf. Figure 4-1.

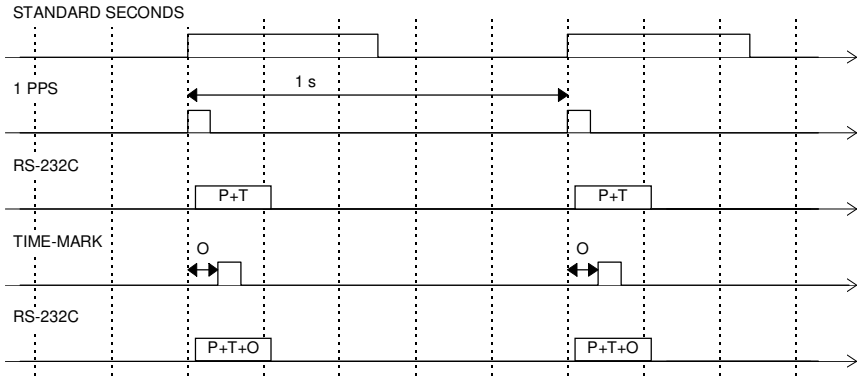


Figure 4-1 GPS Receiver Time Interface

In order to account for antenna cable length, some receivers compensate cable delays at the 1PPS output. For example the Motorola ONCORE receiver allows to adjust the offset of the 1PPS output and the antenna cable delay in 1 ns steps relative to the GPS time or UTC. The rising edge of the 1PPS is placed so that it corresponds to the time indicated by the following equation,

1PPS Rising Edge Time =
Measurement Epoch Offset + 1PPS Offset - 1PPS Cable Delay Equ. 4-1

Some GPS receivers provide a 10 MHz reference frequency derived from their local oscillator and disciplined to the atomic clocks in the SVs. Frequency stability of 10^{-11} (24 hour average) and $5*10^{-12}$ (7 day average) can be achieved (eg Stellar GPS Clock 100).

Most implementations offer GPS-time and UTC, cf. Section 3.1. Pending leap seconds in UTC are indicated several weeks in advance before they go in effect.

There are different ways how a GPS receiver shows its health condition. Some designs use a dedicated signal which goes high or low if the receiver clock is locked to the satellite clock. Some receivers turn on or off their 1PPS output depending on the clock condition. Different strategies are deployed. The 1PPS output is active when tracking at least one satellite, or the 1PPS output is active only when the RAIM algorithm confirms time solution errors within user defined alarm limits. Receiver Autonomous Integrity Monitoring (RAIM) is the term given to those techniques that use the GPS signals from many SVs to provide measure the level of correctness of the individual signals.

The standard GPS time interface found on off the shelf GPS receivers comprises a subset of the following hardware components:

- A bidirectional *RS-232C port* to obtain navigation and time information and to configure the receiver.
- A *1PPS output* or a Time-Mark output disciplined to GPS time or UTC
- A *10 MHz reference* frequency output disciplined to the SV atomic clocks
- A *status line* indicating the GPS receiver health condition (locked/unlocked)

4.3 TIME EXCHANGE PROTOCOLS

The serial GPS interface can run different protocols. Almost every company uses its proprietary home made GPS protocol for configuration and also to output position and time. In addition some industry standards have evolved that are widely accepted by the GPS receiver providers.

4.3.1 Proprietary Solutions

Almost every company implements its own protocol, which can handle all configuration and status messages for their GPS receivers while industry standard protocols by their nature only support a subset. However, these proprietary solutions are often designed to run transparently with industry standard protocols like the NMEA-0183. Some proprietary solutions are listed below.

- MBF: Motorola Binary Format
- TSIP: Trimble Standard Interface Protocol
- TAIP: Trimble ASCII Interface Protocol
- PTEL: ASCII sentence based protocol, used for NavSymm Timing and Frequency Receivers

4.3.2 NMEA-0183

Output of data in NMEA-0183 standard format allows interface via the RS-232C port to an electronic navigation instrument that supports the specific messages. All NMEA messages are formatted in sentences that begin with ASCII $ and end with ASCII <CR><LF>. A five-character address occurs after the ASCII $. The first two characters are the talker ID (which is GP for GPS equipment), and the last three characters are the sentence formatter or message ID from Table 4-1. Any number on fields and an optional checksum can occur in the sentence as long as the total number of characters does not exceed 79. Fields within the message are delimited by the ASCII comma. The checksum is calculated by XORing the 8 data bit of each character in the sentence between, but excluding, the $ and the optional (*) or (CS) checksum. The high and low nibbles of the checksum are sent as ASCII characters.

Message	Message Contents
GPGGA	GPS Fix Data
GPGLL	Geographic Position - Latitude/Longitude
GPGSA	GPS DOP and Active Satellites
GPGSV	GPS Satellites in View
GPRMC	Recommended Minimum Specific GPS/TRANSIT Data
GPVTG	Track Made Good and Ground Speed
GPZDA	Time and Date

Table 4-1 GPS NMEA Output Message Examples

4.3.3 LORAN

This particular output message format is intended to emulate the position and time status message string from a LORAN receiver. It allows to use the GPS receiver to replace the LORAN receiver in embedded positioning or time applications.

4.3.4 RTCM SC-104

The Radio Technical Commission for Maritime Services (RTCM) recommends standards for differential GPS services. "RTCM Recommended Standards For Differential GPS Services", prepared by RTCM Special Committee No. 104 (RTCM SC-104), defines a communication protocol for sending GPS differential corrections from a differential reference station to remote GPS receivers.

4.4 RECEIVER DESIGNS

The architecture of a typical time-based GPS receiver is shown in Figure 4-2. Typically, the manufacturer uses an OEM GPS receiver designed for positioning purposes but adds a *1PPS* output.

The way the *1PPS* output is derived from the OEM product involves jitter from 20 ns to 60 ns rms. A frequency output is generally derived from the jittery 1PPS output by slaving an additional oscillator in a Phase-Locked-Loop (PLL). The performance of the frequency output depends on the 1PPS signal quality, the oscillator and the PLL design.

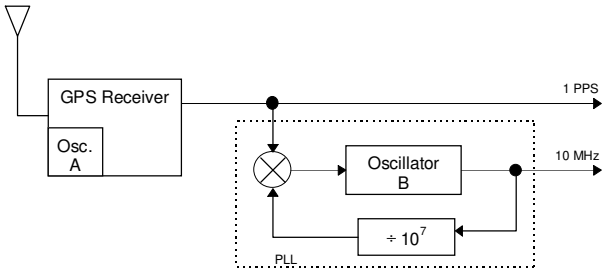


Figure 4-2 Typical Time-Based GPS Receiver Design

A GPS Clock has an architecture optimized for frequency applications, hence it should be *frequency-based* as shown in Figure 4-3. Instead of slaving an oscillator to the 1PPS output of a GPS receiver, the GPS Clock slaves the oscillator of the GPS receiver to the satellites clock and derives the 1PPS from the locked oscillator frequency output. As a result, the GPS Clock output is more stable, more accurate and more precise, cf. [OK93].

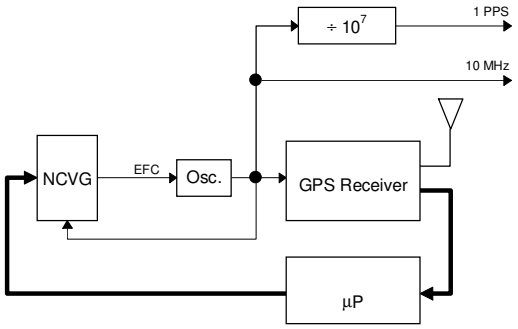


Figure 4-3 GPS Clock Receiver Design

One can observe from the previous sections that GPS time receivers offer only humble timing features. Their superior characteristic, the highly accurate time source, must be upgraded with more advanced timing features to make up a real real-time clock. A list of desirable features follows.

- Timecode generation
- Event timestamping
- Integrity monitoring
- Fault-tolerance

If we find a way how to chain together the UTCSU and a GPS time receiver, we could expect all the advantages one can think of. The GPS receiver provides the accurate time and the UTCSU all other features listed above.

4.5 INTERFACE TO UTCSU

If one steps back to Section 4.2, two types of highly accurate time interfaces were introduced. A 10 MHz frequency output and the 1PPS output. Both types can be used to interface the GPS receiver to the UTCSU.

4.5.1 10 MHz-Interface

Thanks to the adder-based approach of how to maintain local time in the UTCSU, even a decimal frequency (10 MHz) can be used to drive the local clock and still maintain the extended NTP time

format, see Section 3.2.2. Hence the 10 MHz frequency output of the GPS receiver can be used as the primary clock source for the UTCSU. It replaces the quartz crystal oscillator and the whole UTCSU is running from the highly stable GPS reference frequency as shown in Figure 4-4.

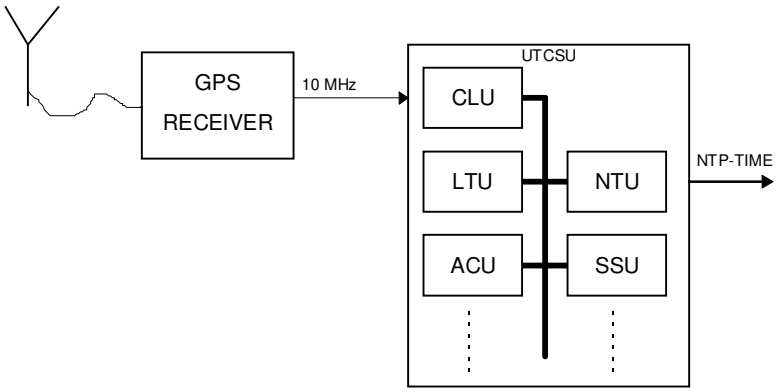


Figure 4-4 Interfacing the 10 MHz Output to the UTCSU

The rate and state adjustable local clock degrades to a state adjustable clock. Rate and state corrections are no longer necessary as the GPS reference frequency is disciplined to the highly accurate SV atomic clocks; but may still be used to efface sporadic GPS outages and during node start-up.

In conjunction with a low jitter GPS-Clock, this arrangement provides the most jitter free timestamps at the NTP-time interface output. The reason therefore is that all flip-flops in the UTCSU are driven by the low jitter 10 MHZ GPS reference frequency. Of course, the chip layouter has to put special attention on the clock network to keep the clock skew low.

The 10 MHz-interface does not provide a global time like UTC or GPS time. If this is desired, the 1PPS-interface, covered in the next section, should be combined with the frequency output.

4.5.2 1PPS-Interface

The 1PPS GPS receiver output is a very powerful interface but requires additional logic in the UTCSU. Basically there are two reasons when to interface to the 1PPS GPS receiver output.

- The GPS receiver in mind only provides a 1PPS output. Section 4.6 will give an overview over industry standard GPS receivers and lists their interface options.
- The application in mind requires a global time notion like UTC. In this case, the 1PPS output delivers the start of the standard second and the RS-232C interface the date and time information.

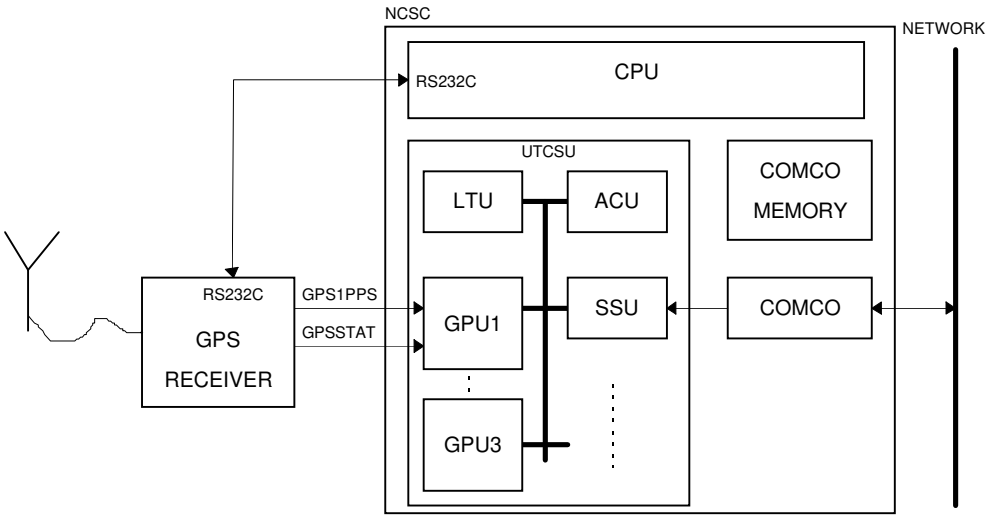


Figure 4-5 Interfacing the 1PPS Output to the UTCSU

A new strategy must be found how to interface the 1PPS output to the UTCSU. One can think of the once per second pulses as external events which should be recorded in order to timestamp these events. As soon as the 1PPS timestamp, which is derived from the local UTCSU time, is recorded, the clock synchronization algorithm in the clock synchronization layer compares the UTCSU second with the 1PPS standard second and computes proper rate and state corrections for the adder-based local clock. Observe that this GPS timestamp behaves for clock synchronization the same way as a clock synchronization packet received from the synchronization subnet with very high accuracy.

The GPS timestamp logic is summarized in the GPS Unit (GPU) and is discussed in Section 8.4. Figure 4-5 shows the 1PPS output connected to the GPS Unit. If available, the GPS status signal (GPSSTAT) can be recorded in the GPU and it could raise an interrupt to indicate a GPS loss.

Actually the UTCSU comprises 3 equally powerful GPS Units (GPU1-GPU3), cf. Figure 4-5. This allows to do clock synchronization with redundant GPS receivers or to set up a GPS evaluation system as discussed in the next section.

4.6 GPS TIME RECEIVER EVALUATION

We are currently carrying out an evaluation of the 6 industry GPS receivers from Table 4-2.

Manufacturer	Model	10 MHz	1PPS SA on	1PPS SA off	1PPS Type	Status
NavStar	NTFR_S	yes	-	-	↑, CMOS	yes
Motorola	Oncore	no	130 ns	-	↑, CMOS	no
Stellar	GPS Clock 100	yes	300 ns	100 ns	↑, TTL	no
Magellan	OEM/PS-5	1 kHz	-	± 100 ns	↑, TTL	no
Trimble	SVeeSix-CM2	no	1 µs	1 µs	↓, open coll.	no
Rockwell	MicroTracker	no	-	± 1 µs	↑, TTL, Timemark	no

Table 4-2 Industry Standard GPS Receivers (Factory Numbers)

The test arrangement is shown in Figure 4-6. A highly accurate temperature controlled rubidium clock provides the reference frequency for determining the errors of the 1PPS receiver outputs. The control unit in the rack system on the left prepares the 1PPS GPS outputs for data aquistion in a hp 16500B based logic analyzer. To achieve a continuous measurement, the hp is equipped with two identical logic analyzer boards, which alternatly capture data and send it to PC 1 via the HP interface bus. Data are transferred to PC 2 for preprocessing and finally stored on a SUN workstation. A technical report on this 72 days evaluation is going to appear in Q2 1996.

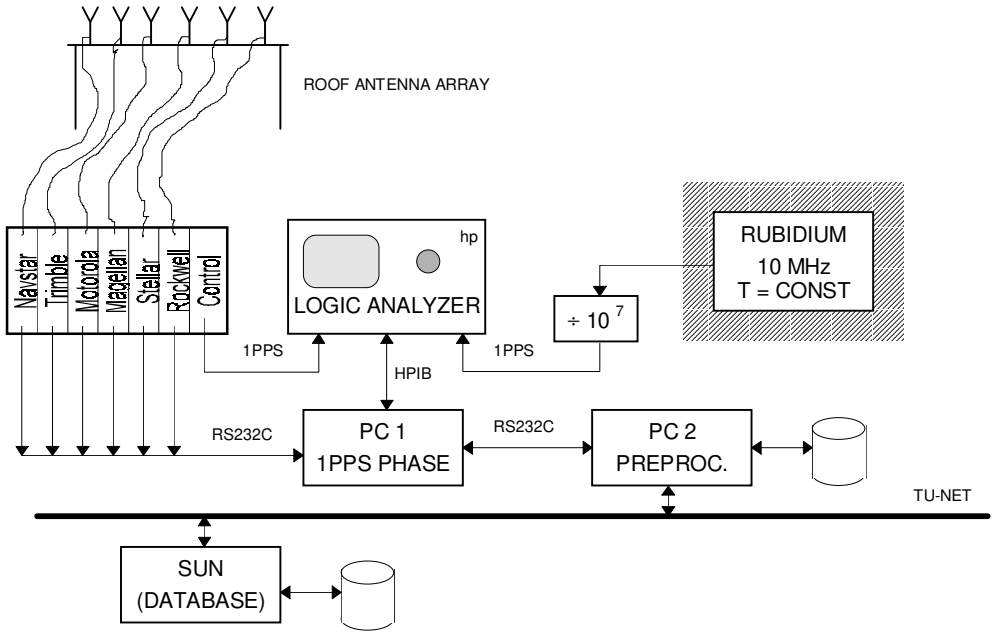


Figure 4-6 GPS Receiver Evaluation

CHAPTER 5

5 BITE & FAULT-TOLERANCE

Abgeredet vor der Zeit, gibt nachher keinen Streit.
Deutsches Sprichwort

BITE is the acronym for Built-In Test Equipment. As the headline already says, this chapter talks about integrated test logic and advanced services to improve correct system behaviour. Concepts of post-fabrication testing and testing during the device integration phase are addressed. Nine test features are implemented in the UTCSU to support the debugging phase as well as fault-tolerant operation.

It starts out that a classification where and when a failure may occur, leads to three phases. Defects on the chip due to process variations during fabrication are detected in the post-fabrication phase. Hundreds of techniques are known in the literature but only a handful is supported with logic synthesis tools. Basically scan-path resp. test-multiplexers to allow chip characterization and dynamic error tracking are inserted.

Next phase is the device integration phase which is best covered by the IEEE 1149.1 boundary-scan standard. The last phase, the operational phase, provides debugging features and support for reliable operation.

5.1 TEST CONCEPT

Testing of a system is an experiment in which the system is exercised and its resulting response is analyzed to ascertain whether it behaved correctly. If incorrect behavior is detected, a second goal may be to diagnose or locate the misbehavior [ABF90].

Physical faults (viz fabrication defects, physical failures, fabrications errors) can be classified according to their stability in time as

- *permanent*, ie being present all the time after their occurrence;
- *intermittent*, ie existing only for some time;
- *transient*, ie a one-time occurrence due to extraordinary environmental factors.

A comprehensive test concept has to address all three types and all three phases of possible error influences as shown in Figure 5-1. Right after chip fabrication at the silicon foundry, the chip goes into a test-house to check for permanent physical faults. The designer has to integrate the test logic and must provide test patterns for post-fabrication testing.

Good devices go to the device integrator for PCB assembly, devices that didn't pass could go back to the design-house for error tracking. Assembled PCBs must be tested for opens, shorts, bridges and bad devices, hence test logic for board testing must be added to the design. Components that didn't pass may be diagnosed and their misbehavior located by means of additional test logic.

During system integration, debugging features are a welcome help to figure out different design specification interpretations or simply to certify the correct behavior. Supervisory facilities can make use of the debugging aids as well. At least 7 test entities are implemented in one UTCSU which support proper system operation. They are listed below.

- An *error detection and correction unit* (EDCU) protects every timestamp that is transferred through the network, to the CPU or to the external world.
- A *version control* register makes sure, that the software version and the chip version match.
- Several UTCSUs can work together in a *redundant synchronous operating* mode. This could improve fault-tolerance in safety-critical applications.
- Two *MISR* (multiple input signature register) compute a signature for a block of time- and accuracystamps.
- Two *blocksum* adders compute the sum for a block of time- and accuracystamps.
- A special *reliable mode* enhances firmness against metastability problems in the input synchronizer stages.
- *External event testing* can simulate external events on primary input pins, hence provides self-test capabilities for some units.
- *Hardware- and software-snapshots* provide information on the current state of the UTCSU.

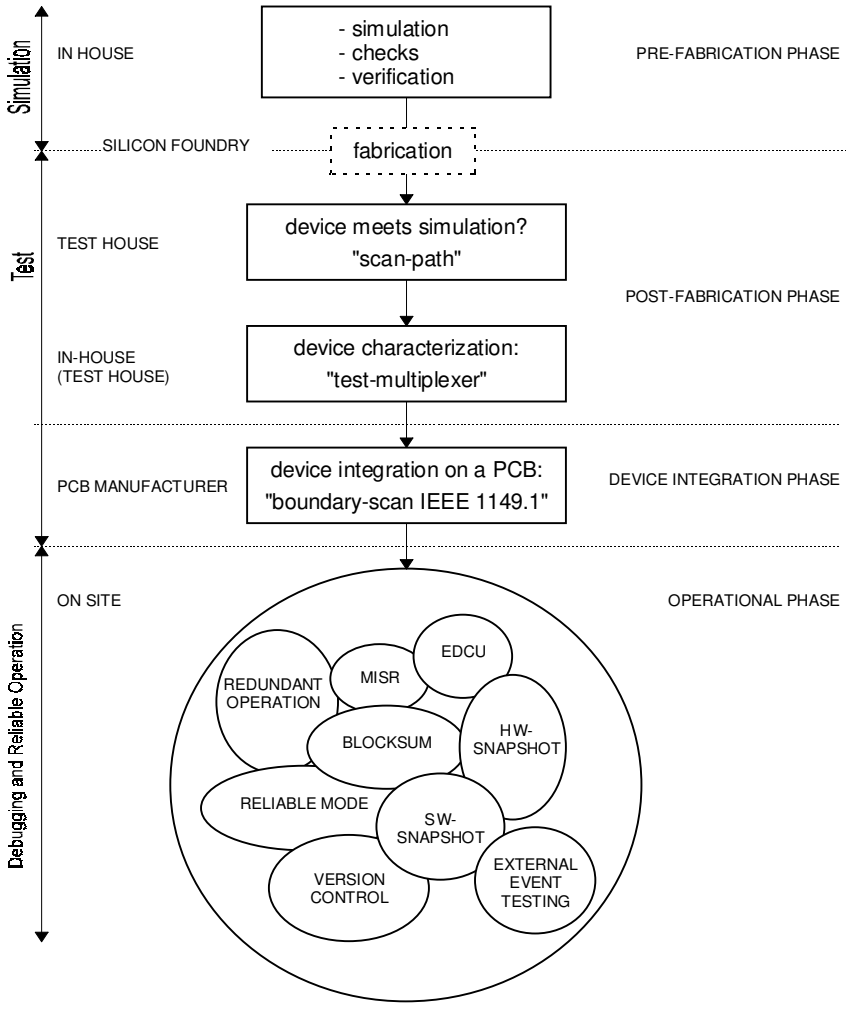


Figure 5-1 Test Concept Overview

5.2 PRE-FABRICATION PHASE

The prefabrication phase is dominated by simulation and verification covered in Chapter 9. Of course, all test logic is implemented, simulated and verified in the prefabrication phase.

5.3 POST-FABRICATION PHASE

When the chip comes back from the silicon foundry, it must be tested for fabrication flaws and characterized to meet all timing and electrical requirements. A test methodology that is supported by most logic synthesis tools is scan-path insertion. Additional test multiplexer ease tracking for design errors.

5.3.1 Scan-Path

Scan design is a comfortable way to achieve high fault-coverage under the burden of a maybe considerable increase in die size. There are many scan styles around like multiplexed flip-flop, clocked scan, Level-Sensitive Scan Design (LSSD), …[ABF90]. One should consider the following questions when choosing the scan style:

- What scan styles (scan cells) are supported by the silicon foundry?
- What is the design style? Is it predominantly edge-triggered or level-sensitive?
- What test methodology will be used? (see next paragraph)
- How robust is the test integration tool?
- What is the area constraint? (scan flip-flops require more area)
- What is the performance constraint? (scan flip-flops have larger delays)

The UTCSU design is flip-flop dominated, hence we should not go with the popular IBM LSSD style. Our two silicon foundries ES2 and AMS only have multiplexed flip-flops in their standard cell synthesis libraries, hence we will be lucky to use them. By the way, Synopsys test compiler we are using for automatic test insertion is most robust for the multiplexed flip-flop scan style. To meet the timing constraints, we give the responsibility to the logic synthesis tool.

Now one should concern about the test methodology. Full scan and partial scan are the two alternatives. Full scan replaces each and every flip-flop in the design with a scannable version of the replaced flip-flop type, connects these scan flip-flops in a chain and feeds the input of the chain to a primary input pin and the output to a primary output pin as shown in Figure 5-2. Test pattern are shifted in at the SCANIN input and shifted out at the SCANOUT output if signal SCANEN (scan-enable) is active, otherwise the scan flip-flops behave as usual. Hence the sequential block is testable and also the combinational block is testable with the test pattern shifted through the scan-chain

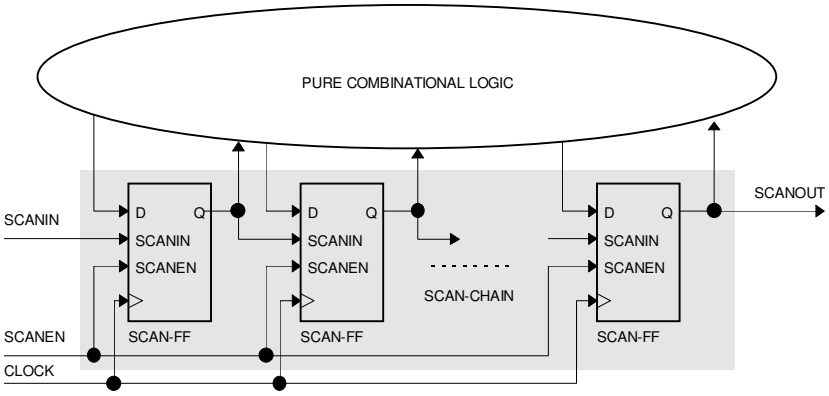


Figure 5-2 Multiplexed Flip-Flop Scan-Path Design

Partial scan only replaces those flip-flops with scan flip-flops, which increase the fault-coverage.

5.3.2 Test Multiplexer

Test multiplexers increase the observability in the UTCSU. Chip internal nodes are mapped to primary output pins when test mode is enabled. Many times test pins and functional pins share the same primary output pins. In the UTCSU, the 48-bit wide NTP-interface volunteers as the test port, cf. Figure 8-151. A list of all observed internal nodes is found in Table 8-52.

5.4 DEVICE INTEGRATION PHASE

Assuming the device passed the production test and was assembled on a printed circuit board (PCB), board level testing is the next step in the production flow. In the past five years, boundary scan testing evolved as the state of the art technology for testing printed circuit boards, cf. Figure 5-3. Pure digital boundary scan is standardized in IEEE 1149.1. Only a subset of the standard is normally implemented, except the mandatory part must be implemented. A performance analysis of the boundary scan test logic in the UTCSU follows in the next subsections.

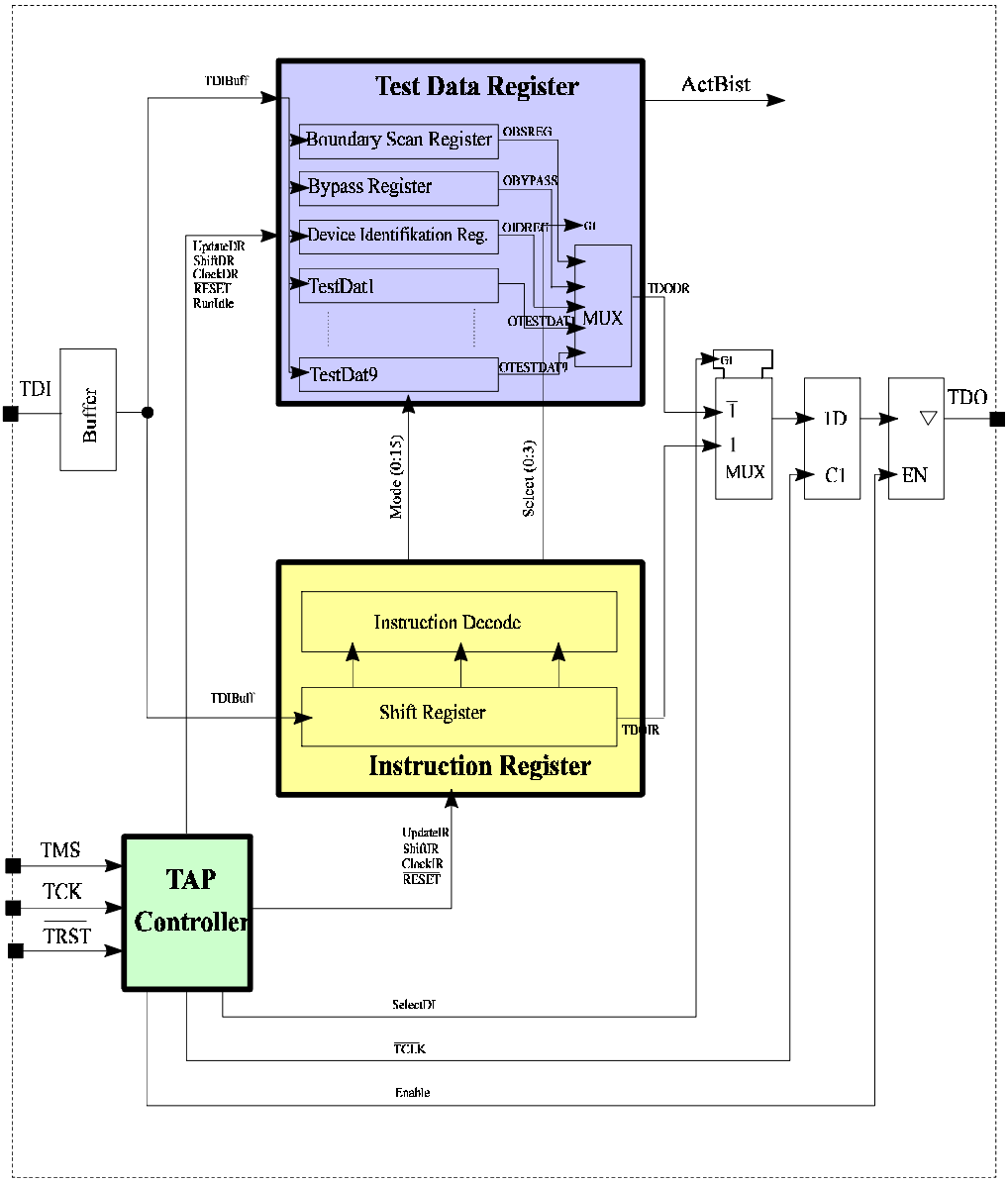


Figure 5-3 Boundary-Scan Test Logic According IEEE 1149.1

The Test Access Port (TAP) Controller is implemented as a state-machine with 16 states. It is clocked by the Test CLock (TCK) input, controlled by the Test Mode Select (TMS) input and controls itself the Test Data Registers and the Instruction Register. Test Data are loaded serially via the Test Data In (TDI) primary input pin into the Test Data Register selected by the Instruction Register. Test instructions as listed below are loaded into the Instruction Register. Test results are shifted out from the Test Data Registers via the Test Data Out (TDO) primary output pin.

The Boundary Scan Test Data Register is made up of a chain of scan flip-flops connected between the primary input/output pins and the chip core logic, hence test stimuli can be applied to the primary outputs and test results can be captured at primary inputs. This configuration allows to test the interconnections between several ASICs on a Printed Circuit Board (PCB) whose TDI and TDO pins are connected in a test chain.

5.4.1 Five Pin Test Access Port

Mandatory is a four pin test access port with pins TMS, TCK, TDI and TDO. The optional TRST (test reset pin) exists in the UTCSU implementation.

5.4.2 Mandatory Instructions

The standard lists 3 mandatory instructions.

- *BYPASS* selects the one bit bypass register and disables the boundary scan test logic for this test cycle.
- *EXTEST* selects the boundary scan register and is used for external tests, such as interconnect testing.
- *SAMPLE/PRELOAD* selects the boundary scan register. In SAMPLE mode input and output signals on primary pins are sampled into the boundary scan register. In PRELOAD mode, test data are loaded into the boundary scan register before EXTEST is selected.

5.4.3 Optional Instructions

The UTCSU makes use of three optional instructions. *INTEST* selects the boundary scan register for internal (core) testing. *IDCODE* selects the device identification register which holds a 32-bit number composed of the 4-bit version number, the 16-bit part number, and the 11-bit manufacturer ID as shown in Figure 5-4. Bit zero (LSB) is always 1.

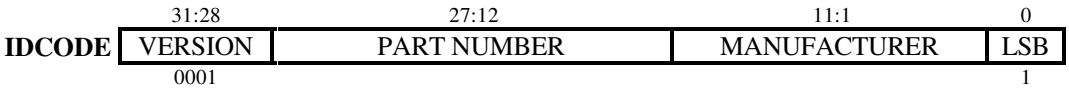


Figure 5-4 IDCODE Register

Because of the JTAG/IEEE 1149.1 requirements, the 11-bit manufacturer code and the 16-bit part number have to be chosen in consultation with ES2, cf. [E02A94]. *RUNBIST* selects a user defined BIST register and invokes device self-test functions.

5.4.4 Instruction Codes

In total the UTCSU boundary scan test logic knows 6 different instructions. Their instruction codes as proposed in [E02A94] are given in Table 5-1. The instruction register length is 4 bits.

Instruction	Instruction Codes
EXTEST	0000
IDCODE	0001
SAMPLE/PRELOAD	0010
INTEST	0011
RUNBIST	0100
reserved	0101 - 1110
BYPASS	1111

Table 5-1 Boundary Scan Instruction Codes

5.5 OPERATIONAL PHASE

When the UTCSU is exposed to the real-world, means must be provided to ensure and monitor proper system operation. The following subsections promote these means.

5.5.1 Timestamp Validation

Timestamps, generated in the adjustable local clock, are the most valuable information kept in the UTCSU, hence they claim extra protection. For an arbitrary circuit with q outputs where all 2^q output combinations can occur, it is impossible to determine whether a fault is present by just observing the outputs of the circuit assuming no knowledge of the inputs. However, if only $k < 2^q$ output combinations can occur during normal operation, the appearance of 2^q - k other output combinations indicates a fault. Hence, faults that manifest in such an illegal output can be detected by a checker. A checker is a circuit, designed to indicate the presence of a fault. What are the requirements for protecting timestamps?

- New timestamps are scheduled every system clock tick. Hence processing time is limited to a single clock tick.
- Protection logic should consume only minimum chip area.
- Protection algorithm must be implementable in hardware and software (for verification).
- 56 data bits have to be protected with a minimal number of check bits. An upper limit is 8 (to fit in 64 bits).
- The number of check bits should be scaleable with the number of protected information bits.
- The all zero and all one condition (open or shorted wires) must be detected.
- Single bit errors should be corrected.

Parity codes, residual codes, Hamming codes, etc. can be used to extend the legal codewords with illegal ones, cf. [ABF90]. They all add redundancy in form of check bits c to the information bits q. The c check bits and q information bits form a codeword with $(c+q)$ bits, $b_{c+q} \cdots b_2 b_1$.

The check bits in the conventional *Hamming code* reside in bits b_{2^i}, $0 \le i \le c\text{-}1$. Let p_j be the set of binary integers B with a 1-value in bit b_j. (ie $p_j = \{B | b_j(B) = 1\}$, where $b_j(B)$ denotes the value of bit j in the binary representation of B. Then the values for the c checkbits are derived from the c parity-check equations 6-1 where the sum is modulo 2.

$$\sum_{k \in p_i} b_k = 0 \qquad i = 1, \ldots c \qquad \text{Equ. 5-1}$$

An error in bit b_j will result in incorrect parity bits for equations where j is in p_i. Conventional Hamming codes correct single-bit errors. In order to protect 56 data bits, six checkbits are required.

Another class of codes are the *residual codes*. They have the property that for the arithmetic operations of addition, subtraction and multiplication, the check bits of the result can be determined from the check bits of the operands. This characteristic is called *independent checking*. In one class of

residual codes the rightmost *q* bits are the check bits. They define a binary number C and the data bits define a binary number N. Then the checkbits are defined so that

$$C = N \text{ modulo } m \qquad m \ldots \text{residue} \qquad \text{Equ. 5-2}$$

where *m* is the residue of the code and $q = \lceil \log_2 m \rceil$.

Berger codes are separable codes. That means *n* data bits can be classified into C check bits and I information bits. The C check bits define a binary number corresponding to the Boolean complement of the number of information bits valued 1. The number of check bits is

$$C = \lceil \log_2 (I + 1) \rceil. \qquad \text{Equ. 5-3}$$

Residual codes require a modulo *m* division, which is not easy to implement in a single clock tick. Berger codes are widely accepted but need a chip area consuming adder. In imitation of the commercial available error detection and correction part Am2960 EDC, the error detection and correction unit (EDCU) in the UTCSU uses a modified Hamming code with Hamming Distance 4 to protect timestamps. Implementation details can be found in Section 8.1.7.

5.5.2 Timestamp Compression

Timestamps resp. accuracystamps fall out of the local time unit resp. the accuracy unit at every single system clock tick. Each timestamp is protected by an eight bit checksum (CS). The checksum does not provide information on the correctness of the timestamp, it only helps to check the validity along a transfer or when the timestamp is stored in memory. Some means are desirable to verify the correctness of a sequence of time- and accuracystamps.

No CPU can deal with the data rate of 110 Mbytes/s required to check the 56-bit timestamp and the two times 16-bit accuracystamp at a clock speed of 10 MHz. Hence compression techniques need to be installed to reduce the very high data rate. One technique is known as multiple-input signature register (MISR).

MISRs are based on the theory of linear feedback shift registers, cf. [PW72, GOL82, ABF90]. Normally a single-input signature analyzer is not attached to every output of a multiple-output circuit because of the high overhead. Of course, a single-input signature analyzer could be time-multiplexed, but this increases test time tremendously and decreases system speed. When applied in an on-line test environment speed problems will occur. The most common technique is to use a multiple-input signature analyzer as the one shown in Figure 5-5. Instead of having a single input, every flip-flop is tapped and serves as a data input.

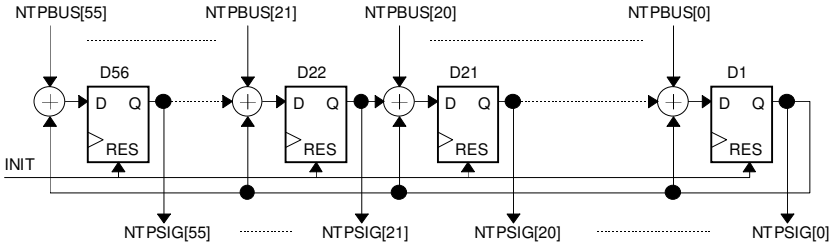


Figure 5-5 MISR, $P^{56}(x) = x^{56} + x^{22} + x^{21} + x^1 + 1$

Note that a generator polynom of degree 56 is used to compress the 56-bit timestamps on the NTP-Bus. Primitive polynoms up to degree 300 are listed in the annex of [YAR93]. A polynom of degree 32 compresses the 2 two times 16-bit accuracystamps, cf. Section 8.7.3.

A MISR having *n* stages has a masking probability approximately equal to 2^{-n} for equally likely error patterns and long data streams. For the polynom of degree 56 one obtains $\approx 13.8 \cdot 10^{-18}$ as the masking probability.

Masking occurs because once an error exists in the shift register, it can be canceled by new errors appearing at the inputs. Periodically inspecting the contents of the MISR further decreases the probability of masking. Frequent comparisons of the contents of the MISR with a reference value informs about the correctness of the produced data. The reference value could result from a redundant UTCSU, see Section 5.5.4, or a CPU. Because of the high data rate and the high computing power

needed to carry out the signature analysis, regrettably no CPU is capable of computing the reference value on-line. Still one can think about a scenario where a physical process will enter a critical phase at time t_{enter} some time in future and will leave this critical phase at time t_{leave}. A CPU can model this situation and pre-compute reference values which are compared with the signatures from the UTCSU during the critical phase. A more elegant way would be to furnish the CPU with hardware to carry out the MISR operation.

5.5.3 Blocksum Signature

One could experience from the previous section, that general purpose CPUs do not have the computing power to cope with the immense data rate and the signature computation algorithm. A more CPU-friendly way is to compute a blocksum rather than a signature. Figure 5-6 illustrates the basic function. Timestamps on the NTP-Bus resp. accuracystamps on the A-Bus are summed up in an 56-bit resp. 32-bit wide adder. The computed blocksums are periodically compared against a reference value from a redundant UTCSU or computed in a general purpose CPU.

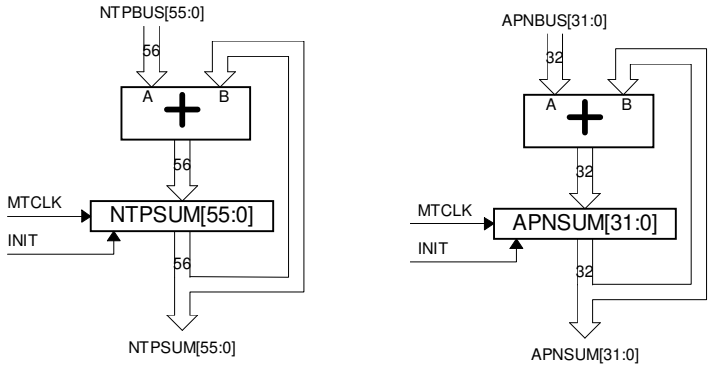


Figure 5-6 Blocksum Signatures for Timestamps and Accuracystamps

Its easier for a CPU to determine the reference value. If, for example, test results are compared at resynchronization interval boundaries, the reference is computed as follows. Let's say during resynchronization interval i, the amortization phase is $n_{A,i}$ clock ticks long, the pure phase $n_{P,i}$ ticks. The clock step during amortization phase is $s_{A,i}$ and during pure phase $s_{P,i}$. At the beginning of the test period, the clock register contents is $c_{0,i}$ and the initial blocksum value is $b_{0,i}$. The width of the adder in bits is given with parameter q. Hence the blocksum contents b_i at the end of the observation period i is composed of the sum of all timestamps during amortization phase c_A+ pure phase c_P + $b_{0,i}$.

$$b_i = \left(b_{0,i} + \sum_{\text{AMORTPHASE}} c_{A,i} + \sum_{\text{PUREPHASE}} c_{P,i} \right) \bmod 2^q \qquad \text{Equ. 5-4}$$

The two sums above are given by

$$\sum_{\text{AMORTPHASE}} c_{A,i} = \left(\sum_{k=1}^{n_{A,i}} \left(c_{A,i} + k \cdot s_{A,i} \right) \right) \bmod 2^q = \left(n_{A,i} \cdot c_{A,i} + \frac{\left(1 + n_{A,i}\right) \cdot n_{A,i} \cdot s_{A,i}}{2} \right) \bmod 2^q \qquad \text{Equ. 5-5}$$

with

$$c_{A,i} = c_{0,i} = c_{P,i-1} + n_{P,i-1} \cdot s_{P,i-1} \qquad \text{Equ. 5-6}$$

and

$$\sum_{\text{PUREPHASE}} c_{P,i} = \left(\sum_{k=1}^{n_{P,i}} \left(c_{P,i} + k \cdot s_{P,i} \right) \right) \bmod 2^q$$
$$= \left(n_{P,i} \cdot c_{P,i} + \frac{\left(1 + n_{P,i}\right) \cdot n_{P,i} \cdot s_{P,i}}{2} \right) \bmod 2^q \quad \text{Equ. 5-7}$$

with

$$c_{P,i} = c_{0,i} + n_{A,i} \cdot s_{A,i}. \quad \text{Equ. 5-8}$$

A total of six multiplication and 5 additions is needed to compute the blocksum of one total resynchronization interval. This shouldn't be a big burden for any CPU running the clock synchronization algorithm. Observe that in order to solve Equation 6-7 the number of clock ticks during pure phase ($n_{P,i}$) must be known. Section 8.7.4 reveals details.

5.5.4 Redundant Operation

Some fault-tolerant applications of the UTCSU are served better with a redundant configuration of multiple chips. Precautions have to be taken during the design cycle to provide means in order to synchronize the local clocks and to periodically observe their results namely their timestamps and accuracystamps. The first issue is addressed in Section 8.1.6.1 and Section 8.2.5.1 and the observation of results was completely talked about in Section 5.5.2.

An active edge on the SYNCRUN primary input pin starts the identically programmed UTCSUs with the same clock edge from the same clock oscillator. Hence all chips produce the same timestamps and accuracystamps, cf. Figure 5-7. One or several CPUs periodically compare the signatures or blocksums and decide about the condition of the clocks.

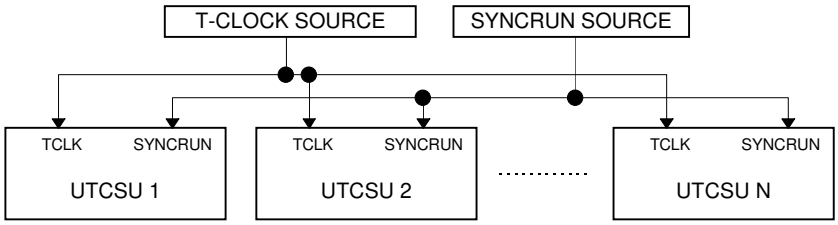


Figure 5-7 Redundant Operation of Multiple UTCSUs

5.5.5 On-Line Event Testing

External events such as Application events, GPS events or network events are observed on primary input pins. Simulating an external event internally would make sense for two reasons.

- The complete data path and control logic between the output of the primary input pin and the CPU interface can be tested, cf. Figure 5-8.
- In the debugging and system integration phase, external events, which are currently not available, can be simulated and reactions on eg raised or pending interrupts verified.

During normal operation events on the primary input pins (EVENT INPUT) are synchronized and fed into the corresponding unit (viz GPU, SSU, APU, SNU). Timestamps are captured and pulled out of the UTCSU through the CPU-interface. In test mode when signal TESTSEL is active, the primary input pin EVENT INPUT is disconnected from the synchronizer input and the internally generated event signal TESTGATE is connected to the synchronizer input. Timestamps captured with the simulated event are pulled out of the UTCSU the same way as during normal operation. The signal path during test is marked with a gray arrow in Figure 5-8.

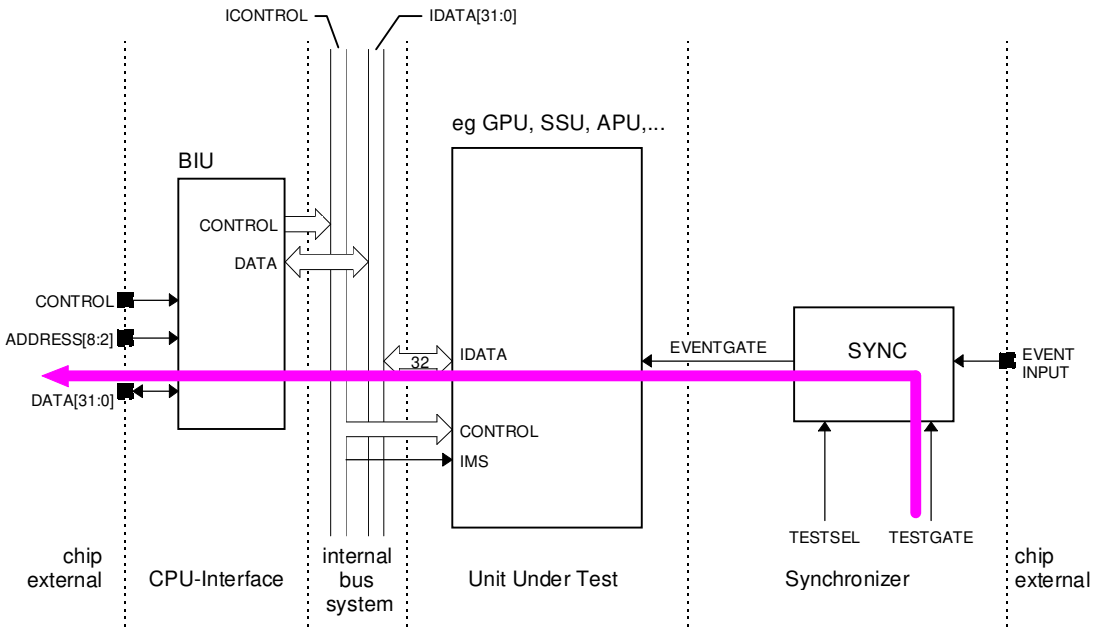


Figure 5-8 Signal Path for On-Line Event Testing

5.5.6 Reliable Mode

The pity with external asynchronous signals is, they must be synchronized to the system clock. This could cause metastability problems in the input-stage flip-flop. Subsequently two strategies for resolving metastability at the flip-flop output are compared.

Figure 5-9 shows a dual stage input architecture with the flip-flops clocked at twice the system clock frequency $2*f_{CLOCK}$.

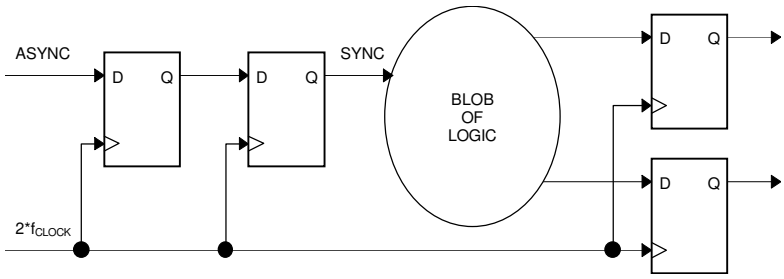


Figure 5-9 Dual Stage Metastability Design

In contrast, Figure 5-10 illustrates a single stage design with the flip-flop clocked at f_{CLOCK}. The latency on the ASYNC input line is the same in both cases. The BLOB of logic represents combinational logic and the two flip-flops at the output the sequential logic of the design.

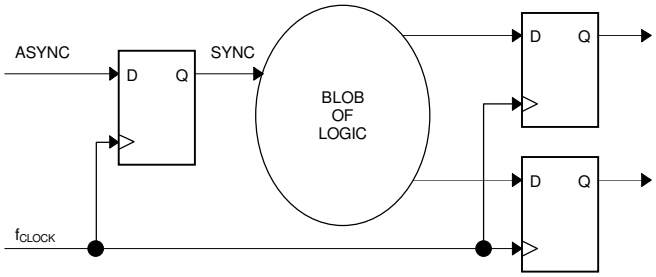


Figure 5-10 Single Stage Metastability Design

Which one of the two concepts will behave better in resolving metastability? I found Equation 6-9 on the Internet in an email from John Cooley, more specifically in ESNUG Post 225 on Friday, 25

August 1995. The article in the email references [KC87] as the source of Equation 6-9. Looking up [KC87, p.8] states the following message 'The fundamental result is that continuous systems cannot consistently make discrete decisions within a finite time based on non discrete or continuously variable inputs'. It further says 'It is generally accepted that a perfect synchronizer cannot be physically realized', and gives a comparable equation on page 12 as the one in 6-9 based on the first-order flip-flop model proposed by [VEEN80]. Equation 6-9 calculates the mean time between failures in terms of system clock, edge rate at the ASYNC input, and flip-flop parameters.

$$MTBF = \frac{e^{\frac{T_r}{\tau}}}{2 * f_{data} * f_{clock} * T_o} \qquad \text{Equ. 5-9}$$

MTBF Mean Time Between Failures
T_r is the time available for the metastability to resolve itself
τ is the resolution rate (experimentally measured). It is a device time constant for the exponential escape from a metastable region of the flip-flop.
$2*f_{data}$ is the input event (edge) rate
f_{clock} is the system clock frequency
T_0 is the metastability aperture window width (experimentally measured). If a data edge occurs within the aperture, the output of the flip-flop remains unresolved longer than a time T [KC87, p.11]. T_0 is a function of T.

For a multi stage design, the equation for MTBF becomes

$$\text{MTBF} = \frac{e^{\frac{T_1}{\tau}}}{2 * f_{data} * f_{clock} * T_o} \qquad \text{Equ. 5-10}$$

where T_1 is the total resolution time available from N stages.

$$T_1 = N * T_{clock} - (N-1) * (T_{pd} + T_{su}) \qquad \text{Equ. 5-11}$$

T_{clock} is the system clock period = $1/f_{clock}$
T_{pd} is the clock to Q propagation delay
T_{su} is the data to clock setup time

Equation 6-11 shows a shorter metastability resolution time T_1 for the multi stage design than for the single stage design with half the system clock frequency, hence the single stage design behaves better in resolving metastability. T_1 is shortened by the second term in 6-11, the setup times plus propagation delays of the multi stage flip-flops. To figure out what we gain with the single stage design, Equation 6-12 compares Equation 6-9 and 6-10.

$$\Delta MTBF = \frac{Equ.5-9}{Equ.5-10} = e^{\frac{T_{su}+T_{pd}}{\tau}} \qquad \text{Equ. 5-12}$$

For the 0.7 µm CMOS process from ES2, one gets the following numbers for a scannable flip-flop: T_{su} = 2.25 ns, T_{pd} = 1.58 ns, and for an Inverter gate: 1 gate delay = 0.24 ns + 2.2 ns/pF. The resolution rate τ is typically an order of magnitude smaller than a gate delay [KC87]. Observe that Equation 6-12 returns a

$$\Delta MTBF = e^{\frac{2.25ns+1.58ns}{0.2ns}} = 207 * 10^6 \qquad \text{Equ. 5-13}$$

times better MTBF. Note that delays in the "BLOB of logic" should be kept as small as possible to obtain maximum resolution time and to achieve best MTBF. Long BLOB delays increase the propagation delay between the flip-flop output and the next flip-flop input, hence shorten the time available to propagate the stable flip-flop output signal. The equations given are based on majority logic.

5.5.7 Version Control

Version control uses a 32-bit identification register to properly match different software versions with different hardware versions of the UTCSU. This 32-bit identification register returns a chip ID and a version number, cf. Figure 5-11. The upper 24 bits hold the acronym "ICT" for "Institute of Computer Technology" and the lower 8 bits hold a major version number, currently 1, in the upper 4-bit block and minor version number, currently 0, in the lower 4-bit block.

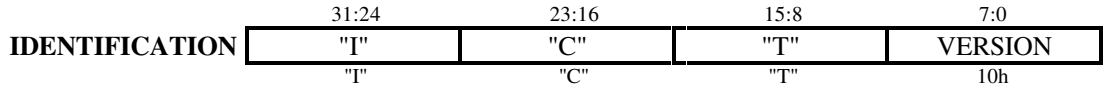


Figure 5-11 IDENTIFICATION - UTCSU Identification Register

5.5.8 Hardware-Snapshot

Snapshots in general are a powerful instrument to examine the current state of the UTCSU. Firing a snapshot captures dynamically changing registers (eg clock, accuracy) into holding registers that can be consulted for system evaluation.

Hardware-snapshots are triggered by either a rising or falling edge (the edge is programmable) at the primary input pin HWSNAP. As a reaction, the current state of the NTP-Bus and the Accuracy-Bus are latched into dedicated hardware-snapshot holding registers. This can happen totally asynchronous to the normal operation of the UTCSU.

5.5.9 Software-Snapshot

Software-snapshots latch the current state of all dynamically changing UTCSU registers into holding registers which are also accessed during normal operation. This snapshot returns a comprehensive view of the complete UTCSU under the expense of a possible interrogation with the normal clock synchronization operation. Hence software-snapshots may only be carried out by software routines familiar with the clock synchronization algorithm. Software-snapshots are executed by a write operation on a dedicated UTCSU register address.

CHAPTER 6

6 UTCSU BLOCK DIAGRAM

Man verliert die meiste Zeit darin, daß man Zeit gewinnen will.
John Steinbeck

6.1 BLOCK DIAGRAM

Please see the enclosed color print on the next page for a block diagram showing the nuts and bolts of the UTCSU. Each colored block represents a unit where the green coloring stands for test logic, the blue coloring for maintaining local time, the red coloring for maintaining accuracies and the yellow blocks finally support the application processes. All units are linked to an internal bus system (left hand) which in turn is connected to a CPU via the bus interface unit (BIU, brown). A synchronization unit (SYU, gray) ties the application units to the external world. Of course, we cannot show the innards here. This is left open to Chapter 8.

6.2 REGISTER MAP

A quick summary of all UTCSU-registers is given in the tables below. Detailed register descriptions are given in Chapter 8.

Start Address	End Address	Unit	Long Name	Description of Function
000h	01Fh	LTU	Local Time Unit	maintains local time, rate + state correction
020h	03Fh	ACU	Accuracy Unit	maintains accuracy (envelopes), bounds
040h	05Fh	CLU	Clock Unit	clock oscillator, reset logic
060h	07Fh	ITU	Interrupt Unit	interrupt processing, configuration & status
080h	09Fh	BTU	Built-In Test Unit	MISR signature, blocksum, test logic
0A0h	0BFh	SNU	Snapshot Unit	hardware snapshot, software snapshot
0C0h	0DFh	APU	Application Unit	external event recording, event generation
0E0h	0FFh	GPU 1	GPS Unit 1	generic interface to GPS receiver #1
100h	11Fh	GPU 2	GPS Unit 2	interface to redundant GPS receiver #2
120h	13Fh	GPU 3	GPS Unit 3	interface to redundant GPS receiver #3
140h	15Fh	SSU 1	Synchronization Subnet Unit 1	XMT & RCV packet-timestamping in SSN1
160h	17Fh	SSU 2	Synchronization Subnet Unit 2	XMT & RCV packet-timestamping in SSN2
180h	19Fh	SSU 3	Synchronization Subnet Unit 3	XMT & RCV packet-timestamping in SSN3
1A0h	1BFh	SSU 4	Synchronization Subnet Unit 4	XMT & RCV packet-timestamping in SSN4
1C0h	1DFh	SSU 5	Synchronization Subnet Unit 5	XMT & RCV packet-timestamping in SSN5
1E0h	1FFh	SSU 6	Synchronization Subnet Unit 6	XMT & RCV packet-timestamping in SSN6

Table 6-1 UTCSU Memory Map and Unit Summary

Table 6-2 summarizes all UTCSU registers, organized in groups representing their unit affiliation. Address locations marked with a "-" don't reference a valid UTCSU register and are reserved for future use. All registers are designed for a 32-bit wide access method. CPUs with data bus width of 16 or 8 bits must care about the big-endian, little-endian format, cf. Section 7.1.1.

Address	Bits	READ-Register Name	Bits	WRITE-Register Name	Unit
000h	[23:0]	*MSGET* - Macro Stamp Get	[23:0]	*MSSET* - Macro Stamp Set	LTU
000h	[31:24]	*CS* - Checksum Get	[31:24]	*STEPLOW* - Step Low	LTU
004h	[31:0]	*TSGET* - Time Stamp Get	[31:0]	*TSSET* - Time Stamp Set	LTU
008h	[31:0]	*USGET* - Micro Stamp Get	[31:0]	*USSET* - Micro Stamp Set	LTU
00Ch	[31:0]	*TSGETL* - Time Stamp Get Latch	[7:0]	*NTPSET* - NTP-Time Set	LTU
010h	[2:0]	*NSGET* - Nano Stamp Get	[31:0]	*STEPAMORT* - Step Amort. Ph.	LTU
014h	[31:0]	-	[31:0]	*STEPPURE* - Step Pure Phase	LTU
018h	[31:0]	*AMORTTIMEGETL*	[31:0]	*AMORTTIMESET*	LTU
01Ch	[31:0]	*AMORTTIMEGET*	[31:0]	*STARTAMORT*	LTU
020h	[15:0]	*STATEPGETL*	[15:0]	*LAMBDAPPURE*	ACU
020h	[31:16]	*STATENGETL*	[31:16]	*LAMBDAPAMORT*	ACU
024h	[15:0]	*STATEPGET*	[15:0]	*LAMBDANPURE*	ACU
024h	[31:16]	*STATENGET*	[31:16]	*LAMBDANAMORT*	ACU
028h	[31:0]	*ALPHAPGETL*	[31:0]	*STATEPSET*	ACU
02Ch	[31:0]	*ALPHANGETL*	[31:0]	*STATENSET*	ACU
030h	[31:0]	*ALPHAPGET*	[15:0]	*BOUNDP*	ACU
030h	-	-	[31:16]	*BOUNDN*	ACU
034h	[31:0]	*ALPHANGET*	[31:0]	*ALPHASET*	ACU
038h	[15:0]	*NALPHAPGET*	[7:0]	*ALPHAPNSET*	ACU
038h	[31:16]	*NALPHANGET*	[31:8]	-	ACU
040h	-	-	[7:0]	*SWRESET*	CLU
060h	[31:0]	*IDENTIFICATION*	[31:0]	*UTCCONF1*	ITU
064h	[31:0]	*UTCSTAT1*	[31:0]	*UTCCONF2*	ITU
068h	[31:0]	*UTCSTAT2*	[31:0]	*UTCINTEN1* - Interrupt Enable 1	ITU
06Ch	[31:0]	*UTCCONF1*	[31:0]	*UTCINTEN2* - Interrupt Enable 2	ITU
070h	[31:0]	*UTCCONF2*	[31:0]	*UTCINTCLEAR1* - Int. Clear 1	ITU
074h	[31:0]	*UTCINTSTAT1*	[31:0]	*UTCINTCLEAR2* - Int. Clear 2	ITU
078h	[31:0]	*UTCINTSTAT2*	-	-	ITU
080h	[23:0]	*MSSIG* - Macro Stamp MISR sign.	[15:0]	*TESTSEL* - Test Select	BTU
084h	[31:0]	*TSSIG* - Time Stamp MISR sign.	[7:0]	*TESTGATE* - Test Gate Pulse	BTU
088h	[31:0]	*ACCSIG* - Accuracy MISR sign.	-	-	BTU
08Ch	[23:0]	*MSSUM* - Macro Stamp Blocksum	-	-	BTU
090h	[31:0]	*TSSUM* - Time Stamp Blocksum	-	-	BTU
094h	[31:0]	*ACCSUM* - Accuracy Blocksum	-	-	BTU
098h	[31:0]	*PUREPHASE* - Pure Phase Counter	-	-	BTU
0A0h	[31:0]	*MSSNU* - Macro Stamp Snapshot	[31:0]	*SWSNAP* - Software Snapshot	SNU
0A4h	[31:0]	*TSSNU* - Time Stamp Snapshot	[31:0]	*DUTYH* - Duty-Timer High Prel	SNU
0A8h	[15:0]	*ACCPSNU* - Accuracy P Snapshot	[15:0]	*DUTYL* - Duty-Timer Low Prel.	SNU
0A8h	[31:16]	*ACCNSNU* - Accuracy N Snapshot	[16]	*E* - Enable Bit Duty-Timer	SNU
0C0h	[31:0]	*MSAPP* - Macro Stamp Appl.	[31:0]	*DUTYH* - Duty-Timer High Prel	APU
0C4h	[31:0]	*TSAPP* - Time Stamp Application	[16:0]	*DUTYL* - Duty-Timer Low Prel.	APU
0C8h	[15:0]	*ACCPAPP* - Accuracy P Appl.	[2:0]	*APPSEL* - Application Select	APU
0C8h	[31:16]	*ACCNAPP* - Accuracy N Appl.	-	-	APU
0CCH	[31:0]	*MSAPPX* - Macro Stamp Appl. X	-	-	APU
0D0h	[31:0]	*TSAPPX* - Time Stamp Appl. X	-	-	APU
0D4h	[15:0]	*ACCPAPPX* - Accuracy P Appl. X	-	-	APU
0D4h	[31:16]	*ACCNAPPX* - Accuracy N Appl. X	-	-	APU
0E0h	[31:0]	*MSGPS* - Macro Stamp GPS	-	-	GPU1
0E4h	[31:0]	*TSGPS* - Time Stamp GPS	-	-	GPU1

Address	Bits	READ-Register Name	Bits	WRITE-Register Name	Unit
100h	[31:0]	*MSGPS* - Macro Stamp GPS	-	-	GPU2
104h	[31:0]	*TSGPS* - Time Stamp GPS	-	-	GPU2
120h	[31:0]	*MSGPS* - Macro Stamp GPS	-	-	GPU3
124h	[31:0]	*TSGPS* - Time Stamp GPS	-	-	GPU3
140h	[31:0]	*MSRCV* - Macro Stamp Receive	[31:0]	*DUTYAH* - Duty-Timer A High	SSU1
144h	[31:0]	*TSRCV* - Time Stamp Receive	[16:0]	*DUTYAL* - Duty-Timer A Low	SSU1
148h	[31:0]	*MSXMT* - Macro Stamp Transmit	[31:0]	*DUTYBH* - Duty-Timer B High	SSU1
14Ch	[31:0]	*TSXMT* - Timestamp Transmit	[16:0]	*DUTYBL* - Duty-Timer B Low	SSU1
150h	[15:0]	*ACCPXMT* - Accuracy P Transmit	-	-	SSU1
150h	[31:16]	*ACCNXMT* - Accuracy N Transmit	-	-	SSU1
154h	[31:0]	*ACCNPNPLXMT*	-	-	SSU1
158h	[31:0]	*ACCNPNPHXMT*	-	-	SSU1
160h	[31:0]	*MSRCV* - Macro Stamp Receive	[31:0]	*DUTYAH* - Duty-Timer A High	SSU2
164h	[31:0]	*TSRCV* - Time Stamp Receive	[16:0]	*DUTYAL* - Duty-Timer A Low	SSU2
168h	[31:0]	*MSXMT* - Macro Stamp Transmit	[31:0]	*DUTYBH* - Duty-Timer B High	SSU2
16Ch	[31:0]	*TSXMT* - Timestamp Transmit	[16:0]	*DUTYBL* - Duty-Timer B Low	SSU2
170h	[15:0]	*ACCPXMT* - Accuracy P Transmit	-	-	SSU2
170h	[31:16]	*ACCNXMT* - Accuracy N Transmit	-	-	SSU2
174h	[31:0]	*ACCNPNPLXMT*	-	-	SSU2
178h	[31:0]	*ACCNPNPHXMT*	-	-	SSU2
180h	[31:0]	*MSRCV* - Macro Stamp Receive	[31:0]	*DUTYAH* - Duty-Timer A High	SSU3
184h	[31:0]	*TSRCV* - Time Stamp Receive	[16:0]	*DUTYAL* - Duty-Timer A Low	SSU3
188h	[31:0]	*MSXMT* - Macro Stamp Transmit	[31:0]	*DUTYBH* - Duty-Timer B High	SSU3
18Ch	[31:0]	*TSXMT* - Timestamp Transmit	[16:0]	*DUTYBL* - Duty-Timer B Low	SSU3
190h	[15:0]	*ACCPXMT* - Accuracy P Transmit	-	-	SSU3
190h	[31:16]	*ACCNXMT* - Accuracy N Transmit	-	-	SSU3
194h	[31:0]	*ACCNPNPLXMT*	-	-	SSU3
198h	[31:0]	*ACCNPNPHXMT*	-	-	SSU3
1A0h	[31:0]	*MSRCV* - Macro Stamp Receive	[31:0]	*DUTYAH* - Duty-Timer A High	SSU4
1A4h	[31:0]	*TSRCV* - Time Stamp Receive	[16:0]	*DUTYAL* - Duty-Timer A Low	SSU4
1A8h	[31:0]	*MSXMT* - Macro Stamp Transmit	[31:0]	*DUTYBH* - Duty-Timer B High	SSU4
1ACh	[31:0]	*TSXMT* - Timestamp Transmit	[16:0]	*DUTYBL* - Duty-Timer B Low	SSU4
1B0h	[15:0]	*ACCPXMT* - Accuracy P Transmit	-	-	SSU4
1B0h	[31:16]	*ACCNXMT* - Accuracy N Transmit	-	-	SSU4
1B4h	[31:0]	*ACCNPNPLXMT*	-	-	SSU4
1B8h	[31:0]	*ACCNPNPHXMT*	-	-	SSU4
1C0h	[31:0]	*MSRCV* - Macro Stamp Receive	[31:0]	*DUTYAH* - Duty-Timer A High	SSU5
1C4h	[31:0]	*TSRCV* - Time Stamp Receive	[16:0]	*DUTYAL* - Duty-Timer A Low	SSU5
1C8h	[31:0]	*MSXMT* - Macro Stamp Transmit	[31:0]	*DUTYBH* - Duty-Timer B High	SSU5
1CCh	[31:0]	*TSXMT* - Timestamp Transmit	[16:0]	*DUTYBL* - Duty-Timer B Low	SSU5
1D0h	[15:0]	*ACCPXMT* - Accuracy P Transmit	-	-	SSU5
1D0h	[31:16]	*ACCNXMT* - Accuracy N Transmit	-	-	SSU5
1D4h	[31:0]	*ACCNPNPLXMT*	-	-	SSU5
1D8h	[31:0]	*ACCNPNPHXMT*	-	-	SSU5
1E0h	[31:0]	*MSRCV* - Macro Stamp Receive	[31:0]	*DUTYAH* - Duty-Timer A High	SSU6
1E4h	[31:0]	*TSRCV* - Time Stamp Receive	[16:0]	*DUTYAL* - Duty-Timer A Low	SSU6
1E8h	[31:0]	*MSXMT* - Macro Stamp Transmit	[31:0]	*DUTYBH* - Duty-Timer B High	SSU6
1ECh	[31:0]	*TSXMT* - Timestamp Transmit	[16:0]	*DUTYBL* - Duty-Timer B Low	SSU6
1F0h	[15:0]	*ACCPXMT* - Accuracy P Transmit	-	-	SSU6
1F0h	[31:16]	*ACCNXMT* - Accuracy N Transmit	-	-	SSU6
1F4h	[31:0]	*ACCNPNPLXMT*	-	-	SSU6
1F8h	[31:0]	*ACCNPNPHXMT*	-	-	SSU6

Table 6-2 UTCSU Register Map

CHAPTER 7

7 EXTERNAL INTERFACES

Benutze redlich deine Zeit!
Goethe

More than 160 pins of the UTCSU interface various system components. A CPU-Port connects the UTCSU with one or several CPUs. The CSUTIME-Port supplies local clock information and test signals to the outside world. Interrupts are signaled via a SYSTEM-Port and external events are processed on an EVENT-Port. Internal test logic is accessed through the TEST-Port.

7.1 CPU-PORT

The CPU-Port establishes a communication path between the hardware clock layer implemented in the UTCSU (eg local time unit (LTU), accuracy unit (ACU), …) and the upper layers consisting of the clock synchronization layer and application layer hosted in the communication CPU. UTCSU-registers make up well defined service access points (SAP) to access the lower layers. SAPs for read and write are supported as well as a prioritized interrupt scheme. Additionally to the core logic, the integrated built-in test-logic is accessed through this interface. Furthermore a communication coprocessor may also request time information via the CPU interface.

Each UTCSU chip is connected to one or more processors ranging from simple 8-bit microcontrollers up to 32-bit high performance CPUs. The CPU interface offers a wide variety of different electrical and functional interface options according to the following criteria.

- Data bus width 8-bit, 16-bit, 32-bit
- Big endian - little endian supported
- Different bus access times supported
- Synchronous - asynchronous bus access
- Different interrupt schemes and interrupt polarities

A two level interface model is implemented to access the UTCSU internal system logic. The bus interface unit (BIU) synchronizes external system bus events with the on-chip system clock, generates the appropriate handshake signals and adapts the different CPU flavors listed above to the internal bus standard. Units are accessed through the module bus interface (MBI) which is part of every design unit. Figure 7-1 shows the data and control path for the described scenario.

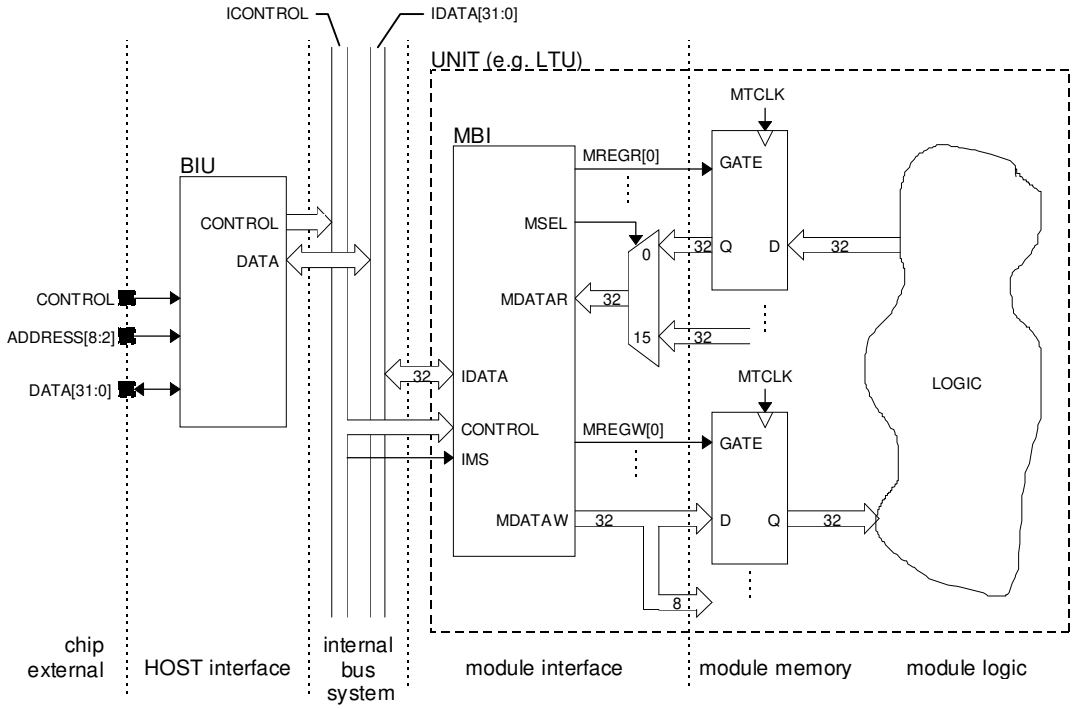


Figure 7-1 A CPU Interfaces with UTCSU-Units through the BIU and MBI

Data enter and leave the chip through the 32-bit wide data bus pins DATA[31:0]. The bus interface unit BIU routes the information onto the internal data bus IDATA[31:0] and maps the bytes to the corresponding operands if the CPU is not a 32-bit architecture. IDATA carries the information to the different units connected to it. A module bus interface MBI links the unit with the internal control bus and the bi-directional data bus. It splits the data path into two one way traffic paths MDATAR for read operations from the unit and MDATAW for write operations to the unit, each 32 bits wide. Furthermore the proper control signals to drive the gates on the latches MREGR[x] for read latches and MREGW[x] for write latches are generated. The number in brackets [x] harmonizes with the module register address. A module data bus multiplexer wires the various input channels to the MBI input data path. It is controlled by the module select signal MSEL generated in the MBI block. Reading from unit register address 0 for example joins multiplexer channel 0 with IDATA[31:0].

7.1.1 Dynamic Bus Sizing

To adapt the CPU interface to various processor architectures a technique called dynamic bus sizing is supported [MOT6890, 7-6]. All access points to the UTCSU internal units are 32 bits wide allowing a standardized interface logic.

The UTCSU dynamically interprets the port size of the CPU or communications chip during each bus access, allowing operand transfers to or from 8-, 16-, and 32-bit ports. The master device must signal its port size (byte, word or long word) at the beginning of a read or write operation. For 8-bit ports BUS8 must be enabled, for 16-bit ports BUS16 must be enabled and for 32-bit ports neither BUS8 nor BUS16 should be active as shown in Table 7-1.

BUS8	BUS16	Port Size
1	0	Byte
0	1	Word
0	0	Long Word
1	1	Long Word

Table 7-1 Size Signal Encoding

Dynamic bus sizing requires that the portion of the data bus used for a transfer to or from a particular port size must be fixed. A 32-bit port must reside on data bus bits 0-31, a 16-bit port must

reside on data bus bits 16-31, and an 8-bit port must reside on data bus bits 24-31. The bytes of operands are designated as shown in Figure 7-2.

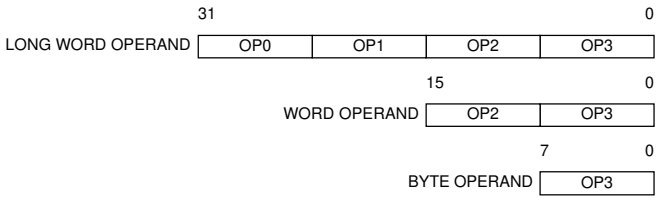


Figure 7-2 Internal Operand Representation

Figure 7-3 shows the required organization of data ports on the UTCSU HOST interface for 8-, 16-, and 32-bit devices. The four operands shown in Figure 7-3 are connected through the internal data bus and data routing to the external data bus. Misaligned operands are not supported with this interface. The data routing module establishes the necessary connections for different combinations of address and data sizes.

The multiplexer takes the four operands (OP0-OP3) and routes them to their required positions. For example, OP3 can be routed to D[7:0] for 32-bit ports or D[23:16] for 16-bit ports or D[31:24] for 8-bit port width. The positioning is determined by the port size (BUS8, BUS16) and the address (BE[3:0]) inputs.

7.1.1.1 32-bit Port Size

Interfacing a 32-bit port to the UTCSU is straight forward. The data bus is connected to the UTCSU external data bus and the transfer width is 32 bits for read and for write operations. Latched read operations are triggered by BE[3] and write operations affect all 32 bits at the same time.

7.1.1.2 16-bit Port Size

Interfacing a 16-bit port to the UTCSU requires special precautions for write operations. A read from a unit always activates all 32 bits on the internal data bus (IDATA). The bus interface unit either propagates bits IDATA[31:16] if BE3=1 or IDATA[15:0] if BE1=1 onto the data port pins D[31:16]. If data have to be latched before reading, e. g. dynamic registers like NTPTIME, the big-endian - little-endian input (BIGEND) decides whether data is latched on OP0 access or OP2 access. If BIGEND=1 data is latched on OP0 access, if BIGEND=0 data is latched on OP2 access.

A 16-bit write claims for 16-bit wide registers. Hence the 32-bit wide UTCSU registers must be configurable for a 16-bit write operation. The multiplexer again determines the routing of OP0-OP3 to D[31:16] by decoding BE[3:0].

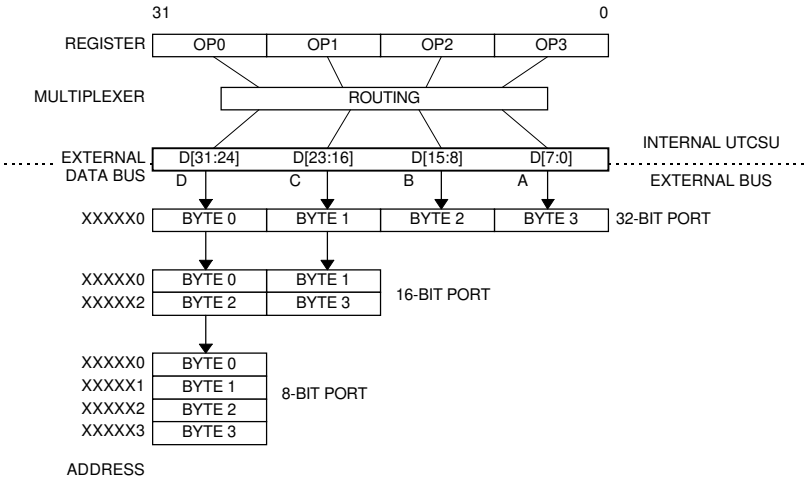


Figure 7-3 UTCSU Interface to Various Port Sizes

7.1.1.3 8-bit Port Size

Interfacing to an 8-bit port requires similar provisions except that only 8 bits of data is transmitted at a time. During a read, IDATA[31:0] carries all 32 bits of data. Depending on BE[3:0] the data multiplexer connects OP0-OP3 to the external data bits D[31:24]. If data have to be latched before reading and BIGEND=1, data is latched on OP0 access. If BIGEND=0 data is latched on OP3 access.

An 8-bit write claims for 8-bit wide registers. Hence the 32-bit wide UTCSU registers must be configurable for an 8-bit write operation. The multiplexer again determines the routing of OP0-OP3 to D[31:24] by decoding BE[3:0].

A summary is given in Table 7-2. The first two rows represent a 32-bit port, the next four rows a 16-bit port and the last four rows the 8-bit port. The section names A, B, C, D are referenced in Figure 7-3.

BUS8	BUS16	BE[3]	BE[2]	BE[1]	BE[0]	D	C	B	A
0	0	x	x	x	x	OP0	OP1	OP2	OP3
1	1	x	x	x	x	OP0	OP1	OP2	OP3
0	1	1	x	x	x	OP0	OP1		
0	1	x	1	x	x	OP0	OP1		
0	1	0	0	1	x	OP2	OP3		
0	1	0	0	x	1	OP2	OP3		
1	0	1	x	x	x	OP0			
1	0	0	1	x	x	OP1			
1	0	0	0	1	x	OP2			
1	0	0	0	0	1	OP3			

Table 7-2 Internal to External Data Bus Multiplexer

As no misaligned transfers are allowed, no special precautions are necessary to handle this situation. Any attempts to make a 16-bit transfer from an odd address will result in wrong data. The same is true for a 32-bit transfer. Only the programmer is responsible for this situation and must be aware of aligning data to match the correct port size. Figure 7-3 shows the allowed address for 32-, 16-, and 8-bit port size.

7.1.2 Multiple CPU Access

In gateway nodes or redundant system configurations, eg triple redundant fault-tolerant networks, multiple CPUs and multiple communication coprocessors (COMCOs) could access the UTCSU registers at the same time. External arbitration logic must guarantee that the first transfer be completed before the next transfer can start. A typical gateway node is shown in Figure 7-4. COMCO1 serves synchronization subnet 1 (SSN1) and COMCO2 serves synchronization subnet 2 (SSN2). UTCSU, COMCO1 and COMCO2 are on the same bus system together with CPU1 and CPU2. All four devices access the UTCSU through the same bus interface. To guarantee atomic CPU read and write operations to registers which are wider than the interface port width, a bus arbiter must lock out intermediate UTCSU accesses. For 32-bit port width only the NTPTIME register (91 bits wide) access must be atomic. On 16-bit and 8-bit ports all read and write operations need arbitration to ensure atomic operation.

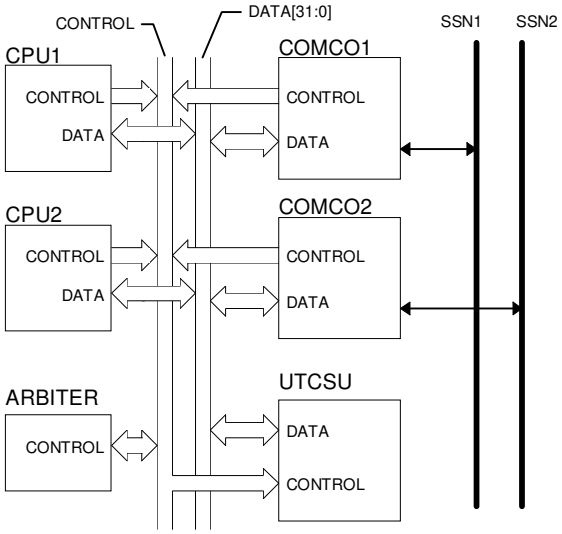


Figure 7-4 Multiple UTCSU Register Access

All communication coprocessors (COMCO1, COMCO2, ...) use dedicated UTCSU register sets (SSUx) allowing an interruptable bus access. The maximum number of COMCOs using the UTCSU as the time source is only limited by the maximum read latency introduced by ordering the different UTCSU accesses one after the other. A single COMCO access (t_{COMACC}) for a 64-bit timestamp is given by the following equation.

$$t_{COMACC} = \frac{64\ bit}{portwidth} * \max\left\{t_{ACCRUL}, t_{COMREAD}\right\} \qquad \text{Equ. 7-1}$$

max{a, b} denotes the maximum of a and b. t_{ACCRUL} represents the UTCSU read access time for unlatched data. $t_{COMREAD}$ is the required read access time for the communication coprocessor. Portwidth defines a 32-, 16-, or 8-bit data port. As t_{ACCRUL} is very small, the total read time is mainly determined by $t_{COMREAD}$. The equation above only shows the best case for a single communication coprocessor (COMCO). The worst case number $t_{COMMACC,WC}$ depends on the number of COMCOs (N) accessing the UTCSU and is specified as

$$t_{COMACC,WC} = (N-1) * t_{COMMACC} \qquad \text{Equ. 7-2}$$

7.2 CSUTIME-PORT

The CSUTIME-port comprises 49 primary output pins plus one primary input pin. Its functionality is twofold. During normal system operation, timestamps and accuracystamps are passed to the outside world. In test mode, test signals can be observed on these pins.

7.2.1 Normal Operation

The current 64-bit timestamp plus 16-bit positive and 16-bit negative accuracystamps are externally available through the CSUTIME-port data pins if signal TESTMUX is low. Observe, that the 96 data bits are multiplexed onto 48 data pins in order to reduce the total pin count. The upper 48 bits are available while signal CSUTIMEPHASE being high and vice versa.

7.2.2 Test Mode

In test mode, when signal TESTMUX is high, 48 chip internal control signals can be observed on these pins, cf. Table 8-52.

7.3 SYSTEM-PORT

7.3.1 System Reset

The chip is reset during power up and if the input pin RESET is forced high.

7.3.2 Interrupt Handling

Three open collector output pins INTN, INTA and INTT carry along interrupts.

7.3.3 System Clock

A quartz crystal in the frequency range between 2 MHz and 20 MHz can directly interface with a two pin on-chip clock oscillator. If a TTL compatible clock signal is available, it must be connected to the OSCIN input. The buffered system clock can be observed on pin TCLK.

7.4 EVENT-PORT

7.4.1 External Event Inputs

External event inputs are synchronized to the system clock in the SYNC unit. The UTCSU processes events from NCSCs (TSRCV[5:0], TSXMT[5:0]), GPS receivers (GPS1PPS[2:0], GPSSTAT[2:0]), application interfaces (TSAPP[8:0]) and debugging aids (HWSNAP).

7.4.2 Event Outputs

The APPDUTY output generates a preprogrammed event, cf. Section 8.5.2.

7.5 TEST-PORT

7.5.1 Scan-Path Port

All units in the UTCSU are prepared for scan-path testing. The test methodology is full-scan with multiplexed flip-flops. Hence a scan enable pin SCANEN, scan-chain inputs SCANIN[9:0] and scan-chain outputs SCANOUT[9:0] for 10 scan-chains must be provided. SCANTESTMODE must be driven high during scan-test to disable the software reset logic and the multiplexer on the CSUTIME-port.

7.5.2 Boundary-Scan Port

The five pin IEEE 1149.1 boundary-scan port comprises pins

- TRST: test reset,
- TMS: test mode select,
- TCK: test clock,
- TDI: test data in, and
- TDO: test data out.

The optional TRST pin is required by the Synopsys Testcompiler automatic test logic insertion and test pattern generation. Supported tests are specified in Section 5.4.4.

CHAPTER 8

8 UNIT SPECIFICATION

Alles Ding währt seine Zeit.
Paul Gerhardt

Chapter 8 talks about the nuts and bolts of the twelve UTCSU units. At the beginning of every subsection, the unit architecture is illustrated and a functional description is given. Subsequently debugging features as well as configuration bits are discussed. A list of all status and interrupt signals follows. All registers are summarized in an agglomerating register map but are also portrayed in a detailed programmers model fashion. Finally the unit test strategy is explored.

8.1 LOCAL TIME UNIT

The local time unit (LTU) maintains standard time based on either the Extended Network Time Protocol (ENTP, Section 3.2.2) format with leap second correction or a Modified NTP format (MENTP, Section 3.2.2) without leap second correction. Rate and state correction (as required by the ICV algorithm [SS95]) are supported in order to serve different frequency oscillators that differ in clock speed and quality.

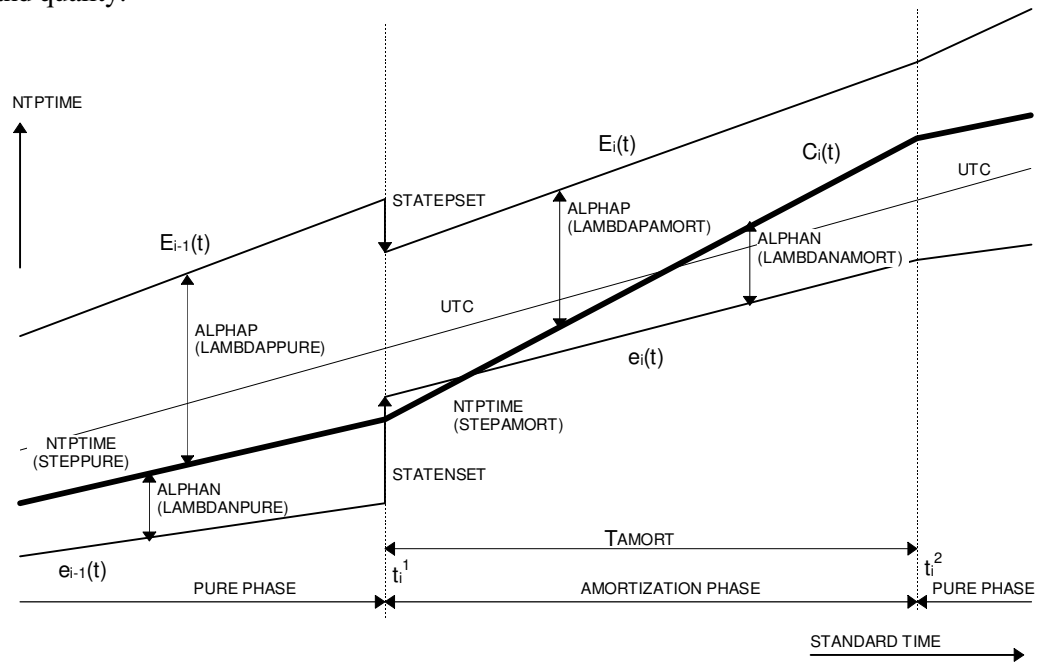


Figure 8-1 Register Components to Maintain Local Time and Accuracy

The heart of this unit consists of the novel adder-based clock from Section 3.6.3. A similar picture as the one shown in Figure 8-1 was presented in section 3.6.3 except that all details including register names were missing.

The internal representation of local time is 91 bits wide spanning the time range from 1.73 as in the least significant bit position NTP [-59] to 68 years in the most significant bit position NTP [+31]. More detailed information on NTP can be found in Section 3.2.1. The UTCSU external representation of local time is 56-bit wide plus an 8-bit error correcting control word (CS), resulting in a 64-bit wide external timestamp on INTPBUS[CS:+31,-24].

Note! Timestamps on the INTPBUS are latched in NTPHOLD, hence delayed by one T-clock.

Figure 8-2 illustrates the architecture of the LTU. A step by step explanation of all sub-blocks follows in the upcoming sections. Introductory one can say that time increments from register STEPAMORT during amortization phase or STEPPUREACT during pure phase are added in the NTPADDER to the locale clock register NTPTIME. The 64-bit wide INTPBUS passes the generated timestamps to the attached units.

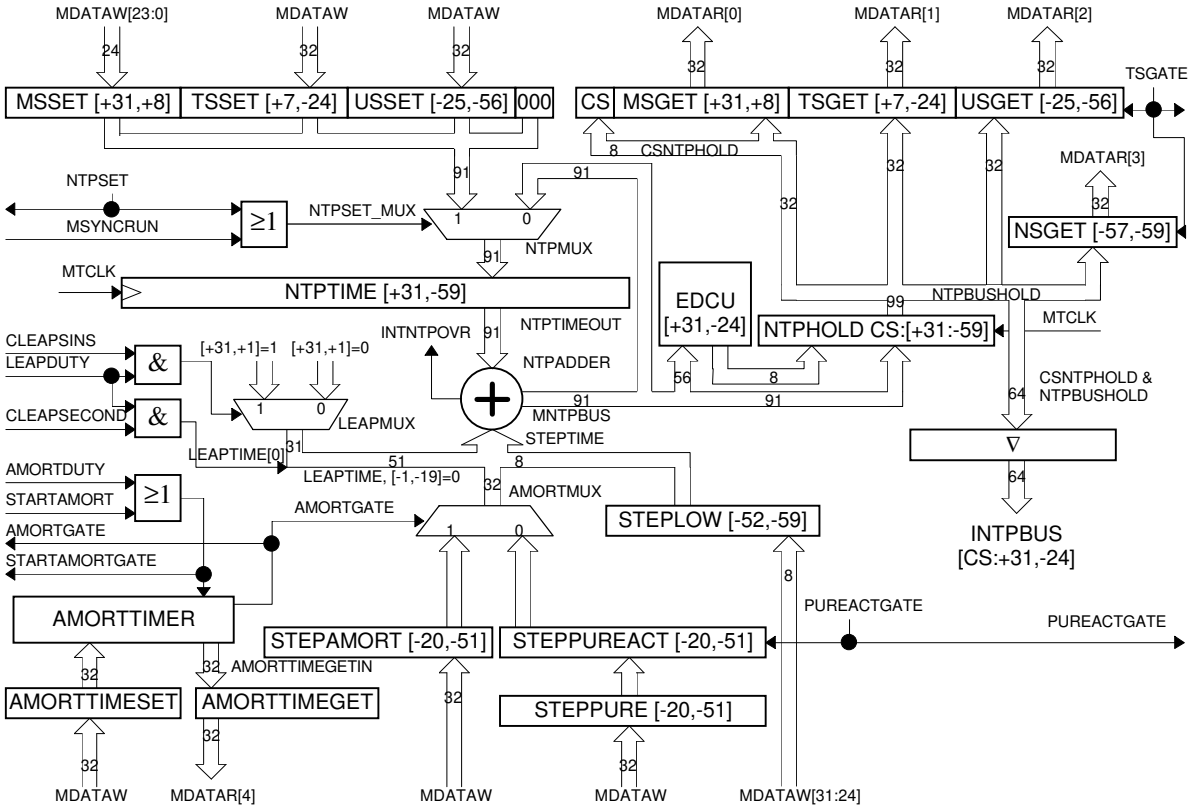


Figure 8-2 Local Time Unit (LTU) Architecture

8.1.1 Validating Register Layout

To justify the configuration given above, we first note that the time granularity G_t is

$$G_t = \frac{1}{f_{osc}}. \qquad \text{Equ. 8-1}$$

Of course, G_t is also the smallest amount of clock correction that can reasonably be applied during one resynchronization interval R. This lower bound on clock correction (G_t) within one resynchronization interval R results in a step granularity G_{step} as given below

$$G_{step} = \frac{G_t}{f_{osc} * R} = \frac{1}{f_{osc}^{\ 2} * R} \qquad \text{Equ. 8-2}$$

Assuming a resynchronization interval R = 30 s and running from a 8.388608 MHz (NTP bit [-23]) oscillator, G_{step} evaluates to

$$G_{step} = \frac{G_t}{f_{osc} * R} = \frac{119.21 ns}{8.388608\,MHz * 30\,s} = 473.697\,as \qquad \text{Equ. 8-3}$$

Looking up this value in Table 3-5 defines NTP bit position [-51] as $G_{step,UTCSU}$. $G_{step,UTCSU}$ = Bit [-51] = 444 as is the smallest possible non zero amount added to NTPTIME at each oscillator tick.

The upper bound for registers STEP is on one hand given by the slowest oscillator frequency and on the other hand by a practical register length. For low resolution system configurations eg time resolution in the 1 µs range, the quantity added to NTPTIME on every timer tick should accommodate 1 µs jumps. Again looking up Table 3-5 returns bit [-19] as the 1.9 µs step. Validating the bit range between [-20] and [-51] consumes a 32 bit wide STEP register.

Knowing the bounds on STEP leads to the range of supportable oscillator frequencies. The lower limit is set by the maximum STEP register content which is

$$f_{osc,low} \approx \frac{1}{2^{-19}\,s} = 524288\,Hz \qquad \text{Equ. 8-4}$$

and the upper limit is given by the smallest STEP register content ($G_{step.UTCSU}$) in order to correct a single G_t within one resynchronization interval R. $G_{step,UTCSU}$ defines the upper frequency for a resynchronization interval length of R = 30 s as

$$f_{osc,high} = \sqrt{\frac{1}{G_{step,UTCSU} * R}} = \sqrt{\frac{1}{444 \cdot 10^{-18}\,s * 30s}} = 8.664587\ \ MHz \qquad \text{Equ. 8-5}$$

and for a resynchronization interval length of R = 3 s as

$$f_{osc,high} = \sqrt{\frac{1}{G_{step,UTCSU} * R}} = \sqrt{\frac{1}{444 \cdot 10^{-18}\,s * 3s}} = 27.399831\ \ MHz \qquad \text{Equ. 8-6}$$

Of course, there is also a (technology-dependent) upper limit. Actually, all simulations on chip level are done for a 25 MHz oscillator clock signal which is equivalent to a 40 ns T-clock period.

G_{sf} denotes the full state granularity. For nominal oscillator frequencies being multiples of 2 (binary frequencies) G_{sf} evaluates to G_{step}. This restricts NTPTIME to NTP bit [-51] in the LSB position. If a very precise oscillator is used whose frequency is not a multiple of 2, eg a 10 MHz GPS output, NTPTIME needs to be expanded to smaller LSBs to satisfy high accurate decimal time increments. For example, converting 100 ns (10 MHz) into NTP-format returns for bits [-24,-64]

$$100\,\text{ns} \xrightarrow{[-24,-64]} 11010110101111111001010011 01[01011110]01010 \qquad \text{Equ. 8-7}$$

Investigating bits [-52, -64] strongly suggests to incorporate eight additional bits [-52, -59]. For this setting full state granularity evaluates to G_{sf} = 1.73 as (corresponding to NTP bit [-59] in NTPTIME), so that truncation errors resulting an oscillator frequency of 10 MHz impair the intrinsic rate of the physical clock by at most

$$IR_f = \frac{G_{sf}}{G_t} = \frac{1.73as}{100\,ns} = 1.73 * 10^{-11} \qquad \text{Equ. 8-8}$$

By contrast, the value of G_{sf} achievable without the additional bits [-52, -59] would give a rate error of

$$IR_f = \frac{G_{sf}}{G_t} = \frac{0.444\text{fs}}{100\,\text{ns}} \approx 4 * 10^{-9} \qquad \text{Equ. 8-9}$$

which would intolerably impair the excellent rate ($10^{-11}..10^{-12}$) provided by GPS receiver for example. Table 8-1 lists the STEPLOW contents for various decimal oscillator frequencies.

Nominal Frequency	STEPLOW [-52, -59]
1 MHz	10101111
5 MHz	10111100
10 MHz	01011110
20 MHz	10101111

Table 8-1 STEPLOW contents for decimal oscillator frequencies

This guides us to a new definition of NTPTIME in the bit range of [+31, -59]. An 8 bit STEPLOW register is appended to STEP, holding the ultrafractional part of STEP.

It is obviously not necessary to deal with 40 bit correction values (STEP). The concept presented above splits STEP into a practicable 32 bit correction register (STEPPURE, STEPAMORT) and a dedicated 8 bit preload register (STEPLOW) which solely depends upon the fixed supplying oscillator frequency. This gives a state granularity G_{sc} of 0.444 fs (corresponding to NTP bit [-51]), so that for the nominal oscillator frequency, the smallest non-zero rate correction

$$IR_c = \frac{G_{sc}}{G_t} = \frac{0.44\,fs}{119.21\,ns} = 0.37 * 10^{-8} \qquad \text{Equ. 8-10}$$

which is equivalent to 1 µs per 270.4 s. Note however that varying f_{osc} (change clock frequency) impacts G_t and therefore IR_f and IR_c computed above.

8.1.2 Initialization and Verification

The NTPTIME register holds local time in NTP time format. After reset this register is cleared to 0. Loading the start up time information into NTPTIME requires 4 write operations to the following registers.

# Write	LTU Address	LTU Register Name	NTP Time Equivalent.
1	0	STEPLOW, MSSET	[-52,-59], [+31,+8]
2	1	TSSET	[+7,-24]
3	2	USSET	[-25,-56]
4	3	NTPSET	-

Table 8-2 Initialization Sequence for NTPTIME Register

The group of SET registers namely Macrostamp Set (MSSET), Timestamp Set (TSSET) and Microstamp Set (USSET) holds the preload value for the NTPTIME register. Writing to LTU address NTPSET enables the preload path as indicated in Figure 8-3 for one T-clock cycle. The start up time is captured into NTPTIME and from now on updated with various time steps from the STEP registers.

Verifying the current NTPTIME contents is possible by an atomic read of MSGET, TSGET, USGET and NSGET registers. Reading MSGET returns a 32 bit number where the upper eight bits reveal a checksum for NTP bits [+31,-24]. Note that all GET registers listed above are used for test purposes too (SW-snapshot in Section 8.6).

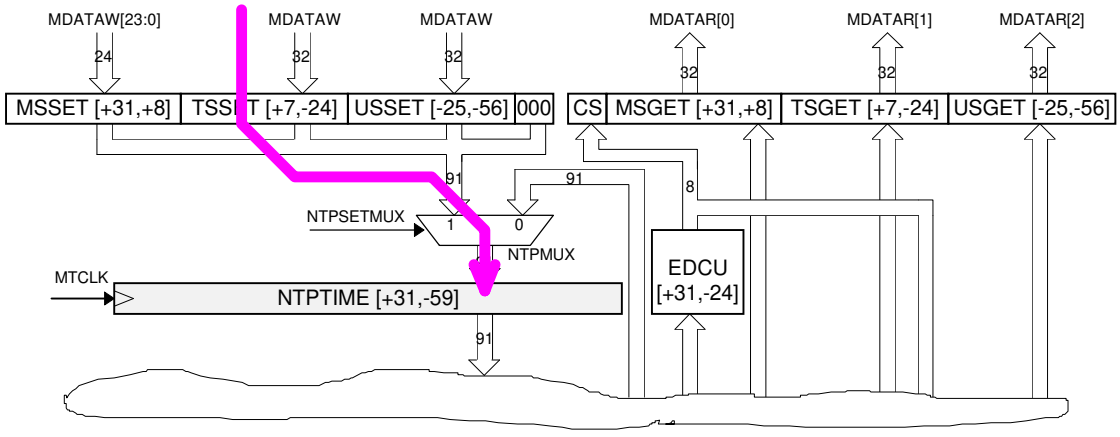


Figure 8-3 Local Time Initialization Data Path

All STEP registers (STEPLOW, STEPPUREACT, STEPPURE, STEPAMORT) are cleared after reset prohibiting any progress in register NTPTIME. Register STEPLOW must subsequently be set according Table 8-1 for non binary nominal frequencies and cleared for binary frequencies. STEPPUREACT is updated with the contents of the preload register STEPPURE only at the beginning of an amortization phase. Hence after reset no progress in time is made in register NTPTIME until the first amortization phase was initiated.

The preload register AMORTTIMESET for the amortization timer is cleared upon reset.

8.1.3 Amortization Phase

Continuous amortization is the process involved with state and rate corrections. The timing behaviour for 2 alternating phases is illustrated in Figure 8-5. First the step register used during amortization (STEPAMORT) is loaded with the step data (AMORT DATA = A) obtained by the clock synchronization algorithm. Next the amortization interval length, specified in T-clock ticks (00000003), is written into register AMORTTIMESET. Interval length between one and $(2^{32}-1)$ T-clocks are programmable. Either writing LTU address 7 (STARTAMORT) or an expiration of duty-timer 2 in unit synchronization subnet 1 (SSU1) triggers signal AMORTGATE and starts the amortization phase. NTPTIME is increased by the value read from STEPAMORT + STEPLOW (A) as illustrated in Figure 8-4 and Figure 8-5, where registers involved in continuous amortization are shaded.

Phase T4+A until phase T6+A in Figure 8-5 show the continuous amortization process. Right after AMORTGATE goes high, STEPPURE is copied into STEPPUREACT by activating PUREACTGATE which updates STEPPUREACT for the upcoming pure phase.

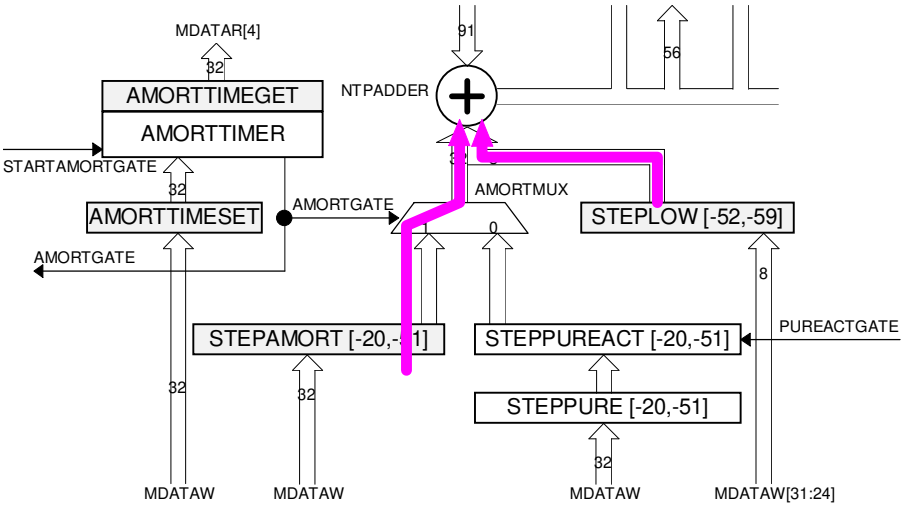


Figure 8-4 Data Path during Continuous Amortization Phase

In parallel the AMORTTIMER starts to decrement the preload value from AMORTTIMESET on every T-clock. Observe that the remaining amortization period is accessed by reading AMORTTIMEGET with two different read semantics. A read operation from LTU address 6 first latches the current amortization timer value into AMORTTIMEGET and then propagates the latched value on to the data bus. Reading LTU address 7 simply returns the contents of register AMORTTIMEGET without latching the current timer value. For details see Section 8.1.6.2.

If the amortization phase expires (AMORTTIMER = 0) then the gate signal AMORTGATE is deactivated and the pure phase is introduced. In the sequel multiplexer (AMORTMUX) switches back to pure register setup and from now on will STEPPUREACT together with STEPLOW supply NTPTIME with time steps (T7+PN in Figure 8-5).

Note! Register STEPAMORT must not be modified during continuous amortization!

An ongoing amortization phase can be untimely terminated by writing the all zero word into register AMORTTIMESET and writing LTU address STARTAMORT. This disables AMORTGATE and switches AMORTMUX back to pure mode. Observe that status bit (SCAMA) in register UTCSTAT reflects the current state of AMORTGATE and provides information about the current phase.

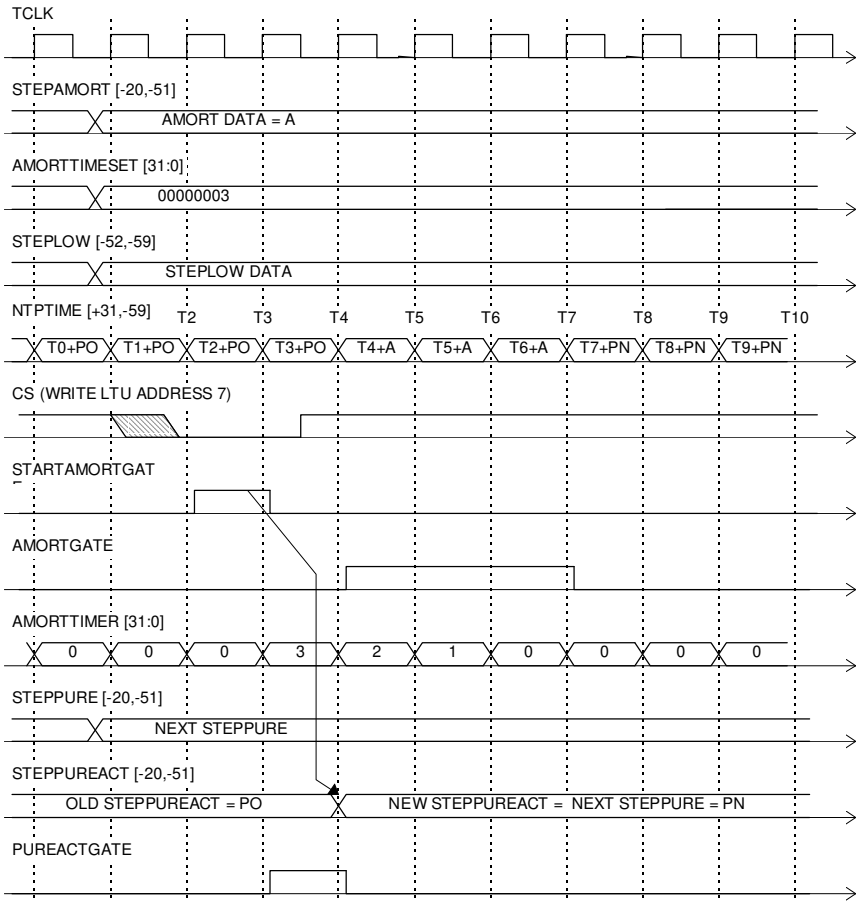


Figure 8-5 Amortization Phase Timing Diagram

8.1.4 Pure Phase

The pure phase follows each amortization phase. When AMORTGATE is low, the multiplexer AMORTMUX switches to STEPPUREACT and time register NTPTIME is increased by time steps read from registers STEPPUREACT + STEPLOW as shown in Figure 8-6. There is no time limit involved with pure phase like an AMORTTIMER but it is terminated by introducing a new amortization phase. Timing information is provided in Figure 8-5. The old pure phase between T0 + PO and T3 + PO (PO...pure old) is taken over by 3 amortization clock cycles at T4. At T7 a new pure phase T7 + PN (PN...pure new) supersedes continuous amortization.

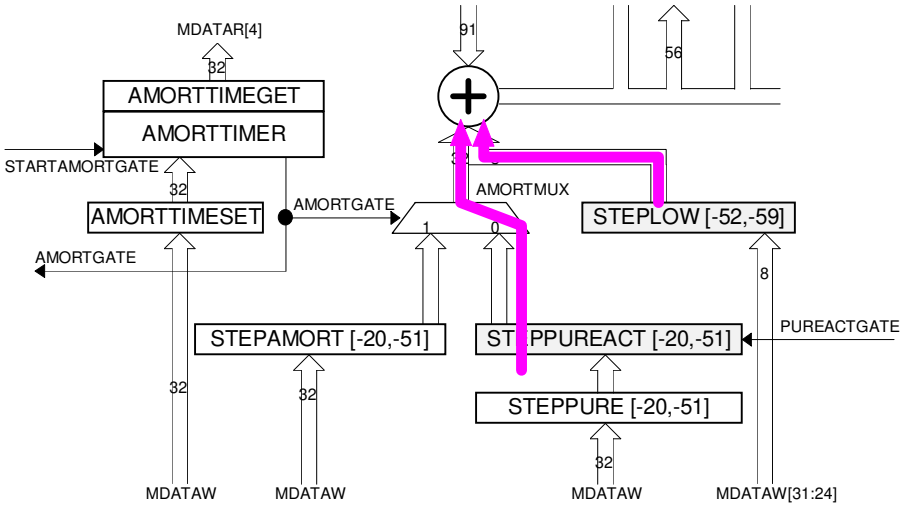


Figure 8-6 Data Path during Pure Phase for the Time Step Data

8.1.5 NTP Time vs. GPS Time

By definition NTP timescale follows the UTC timescale. Every time the second difference between atomic (TAI) and astronomic (UT1) time exceeds 0.7 s, a leap second is inserted or deleted on the last day of June or December. Details are revealed in Section 3.1. Some applications prefer GPS-time which is a chronoscopic time source without any leap second corrections. The UTCSU supports both time notions, atomic time (eg GPS-time) and astronomic time (eg NTP-time).

A mechanism called LEAPDUTY is implemented to allow leap second corrections. Leap seconds can be deleted (CLEAPSECOND) or inserted (CLEAPSINS). Deletion means an addition of 1 s at the beginning of second 23:59:59 thus omitting second 23:59:59. A leap second is inserted following second 23:59:59 by subtracting 1 s, hence second 23:59:59 exists twice. Observe that timestamps generated within this period have an uncertainty of 1 s.

For example; a leap second should be inserted on June 30th 1983, see Table 3-2. First the configuration bit CLEAPSECOND must be set to indicate a pending leap second correction. Next the configuration bit CLEAPSINS must be set to insert a leap second or reset to delete a leap second. Finally the duty timer B in synchronization subnet 3 (see Section 8.3) must be programmed to generate a gate pulse (LEAPDUTY) at time 0:00:00 on July 1st 1983. This triggers the subtraction logic in the LTU to subtract a single second from the NTPTIME register. A detailed timing diagram is illustrated in Figure 8-7. Table 8-3 summarizes the configuration bits dedicated to leap second adjustment.

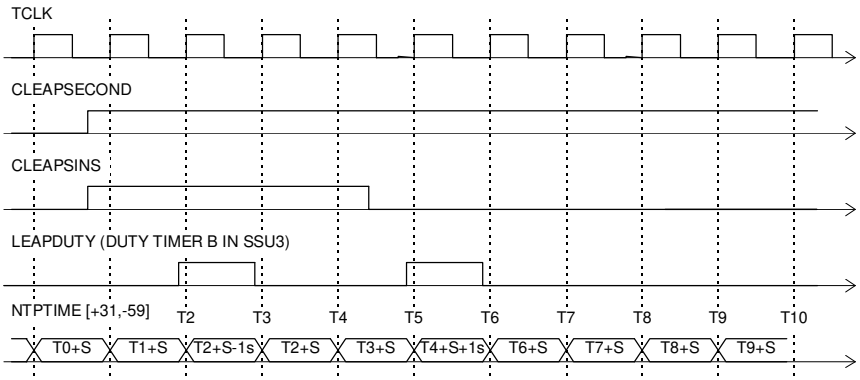


Figure 8-7 Leap Second Insertion and Deletion

Whenever a leap second is inserted, an NTPTIME register overrun interrupt (INTNTPOVR) is fired in addition to the regular NTPTIME overrun once every 136 years.

Note! Leap Second Insertion fires an NTPTIME overrun interrupt (INTNTPOVR)!

Operation	CLEAPSECOND	CLEAPSINS
normal	0	0
delete leap second	1	0
insert leap second	1	1

Table 8-3 Configuration Bits (CLEAPSECOND, CLEAPINS) Control Leap Second Correction

8.1.6 Debug and Test

Debugging this unit in a small network of several UTCSU nodes is a two step process. First all UTCSUs must be set into a common state and started coincidentally. Second all dynamically changing registers need to be read during normal operation for debugging purposes. The first feature is called SYNCRUN and is specified in the next paragraph while the second feature is called snapshot mode and discussed in Section 8.1.6.2.

8.1.6.1 Synchronous Start

An external input signal called SYNCRUN provides a gate signal similar to NTPSET. A rising or falling edge, whatever is programmed, on signal SYNCRUN generates a gate pulse on signal ISYNCRUN which is connected to the MSYNCRUN input of the LTU. MSYNCRUN is connected to

NTPMUX the same way as NTPSET. The active edge on SYNCRUN copies all NTP preload registers MSSET, TSSET, USSET into NTPTIME and initializes the unit.

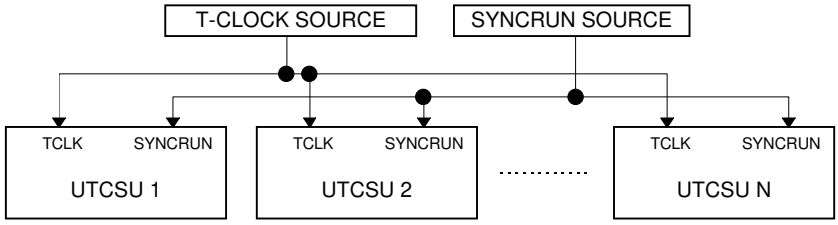


Figure 8-8 Synchronous Start in Debug Mode and for Redundant Operation

8.1.6.2 Snapshot Mode

The LTU supports software snapshot mode (SSNAP) for debug and test purposes. In snapshot mode all dynamically changing UTCSU registers are sampled with signal MSNAPSHOT which is either software or duty-timer controlled.

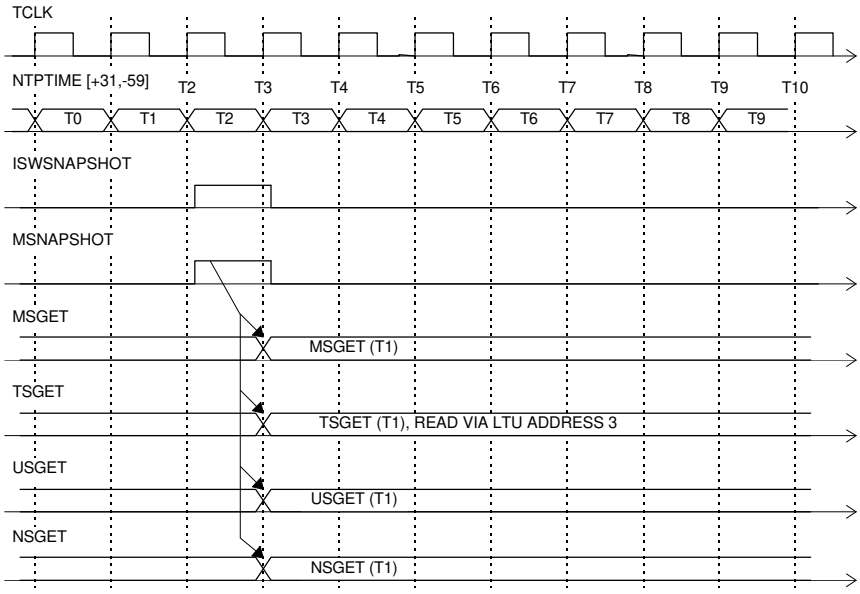


Figure 8-9 NTPTIME Snapshot Mode Timing Diagram

As referenced in Table 8-8 register NTPTIME and the AMORTTIMER change contents every T-clock. All other registers within the LTU are more or less static.

The 91 bits wide NTPTIME register is accessible through latching registers MSGET, TSGET, USGET and NSGET. Right now a read on TSGET performs an atomic latch operation of all 91 NTPTIME bits into the 4 registers listed above. In contrast to TSGET, reading MSGET, USGET and NSGET just return the latched value and no new data is captured. This behaviour is almost ideal to support snapshot mode, however modifications to TSGET are inevitable.

A rising edge on signal MSNAPSHOT acts like a read from register TSGET. All 91 NTPTIME bits are transferred into MSGET, TSGET, USGET and NSGET. Consequently reading MSGET, USGET and NSGET return the snapped time but accessing TSGET would do another capture and the snapshot is lost. Prevention from data loss is possible by providing a non latching read semantics for TSGET. Currently reading LTU address 1 latches NTPTIME and returns TSGET while reading LTU address 3 simply returns the contents of TSGET, see Figure 8-9.

In order to support snapshot mode the amortization timer needs some modifications. A rising edge on signal MSNAPSHOT samples the current state of AMORTTIMER into AMORTTIMEGET, see Figure 8-10. Reading LTU address 6, to obtain the new AMORTTIMER value, would do another latch operation and overwrite the snapshot in AMORTTIMEGET. Hence a new read access for regis-

ter AMORTTIMEGET is introduced. Reading LTU address 7 returns the contents of AMORT-TIMEGET neglecting a preceding latch operation.

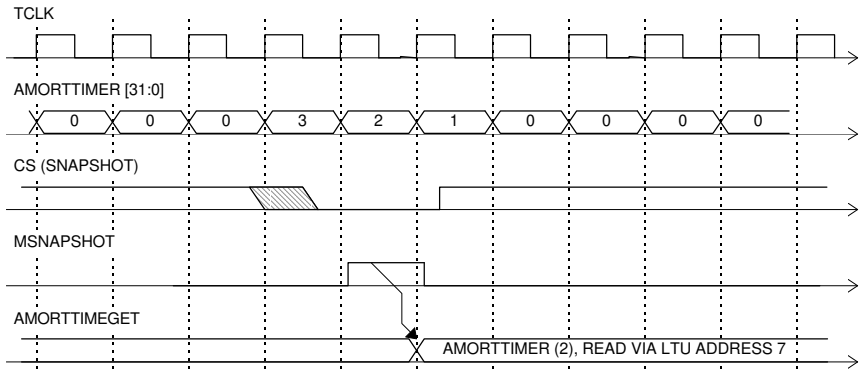


Figure 8-10 Amortization Timer Snapshot Mode Timing Diagram

8.1.7 Error Detection and Correction Unit

An error detection and correction unit (EDCU) provides an 8 bit control word for NTP bits [+31,-24] on the internal and external accessible NTP-Bus. 56 data bits + 8 control bits form the 64 bits wide internal NTP-Bus (INTPBUS) whose layout is illustrated in Figure 8-11.

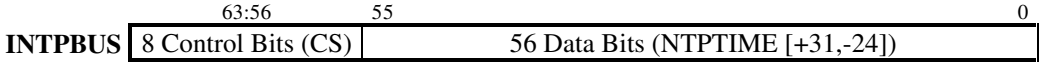


Figure 8-11 Eight Control Bits protect 56 Data Bits

The EDCU uses a modified Hamming Code [KL95] with Hamming Distance HD = 4. Observe that the following errors can be detected and/or corrected. Furthermore is the number of protected data bits a function of the available control bits.

- correct single bit errors; detect all double bit errors and some triple bit errors
- detect the gross error conditions of all 0 or 1
- 6 bit control word protects 20 data bits NTP bits [-5, -24]
- 7 bit control word protects 35 data bits NTP bits [+10, -24]
- 8 bit control word protects 56 data bits NTP bits [+31, -24]

Scalability as introduced in Section 5.5.1 is a fundamental characteristic for this code. Whatever the number of available control bits is, the number of protected data bits can be adapted. Of course, 8 control bits can protect up to 56 data bits while 7 control bits protect up to 35 data bits and 6 control bits up to 20 data bits. A 7 bit down-scaling is used in the GPS unit (GPU) where the most significant bit of the control word is reserved for GPS status information.

The following considerations underlie the construction of the matrix for generating the control bits. Each data bit belongs to 3 different control bits. Assume 8 resp. 7 resp. 6 control bits as shown in Figure 8-11. How many combinations are possible to form different groups of 3 elements if the total number of elements is 8 resp. 7 resp. 6? The answer is

$$\binom{8}{3} = \frac{8 \cdot 7 \cdot 6}{1 \cdot 2 \cdot 3} = 56 \qquad \text{Equ. 8-11}$$

56 resp. 35 resp. 20 different triples can be formed. Hence 8 control bits can protect 56 data bits which is almost ideal if we look at the INTPBUS format. Each of the 56 data bits is a member of 3 control bits. Hence all 21 members $\left(56 * \frac{3}{8}\right)$ of a single control bit are connected to a 21 input XOR

gate. An optimized matrix in terms of gate count under the assumptions made at the beginning of this section is given in Figure 8-12. Reusable blocks of four bits are shaded while blocks of 6 bits have a dark border.

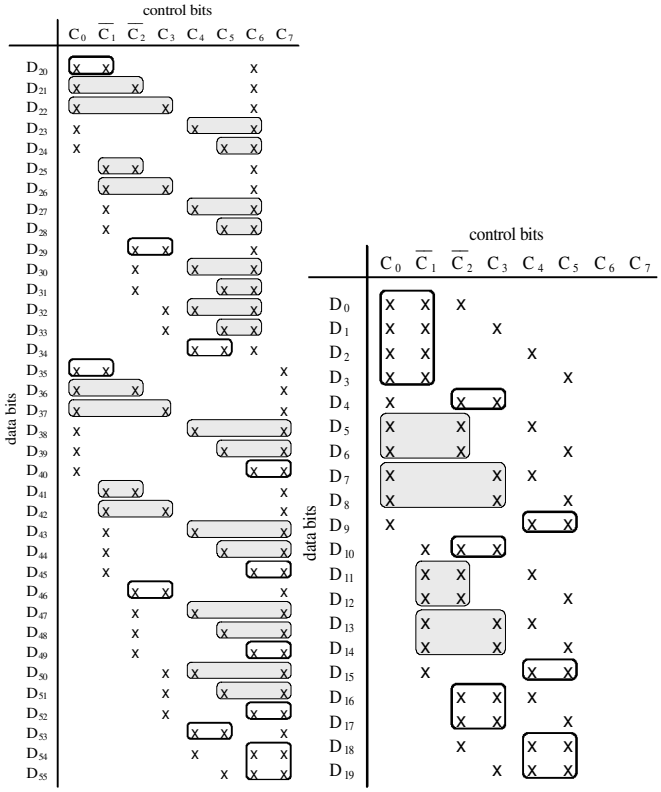


Figure 8-12 The Optimized Matrix requires 116 Two-Input XOR-Gates

All matrixes given above are summarized in the equations of Figure 8-13. The symbol ⊕ represents the XOR function. First of all 4 blocks holding 6 data bits for one control bit are listed. They are named $M_{0,1}$ to $M_{6,7}$ and framed with a dark border in Figure 8-12. Next 8 blocks combining 4 data bits into one block are listed and named $F_{0,2}$ to $F_{5,7}$. In Figure 8-12 they are shaded. Finally the complete equations for the 8 control bits C_0 to C_7 are written down referencing the defined blocks.

$$M_{0,1} = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_{20} \oplus D_{35}$$
$$M_{2,3} = D_4 \oplus D_{10} \oplus D_{16} \oplus D_{17} \oplus D_{29} \oplus D_{46}$$
$$M_{4,5} = D_9 \oplus D_{15} \oplus D_{18} \oplus D_{19} \oplus D_{34} \oplus D_{53}$$
$$M_{6,7} = D_{40} \oplus D_{45} \oplus D_{49} \oplus D_{52} \oplus D_{54} \oplus D_{55}$$

$$F_{0,2} = D_5 \oplus D_6 \oplus D_{21} \oplus D_{36}$$
$$F_{0,3} = D_7 \oplus D_8 \oplus D_{22} \oplus D_{37}$$
$$F_{1,2} = D_{11} \oplus D_{12} \oplus D_{25} \oplus D_{41}$$
$$F_{1,3} = D_{13} \oplus D_{14} \oplus D_{26} \oplus D_{42}$$
$$F_{4,6} = D_{23} \oplus D_{27} \oplus D_{30} \oplus D_{32}$$
$$F_{4,7} = D_{38} \oplus D_{43} \oplus D_{47} \oplus D_{50}$$
$$F_{5,6} = D_{24} \oplus D_{28} \oplus D_{31} \oplus D_{33}$$
$$F_{5,7} = D_{39} \oplus D_{44} \oplus D_{48} \oplus D_{51}$$

$$C_0 = M_{0,1} \oplus F_{0,2} \oplus F_{0,3} \oplus D_4 \oplus D_9 \oplus D_{23} \oplus D_{24} \oplus D_{38} \oplus D_{39} \oplus D_{40}$$
$$\overline{C_1} = M_{0,1} \oplus F_{1,2} \oplus F_{1,3} \oplus D_{10} \oplus D_{15} \oplus D_{27} \oplus D_{28} \oplus D_{43} \oplus D_{44} \oplus D_{45}$$
$$\overline{C_2} = M_{2,3} \oplus F_{0,2} \oplus F_{1,2} \oplus D_0 \oplus D_{18} \oplus D_{30} \oplus D_{31} \oplus D_{47} \oplus D_{48} \oplus D_{49}$$
$$C_3 = M_{2,3} \oplus F_{0,3} \oplus F_{1,3} \oplus D_1 \oplus D_{19} \oplus D_{32} \oplus D_{33} \oplus D_{50} \oplus D_{51} \oplus D_{52}$$
$$C_4 = M_{4,5} \oplus F_{4,6} \oplus F_{4,7} \oplus D_2 \oplus D_5 \oplus D_7 \oplus D_{11} \oplus D_{13} \oplus D_{16} \oplus D_{54}$$
$$C_5 = M_{4,5} \oplus F_{5,6} \oplus F_{5,7} \oplus D_3 \oplus D_6 \oplus D_8 \oplus D_{12} \oplus D_{14} \oplus D_{17} \oplus D_{55}$$
$$C_6 = M_{6,7} \oplus F_{4,6} \oplus F_{5,6} \oplus D_{20} \oplus D_{21} \oplus D_{22} \oplus D_{25} \oplus D_{26} \oplus D_{29} \oplus D_{34}$$
$$C_7 = M_{6,7} \oplus F_{4,7} \oplus F_{5,7} \oplus D_{35} \oplus D_{36} \oplus D_{37} \oplus D_{41} \oplus D_{42} \oplus D_{46} \oplus D_{53}$$

Figure 8-13 Boolean Equations for the Optimized EDC Matrix (⊕...XOR-Gate)

If we analyze the hierarchy of this logic, we will see that no control bit has a longer propagation path than 5 gate delays (routing delay is disregarded). In general an upper bound on the levels of hierarchy t_{min} introduced for this version of the modified Hamming code is given by the height of a binary tree with n leaves.

$$t_{\min} = \mathrm{int}\left(\log_2\left(n-1\right)\right)+1 \qquad \text{Equ. 8-12}$$

Figure 8-14 shows the hierarchy for the modified Hamming code used in the EDCU. No more than 5 gate delays are necessary to generate the control bits.

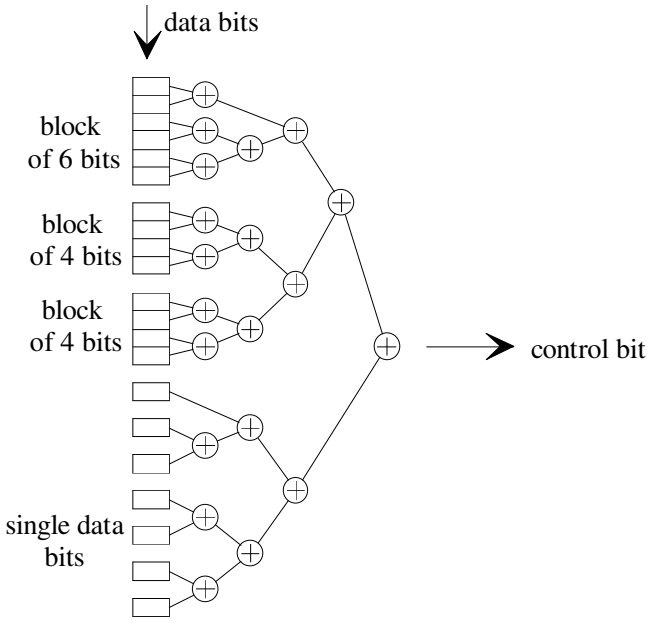


Figure 8-14 Five Levels of Hierarchy in the EDCU

In order to detect and correct bit errors on stored timestamps, the computation of the control bits on the 56 data bits has to be repeated in the analyzing processing unit. Both sets of control bits, the stored C_0 - C_7 and the newly calculated C_{N0} - C_{N7} are xored resulting in syndrome bits S_0 - S_7 as referenced in Table 8-4.

The no fault case is marked with a star, all syndrome bits are zero indicating an error free timestamp. Single bit errors are marked with the number of either the faulty data bit or the faulty control bit. Double bit errors are listed as D and triple bit errors as T. F represents a four bit error which could result in a valid but different data word for HD = 4.

				S_7	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
				S_6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				S_5	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				S_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S_0	S_1	S_2	S_3																	
0	0	0	0		*	C_7	C_6	D	C_5	D	D	34	C_4	D	D	54	D	53	55	D
0	0	0	1		C_3	D	D	52	D	51	33	D	D	50	32	D	19	D	D	T
0	0	1	0		C_2	D	D	49	D	48	31	D	D	47	30	D	18	D	D	T
0	0	1	1		D	46	29	D	17	D	D	T	16	D	D	T	D	T	T	D
0	1	0	0		C_1	D	D	45	D	44	28	D	D	43	27	D	15	D	D	T
0	1	0	1		D	42	26	D	14	D	D	T	13	D	D	T	D	T	T	D
0	1	1	0		D	41	25	D	12	D	D	T	11	D	D	T	D	T	T	D
0	1	1	1		10	D	D	T	D	T	T	D	D	T	T	D	T	D	D	T
1	0	0	0		C_0	D	D	40	D	39	24	D	D	38	23	D	9	D	D	T
1	0	0	1		D	37	22	D	8	D	D	T	7	D	D	T	D	T	T	D
1	0	1	0		D	36	21	D	6	D	D	T	5	D	D	T	D	T	T	D
1	0	1	1		4	D	D	T	D	T	T	D	D	T	T	D	T	D	D	T
1	1	0	0		D	35	20	D	3	D	D	T	2	D	D	T	D	T	T	D
1	1	0	1		1	D	D	T	D	T	T	D	D	T	T	D	T	D	D	T
1	1	1	0		0	D	D	T	D	T	T	D	D	T	T	D	T	D	D	T
1	1	1	1		D	T	T	D	T	D	D	T	T	D	D	T	D	T	T	F

Table 8-4 The Syndrome Table lists Single (#), Double (D) and Triple (T) Faults

8.1.8 Atomic Sampling Across Units

Generating accurate timestamps requires atomic read access of several time and accuracy registers in different units. The local time unit enforces atomic sampling for the following registers if TSGET is read.

- CS:MSGET
- TSGET (read access at LTU address 1)
- USGET
- NSGET
- ALPHAPGET (ACU), NALPHAPGET (ACU)
- ALPHANGET (ACU) , NALPHANGET (ACU)
- STATEPGET (ACU)
- STATENGET (ACU)

8.1.9 Internal NTP-Bus

56 bits of the local time (NTPTIME) maintained in the LTU are available on-chip and off-chip via a 64 bit wide bus structure called INTPBUS[CS:+31,-24]. Bits 0 to 55, as shown in Figure 8-11, represent local time in NTP-format and bits 56 to 63 reflect a checksum on local time. Unit NTP interface (NTU) maps the internal NTP-Bus onto primary output pins to supply peripheral devices with local time information.

8.1.10 Configuration LTU

Configuration bits are required for leap second adjustment. To delete a leap second, configuration bit CLEAPSECOND must be set and CLEAPSINS must be cleared. To insert a leap second, both bits CLEAPSECOND and CLEAPSINS must be set before the LEAPDUTY gate pulse is applied. All configuration bits are listed in Table 8-5.

Name	Signal	CONFIG Reg.	Bit #	Functionality
CLEAPSECOND	CLEAPSECOND	UTCCONF1	12	1: leap second adjustment is activated 0: normal operation if CLEAPSINS = 0
CLEAPSINS	CLEAPSINS	UTCCONF1	13	1: leap second is inserted 0: leap second is deleted

Table 8-5 Configuration Bits in Unit Local Time

8.1.11 Status LTU

All status bits generated and maintained in the local time unit are summarized in Table 8-6.

Name	Signal	STATUS Reg.	Bit #	Functionality
SCAMA	AMORTGATE	UTCSTAT1	0	1: Continuous amortization phase is active 0: Pure phase is active

Table 8-6 Status Bits in Local Time Unit

Status Bit SCAMA (Status Continuous AMortization phase Active) provides information on the current state of the clock synchronization process. If this bit is 1 the current state is amortization, if it is 0 the current state is pure phase.

8.1.12 Interrupt Sources LTU

All interrupt sources due to events within the local time unit are summarized in Table 8-7.

Name	Signal	Interrupt Reg.	Bit #	Active	Functionality
ICAMS	AMORTGATE	UTCINT1	0	r-edge	rising edge: Continuous amortization phase started
ICAMT	AMORTGATE	UTCINT1	1	f-edge	falling edge: Continuous amortization terminated
INTPO	INTNTPOVR	UTCINT1	2	pulse	pulse: NTPTIME has an overrun (year 2036) or a leap second was inserted

Table 8-7 Interrupt Sources in Local Time Unit

Interrupt Source ICAMS (Interrupt Continuous AMortization Started) is activated on every rising edge of signal AMORTGATE indicating the begin of an amortization phase. It is cleared in the interrupt unit.

Interrupt Source ICAMT (Interrupt Continuous AMortization Terminated) is activated on every falling edge of signal AMORTGATE indicating the end of an amortization phase and therefore the begin of a pure phase. It is cleared in the interrupt unit.

Interrupt Source INTPO (Interrupt NTPtime Overrun) is activated by an overrun of adder NTPADDER. The source signal INTNTPOVR is high for one T-clock cycle (pulse) in the year 2036 and when a leap second is inserted. It is cleared in the interrupt unit.

8.1.13 Register Map

All LTU registers and decoded latch signals are listed in Table 8-8. The table header lists the memory element involved, the number of bits needed for that memory element, the access type (read/write/copy), the memory address, the latch element access width for write operations, the byte enable signals tied to and the multiplexer channel used for read operations. Furthermore the memory element gate signals are listed as well as a field indicating a very dynamic or a more or less static (D/S) memory element, the memory contents after reset and the data bus bit positions joining the register. This is a very agglomerating view of all LTU registers. A more detailed one is given in the next section explaining all programming details.

Memory Element	Width	R/WC	Addr	Latch	BE [3:0]	Mux Sel [2:0]	Mux #	Memory Element Gate Signal	D/S	Reset	Data Bus Pos.
STEPLOW	8	W	0	8	3	-	-	MREGW[0] ∧ MBE[3]	S	0	31:24
MSSET	24	W	0	8	2:0		-	MREGW[0] ∧ MBE[2:0]	S	0	23:0
TSSET	32	W	1	8	3:0	-	-	MREGW[1] ∧ MBE[3:0]	S	0	31:0
USSET	32	W	2	8	3:0	-	-	MREGW[2] ∧ MBE[3:0]	S	0	31:0
NTPSET	-	W	3	-	0	-	-	MREGW[3] ∧ MBE[0]	S	-	-
STEPAMORT	32	W	4	8	3:0	-	-	MREGW[4] ∧ MBE[3:0]	S	0	31:0
STEPPURE	32	W	5	8	3:0	-	-	MREGW[5] ∧ MBE[3:0]	S	0	31:0
AMORTTIMESET	32	W	6	8	3:0	-	-	MREGW[6] ∧ MBE[3:0]	S	0	31:0
STARTAMORT	-	W	7	-	0	-	-	MREGW[7] ∧ MBE[0]	S	-	-
CS:MSGET	32	R	0	32	x	000	0	TSGATE = (MREGR[1] ∧ MRWLATCH) ∨ MSNAPSHOT	D	0	31:0
TSGET	32	R	1	32	0	001	1	TSGATE = (MREGR[1] ∧ MRWLATCH) ∨ MSNAPSHOT	D	0	31:0
USGET	32	R	2	32	x	010	2	TSGATE= (MREGR[1] ∧ MRWLATCH) ∨ MSNAPSHOT	D	0	31:0
TSGET	-	R	3	no	x	011	1	no latch operation	S	0	31:0
NSGET	3	R	4	3	x	100	3	TSGATE= (MREGR[1] ∧ MRWLATCH) ∨ MSNAPSHOT	S	0	2:0
AMORTTIMEGET	32	R	6	32	x	110	4	(MREGR[6] ∧ MRWLATCH) ∨ MSNAPSHOT	D	0	31:0
AMORTTIMEGET	-	R	7	no	x	111	4	no latch operation	S	0	31:0
STEPPUREACT	32	C	-	32	-	-	-	PUREACTGATE = STARTAMORTGATE delay TCLK	D	0	31:0
NTPTIME	91	-	-	91	-	-	-	not gated	D	0	-
NTPHOLD	99	-	-	99	-	-	-	not gated	D	0	-

Table 8-8 Local Time Unit Register Map

8.1.14 Programmers Model

A very detailed view of all Local Time Unit registers is given in this section intended for hardware level programmers. The list order is increasing memory address.

8.1.14.1 NTP Time Registers

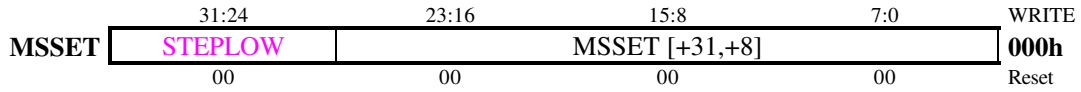


Figure 8-15 MSSET - Macro Stamp Set Register

This 24-bit register holds a preload value for the NTPTIME register. Bits 24 to 31 are reserved for a step register explained later on. Macro Stamp holds bits [+31,+8] in NTP-format. Upon reset MSSET is cleared to all 0. Writing to MSSET does not immediately affect NTPTIME.

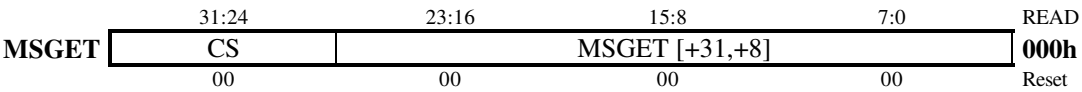


Figure 8-16 MSGET - Macro Stamp Get Register

Reading this register returns the previously latched integer part of NTPTIME in bits 0 to 23 and an 8 bit checksum for NTPTIME [+31,-24] in bits 24 to 31. Two methods of latching data into MSGET are known. A read access on address 004h (TSGET) or a snapshot capture the upper bits of NTPTIME into the MSGET register. MSGET and CS are cleared upon reset.

Figure 8-17 TSSET - Timestamp Set Register

This 32-bit register holds a preload value for the NTPTIME register. Timestamp Set holds bits [+7,-24] in NTP-format. Upon reset TSSET is cleared. Writing TSSET doesn't immediately affect NTPTIME.

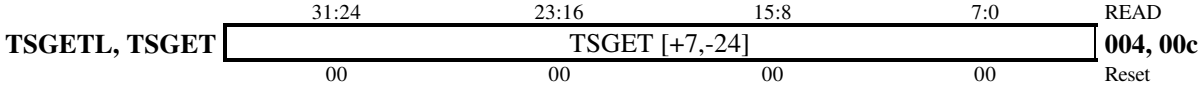


Figure 8-18 TSGET - Timestamp Get Register

Reading this register on address 004h (TSGETL) latches and returns the fractional part of NTPTIME. A read operation on address 004h additionally latches new NTPTIME data into MSGET, TSGET, USGET and NSGET. In snapshot mode TSGET is read from address 00ch (TSGET) to prevent overwriting snapshot data in all four timestamp GET registers. TSGET is cleared on reset.

Figure 8-19 USSET - Micro Stamp Set Register

This 32-bit register holds the preload data for the NTPTIME register. Micro Stamp Set holds bits [-25,-56] in NTP-format. Upon reset USSET is cleared. Writing to USSET doesn't immediately affect NTPTIME.

Figure 8-20 USGET - Micro Stamp Get Register

Reading this register returns the previously latched ultrafractional part of NTPTIME. Two methods of latching data into USGET are known. A read access on address 004h (TSGET) or a snapshot capture the lower bits of NTPTIME into the USGET register. USGET is cleared on reset.

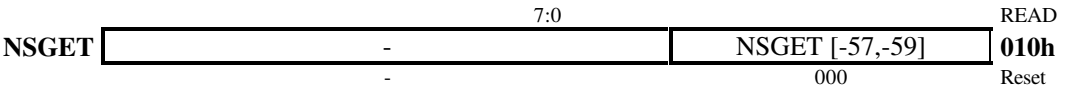


Figure 8-21 NSGET - Nano Stamp Get Register

Reading this register returns the previously latched nanofractional part of NTPTIME. Two methods of latching data into NSGET are known. A read access on address 004h (TSGET) or a snapshot capture the lower bits of NTPTIME into the NSGET register. NSGET is cleared on reset.

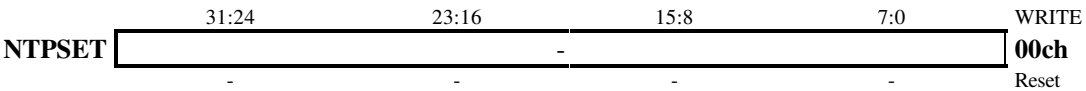


Figure 8-22 NTPSET - NTPTIME Set Register

This virtual register has no memory logic involved. Writing byte 0 at address 00ch copies the preload data from MSSET, TSSET and USSET into register NTPTIME.

8.1.14.2 STEP Registers

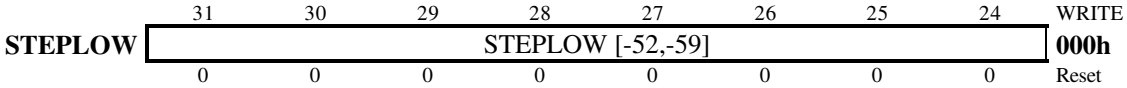


Figure 8-23 STEPLOW - Step Low Register

The STEPLOW register is part of the MSSET register. Bits 24 to 31 hold the ultrafractional NTP bits [-52,-59] to compensate non binary nominal oscillator frequencies. Writing STEPLOW immediately influences the NTPTIME increment for the next T-clock cycle. Numbers for 4 different clock frequencies are given in Table 8-1.

Figure 8-24 STEPAMORT - Step Register for Continuous Amortization Phase

This 32-bit register holds the step value used during continuous amortization. STEPAMORT holds bits [-20,-51] in NTP-format. Upon reset STEPAMORT is cleared. Writing to STEPAMORT is only save during pure phase. For references see Section 8.1.3.

Figure 8-25 STEPPURE - Step Register for Pure Phase

This 32-bit register holds the step value used in pure phase. STEPPURE holds bits [-20,-51] in NTP-format. Upon reset STEPPURE is cleared. Writing to STEPPURE is always save, but is supposed to be done before starting the amortization phase. STEPPURE is copied into STEPPUREACT at the beginning of continuous amortization. For references see Section 8.1.4.

8.1.14.3 Continuous Amortization Registers

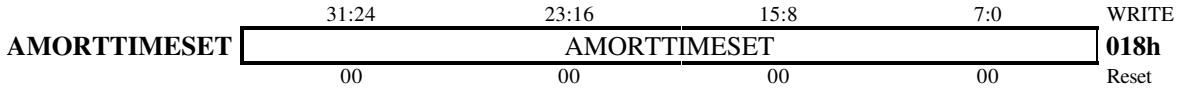


Figure 8-26 AMORTTIMESET - Preload Value for Amortization Timer

This 32-bit register initializes the amortization timer at the beginning of each amortization phase. Valid timer values are between 1 and 2^{32}-1 T-clock ticks while zero terminates the amortization phase. Similar to setting NTPTIME, writing AMORTTIMESET does not automatically start the timer. A dedicated start command is required. AMORTTIMESET is cleared on reset.

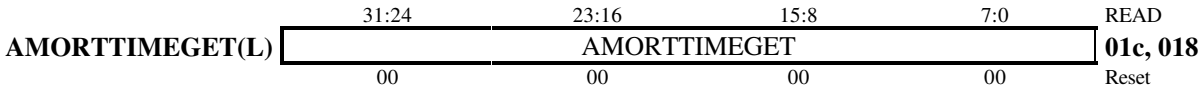


Figure 8-27 AMORTTIMEGET - Read Current Amortization Timer

This 32-bit register has two access semantics. Reading address 0018h (AMORTTIMEGETL) latches the current amortization timer value into AMORTTIMEGET and returns the number of T-clocks remaining in this amortization phase. Snapshot mode requires a read from address 001ch (AMORTTIMEGET) returning AMORTTIMEGET without latching new data. AMORTTIMEGET is cleared on reset.

	31:24	23:16	15:8	7:0	WRITE
STARTAMORT	-				**01ch**
	-	-	-	-	Reset

Figure 8-28 STARTAMORT - Start the Amortization Timer

After loading the proper timer value into AMORTTIMESET, a write operation to START-AMORT starts the amortization timer. No memory hardware is involved with this register.

8.1.15 Initialization Phase

Synchronizing the local clock to a global time standard requires write capabilities to register NTPTIME, which is a 91 bits wide memory element. First register STEPPUREACT must be programmed with time steps appropriate for the driving clock oscillator following the instruction flow diagram in Figure 8-29. In order to load register STEPPUREACT, a short amortization phase is necessary.

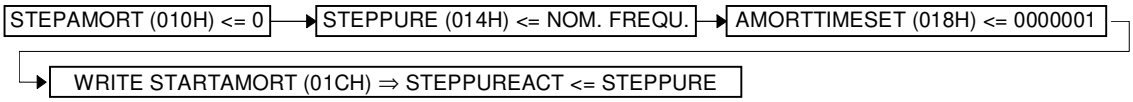


Figure 8-29 Initialization Procedure for STEPPURE Register

Following the initialization of STEPPUREACT, register NTPTIME must be adapted to the current global time and the ultrafractional part of STEP needs initialization. The sequence given in Figure 8-30 is practicable. Right after writing address NTPSET, pure phase is started and the local time register NTPTIME is updated with time steps from register STEPPUREACT and STEPLOW.

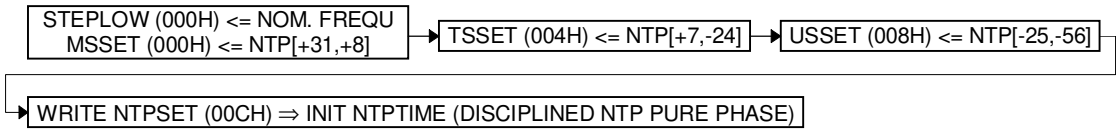


Figure 8-30 Initialization of NTPTIME Register

A timing diagram illustrating the details of NTPTIME initialization and the relationship between the contents of register NTPTIME and the internal NTP-Bus (INTPBUS) is given in Figure 8-31.

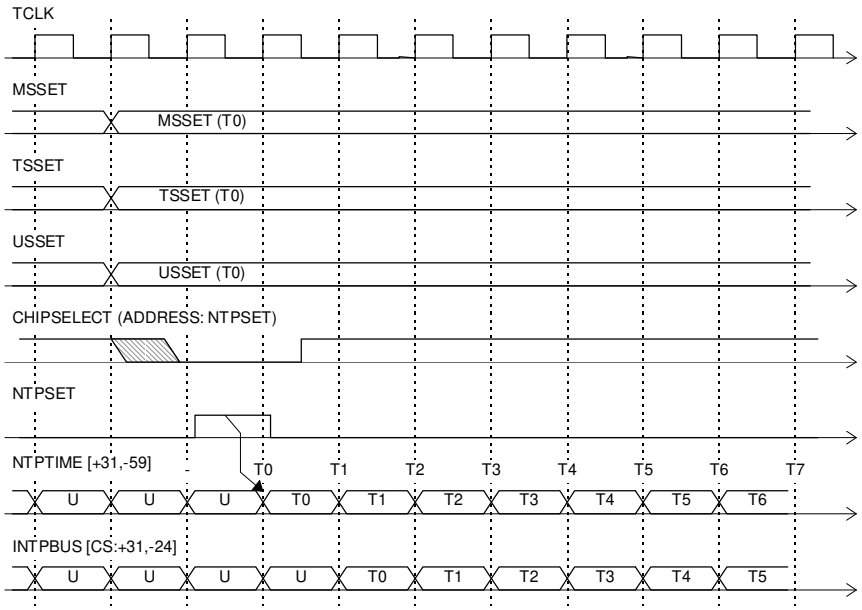


Figure 8-31 NTPTIME-Initialization Timing Diagram

8.1.16 LTU Test Strategy

The basic test concept is elucidated in Chapter 5.1. Testing through the bus interface unit (BIU) is accomplished here. In addition scan path is inserted for all memory elements in the local time unit.

Furthermore some of the unit internal control signals are mapped to a unit test port and further on to primary output pins. Table 8-9 lists the control signals and the primary output pins they are mapped to. Figure 8-32 illustrates the local time unit test port.

LTU Signal Name	Test Pin #	CSUTIME Pin #	Comments
AMORTGATE	0	0	Indicates an active amortization phase
PUREACTGATE	1	1	Gate signal for STEPPUREACT register
TSGATE	2	2	Gate signal triggered by reading TSGET
NTPSET_MUX	3	3	Select signal for NTPMUX

Table 8-9 Various Observed Control Signals in Unit Local Time

One input port bit TESTSEL and one output port bus TESTBUS[y:0] is required. Activating TESTSEL puts the local time unit in test mode. Observable unit internal signals are propagated to primary output pins through the LTU TESTBUS port and the ITESTBUS bus system.

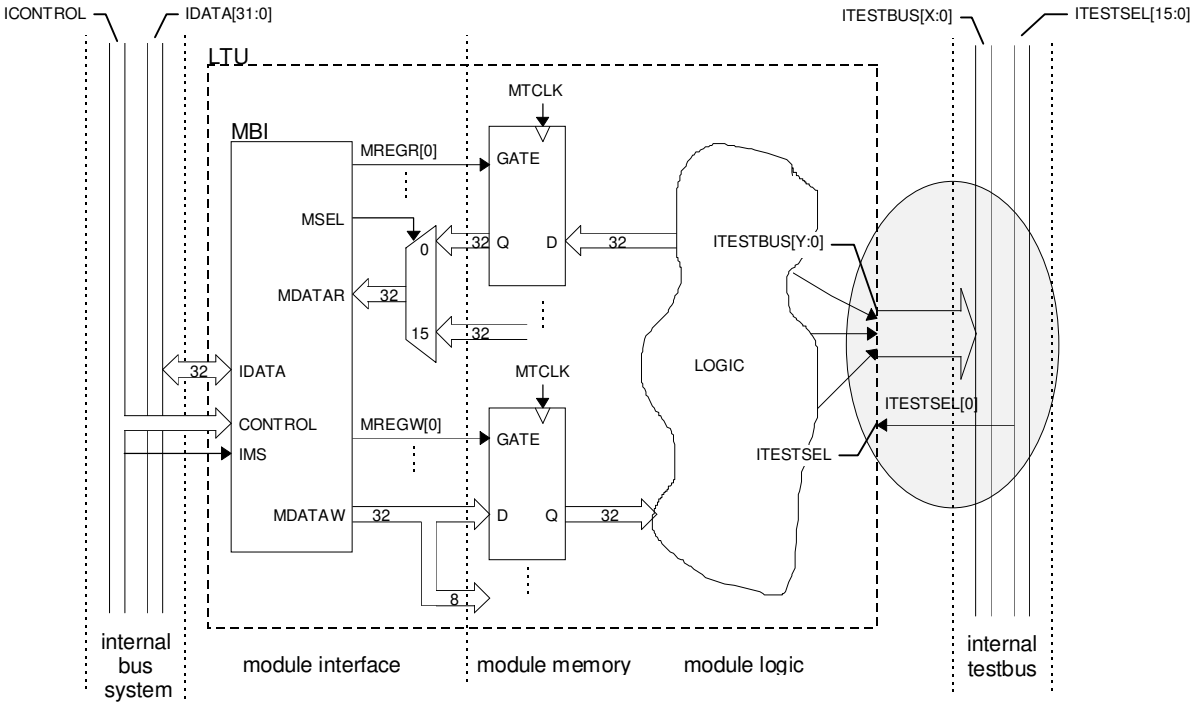


Figure 8-32 Local Time Unit Test Port Layout

8.2 ACCURACY UNIT

The accuracy unit (ACU) provides information on upper (positive P) accuracy (ALPHAP) and lower (negative N) accuracy (ALPHAN) along with guarding information on upper bounds. ALPHAP and ALPHAN follow the envelopes drawn in Figure 2-9. To deal with state intervals, the full length of ALPHAP and ALPHAN is not required. Hence, only the upper 16 bits of ALPHAP constitute the internal 16-bit IAPBUS (Internal Accuracy Positive BUS) and the same holds for the 16-bit IANBUS. The basic register layout for the upper or lower accuracy is illustrated in Figure 8-33.

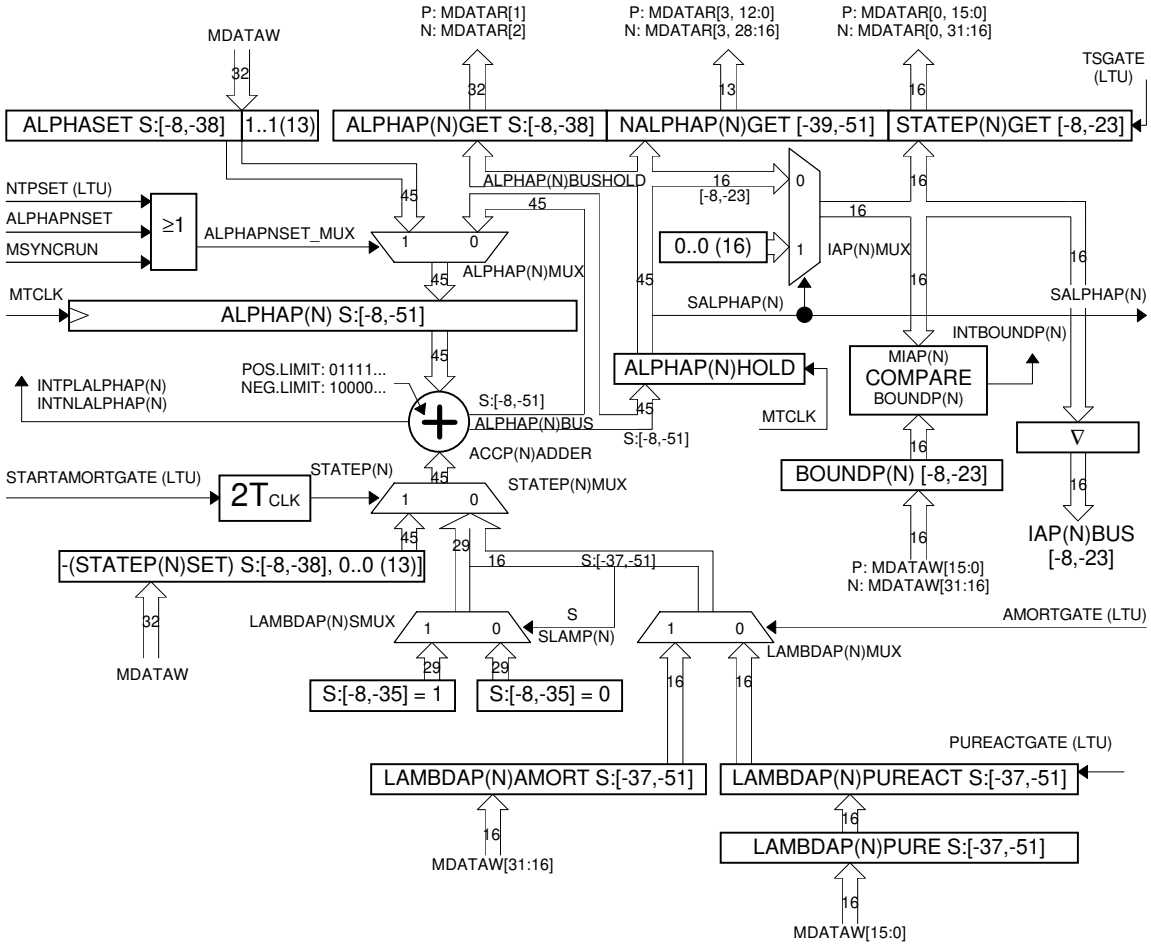


Figure 8-33 Accuracy Unit (ACU) Architecture

ALPHAP follows the upper envelope, ALPHAN the lower envelope. The resolution is given by bit [-51] in NTP format which is equivalent to 444 as.

During pure phase, register ALPHAP(N) is either increased or decreased on every T-clock by the contents of register LAMBDAP(N)PUREACT which holds a 16-bit number and affects the lower 16 bits in ALPHAP(N). Multiplexer LAMBDAP(N)MUX, controlled with signal AMORTGATE from the local time unit, switches to channel 0 and allows data from LAMBDAP(N)PUREACT to pass. Another multiplexer LAMBDAP(N)***S***MUX, ***S*** denotes the sign functionality, generates the 29 sign bits S:[-8,-35] for the LAMBDAP(N)x registers. Together with LAMBDAP(N)PUREACT these two registers provide a 45-bit ALPHAP(N) compatible number which is fed into the 45 bits wide ACCP(N)ADDER.

At the beginning of each amortization phase, both envelopes must be realigned, cf. Figure 2-9. Only relative operations are allowed to manipulate ALPHAP and ALPHAN. To adjust the envelope, register STATEP(N)SET is loaded with the 2's complement between the old and new ALPHAP(N) content. The start of the amortization period is indicated by signal STARTAMORTGATE from local time unit, which selects path STATEP(N)SET in multiplexer STATEP(N)MUX for one clock cycle. The precalculated difference stored in STATEP(N)SET is subtracted from ALPHAP(N) and the enve-

lope jumps to its new position. Envelopes are not influenced by LAMBDA (neither LAMBDAP(N)PUREACT nor LAMBDAP(N)AMORT) during that specific T-clock cycle. During the following ALPHAP(N) updates, deterioration values are read from register LAMDAPAMORT respectively LAMBDANAMORT. Deactivating signal AMORTGATE indicates the end of the amortization phase and multiplexer LAMBDAP(N)MUX switches back to pure phase.

Only the upper 16 data bits of register ALPHAP(N) are used for dealing with state intervals forming the internal 16 bit bus systems IAPBUS and IANBUS. On this bus, negative envelope numbers should be read as zero to save the sign bit. Multiplexer IAP(N)MUX passes along ALPHAP(N) bits [-8,-23] if it is a positive number, otherwise an all 0 word is generated and passed along.

Note! Accuracystamps on IAPBUS and IANBUS are latched in ALPHAP(N)HOLD), hence delayed by one T-clock compared to the contents of ALPHAP(N).

The state intervals are compared against a bound captured in register BOUNDP(N). If the value on IAP(N)BUS exceeds the bound, an interrupt INTBOUNDP(N) is generated. Bounds holding the value FFFF hex disable the guarding functionality.

Adder ACCP(N)ADDER is 45 bits wide. It is specially implemented generating interrupts on positive overflows and negative underflows and clipping to upper and lower limits. Table 8-10 shows details.

ALPHAP(N)BUS new	LIMIT Result to	INTPLALPHAP(N)	INTNLALPHAP(N)
< 0:111...1	-	0	0
≥ 0:111...1	0:111...1	1	0
> 1:000...0	-	0	0
≤ 1:000...0	1:000...0	0	1

Table 8-10 Limits and Interrupts for Adder ACCP and ACCN

If, for any reason, adder ACCPADDER or ACCNADDER overrun, the adder output ALPHAP(N)BUS is limited to the largest positive number 0:111...1 and the interrupt signal INTPLALPHAP(N) (INTerrupt Positive Limit) is set high and kept high until ALPHAP(N)BUS goes below 0:111...1. If adder ACCPADDER or ACCNADDER underflow, the adder output ALPHAP(N)BUS is limited to the smallest negative number 1:000...0 and the interrupt line INTNLALPHAP(N) (INTerrupt Negative Limit) is set high and kept high until ALPHAP(N)BUS goes above 1:000...0. Figure 8-34 indicates this behaviour on the number scale.

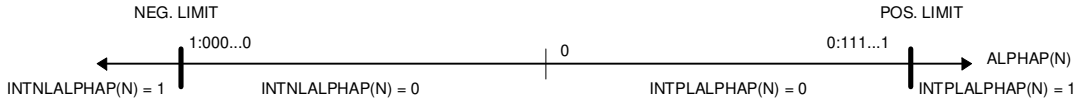


Figure 8-34 Number Scale applied to Accuracy Envelopes ALPHAP(N)

8.2.1 Validating Register Layout

To qualify the configuration shown above, we first denote the nominal oscillator frequency to be in the range specified in 8.1.1. This is 524288 Hz on the lower end and 20 MHz on the upper end. Registers LAMBDAPx and LAMBDANx contain the signed value of deterioration per T-clock where x is replaced by either PUREACT or AMORT. At each oscillator tick, the contents of registers LAMBDAPx and LAMBDANx are added to the corresponding ALPHAP(N). The maximum deterioration per tick for a LAMBDAP(N)x register length of 16 bits is given by

$$2^{-36}\,\mathrm{s/tick} = 14.6\,\mathrm{ps/tick} \xrightarrow{\mathrm{TCLK=8.388\,MHz}} 2^{-36}\,\mathrm{s/tick} \cdot 2^{23}\,\mathrm{ticks/s} = 122\,\mu\mathrm{s/s} = 1.22 * 10^{-4}$$

which is approximately 122 µs/s for the nominal oscillator frequency of 8.388 MHz. This should be sufficient even for worst quartz oscillators.

The minimal (non 0) accuracy deviation (ALPHAP(N)) within a second is computed by

$$2^{-51}\,\mathrm{s/tick} = 444\,\mathrm{as/tick} \xrightarrow{\mathrm{TCLK=8.388\,MHz}} 2^{-51}\,\mathrm{s/tick} \cdot 2^{23}\,\mathrm{ticks/s} = 3.72\,\mathrm{ns/s} \approx 4 * 10^{-9}$$

The maximum possible ALPHAP(N) contents at the end of a single resynchronization interval with R = 30 s, assuming maximum deterioration (122 µs/s), is given by

$$2^{-36}\,\mathrm{s/tick} = 14.6\,\mathrm{ps/tick} \xrightarrow{\mathrm{TCLK=8.388\,MHz,R=30\,s}} 2^{-36}\,\mathrm{s/tick} \cdot 2^{23}\,\mathrm{ticks/s} \cdot 30\,\mathrm{s} = 3.66\,\mathrm{ms}$$

Scanning Table 3-5 for a bit valence close to 3.66 ms returns bit [-8] (3.81 ms) as the upper bound on ALPHAP(N). Restricting register ALPHAP(N) to 44 bits plus sign bit S:[-8,-51] limits the length of the resynchronization interval R to

$$R = \frac{\mathrm{ALPHAP(N)_FULL_RANGE}}{\mathrm{LAMBDAP(N)_FULL_RANGE} * f_{osc}} = \frac{2^{-7}\,\mathrm{s}}{2^{-36}\,\mathrm{s/tick} \cdot 2^{23}\,\mathrm{ticks/s}} = 64\,\mathrm{s}$$

8.2.2 Initialization after Reset

All registers, except the low part of ALPHASET and the two bound registers, are cleared after reset. Both positive and negative envelopes are set at a time. Therefore register ALPHASET holds the preload data and writing to address ALPHAPNSET copies the preload data into ALPHAP and ALPHAN.

# Write	ACU Address	ACU Register Name	NTP-Time Equivalence
1	6	ALPHASET	S:[-8,-38]
2	0	ALPHAPNSET	-

Table 8-11 Initialization Sequence for ALPHAP and ALPHAN Registers

All LAMBDA registers (LAMBDAP(N)AMORT, LAMBDAP(N)PUREACT, LAMBDAP(N)PURE) are cleared after reset, what allows no change in envelopes. LAMBDAP(N)PUREACT is updated with the contents of the preload register LAMBDAP(N)PURE only at the beginning of an amortization period. Envelopes are not changed after reset until the first amortization phase is initiated and LAMBDAP(N)AMORT is set.

The two bound registers BOUNDP for the upper envelope and BOUNDN for the lower envelope are set to all ones after reset. This disables their guard functionality. Both bounds should only be set after ALPHAP(N) are stable and hold valid envelopes.

8.2.3 Amortization Phase

Similar to Section 8.1.3 the amortization phase is announced by signal AMORTGATE going high. During that period envelopes are increased by the values read from LAMBDAP(N)AMORT. Additionally both upper and lower envelopes (ALPHAP and ALPHAN) need adjustments at the beginning of each amortization phase to realign the envelopes with the calculated numbers resulting from the previous clock synchronization computations based on the ICV algorithm.

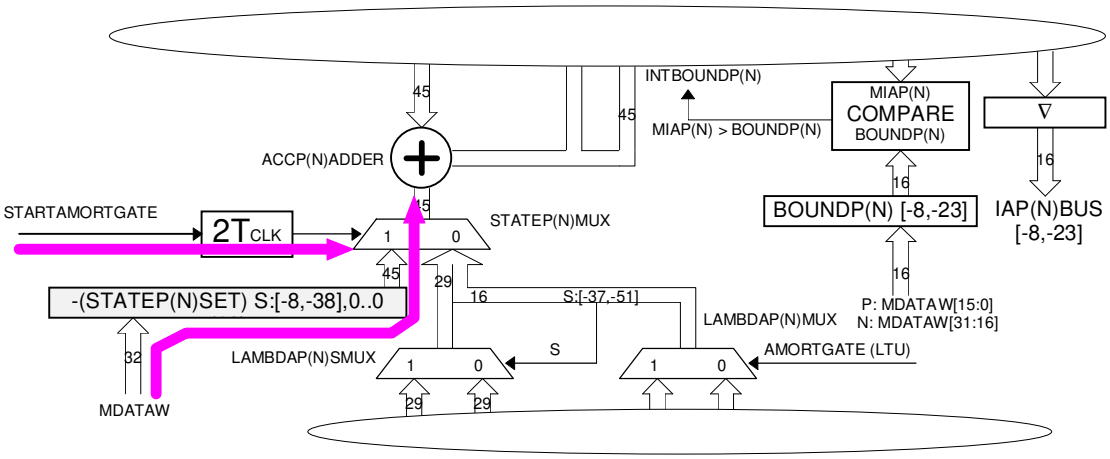


Figure 8-35 Realigning Envelopes at Start of Amortization Phase

The very first clock cycle during the amortization phase is spent on realigning the envelopes by subtracting the contents of STATEP(N)SET from ALPHAP(N). The data loaded into STATEP(N)SET must be the two's complement of the absolute difference between old an new envelopes as the ACCP(N)ADDER can only subtract by adding the two's complement. Therefore STATEP(N)MUX switches to channel 1 for a single T-clock as shown in Figure 8-35.

All registers involved in adjusting the envelopes are shaded. The remaining time is used to track the envelopes according to the deterioration values specified in LAMBDAP(N)AMORT

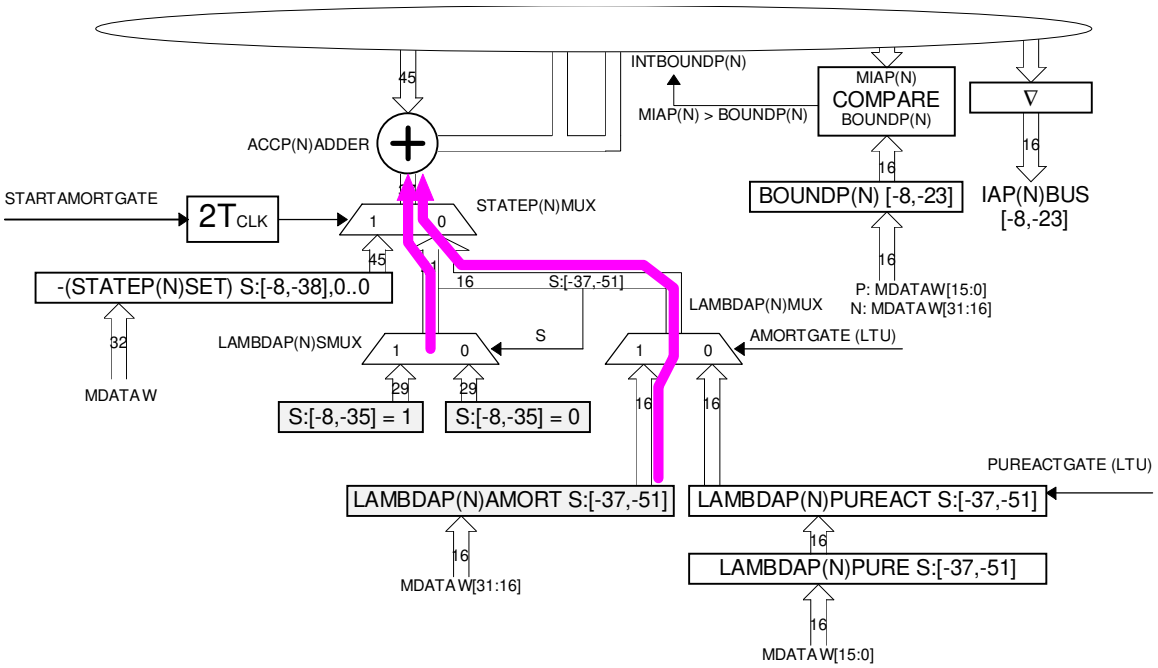


Figure 8-36 Accuracy Tracking during Continuous Amortization

Figure 8-36 illustrates the amortization data flow. When the first amortization clock cycle has passed by, multiplexer STATEP(N)MUX switches back to channel 0 but multiplexer LAMBDAP(N)MUX holds channel 1 and allows data from register LAMDAP(N)AMORT to pass along. Sign extension achieves a 45 bit number out of the 16 bit LAMBDA values with multiplexer LAMBDAP(N)SMUX. For negative LAMBDA the upper 29 bits [-8,-35] are set to one and for positive numbers these bits are cleared to give a 45 bit ALPHAP(N) compatible number.

Note! Register LAMBDAP(N)AMORT must not be written during amortization!

A detailed view on amortization timing in unit accuracy is portrayed in Figure 8-37. Before amortization can start, several registers need to be updated. The new deterioration value during amortization is loaded into LAMBDAP(N)AMORT, the preload data for LAMBDAP(N)PUREACT is written into LAMBDAP(N)PURE and the absolute difference in twos complement for adjusting both envelopes at the beginning of continuous amortization are loaded into STATEP(N)SET. Starting the amortization period by writing to LTU address 7 activates signal STARTAMORT for one T-clock. This pulse is delayed by 2 clock cycles (block 2 TCLK) for the multiplexer STATEP(N)MUX. STATEP(N)SET is subtracted from register ALPHAP(N), see cycle A4-STAT in Figure 8-37. To remind the programmer, this cycle is lost for amortization, envelopes are only aligned to their new distance. All the following ticks expand or compand envelopes with deterioration numbers read from LAMBDAP(N)AMORT.

Phase A6+LA terminates the continuous amortization and the accuracy unit falls back into pure phase. As a consequence registers LAMBDAP(N)PUREACT provide the new deterioration numbers for the envelopes.

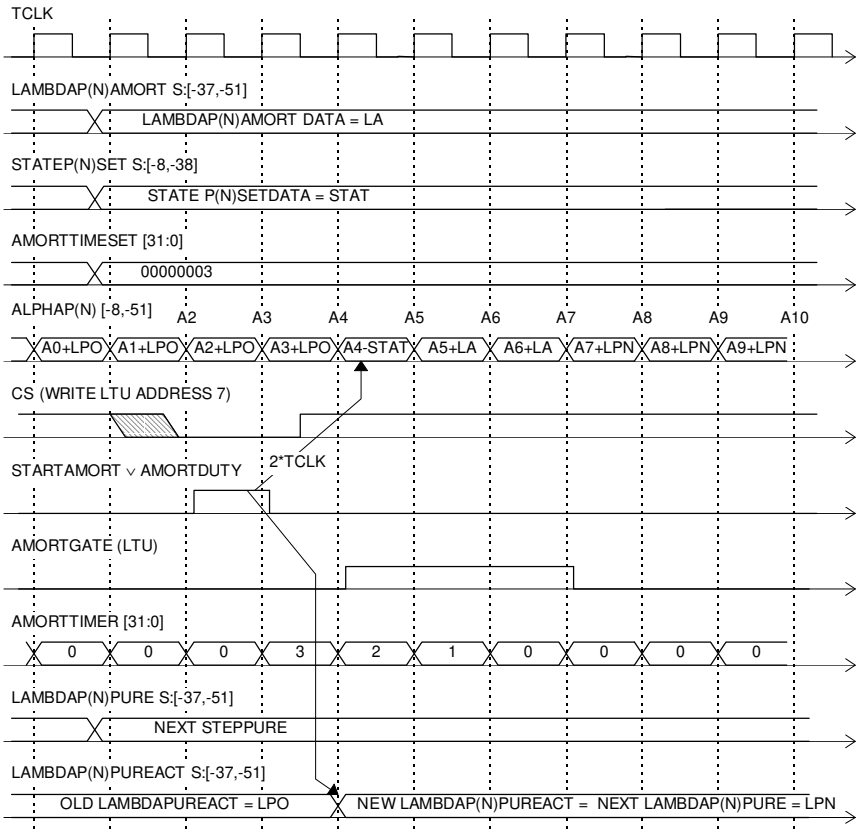


Figure 8-37 Envelope Evolution during Amortization and Pure Phase

8.2.4 Pure Phase

Pure phase follows an amortization phase. When AMORTGATE (propagated from LTU) is low, multiplexer LAMBDAP(N)MUX switches to LAMBDAP(N)PUREACT and envelope register ALPHAP(N) is incrementally modified by deterioration data from LAMBDAP(N)PUREACT as shown in Figure 8-38. Again all touched registers are shaded.

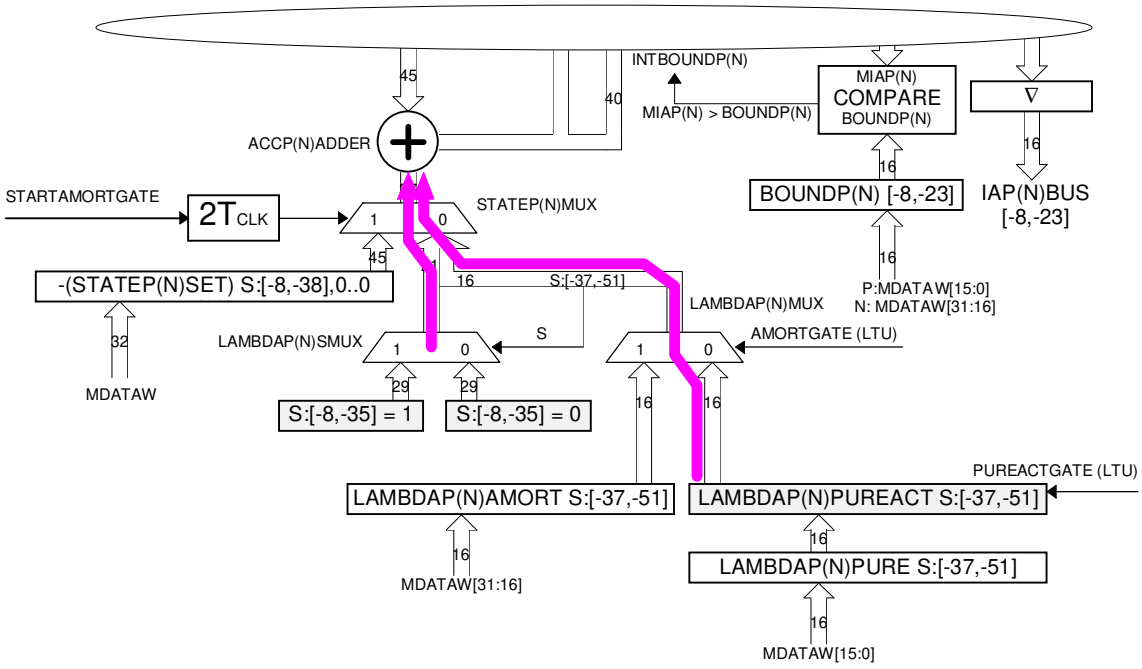


Figure 8-38 Envelope Maintenance during Pure Phase

No watchdog timer is involved during pure phase, hence pure phase is terminated by introducing the next amortization phase. Timing information is also provided in Figure 8-37. Pure phase between A0+LPO till A3+LPO (LPO...Lambda Pure Old) turns into continuous amortization for 3 T-clocks. At time A7, the new pure phase (A7+LPN) supersedes continuous amortization.

8.2.5 Debug and Test

Debugging this unit in a small network of several UTCSU nodes is a two step process. First all UTCSUs must be set into a common state and started synchronously. Second all dynamically changing registers should provide an atomic read operation. The first feature is called SYNCRUN and is specified in the next paragraph the 2nd feature is called snapshot mode and presented in Section 8.2.5.2.

8.2.5.1 Synchronous Start

An external input signal called SYNCRUN offers a gate pulse similar to ALPHAPNSET. In unit ACU it is called MSYNCRUN and it is connected to ALPHAP(N)MUX the same way as ALPHAPNSET. A rising edge on signal SYNCRUN copies the ALPHAP(N) preload register ALPHASET into ALPHAP(N) and initializes the unit.

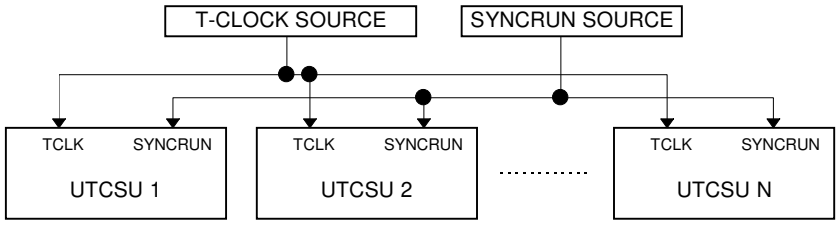


Figure 8-39 Synchronous Start of the ACU

8.2.5.2 Snapshot Mode

ACU snapshots are implemented as software snapshots (SSNAP). In snapshot mode, all dynamically changing registers are captured into holding registers. This is done on the rising edge of signal MSNAPSHOT. As referenced in Table 8-14 registers ALPHAP and ALPHAN change their contents every T-clock. All other registers are more or less static from the programmers view.

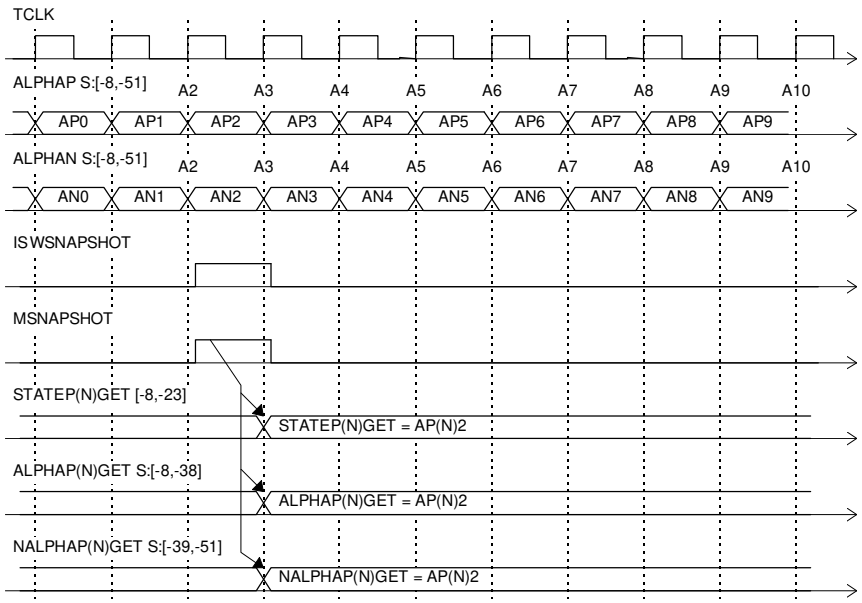


Figure 8-40 ACU Snapshot Mode Timing Diagram

A very similar technique to the one used for the LTU snapshot mode is applied here. Accessing the same memory element on two different memory addresses either latches data before reading the

memory element or just returns the current memory element contents. For full accuracy resolution registers ALPHAP(N)GET and NALPHAP(N)GET capture 45 bits accuracy information, for low resolution STATEP(N)GET captures the upper 16 bits only. Results can be verified by reading register STATEP(N)GET at address 024h, register ALPHAP(N)GET at address 030h resp. 034h and register NALPHAP(N)GET at address 038h. Figure 8-40 shows timing details.

8.2.6 Atomic Sampling Across Units

Generating accurate timestamps requires atomic sampling of several time and accuracy registers in different units. The ACU enforces atomic sampling for the following registers if register TSGET (LTU) is read at address 004h.

- CS:MSGET (LTU)
- TSGET (LTU, read access at address 004h)
- USGET (LTU), NSGET (LTU)
- ALPHAPGET, NALPHAPGET
- ALPHANGET, NALPHANGET
- STATEPGET
- STATENGET

8.2.7 Configuration ACU

All configuration elements are addressed and generated within the ACU block and already outlined in the previous sections. No configuration signals from other units are required.

8.2.8 Status ACU

All status bits generated and maintained in the accuracy unit are summarized in Table 8-12.

Name	Signal	STATUS Reg.	Bit #	Functionality
SAPN	SALPHAP	UTCSTAT1	2	1: ALPHAP holds a negative number 0: ALPHAP holds a positive number
SANN	SALPHAN	UTCSTAT1	3	1: ALPHAN holds a negative number 0: ALPHAN holds a positive number
SAPPL	INTPLALPHAP	UTCSTAT1	4	1: ALPHAP holds the positive limit 0: ALPHAP < positive limit
SAPNL	INTNLALPHAP	UTCSTAT1	5	1: ALPHAP holds the negative limit 0: ALPHAP > negative limit
SANPL	INTPLALPHAN	UTCSTAT1	6	1: ALPHAN holds the positive limit 0: ALPHAN < positive limit
SANNL	INTNLALPHAN	UTCSTAT1	7	1: ALPHAN holds the negative limit 0: ALPHAN > negative limit
SBNDP	INTBOUNDP	UTCSTAT1	8	1: ALPHAP > BOUNDP 0: ALPHAP ≤ BOUNDP
SBNDN	INTBOUNDN	UTCSTAT1	9	1: ALPHAN > BOUNDN 0: ALPHAN ≤ BOUNDN

Table 8-12 Status Bits in Accuracy Unit

Status bit SAPN (Status AlphaP Negative) provides information on the current sign bit in register ALPHAP. A high in this bit field represents a negative number in register ALPHAP, a low represents a positive number.

Status bit SANN (Status AlphaN Negative) provides information on the current sign bit in register ALPHAN. A high in this bit field represents a negative number in register ALPHAN, a low represents a positive number.

Status bit SAPPL (Status AlphaP Positive Limit) provides information on the current ALPHAP contents. If SAPPL is high, register ALPHAP holds the largest positive number possible, a positive overflow is very likely.

Status bit SAPNL (Status AlphaP Negative Limit) provides information on the current ALPHAP contents. If SAPNL is high, register ALPHAP holds the smallest negative number possible, a negative underflow is very likely.

Status bit SANPL (Status AlphaN Positive Limit) provides information on the current ALPHAN contents. If SANPL is high, register ALPHAN holds the largest positive number possible, a positive overflow is very likely.

Status bit SANNL (Status AlphaN Negative Limit) provides information on the current ALPHAN contents. If SANNL is high, register ALPHAN holds the smallest negative number possible, a negative underflow is very likely.

Status bit SBNDP (Status BouNDP) provides information on the current compare operation between BOUNDP and IAPBUS. If IAPBUS exceeds BOUNDP this bit is set otherwise it is cleared.

Status bit SBNDN (Status BouNDN) provides information on the current compare operation between BOUNDN and IANBUS. If IANBUS exceeds BOUNDN this bit is set otherwise it is cleared.

8.2.9 Interrupt Sources ACU

All interrupt sources due to events within the accuracy unit are summarized in Table 8-13.

Name	Signal	Interrupt Reg.	Bit #	Active	Functionality
IBNDP	INTBOUNDP	UTCINT1	4	pulse	IAPBUS > BOUNDP
IBNDN	INTBOUNDN	UTCINT1	5	pulse	IANBUS > BOUNDN
IAPPL	INTPLALPHAP	UTCINT1	6	pulse	ALPHAP positive overflow
IAPNL	INTNLALPHAP	UTCINT1	7	pulse	ALPHAP negative underflow
IANPL	INTPLALPHAN	UTCINT1	8	pulse	ALPHAN positive overflow
IANNL	INTNLALPHAN	UTCINT1	9	pulse	ALPHAN negative underflow

Table 8-13 Interrupt Sources in Accuracy Unit

Interrupt Source IBNDP (Interrupt BouNDP) is activated if IAPBUS exceeds BOUNDP indicating a violation of the limit (BOUNDP) for the upper envelope. To clear the interrupt the pending interrupt must be cleared by writing register UTCINTCLEAR1[4] = 1 in the ITU.

Interrupt Source IBNDN (Interrupt BouNDN) is activated if IANBUS exceeds BOUNDN indicating a violation of the limit (BOUNDN) set for the lower envelope. To clear the interrupt the pending interrupt must be cleared by writing register UTCINTCLEAR1[5] = 1 in the ITU.

Interrupt Source IAPPL (Interrupt AlphaP Positive Limit) is activated if signal ALPHAPBUS reaches or exceeds the upper positive accuracy limit. To clear the interrupt the pending interrupt must be cleared by writing register UTCINTCLEAR1[6] = 1 in the ITU.

Interrupt Source IAPNL (Interrupt AlphaP Negative Limit) is activated if signal ALPHAPBUS reaches or underflows the lower negative accuracy limit. To clear the interrupt the pending interrupt must be cleared by writing register UTCINTCLEAR1[7] = 1 in the ITU.

Interrupt Source IANPL (Interrupt AlphaN Positive Limit) is activated if signal ALPHANBUS reaches or exceeds the upper positive accuracy limit. To clear the interrupt the pending interrupt must be cleared by writing register UTCINTCLEAR1[6] = 1 in the ITU.

Interrupt Source IAPNL (Interrupt AlphaN Negative Limit) is activated if signal ALPHANBUS reaches or underflows the lower negative accuracy limit. To clear the interrupt the pending interrupt must be cleared by writing register UTCINTCLEAR1[7] = 1 in the ITU.

8.2.10 Register Map

All ACU registers and decoded latch signals are listed in Table 8-14. The table header lists the memory element involved, the number of bits needed for that memory element, the access type (read/write/copy), the memory address, the latch element access width for write operations, the byte enable signals tied to and the multiplexer channel used for read operations. Furthermore the memory element gate signals are listed as well as a field indicating a very dynamic or a more or less static (D/S) memory element, the memory contents after reset and the data bus bit positions joining the register.

This is a very agglomerating view of all ACU registers. A more detailed one is given in the next section explaining all programming details.

Memory Element	Width	R/W/C	Addr	Latch	BE 3:0	Mux Sel 2:0	Mux #	Memory Element Gate Signal	D/S	Reset	Data Bus Pos.
LAMBDAPAMORT	16	W	0	8	3:2	-	-	MREGW[0] ∧ MBE[3:2]	S	0	31:16
LAMBDAPPURE	16	W	0	8	1:0	-	-	MREGW[0] ∧ MBE[1:0]	S	0	15:0
LAMBDANAMORT	16	W	1	8	3:2	-	-	MREGW[1] ∧ MBE[3:2]	S	0	31:16
LAMBDANPURE	16	W	1	8	1:0	-	-	MREGW[1] ∧ MBE[1:0]	S	0	15:0
STATEPSET	32	W	2	8	3:0	-	-	MREGW[2] ∧ MBE[3:0]	S	0	31:0
STATENSET	32	W	3	8	3:0	-	-	MREGW[3] ∧ MBE[3:0]	S	0	31:0
BOUNDP	16	W	4	8	1:0	-	-	MREGW[4] ∧ MBE[1:0]	S	1	15:0
BOUNDN	16	W	4	8	3:2	-	-	MREGW[4] ∧ MBE[3:2]	S	1	31:16
ALPHASET	32	W	5	8	3:0	-	-	MREGW[5] ∧ MBE[3:0]	S	1	31:0
ALPHAPNSET	-	W	6	-	0	-	-	MREGW[6] ∧ MBE[0]	S	-	-
STATEPGET	16	R	0	16	x	000	0	(MREGR[0] ∧ MRWLATCH) ∨ TSGATE (LTU)	D	0	15:0
STATENGET	16	R	0	16	x	000	0	(MREGR[0] ∧ MRWLATCH) ∨ TSGATE (LTU)	D	0	31:16
STATEPGET	16	R	1	16	x	001	0	no latch	S	0	15:0
STATENGET	16	R	1	16	x	001	0	no latch	S	0	31:16
ALPHAPGET	32	R	2	32	x	010	1	(MREGR[2] ∧ MRWLATCH) ∨ TSGATE (LTU)	D	0	31:0
ALPHANGET	32	R	3	32	x	011	2	(MREGR[3] ∧ MRWLATCH) ∨ TSGATE (LTU)	D	0	31:0
ALPHAPGET	-	R	4	32	x	100	1	no latch	S	0	31:0
ALPHANGET	-	R	5	32	x	101	2	no latch	S	0	31:0
NALPHAPGET	13	R	6	13	x	110	3	(MREGR[2] ∧ MRWLATCH) ∨ TSGATE (LTU)	S	0	12:0
NALPHANGET	13	R	6	13	x	110	3	(MREGR[3] ∧ MRWLATCH) ∨ TSGATE (LTU)	S	0	28:16
LAMBDAPUREACT	16	C	-	16	-	-	-	PUREACTGATE (LTU)	D	0	-
LAMBDANUREACT	16	C	-	16	-	-	-	PUREACTGATE (LTU)	D	0	-
ALPHAP	45	-	-	45	-	-	-	not gated	D	0	-
ALPHAN	45	-	-	45	-	-	-	not gated	D	0	-
ALPHAPHOLD	45	-	-	45	-	-	-	not gated	D	0	-
ALPHANHOLD	45	-	-	45	-	-	-	not gated	D	0	-

Table 8-14 Accuracy Unit Register Map

8.2.11 Programmers Model

A very detailed view of all accuracy unit registers is given in this section intended for low level programmers. The list order is increasing memory address.

8.2.11.1 Deterioration Registers

	31:24	23:16	15:8	7:0	WRITE
LAMBDAPAMORT	LAMBDAPAMORT S:[-37,-51]		LAMBDAPPURE S:[-37,-51]		**020h**
	00	00	00	00	Reset

Figure 8-41 LAMBDAPAMORT - Deterioration during Amortization for Upper Envelope

This 16-bit register holds the deterioration value used during amortization phase for the upper envelope. Bits 0 to 15 are reserved for the deterioration during pure phase explained later on. LAMBDAPAMORT holds bits S:[-37,-51] in NTP-format. It is a signed number allowing companding envelope phases. Upon reset LAMBDAPAMORT is cleared to all 0.

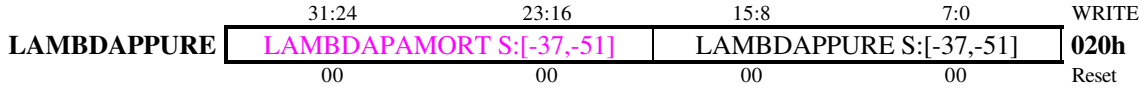


Figure 8-42 LAMBDAPPURE - Deterioration during Pure Phase for Upper Envelope

This 16-bit register holds the deterioration value used during pure phase for the upper envelope. Bits 16 to 31 are reserved for the deterioration during amortization phase explained earlier. LAMBDAPPURE holds bits S:[-37,-51] in NTP-format. It is a signed number allowing companding envelope phases. Upon reset LAMBDAPPURE is cleared to all 0.

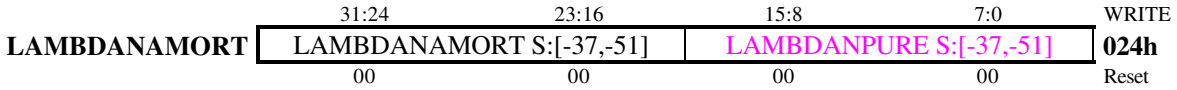


Figure 8-43 LAMBDANAMORT - Deterioration during Amortization for Lower Envelope

This 16-bit register holds the deterioration value used during amortization phase for the lower envelope. Bits 0 to 15 are reserved for the deterioration during pure phase explained later on. LAMBDANAMORT holds bits S:[-37,-51] in NTP-format. It is a signed number allowing companding envelope phases. Upon reset LAMBDANAMORT is cleared to all 0.

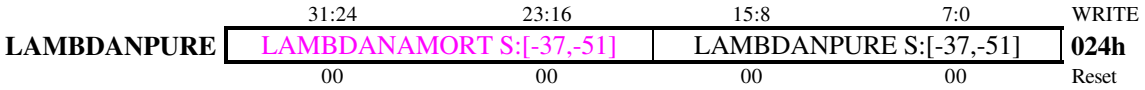


Figure 8-44 LAMBDANPURE - Deterioration during Pure Phase for Lower Envelope

This 16-bit register holds the deterioration value used during pure phase for the lower envelope. Bits 0 to 15 are reserved for the deterioration during amortization phase explained earlier. LAMBDANPURE holds bits S:[-37,-51] in NTP-format. It is a signed number allowing companding envelope phases. Upon reset LAMBDANPURE is cleared to all 0.

8.2.11.2 State Registers and Bounds

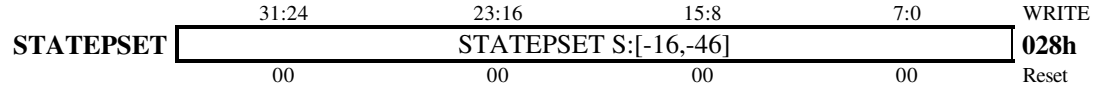


Figure 8-45 STATEPSET - Upper Envelope Adjustment at Start of Amortization Phase

This 32-bit register holds the absolute difference between old and new ALPHAP in two's complement format which is added to ALPHAP at the beginning of each amortization phase. STATEPSET holds bits S:[-8,-38] in NTP-format. It is a signed number allowing adjustments in both directions. Upon reset STATEPSET is cleared to all 0.

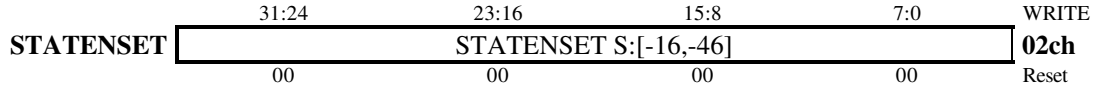


Figure 8-46 STATENSET - Lower Envelope Adjustment at Start of Amortization Phase

This 32-bit register holds the absolute difference between old and new ALPHAN in two's complement format which is added to ALPHAN once at the beginning of each amortization phase. STATENSET holds bits S:[-8,-38] in NTP-format. It is a signed number allowing adjustments in both directions. Upon reset STATENSET is cleared to all 0.

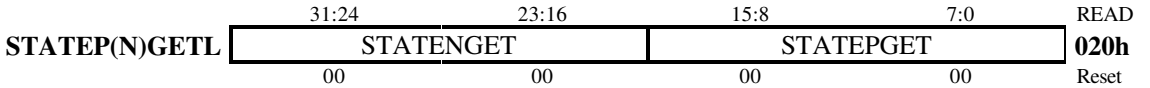


Figure 8-47 STATEP(N)GETL - Upper and Lower Envelope Values with Latch Operation

These two 16-bit registers hold the current accuracy latched from ALPHAP(N) when read. They represent the upper and the lower envelopes. These two registers are also accessible via address 001h but then no latch operation is executed. STATEP(N)GETL hold bits [-8,-23] in NTP-format. Negative numbers in ALPHAP(N) are read as zero. Upon reset STATEP(N)GET is cleared to all 0.

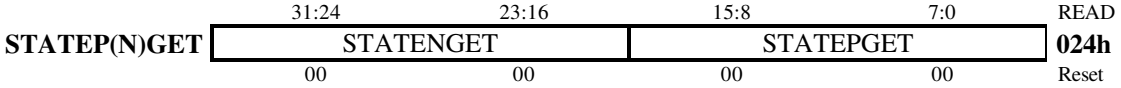


Figure 8-48 STATEP(N)GET - Upper and Lower Envelope Values; No Latch Operation

These two 16-bit registers hold the accuracy latched from ALPHAP(N) during either accessing register TSGET (LTU) or snapshot mode or a previous read from address 020h. They represent the upper and the lower envelope. No latch operation is involved. These two registers are also accessible via address 000h but then a latch operation is executed first overwriting previously latched data. STATEP(N)GET hold bits [-8,-23] in NTP-format. Negative numbers in ALPHAP(N) are read as zero. Upon reset STATEP(N)GET is cleared to all 0.

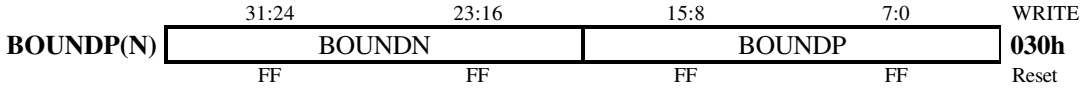


Figure 8-49 BOUNDP(N) - Bounds for Upper and Lower Envelopes

These two 16-bit registers hold the bounds on upper and lower envelopes. The 16 most significant bits in ALPHAP(N) are compared against these bounds and interrupts are generated if bounds are crossed. BOUNDP(N) hold bits [-8,-23] in NTP-format. Programming FFhex into BOUNDP(N) disables the bounds. Upon reset BOUNDP(N) is set to all 1.

8.2.11.3 Accuracy Registers

Figure 8-50 ALPHASET - Preload Data for Envelope Registers ALPHAP(N)

This 32-bit register holds the preload data for both envelope registers ALPHAP and ALPHAN. ALPHASET holds bits S:[-8,-38] in NTP-format. The remaining 13 bits [-39,-51] enforce compatibility with register ALPHAP(N) and are set to 1. Writing to ALPHASET does not immediately affect ALPHAP(N). A write operation to address ALPHAPNSET (038h) copies data from ALPHASET into ALPHAP and ALPHAN. Upon reset ALPHASET is cleared to all 0.

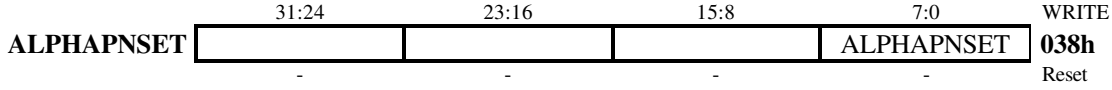


Figure 8-51 ALPHAPNSET - Load ALPHAP(N) with Data from ALPHASET

This virtual register has no memory logic involved. Writing byte 0 at address 038h copies data from register ALPHASET into registers ALPHAP and ALPHAN. Both accuracy registers are fed with the same data. Hence accuracy intervals are symmetric after initialization.

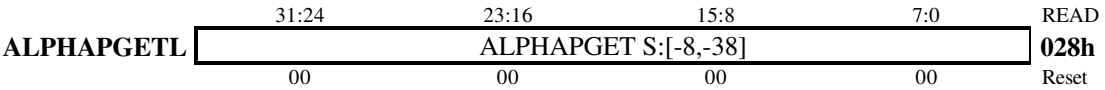


Figure 8-52 ALPHAPGETL - Read Envelope Register ALPHAP with Latch

This 32-bit register latches and returns the current value from register ALPHAP. ALPHAPGETL is a different access semantic for memory element ALPHAPGET. ALPHAPGET holds bits S:[-8,-38] in NTP-format. Upon reset ALPHAPGET is cleared to all 0.

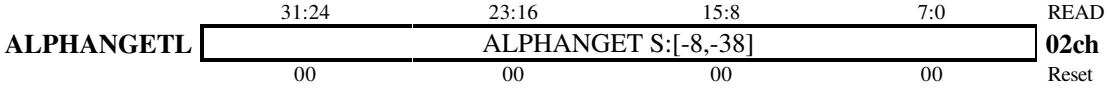


Figure 8-53 ALPHANGETL - Read Envelope Register ALPHAN with Latch

This 32-bit register latches and returns the current value from register ALPHAN. ALPHANGETL is a different access semantic for memory element ALPHANGET. ALPHANGET holds bits S:[-8,-38] in NTP-format. Upon reset ALPHANGET is cleared to all 0.

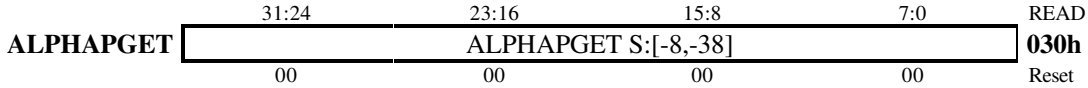


Figure 8-54 ALPHAPGET - Read Envelope Register ALPHAP

This 32-bit register holds data latched from register ALPHAP in a previous event. This event can either be a read at address 028h or a read from LTU register TSGET at address 004h or a software snapshot. No latch operation is implied with this access. ALPHAPGET holds bits S:[-8,-38] in NTP-format. Upon reset ALPHAPGET is cleared to all 0.

Figure 8-55 ALPHANGET - Read Envelope Register ALPHAN

This 32-bit register holds data latched from register ALPHAN in a previous event. This event can either be a read at address 02ch or a read from LTU register TSGET at address 004h or a software snapshot. No latch operation is implied with this access. ALPHANGET holds bits S:[-8,-38] in NTP-format. Upon reset ALPHANGET is cleared to all 0.

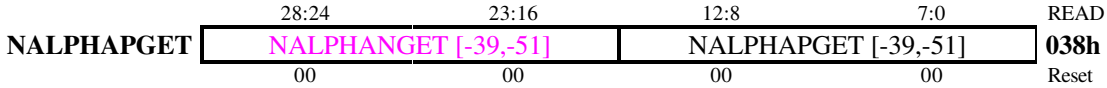


Figure 8-56 NALPHAPGET - Read Envelope Register NALPHAP

This 13-bit register holds the nanofractional part latched from register ALPHAP in a previous event. This event can either be a read at address 028h or a read from LTU register TSGET at address 004h or a software snapshot. No latch operation is implied with this access. NALPHAPGET holds bits [-39,-51] in NTP-format. Upon reset NALPHAPGET is cleared to all 0.

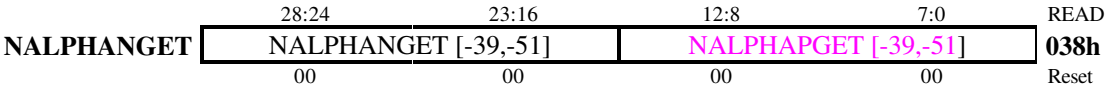


Figure 8-57 NALPHANGET - Read Envelope Register NALPHAN

This 13-bit register holds the nanofractional part latched from register ALPHAN in a previous event. This event can either be a read at address 02ch or a read from LTU register TSGET at address 004h or a software snapshot. No latch operation is implied with this access. NALPHANGET holds bits [-39,-51] in NTP-format. Upon reset NALPHANGET is cleared to all 0.

8.2.12 Initialization Phase

Both accuracies are set to zero after reset, except bits [-39,-51] are set to one. Register LAMBDAP(N)PUREACT must be loaded with deterioration coefficients adequate for this type of oscil-

lator following the instruction flow diagram in Figure 8-58. In order to load register LAMBDAP(N)PUREACT, an amortization phase of at least one T-clock is mandatory.

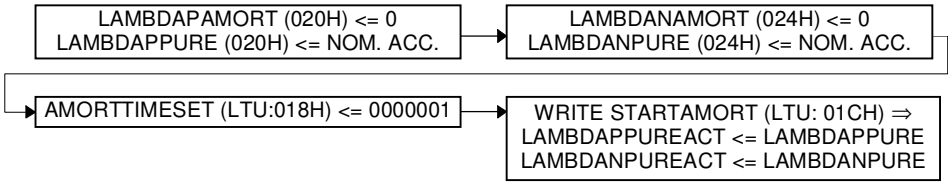


Figure 8-58 Initialization Procedure for LAMBDAP(N)PURE Register

Following the initialization of LAMBDAP(N)PUREACT, registers ALPHAP and ALPHAN must be set to accuracies specific for the crystal oscillator in use. The sequence given in Figure 8-59 should be used. Right after writing address ALPHAPNSET, registers ALPHAP and ALPHAN are both loaded with nominal oscillator accuracies specified in ALPHASET. Although the unit is active, since it was powered up, from now on envelopes are disciplined.

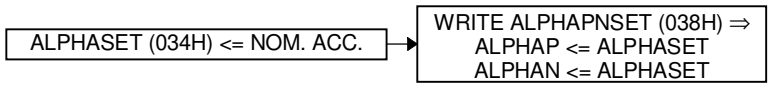


Figure 8-59 Initialization of ALPHAP(N) Registers

A timing diagram illustrating the details of ALPHAP and ALPHAN initialization is given in Figure 8-60. It also shows the relationship between the contents of registers ALPHAP(N) and the internal IAP(N)BUS.

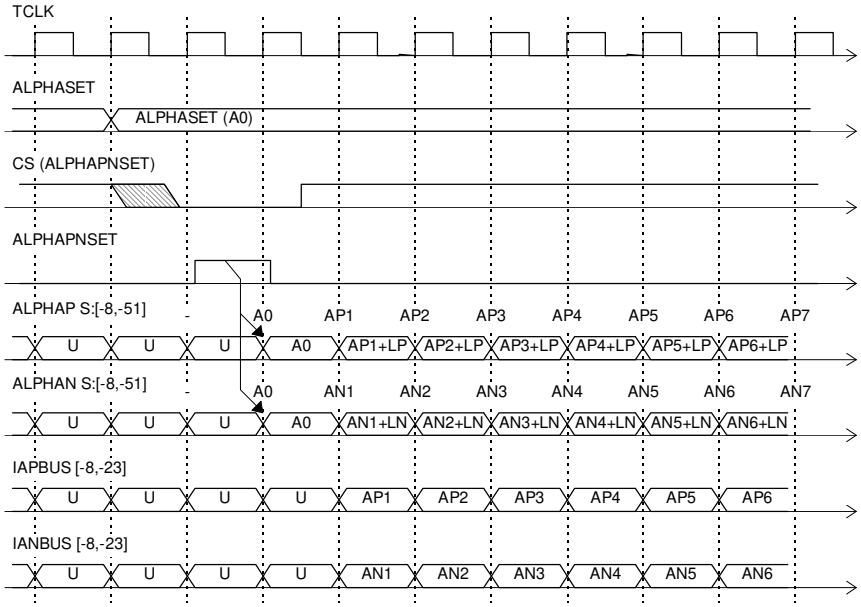


Figure 8-60 ALPHAP(N) Initialization Timing Diagram

8.2.13 ACU Test Strategy

The basic test concept was elucidated in Chapter 5.1. Testing through the bus interface unit (BIU) is accomplished here. In addition scan path is inserted for all memory elements in the accuracy unit.

Furthermore some of the unit internal control signals are mapped to primary output pins. Table 8-15 lists the control signals and the primary output pins they are mapped to.

ACU Signal Name	TestPin #	CSUTIME Pin #	Comments
SALPHAP	0	4	Sign bit at IAPBUS
SALPHAN	1	5	Sign bit at IANBUS
SLAMP	2	6	Sign bit at LAMBDAPMUX output
SLAMN	3	7	Sign bit at LAMBDANMUX output
ALPHAPNSET_MUX	4	8	Load ALPHAP(N) with preload data from ALPHASET

Table 8-15 Various Observed Control Signals in Unit Accuracy

One input port TESTSEL and one output port TESTBUS[y:0] is required as illustrated in Figure 8-61. Observable unit internal signals are propagated to primary output pins through the ACU TESTBUS port and the ITESTBUS bus system.

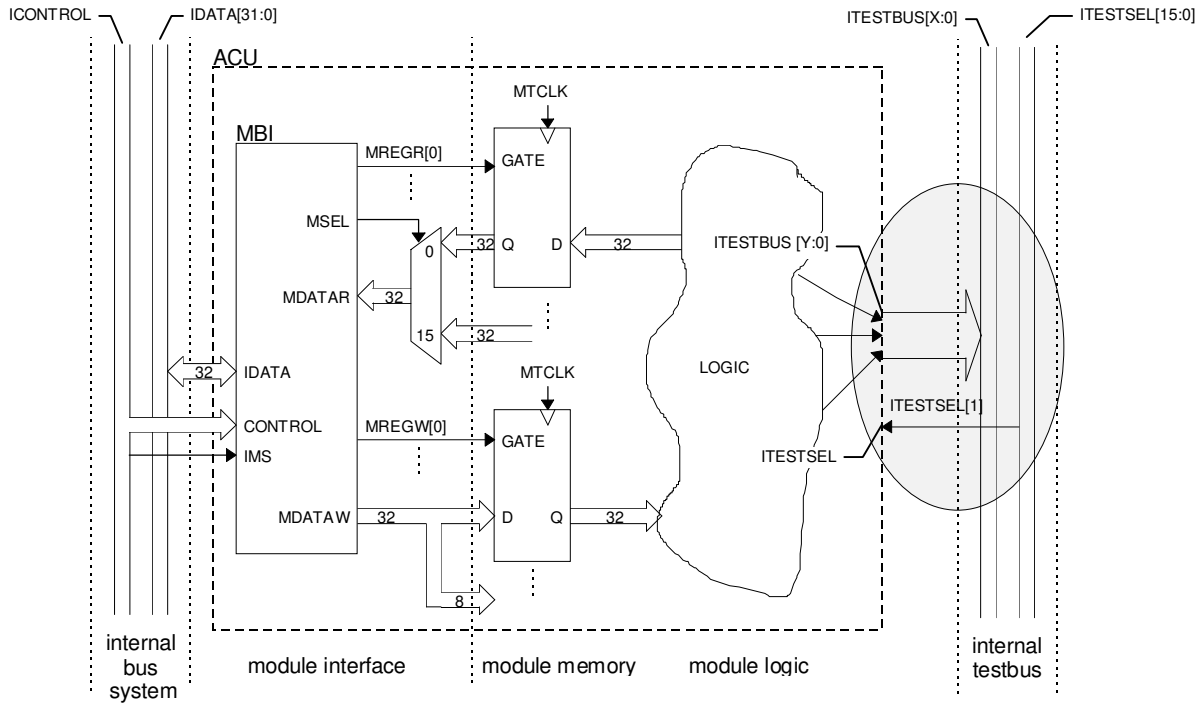


Figure 8-61 Accuracy Unit Test Port Layout

8.3 SYNCHRONIZATION SUBNET UNIT

The synchronization subnet unit samples NTP timestamps on external events. External events can either be an incoming data packet on the network media (receive block) or an outgoing data packet (transmit block). In addition some basic synchronization subnet maintenance features are implemented in this unit. In fact, there are two duty timers each comparing NTP time against a 48 bit NTP preload value, firing whenever NTP time equals or exceeds the preload value. One duty timer is responsible for periodically triggering S/R/D/A-duties, see [SS95]. Duty timer B is spare, except duty timer B in synchronization subnet 1 (SSU1) is reserved for amortization duties, timer B in SSU2 is reserved for termination (T) duties and timer B in SSU3 is reserved for leap second insertion and deletion duties. The upper SYNC-block in Figure 8-62 synchronizes the asynchronous external timestamp receive signal (TSRCV) to the internal T-clock. The external signal timestamp transmit (TSXMT) is synchronized in the lower SYNC-block. Physically both SYNC-blocks are not part of this unit, but logically they are and therefore specified here for better understanding. A register layout is illustrated in Figure 8-62.

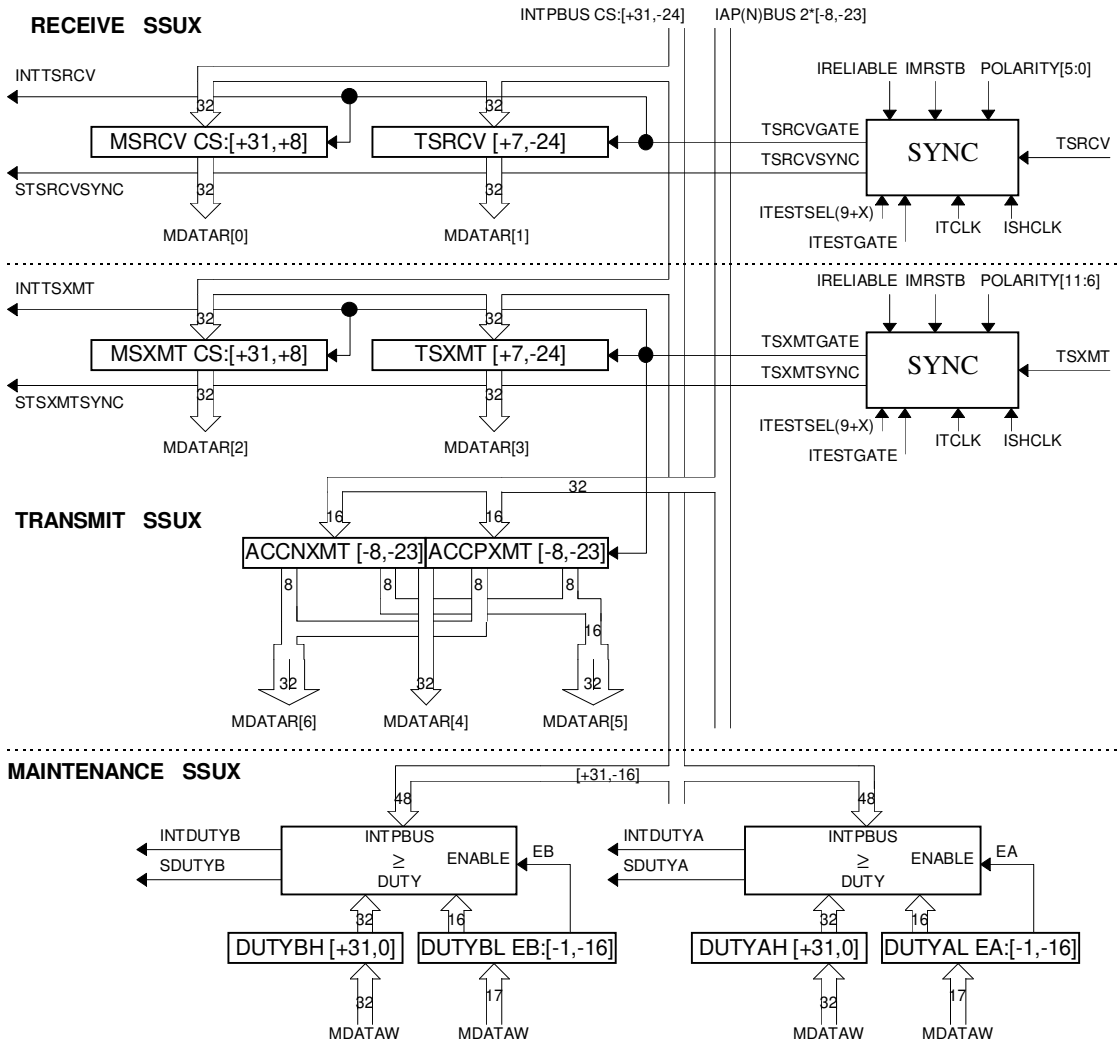


Figure 8-62 Synchronization Subnet Unit (SSU) Data Flow Diagram

Received data packets generate receive timestamps incorporating NTP bits [+31,-24] as well as an 8 bit checksum. Two 32-bit registers MSRCV and TSRCV store the timestamp information. On packet reception a receive status signal (TSRCV) is synchronized to the internal T-clock in the SYNC-block and a gate signal (TSRCVGATE) controlling both registers is created. For CPU interaction status signal (TSRCVSYNC) and interrupt signal (INTTSRCV) are generated.

Outgoing data packets deduct a more complex timestamp embodying NTP time information (bits [+31,-24]), 8 bit checksum and two times 16 bit accuracy information for upper and lower enve-

lopes. Managing upper and lower accuracy individually allow non symmetric accuracy intervals. This information is captured in registers MSXMT, TSXMT, ACCPXMT and ACCNXMT. The two registers ACCPXMT and ACCNXMT are both 16 bits wide. Three different 32 bit wide access strategies are available for optimized communication coprocessor access. On packet transmission a transmit event signal (TSXMT) is synchronized to the internal T-clock in the SYNC-block and a gate signal controlling the registers is created (TSXMTGATE). Status signal (TSXMTSYNC) and interrupt signal (INTTSXMT) are generated for CPU interaction

8.3.1 Receive Action

A received data packet should trigger the SSU block by a rising or falling edge on input pin TSRCV. If POLARITY in the corresponding SYNC-block is high, the rising edge is the active edge and vice versa. An exact timing diagram is given in Figure 8-63.

The SYNC-block generates two output signals, TSRCVSYNC and TSRCVGATE. TSRCVSYNC is the synchronized and maybe inverted version of TSRCV fed directly into the corresponding status bit. TSRCVGATE is the gate signal for registers MSRCV and TSRCV which latch the current NTP-time. TSRCVGATE also serves as an interrupt source signal indicating an incoming data packet. For an analysis of the worst case uncertainty in the SYNC-block see Section 8.12.

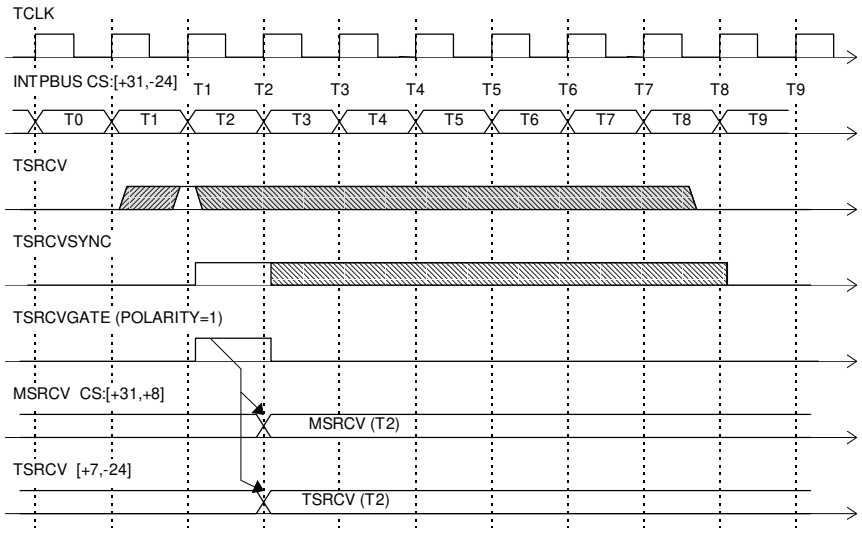


Figure 8-63 Receive Timestamp Timing Diagram

8.3.2 Transmit Action

The transmit action is very similar to the receive action. On data packet transmission, the communication coprocessor triggers the SSU block by a rising or falling edge on input pin TSXMT. If POLARITY in the corresponding SYNC-block is high, the rising edge is the active edge and vice versa. An exact timing diagram is given in Figure 8-64.

The SYNC-block generates two output signals, TSXMTSYNC and TSXMTGATE. TSXMTSYNC is the synchronized and maybe inverted (POLARITY = 1) version of TSXMT fed directly into the corresponding status bit. TSXMTGATE is the gate signal for registers MSXMT, TSXMT, ACCPXMT and ACCNXMT which latch the current NTP-time as well as accuracy information. TSXMTGATE also serves as an interrupt source signal indicating outgoing data packets. For an analysis of the worst case uncertainty in the SYNC-block see Section 8.12.

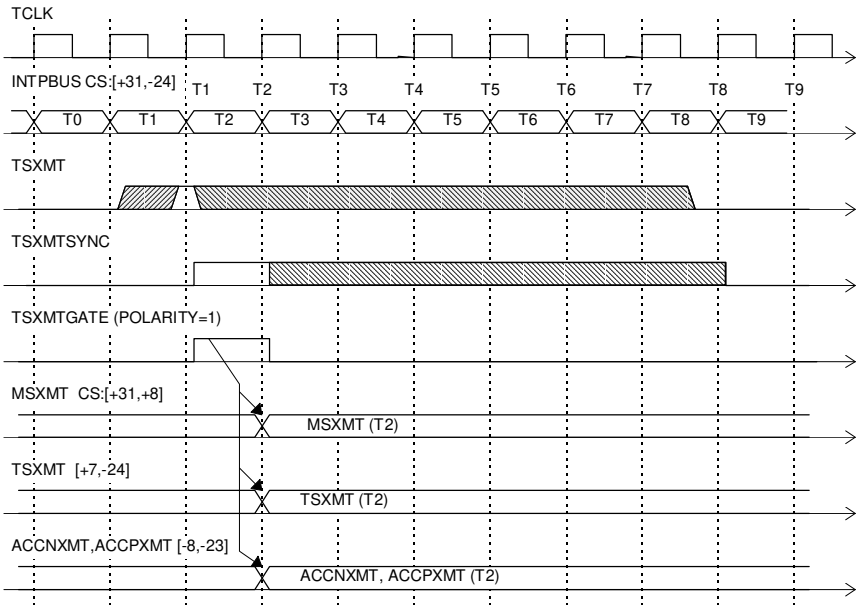


Figure 8-64 Transmit Timestamp Timing Diagram

8.3.3 Duty Timer Operation

Each duty-timer compares 48 bits (DUTY), organized as a 32 bits high part register DUTYA(B)H and a 16 bits low part register DUTYA(B)L, against the NTP-Bus. The time resolution of the comparison is 15 µs which is equivalent to NTP bit [-16]. An enable bit (EA, EB) is appended to the low part register (bit 17) to arm and disarm the duty-timer.

The duty-timer reference time (DUTY) is continuously compared against the NTP-Bus (INTPBUS). If a duty timer is armed and there is or was a transition from DUTY > INTPBUS to DUTY ≤ INTPBUS, a dedicated interrupt INTDUTYA resp. INTDUTYB is raised. Observe that only the pending interrupt bit in the ITU has to be cleared to service this interrupt. A status signal SDUTYA resp. SDUTYB is set high if the duty-timer is armed and DUTY ≤ INTPBUS.

First the high order part DUTYAH resp. DUTYBH is loaded then the low order part together with the ENABLE bit is written into EA:DUTYAL resp. EB:DUTYBL. Note that all 48 bits of DUTY must be set before setting ENABLE high. To disable a duty-timer the corresponding enable bit (EA resp. EB) is cleared.

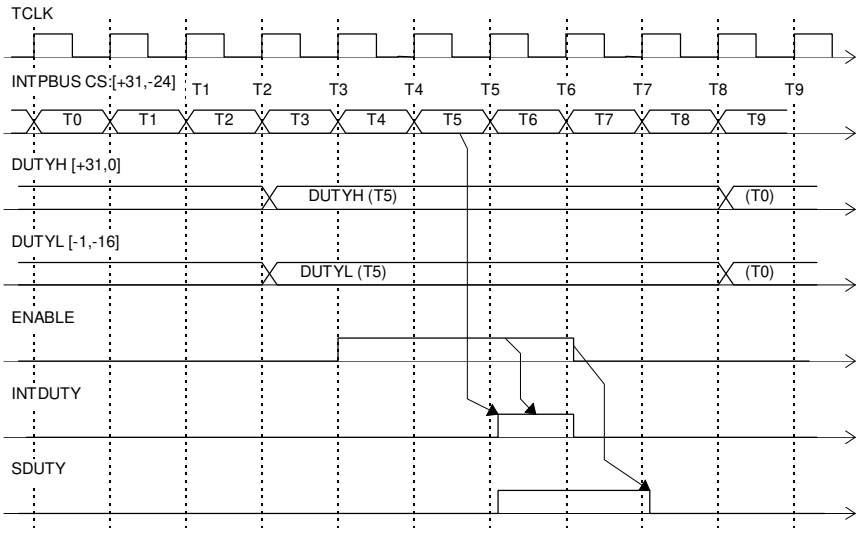


Figure 8-65 Duty-Timer Operation Timing Diagram

Both duty-timers A and B are equivalent. The duty-timer DUTYA is responsible to periodically trigger S/R/D/A-duties. With the exception of the first three synchronization subnet units SSU1, SSU2 and SSU3, DUTYB has no dedicated application yet.

Duty-timer B in synchronization subnet 1 can be programmed to start the amortization phase in units LTU and ACU. Hence, signal INTDUTYB in SSU1 is hardwired to signal AMORTDUTY in the LTU (see Figure 8-66). Duty-timer B in synchronization subnet unit 2 (SSU2) triggers termination duties (T-duty) at the end of a CSP reception round. Duty-timer B in synchronization subnet unit 3 (SSU3) controls leap second correction in unit local time. It is hardwired to LTU.

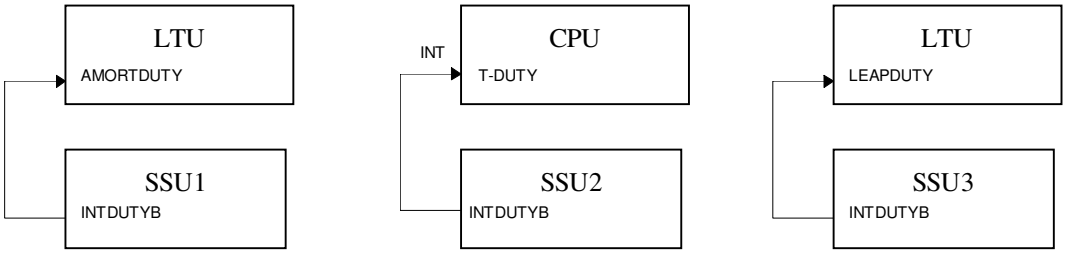


Figure 8-66 Duty-Timer B in SSU1 Starts Continuous Amortization, SSU2 Triggers T-Duties and SSU3 Triggers Leap Second Correction in Local Time Unit

8.3.4 Configuration SSU

Configuration bits are required to select the proper polarity for the external input signals TSRCV and TSXMT. To activate the rising edge, POLARITY on the SYNC-block must be high and low for the falling edge. All configuration bits are listed in the following Table 8-16.

Name	Signal	CONFIG Reg.	Bit #	Functionality
CPOLRCV1-6	POLARITY[0:5]	UTCCONF2	20-25	1: rising edge is active edge on TSRCV 0: falling edge is active edge on TSRCV
CPOLXMT1-6	POLARITY[6:11]	UTCCONF2	26-31	1: rising edge is active edge on TSXMT 0: falling edge is active edge on TSXMT

Table 8-16 Configuration Bits in Unit Synchronization Subnet

8.3.5 Status SSU

All status bits generated and maintained in the synchronization subnet unit are summarized in the following Table 8-17.

Name	Signal	STATUS Reg.	Bit #	Functionality
STSRCV1-6	TSRCVSYNC	UTCSTAT2	8-13	1: TSRCV = 1 0: TSRCV = 0
STSXMT1-6	TSXMTSYNC	UTCSTAT2	14-19	1: TSXMT = 1 0: TSXMT = 0
SDUTYA1-6	SDUTYA	UTCSTAT2	20-25	1: INTPBUS ≥ DUTYA and EA = 1 0: INTPBUS < DUTYA or EA = 0
SDUTYB1-6	SDUTYB	UTCSTAT2	26-31	1: INTPBUS ≥ DUTYB and EB = 1 0: INTPBUS < DUTYB or EB = 0

Table 8-17 Status Bits in Unit Synchronization Subnet

Status Bit STSRCV1-6 (Status TSRCV signal in ssu1-6) provides information on the current state of the external input signal TSRCV. If this bit is one, TSRCV carries high level.

Status Bit STSXMT1-6 (Status TSXMT signal in ssu1-6) provides information on the current state of the external input signal TSXMT. If this bit is one, TSXMT carries high level.

Status Bit SDUTYA1-6 (Status DUTY-timer A ssu1-6) provides information on the current NTP-time compared to duty-timer A. If the duty-timer is enabled and this bit is 1 the current NTP-time is larger than or equal to the time reference in DUTYA.

Status Bit SDUTYB1-6 (Status DUTY-timer B ssu1-6) provides information on the current NTP-time compared to duty-timer B. If the duty-timer is enabled and this bit is 1 the current NTP-time is larger than or equal to the time reference in DUTYB.

8.3.6 Interrupt Sources SSU

All interrupt sources within the synchronization subnet unit are summarized in Table 8-18.

Name	Signal	Interrupt Reg.	Bit #	Active	Functionality
ITSRCV1-6	INTTSRCV	UTCINT2	8-13	pulse	receive timestamp generated
ITSXMT1-6	INTTSXMT	UTCINT2	14-19	pulse	transmit timestamp generated
IDUTYA1-6	INTDUTYA	UTCINT2	20-25	pulse	INTPBUS ≥ DUTYA and EA = 1
IDUTYB1-6	INTDUTYB	UTCINT2	26-31	pulse	INTPBUS ≥ DUTYB and EB = 1

Table 8-18 Interrupt Sources in SSU

Interrupt ITSRCV1-6 (Interrupt TSRCV in ssu1-6) is activated on every active edge of signal TSRCV indicating a received data packet. The interrupt is cleared in ITU.

Interrupt ITSXMT1-6 (Interrupt TSXMT in ssu1-6) is activated on every active edge of signal TSXMT indicating a transmitted data packet. The interrupt is cleared in ITU.

Interrupt IDUTYA1-6 (Interrupt DUTYA in ssu1-6) is activated if EA = 1 and there is or was a transition from INTPBUS < DUTYA to INTPBUS ≥ DUTYA. The interrupt is cleared in ITU.

Interrupt IDUTYB1-6 (Interrupt DUTYB in ssu1-6) is activated if EB = 1 and there is or was a transition from INTPBUS < DUTYB to INTPBUS ≥ DUTYB. The interrupt is cleared in ITU.

8.3.7 Register Map

All SSU registers and decoded latch signals are listed in Table 8-19. The table header lists the memory element involved, the number of bits needed for that memory element, the access type (read/write/copy), the memory address, the latch element access width for write operations, the byte enable signals tied to and the multiplexer channel used for read operations. Furthermore the memory element gate signals are listed as well as a field indicating a very dynamic or a more or less static (D/S) memory element, the memory contents after reset and the data bus bit positions joining the register. This is a very agglomerating view of all SSU registers.

Memory Element	Width	R/W/C	Addr	Latch	BE [3:0]	Mux Sel [2:0]	Mux #	Memory Element Gate Signal	D/S	Reset	Data Bus Pos.
DUTYAH	32	W	0	8	3:0	-	-	MREGW[0] ∧ MBE[3:0]	S	0	31:0
DUTYAL	17	W	1	8	2:0		-	MREGW[1] ∧ MBE[2:0]	S	0	16:0
DUTYBH	32	W	2	8	3:0	-	-	MREGW[2] ∧ MBE[3:0]	S	0	31:0
DUTYBL	17	W	3	8	2:0		-	MREGW[3] ∧ MBE[2:0]	S	0	16:0
MSRCV	32	R	0	32	x	000	0	TSRCVGATE	S	0	31:0
TSRCV	32	R	1	32	x	001	1	TSRCVGATE	S	0	31:0
MSXMT	32	R	2	32	x	010	2	TSXMTGATE	S	0	31:0
TSXMT	32	R	3	32	x	011	3	TSXMTGATE	S	0	31:0
ACCNXMT &	32	R	4	32	x	100	4	TSXMTGATE	S	0	31:16
ACCPXMT									S	0	15:0
ACCNXMT[7:0] &	32	R	5	32	x	101	5	TSXMTGATE	S	0	31:0
ACCPXMT[7:0] &									S	0	23:16
ACCNXMT[7:0] &									S	0	15:8
ACCPXMT[7:0]									S	0	7:0
ACCNXMT[15:8] &	32	R	6	32	x	110	6	TSXMTGATE	S	0	31:0
ACCPXMT[15:8] &									S	0	23:16
ACCNXMT[15:8] &									S	0	15:8
ACCPXMT[15:8]									S	0	7:0

Table 8-19 Synchronization Subnet Unit Register Map

8.3.8 Programmers Model

A very detailed view of all SSU registers is given in this section and is intended for hardware level programmers. The list order is increasing memory address. All addresses listed are referenced for SSU1. Add offsets from Table 8-20 to obtain the address for SSUx.

SSUx	Add Offset	Start Address	End Address
1	00h	140h	15fh
2	20h	160h	17fh
3	40h	180h	19fh
4	60h	1a0h	1bfh
5	80	1c0h	1dfh
6	a0h	1e0h	1ffh

Table 8-20 SSUx Memory Address Locations

8.3.8.1 Receive Timestamp Registers

Figure 8-67 MSRCV - Macro Stamp Receive Register

This 32-bit register holds the upper 32 bits of INTPBUS latched on the active edge of signal TSRCV. Bits 24 to 31 represent a control word for the 56 bit timestamp information stored in bits 0 to 23 and register TSRCV. Macro Stamp holds bits [+31,+8] in NTP-format. Upon reset MSRCV is cleared to all 0.

Figure 8-68 TSRCV - Timestamp Receive Register

This 32-bit register holds the lower 32 bits of INTPBUS latched on the active edge of signal TSRCV. Timestamp holds bits [+7,-24] in NTP-format. Upon reset TSRCV is cleared to all 0.

Figure 8-69 MSXMT - Macro Stamp Transmit Register

This 32-bit register holds the upper 32 bits of INTPBUS latched on the active edge of signal TSXMT. Bits 24 to 31 represent a control word for the 56 bit timestamp information stored in bits 0 to 23 and register TSXMT. Macro Stamp holds bits [+31,+8] in NTP-format. Upon reset MSXMT is cleared to all 0.

Figure 8-70 TSXMT - Timestamp Transmit Register

This 32-bit register holds the lower 32 bits of INTPBUS latched on the active edge of signal TSXMT. Timestamp holds bits [+7,-24] in NTP-format. Upon reset TSXMT is cleared to all 0.

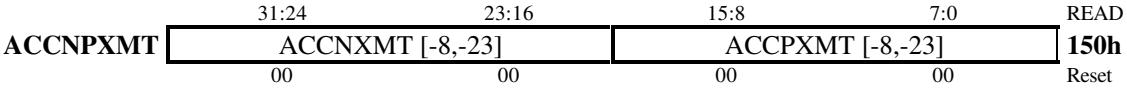


Figure 8-71 ACCNPXMT - Accuracy N and P Register

This 32-bit register holds the 16-bit IANBUS and IAPBUS, latched on the active edge of signal TSXMT. ACCNPXMT holds bits [-8,-23] in NTP-format. Upon reset, ACCNPXMT is cleared to all 0.

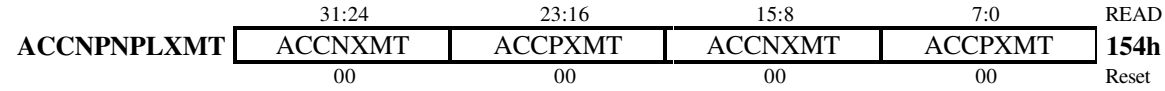


Figure 8-72 ACCNPNPLXMT - Accuracy N and P Register, Lower 8 bits

This 32-bit register holds the lower 8 bits from IANBUS and IAPBUS, latched on the active edge of signal TSXMT. ACCNPNPLXMT holds bits [-16,-23] in NTP-format. Upon reset, ACCNPNPLXMT is cleared to all 0.

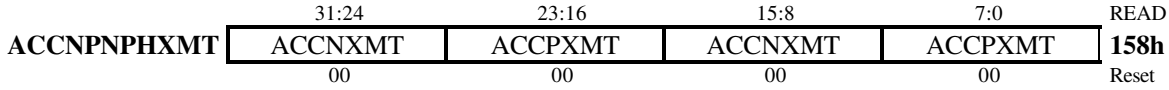


Figure 8-73 ACCNPNPHXMT - Accuracy N and P Register, Higher 8 bits

This 32-bit register holds the upper 8 bits from IANBUS and IAPBUS, latched on the active edge of signal TSXMT. ACCNPNPHXMT holds bits [-16,-23] in NTP-format. Upon reset, ACCNPNPHXMT is cleared to all 0.

Figure 8-74 DUTYAH - Duty-Timer A Higher Preload Register

This 32-bit register holds the higher 32 bits of the time reference. It is compared against the INTPBUS and if the contents in DUTYAH plus DUTYAL is larger than or equivalent to the INTPBUS and EA = 1, interrupt INTDUTYA is raised. DUTYAH holds bits [+31,0] in NTP-format. Upon reset, DUTYAH is cleared to all 0.

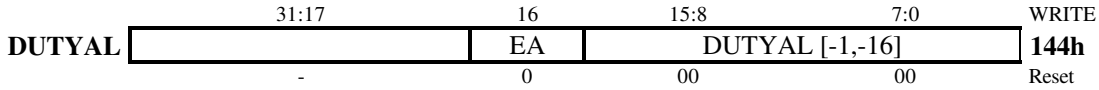


Figure 8-75 DUTYAL - Duty-Timer A Lower Preload Register

This 17-bit register holds the NTP time reference [-1,-16] in bits [15:0] and an enable bit EA in bit [16]. The lower 16 bits are compared against the INTPBUS and if the contents in DUTYAH plus DUTYAL is larger than or equivalent to the INTPBUS, interrupt INTDUTYA is raised. Upon reset, DUTYAL is cleared to all 0.

Figure 8-76 DUTYBH - Duty-Timer B Higher Preload Register

This 32-bit register holds the higher 32 bits of the NTP time reference. It is compared against the INTPBUS and if the contents in DUTYBH plus DUTYBL is larger than or equivalent to the INTPBUS, interrupt INTDUTYB is raised. DUTYBH holds bits [+31,0] in NTP-format. Upon reset, DUTYBH is cleared to all 0.

	31:17	16	15:8	7:0	WRITE
DUTYBL		EB	DUTYBL [-1,-16]		**14ch**
	-	0	00	00	Reset

Figure 8-77 DUTYAL - Duty-Timer B Lower Preload Register

This 17-bit register holds the NTP time reference [-1,-16] in bits [15:0] and an enable bit EB in bit [16]. It is compared against the INTPBUS and if the contents in DUTYBH plus DUTYBL is larger than or equivalent to the INTPBUS, interrupt INTDUTYB is raised. Upon reset, DUTYBL is cleared to all 0.

8.3.9 SSU Test Strategy

The basic test concept is elucidated in Chapter 5.1. Testing through the bus interface unit (BIU) is accomplished here. Some of the unit internal control signals are mapped to a unit test port and further to primary output pins. Table 8-21 lists the control signals and the primary output pins they are mapped to. Figure 8-78 illustrates the SSU test port.

SSU Signal Name	Testbus Pin #	CSUTIME Pin #	Comments
TSRCVGATE	0	35	indicates receive timestamp generation
TSRCVSYNC	1	36	reflects the synchronized TSRCV input signal
TSXMTGATE	2	37	indicates transmit timestamp generation
TSXMTSYNC	3	38	reflects the synchronized TSXMT input signal
SDUTYA	4	39	INTPBUS ≥ DUTYA and EA = 1
SDUTYB	5	40	INTPBUS ≥ DUTYB and EA = 1
EA	6	41	enable signal for duty-timer A
EB	7	42	enable signal for duty-timer B

Table 8-21 Various Observed Control Signals in SSU

One input port bit TESTSEL and one output port bus TESTBUS[y:0] is required. Activating TESTSEL puts the synchronization subnet unit in test mode. Observable unit internal signals are propagated to primary output pins through the SSU TESTBUS port and the ITESTBUS bus system.

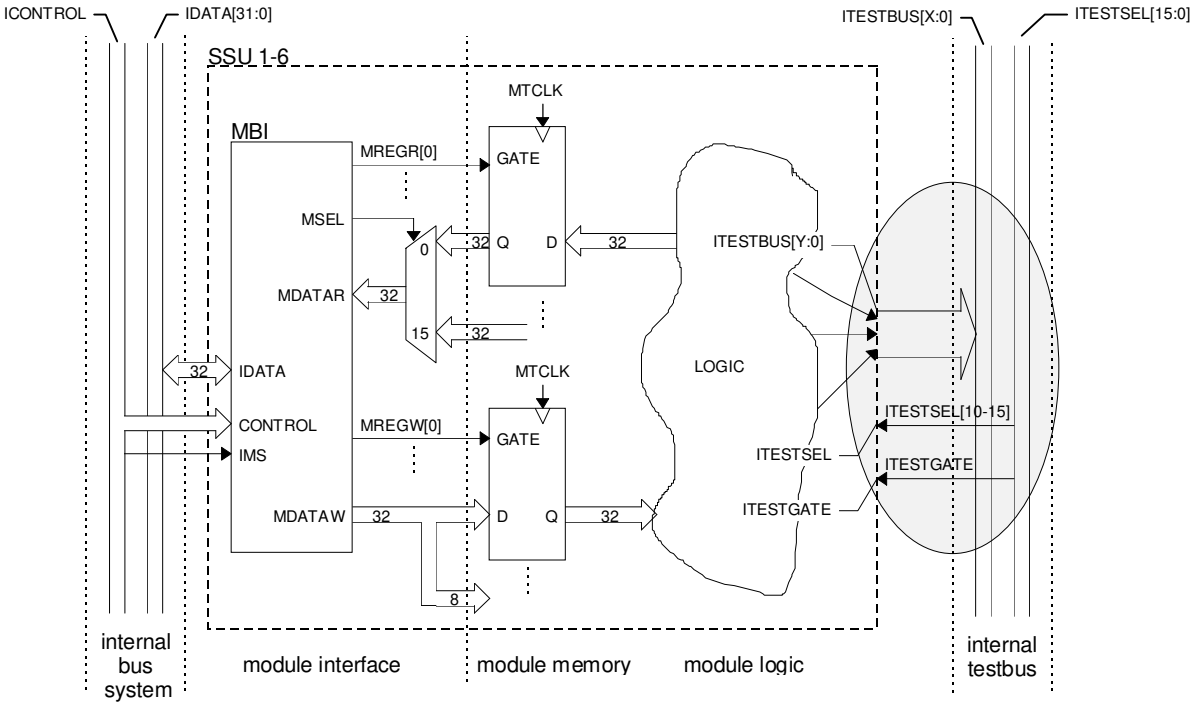


Figure 8-78 SSU Test Port Layout

8.3.10 Self-Test Option

The SSU has self-test capability. The SYNC-block in Figure 8-62 shows two input pins ITESTGATE and ITESTSEL. Input ITESTSEL from SSU1 is connected to ITESTSEL[10], ITESTSEL from SSU2 to ITESTSEL[11] and so on for all 6 SSUs, cf. Figure 8-78.

Signal ITESTSEL enables a test mode in the SYNC-block. Whenever signal TESTGATE carries a gate pulse generated by a HOST write operation to BTU (Built-In Test Unit) address 084h, TSRCVGATE and TSXMTGATE carry a gate signal for registers MSRCV, TSRCV, MSXMT, TSXMT, ACCNXMT and ACCPXMT which latch the current NTP-time as well as accuracy information. This behaviour is equivalent to an active edge on external inputs TSRCV and TSXMT. The complete SSU, including the SYNC-block, can be tested without taking any external test precautions. While the SSU is set into test mode (ITESTSEL[10-15] = 1), events on signals TSRCV and TSXMT are ignored. The test sequence is illustrated in Figure 8-79.

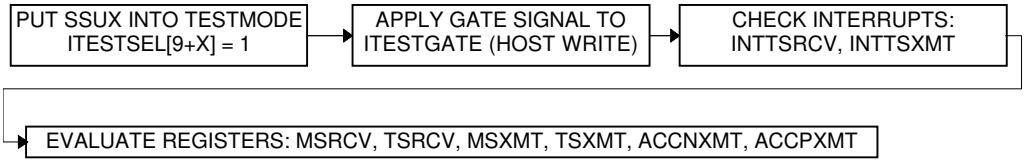


Figure 8-79 Test Sequence for the SSU Self-Test

8.4 GPS UNIT

The Global Positioning Unit (GPU) samples NTP timestamps on GPS events. In addition a status signal indicating the GPS health condition is sampled together with the timestamp. It is stored in the highest bit position of the 8 bit checksum CS. Hence only 7 control bits are available to protect the timestamp, ie only the 35 lower bits are protected by the checksum, cf. Section 8.1.7. The upper SYNC-block in Figure 8-80 synchronizes the asynchronous external one pulse per second (1PPS) signal from the GPS receiver (GPS1PPS) to the internal T-clock. The status signal (GPSSTAT), if available from the GPS receiver, is synchronized in the lower SYNC-block. Physically both SYNC-blocks are not part of this unit, but logically they are and therefore specified here for better understanding. A basic register layout is illustrated in Figure 8-80.

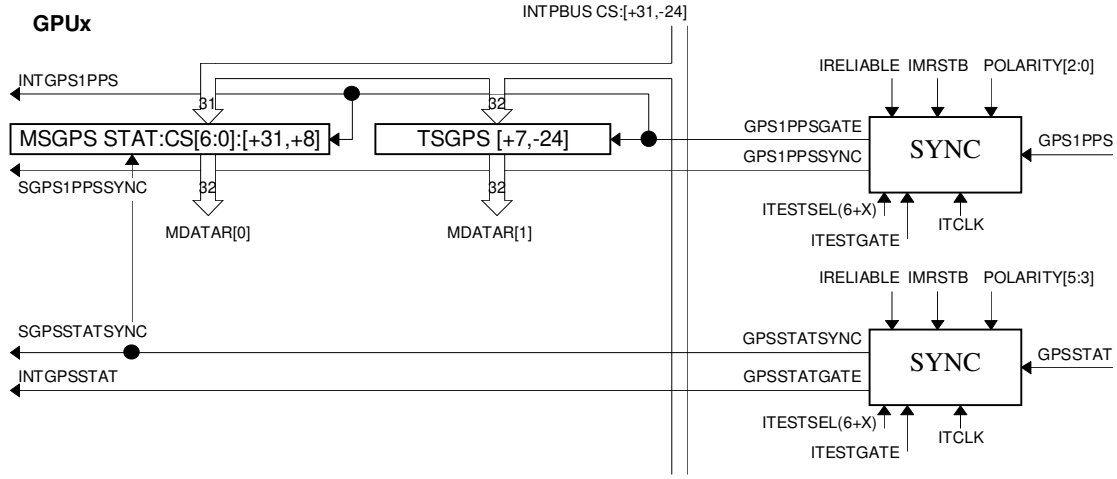


Figure 8-80 Global Positioning Unit Data Flow Diagram

Every second a GPS event (GPS1PPS) generates a GPS timestamp incorporating NTP bits [+31,-24] as well as an 7 bit checksum and a 1 bit receiver status information. Two 32-bit registers MSGPS and TSGPS save the timestamp. On a GPS event (1PPS) a signal (GPS1PPS) is synchronized to the internal T-clock in the SYNC-block and a gate signal (GPS1PPSGATE) controlling both registers is created. For CPU interaction a status signal (GPS1PPSSYNC) and an interrupt signal (INTGPS1PPS) are generated. If status information is retrievable from the GPS receiver, this information is synchronized in the lower SYNC-block and fed directly into the MSGPS register as bit 31. It is also provided to ITU as signal SGPSSTATSYNC for status information and INTGPSSTAT for interrupt service.

8.4.1 One Pulse Per Second Event

GPS time receivers usually provide a 1PPS output signal. This signal is connected to the GPS1PSS input pin. If POLARITY is high, timestamps are generated on every rising edge, if low on every falling edge of GPS1PPS. A timing diagram is shown in Figure 8-81. The SYNC-block generates two output signals, GPS1PPSSYNC and GPS1PPSGATE. GPS1PPSSYNC is the synchronized version of GPS1PSS fed directly into the corresponding status bit. GPS1PPSGATE is the gate signal for registers MSGPS and TSGPS latching the current NTP time. GPS1PPSGATE also serves as an interrupt source signal indicating the begin of a standard second, hence telling the HOST to pick up the new timestamp. A special self-test mode is discussed in a following section.

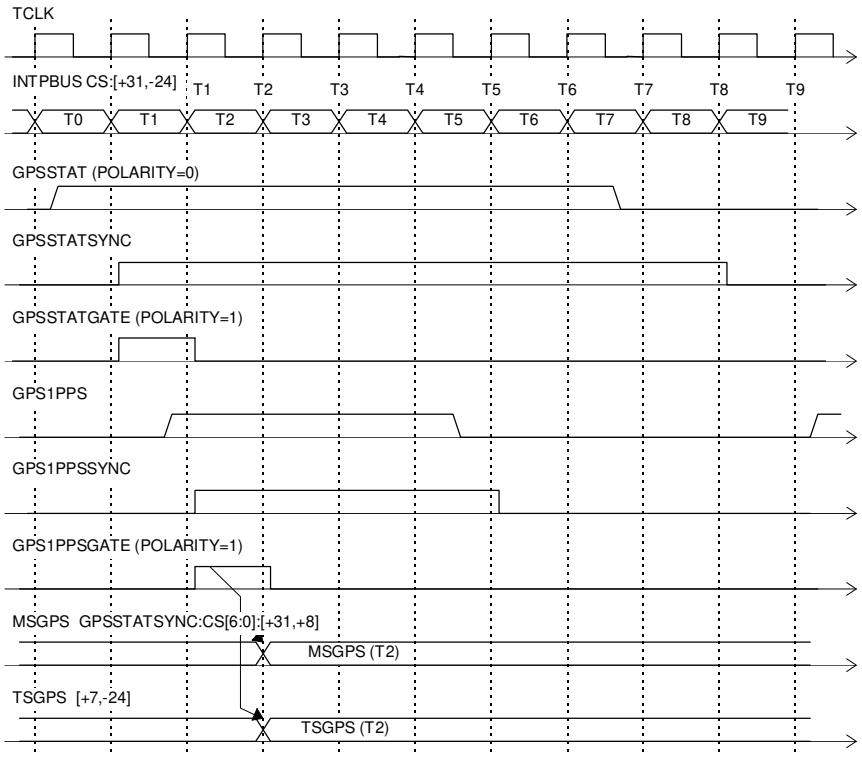


Figure 8-81 GPS Timestamp Timing Diagram

8.4.2 Configuration GPU

Configuration bits are required to select the proper polarity for the external input signals GPS1PPS and GPSSTAT. To select the rising edge, signal POLARITY on the SYNC-block must be high and low for the falling edge. All configuration bits are listed in the following Table 8-22.

Name	Signal	CONFIG Reg.	Bit #	Functionality
CPOLGPS1PPS1-3	POLARITY[0:2]	UTCCONF2	12-14	1: rising edge is active edge on GPS1PPS 0: falling edge is active edge on GPS1PPS
CPOLGPSSTAT1-3	POLARITY[3:5]	UTCCONF2	15-17	1: rising edge is active edge on GPSSTAT 0: falling edge is active edge on GPSSTAT

Table 8-22 Configuration Bits in Unit GPS

8.4.3 Status GPU

All status bits generated and maintained in the GPS unit are summarized in the following Table 8-23.

Name	Signal	STATUS Reg.	Bit #	Functionality
SGPS1PPS1-3	GPS1PPSSYNC	UTCSTAT2	0-2	1: Signal GPS1PPS = 1 0: Signal GPS1PPS = 0
SGPSSTAT1-3	GPSSTATSYNC	UTCSTAT2	3-5	1: Signal GPSSTAT = 1 0: Signal GPSSTAT = 0

Table 8-23 Status Bits in Unit GPS

Status Bit SGPS1PPS1-3 (Status GPS1PPS signal in gpu1-3) provides information on the current state of the external input signal GPS1PPS. If this bit is one, GPS1PPS carries high level.

Status Bit SGPSSTAT1-3 (Status GPSSTAT signal in gpu1-3) provides information on the current state of the external input signal GPSSTAT. If this bit is one, GPSSTAT carries high level.

8.4.4 Interrupt Sources GPU

All interrupt sources in the synchronization subnet unit are summarized in Table 8-24.

Name	Signal	Interrupt Reg.	Bit #	Active	Functionality
IGPS1PPS1	INTGPS1PPS	UTCINT2	0	pulse	GPS timestamp captured
IGPS1PPS2	INTGPS1PPS	UTCINT2	1	pulse	GPS timestamp captured
IGPS1PPS3	INTGPS1PPS	UTCINT2	2	pulse	GPS timestamp captured
IGPSSTAT1	INTGPSSTAT	UTCINT2	3	pulse	GPS status changed
IGPSSTAT2	INTGPSSTAT	UTCINT2	4	pulse	GPS status changed
IGPSSTAT3	INTGPSSTAT	UTCINT2	5	pulse	GPS status changed

Table 8-24 Interrupt Sources in GPU

Interrupt IGPS1PPS1-3 (Interrupt GPS1PPS in gpu1-3) is activated on every active edge of signal GPS1PPS. The interrupt is cleared in ITU.

Interrupt IGPSSTAT1-3 (Interrupt GPSSTAT in gpu1-3) is activated on every active edge of signal GPSSTAT. The interrupt is cleared in ITU.

8.4.5 Register Map

All GPU registers and decoded latch signals are listed in Table 8-25. The table header lists the memory element involved, the number of bits needed for that memory element, the access type (read/write/copy), the memory address, the latch element access width for write operations, the byte enable signals tied to and the multiplexer channel used for read operations. Furthermore the memory element gate signals are listed as well as a field indicating a very dynamic or a more or less static (D/S) memory element, the memory contents after reset and the data bus bit positions joining the register. This is a very agglomerating view of all GPU registers. A more detailed one is given in the next section explaining all programming details.

Memory Element	Width	R/WC	Addr	Latch	BE [3:0]	Mux Sel [2:0]	Mux #	Memory Element Gate Signal	D/S	Reset	Data Bus Pos.
MSGPS	32	R	0	32	x	000	0	GPS1PPSGATE	S	0	31:0
TSGPS	32	R	1	32	x	001	1	GPS1PPSGATE	S	0	31:0

Table 8-25 Global Positioning Unit Register Map

8.4.6 Programmers Model

A very detailed view of all GPU registers is given in this section, intended for hardware level programmers. The list order is increasing memory address. All addresses listed are referenced for GPU1. Add offsets from Table 8-26 to obtain the address for GPUx.

GPUx	Add Offset	Start-Address	End-Address
1	00h	0e0h	0ffh
2	20h	100h	11fh
3	40h	120h	13fh

Table 8-26 GPUx Memory Address Locations

Figure 8-82 MSGPS - Macro Stamp GPS Register

This 32-bit register holds the GPS status signal in bit position [31], CS[6:0] in bits [30:24] and NTP bits [+31,+8] from the INTPBUS in bits [23:0]. Data is latched on the active edge of signal GPS1PPS. Bits 24 to 30 represent a 7-bit reduced control word for the 56-bit timestamp stored in bits 0 to 23 plus register TSGPS. Observe that only the lower three bits in MSGPS are protected by the 7-bit control word. Status bit [31] reflects the health condition of the GPS receiver, ie holds the logic level of signal GPSSTAT at sampling time. Upon reset, MSGPS is cleared to all 0.

Figure 8-83 TSGPS - Time Stamp GPS Register

This 32-bit register holds the lower 32 bits of the INTPBUS latched at the active edge of signal GPS1PPS. Upon reset, TSGPS is cleared to all 0.

8.4.7 GPU Test Strategy

The basic test concept was elucidated in Chapter 5.1. Testing through the bus interface unit (BIU) is accomplished here. Some of the unit internal control signals are mapped to a unit test port and further to primary output pins. Table 8-27 lists the control signals and the primary output pins they are mapped to. Figure 8-84 illustrates the GPU test port.

GPU1 Signal Name	Testbus Pin #	CSUTIME Pin #	Comments
GPS1PPSGATE	0	32	indicates GPS timestamp generation
GPS1PPSSYNC	1	33	reflects the synchronized GPS1PPS input signal (POL = 0)
GPSSTATSYNC	2	34	reflects the synchronized GPSSTAT input signal (POL = 0)

Table 8-27 Various Observed Control Signals in GPU

One input port bit TESTSEL and one output port bus TESTBUS[y:0] is required. Activating TESTSEL puts the GPU in test mode. Observable unit internal signals are propagated to primary output pins through the GPU TESTBUS port and the ITESTBUS bus system.

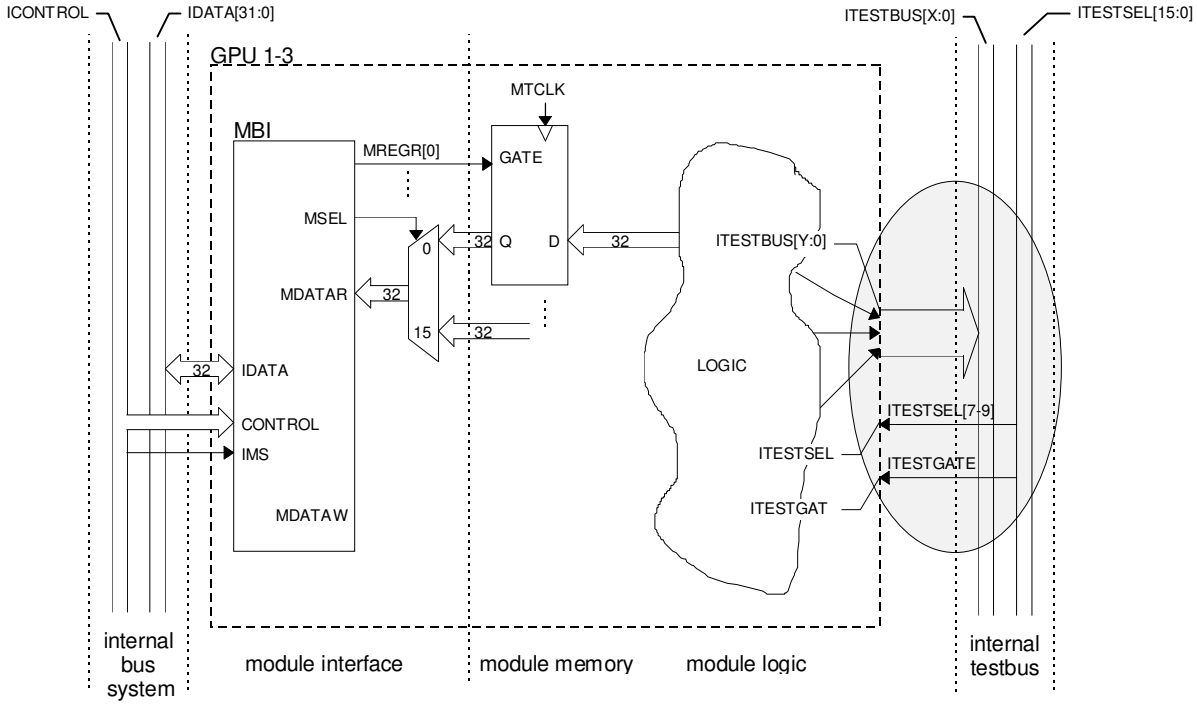


Figure 8-84 GPU Test Port Layout

8.4.8 Self-Test Option

The GPU has self-test capability. The SYNC-block in Figure 8-80 shows two input signals ITESTGATE and ITESTSEL. ITESTSEL in GPU1 is connected to ITESTSEL[7], ITESTSEL in GPU2 to ITESTSEL[8] resp. ITESTSEL in GPU3 to ITESTSEL[9], cf. Figure 8-84. ITESTSEL enables a test mode in the SYNC-block. Whenever signal ITESTGATE inputs a gate pulse generated by a HOST write operation to BTU (Built-In Test Unit) address 084h, TSRCVGATE and TSXMTGATE carry a gate signal for registers MSGPS and TSGPS which latch the current NTP-time. This behaviour is equivalent to an active edge on external input GPS1PPS. All logic in the GPU, including the SYNC-block, can be tested internally without taking any external test precautions. While GPU is set into test mode (ITESTSEL[7-9] = 1), events on signals GPS1PPS and GPSSTAT are ignored. The test steps are illustrated in Figure 8-85.

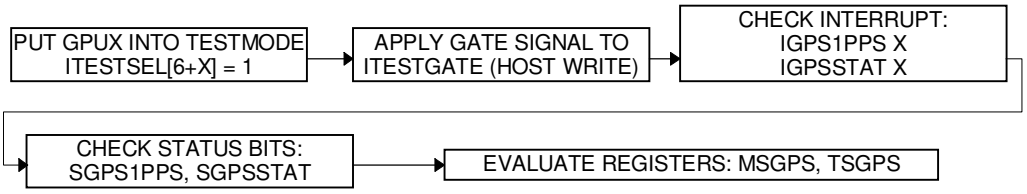


Figure 8-85 Test Sequence for the GPU Self-Test

8.5 APPLICATION UNIT

The application unit offers two services. One is timestamping of 9 different external events and the other is the generation of a timed UTCSU external event. Timestamps triggered by a level programmable edge on signal TSAPP[x] are captured in 9 register sets MSAPP, TSAPP and ACCP(N)APP. Every read access of these nine register sets is indicated in status bit SAPPREF.

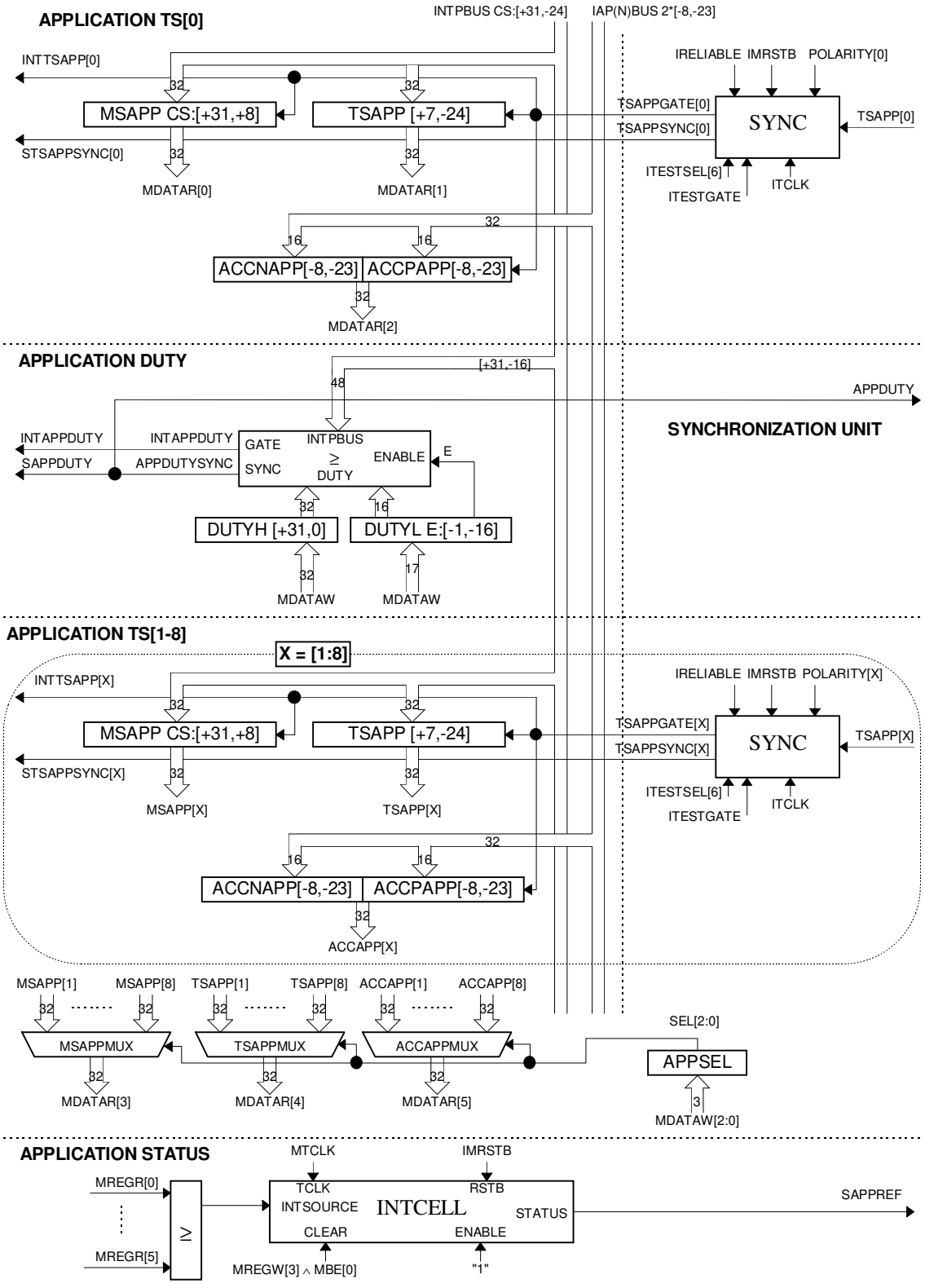


Figure 8-86 Application Unit Data Flow Diagram

An NTP-time reference value can be programmed into the duty timer. Whenever the internal INTPBUS exceeds this number and ENABLE (E) in the duty timer is set high, the internal status signal SAPPDUTY as well as the external accessible signal APPDUTY are set high. Signal INTAPPDUTY generates an application interrupt pulse. For information on the operation of the duty timer see section 8.3.3. The basic APU functionality is illustrated in Figure 8-86.

8.5.1 Event Recording

Nine different external input signals TSAPP[0-8] can be recorded. Every active edge of TSAPP[0-8], which is either the rising edge or the falling edge whatever is set in the configuration register, samples timestamps into nine register sets, each consisting of a macrostamp, a timestamp and an accuracy register. The SYNC-blocks in Figure 8-86 synchronize the asynchronous external input pins TSAPP[0-8] with the internal T-clock.

A CPU can read the sampling register sets in two, slightly different, ways. The register set, connected to input signal TSAPP[0], is read at memory location 0c0h through 0cbh while the remaining eight sets, which are connected to input signals TSAPP[1-8], follow a different access semantics. Because of too few available memory address locations in this unit, the eighth sets must share the same memory address space between 0cch and 0d7h. Hence, before reading a register set, the set must be selected by a write to memory location 0c8h (APPSEL) with the set number minus one. This selects the set for read access through address space 0cch-0d7h. As an example, to read the register set connected to input signal TSAPP[5] follow the instructions in Figure 8-87.

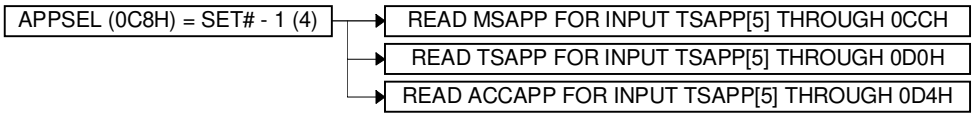


Figure 8-87 CPU-Access Procedure for the Application Recording Register Set 5

Observe that if one of the timestamp registers MSAPP, TSAPP or ACCAPP in any set is read by the CPU, a status flag SAPPREF in the status register UTCSTAT1 is set. This flag can be used in case of synchronization loss to report any, maybe faulty, application timestamp and to call a recovery process. SAPPREF is cleared by writing address APPCLEAR.

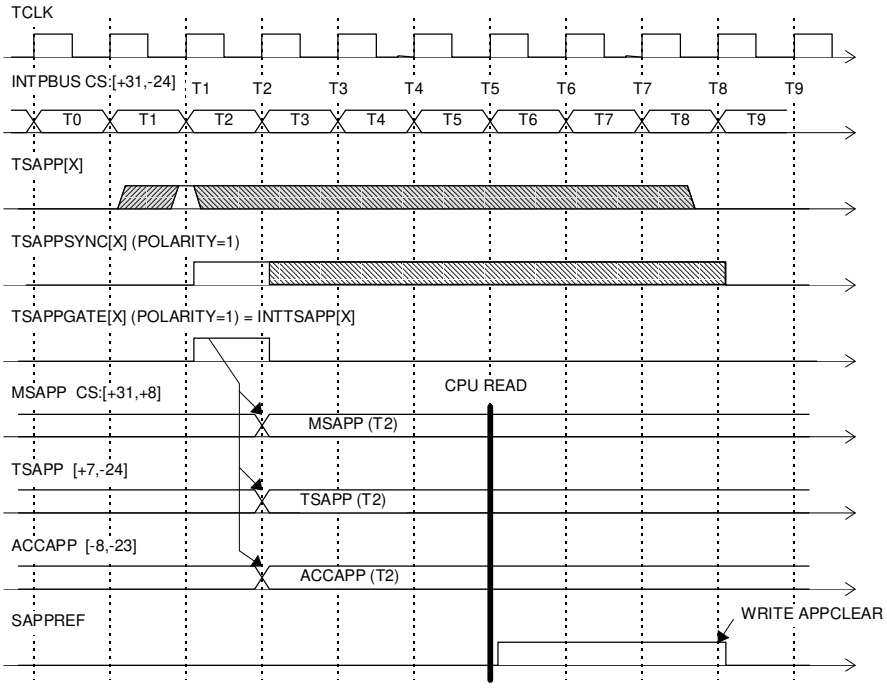


Figure 8-88 Application Event Recording

Figure 8-88 illustrates a timing diagram when data is sampled into one application register set. The active edge (rising if POLARITY = 1) of the external input signal TSAPP[x] generates a latch

pulse TSAPPGATE[x] for the application registers MSAPP, TSAPP and ACCAPP. In addition the interrupt ITSAPPx is reported to the CPU and the status flag SAPPREF is set when the CPU reads the registers.

8.5.2 Event Generation

The APU hosts one output channel which can trigger a timed external event. The event time is loaded into the application duty-timer reference registers DUTYH and DUTYL. If the INTPBUS is equal to or larger than this time reference, output signal SAPPDUTY goes high and stays high until the duty-timer is disabled (E = 0) or a later time reference is loaded. Signal INTAPPDUTY reports the application interrupt IAPPDUTY. The operation of the duty-timer is exemplified in Section 8.3.3 and a timing diagram displaying an application event generation is illustrated in Figure 8-89.

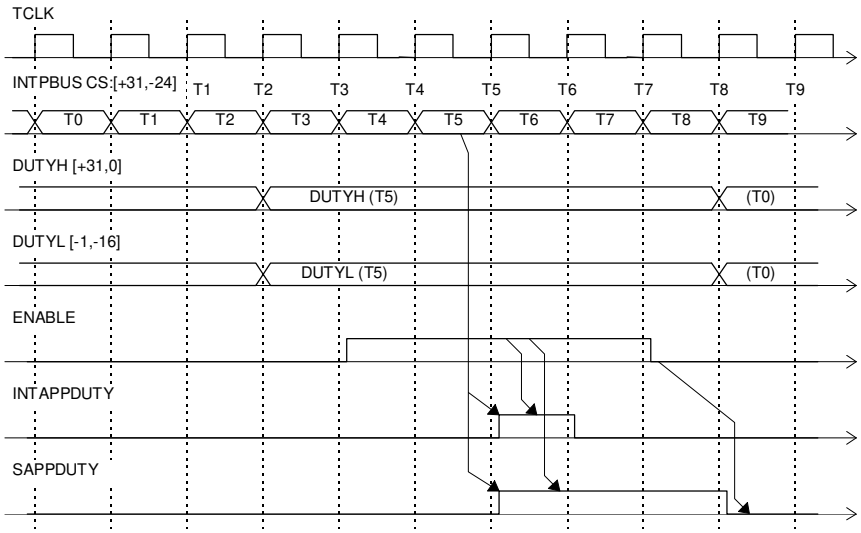


Figure 8-89 Application Event Generation

8.5.3 Configuration APU

Configuration bits are required to select the proper polarity for the external input signals TSAPP[x]. To activate the rising edge, POLARITY on the SYNC-block must be high and low to select the falling edge. All configuration bits are listed in Table 8-28.

Name	Signal	CONFIG Reg.	Bit #	Functionality
CPOLAPP	POLARITY[0]	UTCCONF2	0	1: rising edge is active edge on TSAPP[0] 0: falling edge is active edge on TSAPP[0]
CPOLAPP1-8	POLARITY[1-8]	UTCCONF2	1-8	1: rising edge is active edge on TSAPP[x] 0: falling edge is active edge on TSAPP[x]

Table 8-28 Configuration Bits in the Application Unit

8.5.4 Status APU

All status bits making their debut in the application unit are summarized in Table 8-29.

Name	Signal	STATUS Reg.	Bit #	Functionality
STSAPP	TSAPPSYNC[0]	UTCSTAT1	20	1: TSAPP[0] = 1; 0: TSAPP[0] = 0
STSAPP1-8	TSAPPSYNC[1-8]	UTCSTAT1	21-28	1: TSAPP[x] = 1; 0: TSAPP[x] = 0
SAPPDUTY	SAPPDUTY	UTCSTAT1	29	1: DUTY ≥ INTPBUS and E = 1 0: DUTY < INTPBUS or E = 0
SAPPREF	SAPPREF	UTCSTAT1	30	1: APU register was referenced 0: APU register was not referenced

Table 8-29 Status Bits in Application Unit

Status Bit STSAPP (Status TimeStamp APPlication) provides information on the current state of the external input signal TSAPP[0]. If this bit is one, TSAPP[0] carries high level.

Status Bit STSAPP1-8 (Status TimeStamp APPlication 1-8) provides information on the current state of the external input signal TSAPP[x]. If this bit is one, TSAPP[x] carries high level.

Status Bit SAPPDUTY (Status APPlication snapshot DUTY-timer) provides information on the current state of the application duty-timer. If this bit is one the duty-timer is enabled and expired.

Status Bit SAPPREF (Status APPlication REFerenced) indicates an APU read access since this status bit was last cleared by a CPU write at address 0cch. If SAPPREF is one, an application timestamp register was accessed by a CPU read operation.

8.5.5 Interrupt Sources APU

All interrupt sources due to events within the application unit are summarized in Table 8-30.

Name	Signal	Interrupt Reg.	Bit #	Active	Functionality
ITSAPP	INTTSAPP[0]	UTCINT1	20	pulse	timestamp captured at TSAPP[0].
ITSAPP1-8	INTTSAPP[1-8]	UTCINT1	21-28	pulse	timestamp captured at TSAPP[1-8]
IAPPDUTY	INTAPPDUTY	UTCINT1	29	pulse	INTPBUS ≥ DUTY and E = 1

Table 8-30 Interrupt Sources in APU

Interrupt ITSAPP (Interrupt TimeStamp APPlication) is raised on every active edge of signal TSAPP[0] requesting an application timestamp. The interrupt is cleared in ITU.

Interrupt ITSAPP1-8 (Interrupt TimeStamp APPlication 1-8) is raised on every active edge of signal TSAPP[1-8] requesting an application timestamp. The interrupt is cleared in ITU.

Interrupt IAPPDUTY (Interrupt APPlication DUTY-timer expired) is raised whenever the application duty-timer is enabled and expires. The interrupt is cleared in the ITU.

8.5.6 Register Map

All APU registers and decoded latch signals are listed in Table 8-31. The table header lists the memory element involved, the number of bits needed for that memory element, the access type (read/write/copy), the memory address, the latch element access width for write operations, the byte enable signals tied to and the multiplexer channel used for read operations. Furthermore the memory element gate signals are listed as well as a field indicating a very dynamic or a more or less static (D/S) memory element, the memory contents after reset and the data bus bit positions joining the register. This is a very agglomerating view of all APU registers. A more detailed one is given in the next section explaining all programming details.

Memory Element	Width	R/W/C	Addr	Latch	BE [3:0]	Mux Sel [2:0]	Mux #	Memory Element Gate Signal	D/S	Reset	Data Bus Pos.
DUTYH	32	W	0	8	3:0	-	-	MREGW[0] ∧ MBE[3:0]	S	0	31:0
DUTYL	17	W	1	8	2:0	-	-	MREGW[1] ∧ MBE[2:0]	S	0	16:0
APPSEL	3	W	2	8	0	-	-	MREGW[2] ∧ MBE[0]	S	0	2:0
APPCLEAR	-	W	3	8	0	-	-	MREGW[3] ∧ MBE[0]	S	0	-
MSAPP	32	R	0	32	x	000	0	TSAPPGATE[0]	S	0	31:0
TSAPP	32	R	1	32	x	001	1	TSAPPGATE[0]	S	0	31:0
ACCAPP	32	R	2	32	x	010	2	TSAPPGATE[0]	S	0	31:0
MSAPPX	32	R	3	32	x	011	3	TSAPPGATE[x]	S	0	31:0
TSAPPX	32	R	4	32	x	100	4	TSAPPGATE[x]	S	0	31:0
ACCAPPX	32	R	5	32	x	101	5	TSAPPGATE[x]	S	0	31:0

Table 8-31 Application Unit Register Map

8.5.7 Programmers Model

A very detailed view of all Application Unit registers is given in this section, intended for hardware level programmers. The list order is increasing memory address.

8.5.7.1 Application Timestamp Registers

Figure 8-90 MSAPP - Macro Stamp Application Register

This 32-bit register holds the upper 32 bits of the INTPBUS latched on the active edge of signal TSAPP[0]. Bits 24 to 31 represent a control word for the 56-bit timestamp stored in bits 0 to 23 plus register TSAPP. Macro Stamp holds bits [+31,+8] in NTP-format. Upon reset, MSAPP is cleared.

Figure 8-91 TSAPP - Timestamp Application Register

This 32-bit register holds the lower 32 bits of the INTPBUS latched on the active edge of signal TSAPP[0]. Time Stamp holds bits [+7,-24] in NTP-format. Upon reset, TSAPP is cleared.

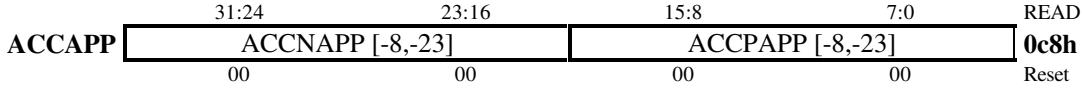


Figure 8-92 ACCAPP - Accuracy Application Register

This 32-bit register holds a 16 bit value from IANBUS and a 16 bit value from IAPBUS latched on the active edge of signal TSAPP[0]. Upon reset, ACCAPP is cleared.

Figure 8-93 MSAPPX - Macro Stamp Application Register X

This memory address can access eighth different macrostamp registers MSAPP1-8. One register out of eight possible must be selected by a write to address 0cch. MSAPP1-8 hold the upper 32 bits of the INTPBUS latched on the active edge of signal TSAPP[1-8]. Bits 24 to 31 represent a control word for the 56-bit timestamp stored in bits 0 to 23 plus register TSAPPX. Macro Stamp holds bits [+31,+8] in NTP-format. Upon reset, all MSAPP1-8 registers are cleared.

Figure 8-94 TSAPPX - Timestamp Application Register X

This memory address can access eight different timestamp registers TSAPP1-8. One register out of eight possible must be selected by a write to address 0cch. TSAPP1-8 hold the lower 32 bits of the INTPBUS latched on the active edge of signal TSAPP[1-8]. TSAPP1-8 holds bits [+7,-24] in NTP-format. Upon reset, all TSAPP1-8 registers are cleared.

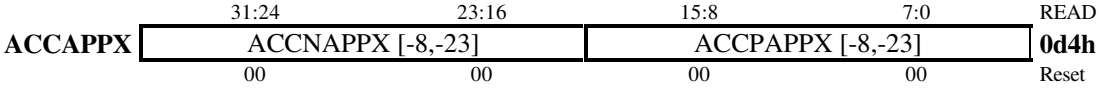


Figure 8-95 ACCAPPX - Accuracy Application Register X

This memory address can access eight different accuracy registers ACCAPP1-8. One register out of eight possible is selected by a write to address 0cch. ACCAPP1-8 hold a 16-bit accuracystamp from the IANBUS and a 16 bit accuracystamp from the IAPBUS, latched on the active edge of signal TSAPP[1-8]. Upon reset, all ACCAPP1-8 registers are cleared.

8.5.7.2 Application Event Registers

Figure 8-96 DUTYH - Duty-Timer Higher Preload Register

This 32-bit register holds the higher 32 bits of the reference time. It is compared against the INTPBUS and if the value in DUTYH plus DUTYL is larger than or equal to INTPBUS, interrupt INTAPPDUTY is raised. DUTYH holds bits [+31,0] in NTP-format. Upon reset, DUTYH is cleared.

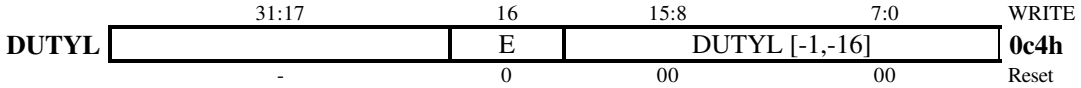


Figure 8-97 DUTYL - Duty-Timer Lower Preload Register

This 17-bit register holds the reference time bits [-1,-16] in bits [15:0] and an enable bit E in bit [16]. The lower 16 bits are compared against the INTPBUS and if the contents in DUTYH plus DUTYL is larger than or equal to the INTPBUS, interrupt INTAPPDUTY is raised. Upon reset, DUTYL is cleared.

8.5.7.3 Application Select Register

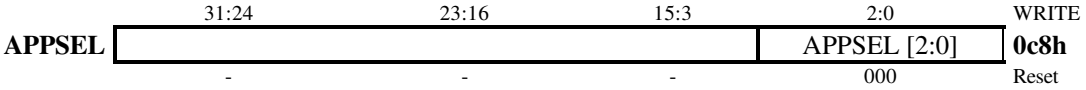


Figure 8-98 APPSEL - Application Register Set Select Register

This 3-bit register selects one out of 8 different application timestamp register sets. Upon reset, APPSEL is cleared and register set 1 is mapped into the address space 0cch, 0d0h and 0d4h.

8.5.7.4 Application Status Register

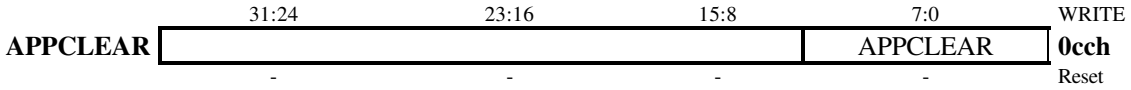


Figure 8-99 APPCLEAR - Clear Application Referenced Status Bit

A write to this memory address (0cch) clears the application referenced status bit SAPPREF in UTCSTAT1. The referenced status bit SAPPREF is set whenever a read access in the application unit occurs.

8.5.8 APU Test Strategy

The basic test concept was elucidated in Chapter 5.1. Testing through the bus interface unit (BIU) is accomplished here. Some of the unit internal control signals are mapped to a unit test port and further to primary output pins. Table 8-32 lists the control signals and the primary output pins they are mapped to. Figure 8-100 illustrates the application unit test port.

APU Signal Name	Testbus Pin #	CSUTIME Pin #	Comments
TSAPPGATE	0	27	application timestamp gate signal
TSAPPSYNC	1	28	reflects the synchronized TSAPP input signal
APPDUTYSYNC	2	29	1 if INTPBUS ≥ DUTY and E = 1
INTAPPDUTY	3	30	application duty-timer interrupt raised
E	4	31	reflects the internal duty-timer enable signal

Table 8-32 Various Observed Control Signals in APU

One input port bit ITESTSEL and one output port bus ITESTBUS[y:0] makes up the test port. Activating ITESTSEL[6] puts the application unit in test mode. Observed internal signals are propagated to primary output pins through the ITESTBUS port and the ITESTBUS bus system.

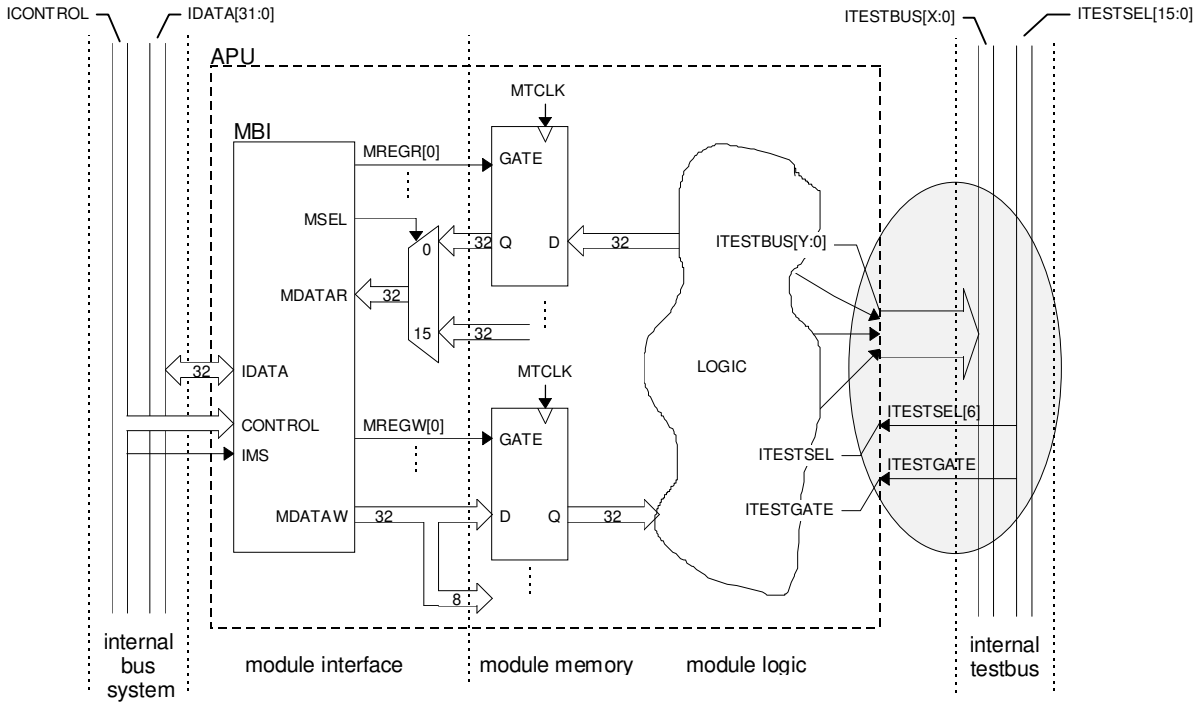


Figure 8-100 APU Test Port Layout

8.5.9 Self-Test Option

The APU has self-test capability. The SYNC-block in Figure 8-86 shows two input pins TESTGATE and TESTSEL. TESTSEL is connected to ITESTSEL[6], cf. Figure 8-100. TESTSEL enables a test mode in the SYNC-block. Whenever signal ITESTGATE carries a gate pulse generated by a HOST write to BTU (Built-In Test Unit) address 084h, HWSNAPGATE carries a gate signal for registers MSAPP, TSAPP and ACCAPP, which latch the current NTP-time as well as accuracy data. This behaviour is equivalent to an active edge on external input TSAPP[x]. The complete APU including the SYNC-block can be tested without taking any external test precautions. While the APU is set into test mode (ITESTSEL[6] = 1), events on signal TSAPP[x] are ignored. All test steps are illustrated in Figure 8-101.

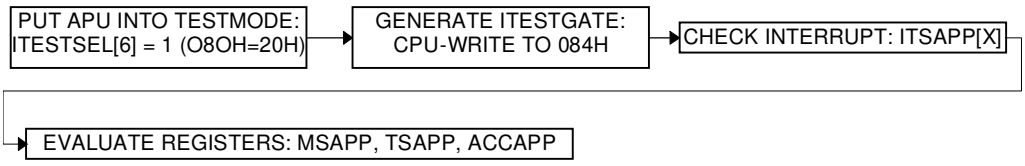


Figure 8-101 Test Sequence for the APU Self-Test

8.6 SNAPSHOT UNIT

Three UTCSU supporting services are implemented in the snapshot unit (SNU). All of them are used for debugging and test purposes.

- ♦ Hardware-Snapshot
- ♦ Software-Snapshot
- ♦ Synchronous Operation

Basically the snapshot unit samples NTP timestamps and accuracy information triggered either by internal or external events. Two different snapshot modes are distinguished. A hardware snapshot to retrieve coarse information and the software snapshot to obtain a detailed view of the current UTCSU internal register image.

The third service is called synchronous UTCSU operation or for short SYNCRUN. It is intended for test purposes and fault-tolerant UTCSU operation to allow several UTCSU chips to start at the same time with the same state. The SNU basic functionality is illustrated in Figure 8-102.

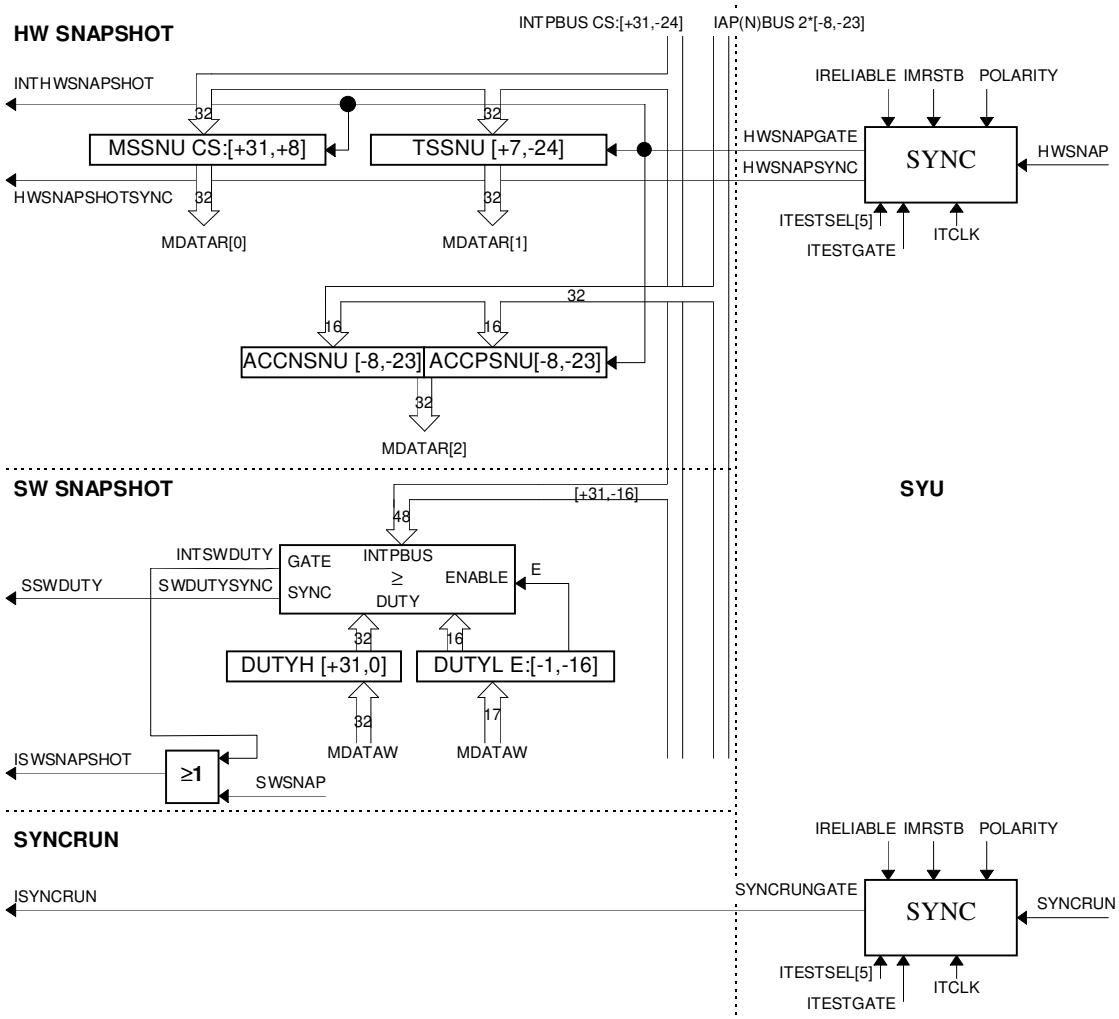


Figure 8-102 Snapshot Unit Data Flow Diagram

8.6.1 Hardware-Snapshot

The hardware snapshot (HWSNAP) is triggered by an external event. An active edge on the HWSNAP input pin (polarity is programmable) captures the internal NTP-Bus and accuracy information into dedicated snapshot registers located in this unit. This debugging aid can be activated any time,

even during normal operation. Although there is only the basic timestamp information delivered to the outside world, this snapshot mode is preferably used for monitoring purposes operating completely asynchronous to the application software.

Upon detecting an active edge on signal HWSNAP, NTP time and accuracy information are latched into registers MSSNU, TSSNU, ACCNSNU and ACCPSNU. An interrupt event is signaled on INTHWSNAPSHOT. The synchronized current signal level on pin HWSNAP can be monitored through signal HWSNAPSYNC. A self-test method for the path HWSNAP to the HOST is explained later on. Figure 8-103 shows timing information.

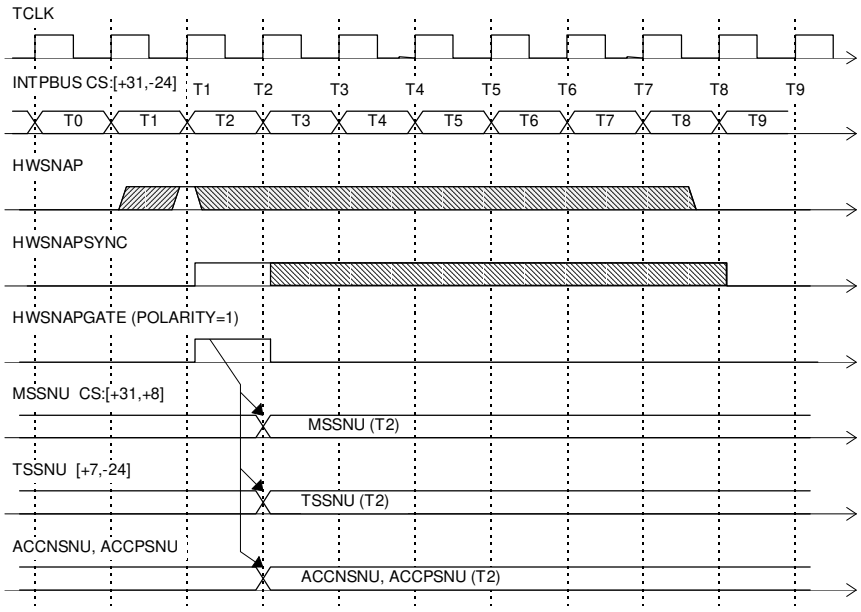


Figure 8-103 Hardware Snapshot Timing Diagram

8.6.2 Software-Snapshot

The software-snapshot provides very detailed information about the current state of the complete chip. It is intended for the debugging phase during system integration. A comprehensive knowledge of the chip is prerequisite.

Executing a software-snapshot copies all chip-wide dynamically changing registers into holding registers. These holding registers are sometimes shared with services used during normal operation, so special care must be taken when using software-snapshots.

Note! Software-snapshot may interfere with normal system operation!

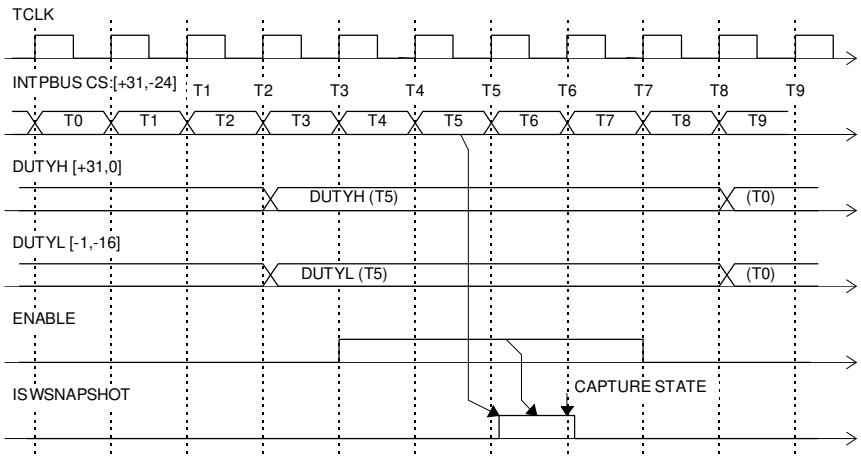


Figure 8-104 Programmed Software-Snapshot Timing Diagram

A software-snapshot can be triggered by two different events. One is called the direct snapshot, where a write operation to memory address 0a0h in the snapshot unit captures the current state of the UTCSU into holding registers. These holding registers are in the LTU: MSGET, TSGET, USGET and NSGET and in the ACU: ALPHAP(N)GET, NALPHAP(N)GET and STATEP(N)GET.

The other one is the programmed snapshot. A duty-timer in the snapshot unit can be programmed with a reference-time (DUTY) to capture the current UTCSU state when INTPBUS ≥ DUTY. The function of the duty-timer was explained in Section 8.3.3. In Figure 8-104 the snapshot duty-timer is programmed with time T5 to capture the UTCSU state at time T6..

8.6.3 Synchronous Operation

An active edge on external input SYNCRUN generates a gate pulse on signal ISYNCRUN, which is connected to MSYNCRUN in LTU and ACU. This gate pulse copies the LTU preload registers MSSET, TSSET, USSET into register NTPTIME and the ACU preload register ALPHASET into registers ALPHP(N). Both units are initialized and start working at the same time. Of course, a couple of UTCSUs can be connected on a single PCB (Printed Circuit Board) as shown in Figure 8-105. All UTCSUs in the set start working at the same time and if they are initialized with the same preload data, they run totally synchronous. One can exploit synchronous operation during the debug phase and together with blocksum computation, elucidated in Section 8.7, for fault tolerant operation of two or three UTCSUs sitting on a single PCB.

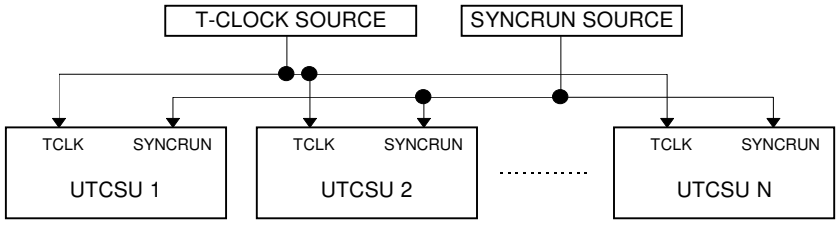


Figure 8-105 Synchronous Operation of Several UTCSU Chips

8.6.4 Configuration SNU

Configuration bits are required to select the proper polarity for the external input signals HWSNAP and SYNCRUN. To select the rising edge on these pins, POLARITY on the SYNC-block must be high and low to select the falling edge. All configuration bits are listed in Table 8-33.

Name	Signal	CONFIG Reg.	Bit #	Functionality
CPOLHWSNAP	POLARITY[0]	UTCCONF1	28	1: rising edge is active edge on HWSNAP 0: falling edge is active edge on HWSNAP
CPOLSYNCRUN	POLARITY[1]	UTCCONF1	29	1: rising edge is active edge on SYNCRUN 0: falling edge is active edge on SYNCRUN

Table 8-33 Configuration Bits in Unit Snapshot

8.6.5 Status SNU

All status bits generated and maintained in the snapshot unit are summarized in Table 8-34.

Name	Signal	STATUS Reg.	Bit #	Functionality
SHWSNAP	HWSNAPSHOTSYNC	UTCSTAT1	18	1: HWSNAP = 1 0: HWSNAP = 0
SSWDUTY	SSWDUTY	UTCSTAT1	19	1: INTPBUS ≥ DUTY and E = 1 0: INTPBUS < DUTY or E = 0

Table 8-34 Status Bits in Snapshot Unit

Status Bit SHWSNAP (Status HWSNAP) provides information on the current state of the external input signal HWSNAP. If this bit is one, HWSNAP carries high level.

Status Bit SSWDUTY (Status SoftWare snapshot DUTY-timer) provides information on the current state of the software snapshot duty timer. The duty-timer is enabled and expired, if its one.

8.6.6 Interrupt Sources SNU

All interrupt sources due to events within the snapshot unit are summarized in Table 8-35.

Name	Signal	Interrupt Reg.	Bit #	Active	Functionality
IHWSNAP	INTHWSNAPSHOT	UTCINT2	16	pulse	hardware-snapshot event
ISWSNAP	ISWSNAPSHOT	UTCINT2	17	pulse	software-snapshot event
ISYNCRUN	ISYNCRUN	UTCINT2	18	pulse	restart LTU & ACU operation

Table 8-35 Interrupt Sources in SNU

Interrupt IHWSNAP (Interrupt HardWare SNAPshot) is activated on every active edge of external input signal HWSNAP, which indicates a hardware-snapshot. The interrupt is cleared in ITU.

Interrupt ISWSNAP (Interrupt SoftWare SNAPshot) is activated whenever the duty-timer is enabled and expires or after a write operation to address 0a0h. The interrupt is cleared in ITU.

Interrupt ISYNCRUN (Interrupt SYNCRUN) is activated on every active edge of external input signal SYNCRUN. The interrupt is cleared in ITU.

8.6.7 Register Map

All SNU registers and decoded latch signals are listed in Table 8-36. The table header lists the memory element involved, the number of bits needed for that memory element, the access type (read/write/copy), the memory address, the latch element access width for write operations, the byte enable signals tied to and the multiplexer channel used for read operations. Furthermore the memory element gate signals are listed as well as a field indicating a very dynamic or a more or less static (D/S) memory element, the memory contents after reset and the data bus bit positions joining the register. This is a very agglomerating view of all SNU registers. A more detailed one is given in the next section explaining all programming details.

Memory Element	Width	R/W/C	Addr	Latch	BE [3:0]	Mux Sel [2:0]	Mux #	Memory Element Gate Signal	D/S	Res	Data Bus Pos.
SWSNAP	-	W	0	-	0	-	-	SWSNAP = MREGW[0] ∧ MBE[0]	S	0	-
DUTYH	32	W	1	8	3:0	-	-	MREGW[1] ∧ MBE[3:0]	S	0	31:0
DUTYL	17	W	2	8	2:0	-	-	MREGW[2] ∧ MBE[2:0]	S	0	16:0
MSSNU	32	R	0	32	x	000	0	HWSNAPGATE	S	0	31:0
TSSNU	32	R	1	32	x	001	1	HWSNAPGATE	S	0	31:0
ACCNSNU &	32	R	2	32	x	010	2	HWSNAPGATE	S	0	31:16
ACCPSNU									S	0	15:0

Table 8-36 Snapshot Unit Register Map

8.6.8 Programmers Model

A very detailed view of all snapshot unit registers is given in this section, intended for hardware level programmers. The list order is increasing memory address.

8.6.8.1 Hardware-Snapshot Registers

Figure 8-106 MSSNU - Macro Stamp Snapshot Register

This 32-bit register holds the upper 32 bits of INTPBUS, latched at the active edge of signal HWSNAP. Bits 24 to 31 represent a control word for the 56-bit timestamp stored in bits 0 to 23 plus register TSSNU. Macro Stamp holds bits [+31,+8] in NTP-format. Upon reset, MSSNU is cleared.

Figure 8-107 TSSNU - Time Stamp Snapshot Register

This 32-bit register holds the lower 32 bits of INTPBUS, latched at the active edge of signal HWSNAP. Timestamp holds bits [+7,-24] in NTP-format. Upon reset, TSSNU is cleared to all 0.

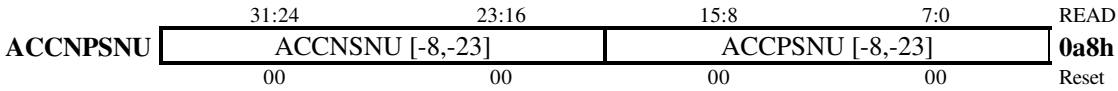


Figure 8-108 ACCNPSNU - Accuracy N and P Register

This 32-bit register holds the 16 bits IANBUS and IAPBUS latched on the active edge of signal HWSNAP. ACCNPSNU holds bits [-8,-23] in NTP-format. Upon reset ACCNPSNU is cleared to 0.

8.6.8.2 Software-Snapshot Registers

Figure 8-109 SWSNAP - Fire Software Snapshot

This memory location has no memory element involved. Writing to this address simply triggers a software-snapshot.

Figure 8-110 DUTYH - Duty-Timer Higher Preload Register

This 32-bit register holds a reference time for the higher 32 bits of the NTP-Bus. It is compared against the INTPBUS and if the contents in DUTYH plus DUTYL is larger than or equivalent to the INTPBUS, interrupt ISWSNAP is raised. DUTYH holds bits [+31,0] in NTP-format. Upon reset, DUTYH is cleared to all 0.

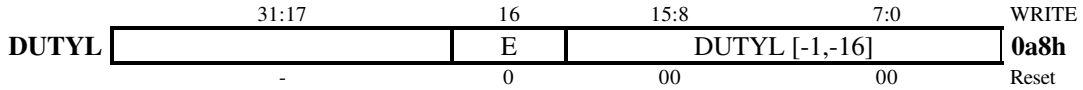


Figure 8-111 DUTYL - Duty-Timer Lower Preload Register

This 17-bit register holds a reference time for the NTP-Bus bits [-1,-16] in bits [15:0] and a high active enable bit E in bit [16]. The lower 16 bits are compared against the INTPBUS and if the contents in DUTYH plus DUTYL is larger than or equivalent to the INTPBUS, interrupt ISWSNAP is raised. Upon reset, DUTYL is cleared to all 0.

8.6.9 SNU Test Strategy

The basic test concept was elucidated in Chapter 5.1. Testing through the bus interface unit (BIU) is accomplished here. Some of the unit internal control signals are mapped to a unit test port and further to primary output pins. Table 8-37 lists the control signals and the primary output pins they are mapped to. Figure 8-112 illustrates the snapshot unit test port.

SNU Signal Name	Testbus Pin #	CSUTIME Pin #	Comments
HWSNAPGATE	0	21	indicates a fired hardware snapshot
HWSNAPSYNC	1	22	reflects the synchronized HWSNAP input signal
SSWDUTY	2	23	1 if INTPBUS ≥ SWDUTY and duty-timer is armed
ISWSNAPSHOT	3	24	reflects the internal software-snapshot signal ISWSNAPSHOT
ISYNCRUN	4	25	reflects the internal SYNCRUN input signal
E	5	26	reflects the internal duty-timer enable signal E

Table 8-37 Various Observed Control Signals in SNU

Observe that for the test port one input bit ITESTSEL and one output bus ITESTBUS[y:0] are required. Setting ITESTSEL high puts the snapshot unit into test mode. Observed signals are propagated to primary output pins through the SNU ITESTBUS port and the ITESTBUS bus system.

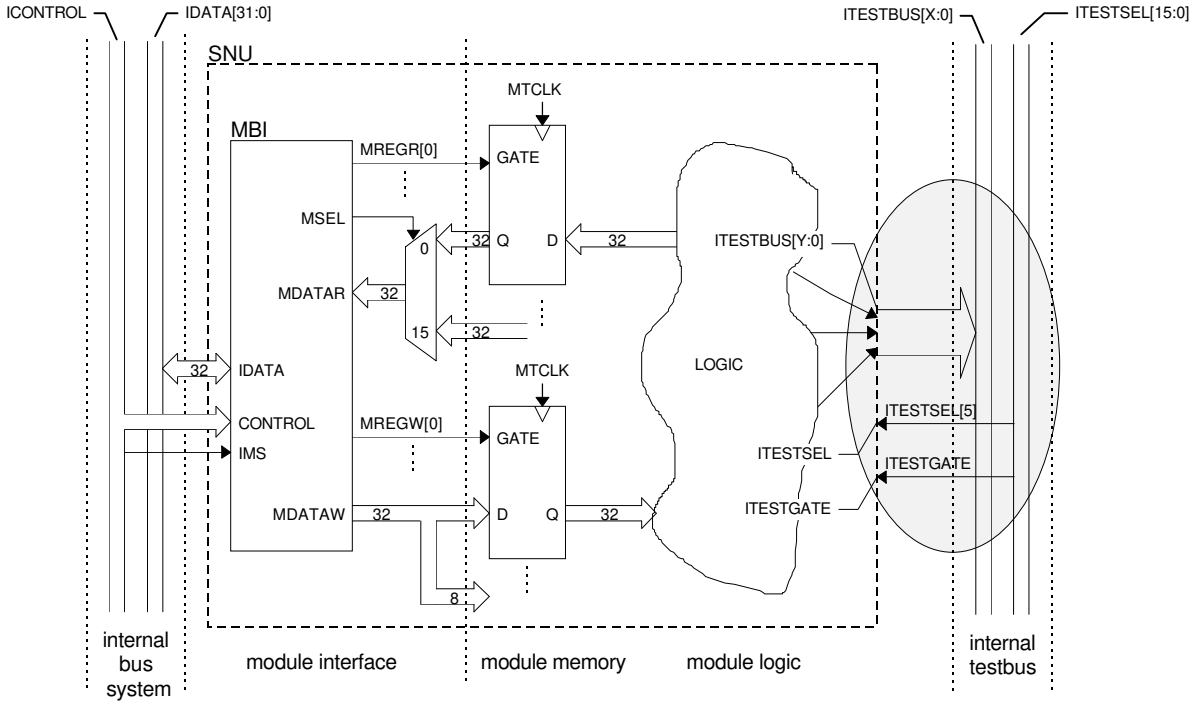


Figure 8-112 SNU Test Port Layout

8.6.10 Self-Test Option

The two SYNC-blocks in Figure 8-102 show two input pins ITESTGATE and ITESTSEL. ITESTSEL is connected to ITESTSEL[5], cf. Figure 8-112. ITESTSEL set high enables a test mode in the SYNC-block. Whenever signal ITESTGATE carries a gate pulse generated by a HOST write operation to BTU (Built-In Test Unit) address 084h, signal HWSNAPGATE carries a gate signal for registers MSSNU, TSSNU, ACCNSNU and ACCPSNU which capture the current NTP-time as well as accuracy information. This behaviour is equivalent to an active edge on external input HWSNAP. The complete SNU, including the SYNC-block, can be tested without taking any external test precautions. While the SNU is set into test mode (ITESTSEL[5] = 1), events on input pins HWSNAP and SYNCRUN are ignored. The test sequence is illustrated in Figure 8-113.

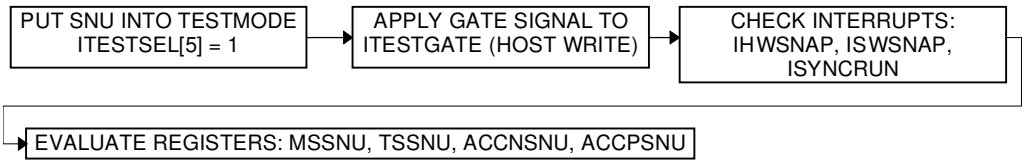


Figure 8-113 Test Sequence for the SNU Self-Test

8.7 BUILT-IN TEST UNIT

The Built-In Test Unit (BTU) provides test and self-test services. Testing on two different levels is supported.

- Test at the unit-level
- Test at the chip-level

Chip-level testing, as it is implemented here, is also known as on-line-testing in the scientific world. We use a true transparent BIST-architecture based on test response compression [YAR93, ABF90, FUJ90].

A test at the unit-level puts all selected units into test mode and runs functional tests with CPU support. Of course, the units under test (UUT) must be set off-line during the test and hence the operation of the UTCSU is cut down.

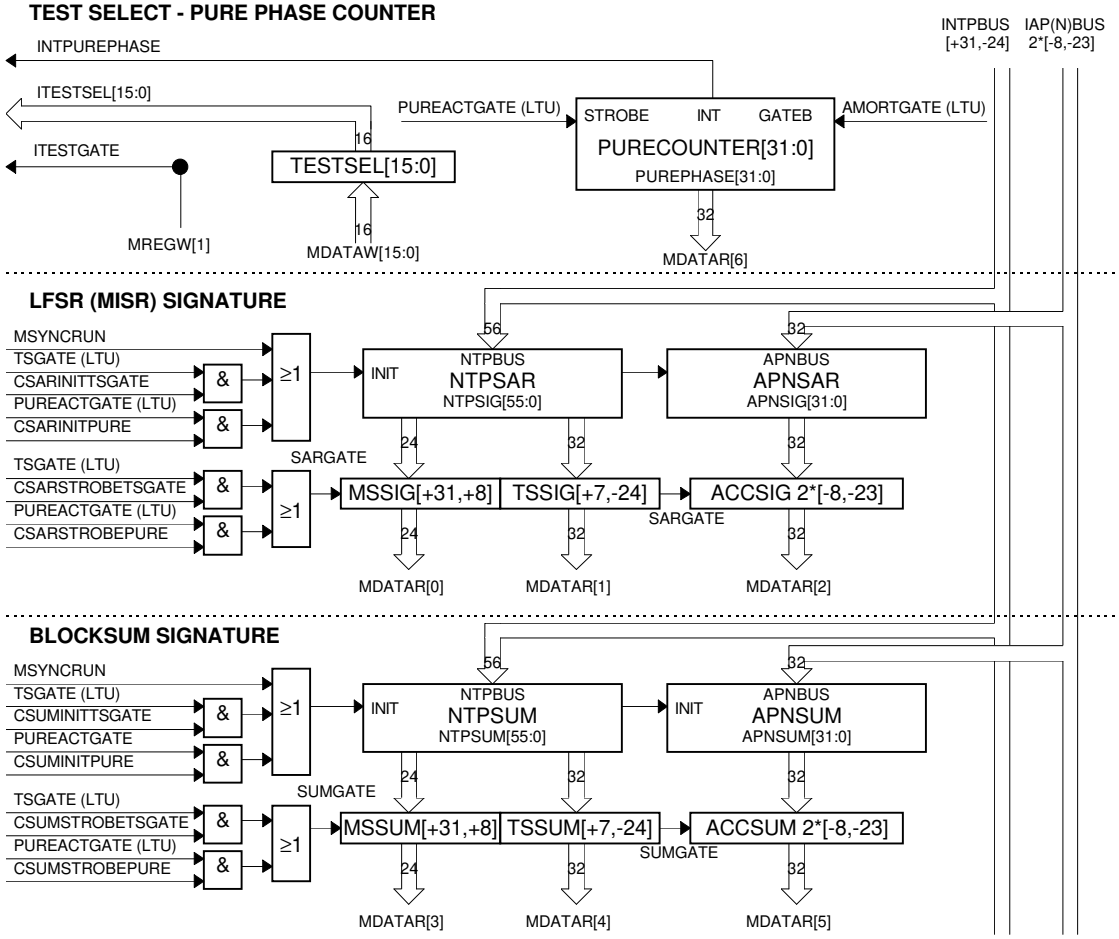


Figure 8-114 Built-In Self-Test Unit Data Flow Diagram

8.7.1 Test at the Unit-Level

The units SSUx, GPUx and APU can be put into test mode by activating the input signal ITESTSEL, which is connected to one of the TESTSEL bus lines according the block diagram in Section 6.1. Setting the related bit in register TESTSEL selects the ITESTSEL[15:0] signals and puts the units into test mode.

The unit internal test logic can simulate external events on input pins without actual interaction from the outside world. Observe that the test logic disconnects the external input pin from the unit input and a simulated test pulse on signal ITESTGATE, generated in the BTU, is applied instead. Hence the overall signal path beginning at the external input pin and ending at the bus interface is tested. The logic, called Test Select, is shown in Figure 8-114,.

8.7.2 Test at the Chip-Level

Here we compute a signature, actually we compute two different signatures, covering all timestamps and accuracystamps on the internal INTPBUS and IAP(N)BUS within a well defined time window. The start time and the end time of the signature computation are programmable. The first is shown in Figure 8-115 and Figure 8-116.

A multiple input linear feedback shift register (MISR) calculates the signature for a set of timestamps and accuracystamps. In our arrangement timestamps are 56 bits wide, disregarding the 8-bit checksum, hence a primitive generator polynom of degree 56 is used. Yarmolik lists primitive polynoms up to degree 300 in the annex of [YAR93]. The listed polynom of degree 56 looks like $P_{56}(x) = x^{56} + x^{22} + x^{21} + x^{1} + 1$ and is illustrated in Figure 8-115. Note that the ⊕ sign denotes the Boolean XOR operator.

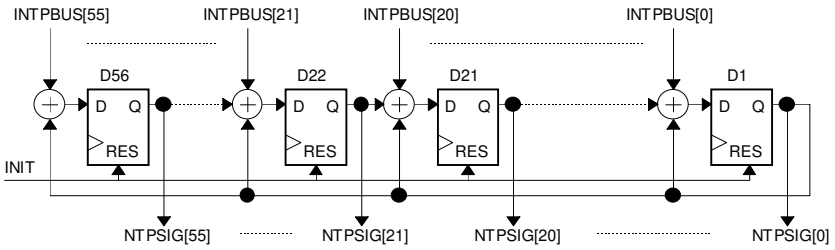


Figure 8-115 Timestamp Signature Analysis $P_{56}(x) = x^{56} + x^{22} + x^{21} + x^{1} + 1$ [YAR93]

A primitive generator polynom of degree 32 can also be found in [YAR93]. It goes like $P_{32}(x) = x^{32} + x^{28} + x^{27} + x^{1} + 1$ and signs the accuracystamps. Figure 8-116 shows $P_{32}(x)$.

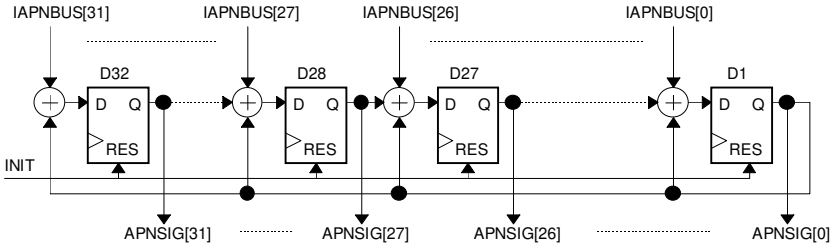


Figure 8-116 Accuracystamp Signature Analysis $P_{32}(x) = x^{32} + x^{28} + x^{27} + x^{1} + 1$ [YAR93]

The second implemented algorithm is a blocksum calculation technique where all timestamps and all accuracystamps are summed up within a specified time window. The data-flow diagram for both timestamp and accuracystamp blocksum calculation is shown in Figure 8-117. Signal INIT in Figure 8-117 clears the NTPSUM and the APNSUM registers and initializes a new time window. Every T-clock a new timestamp resp. accuracystamp is added to these 56-bit resp. 32-bit blocksums.

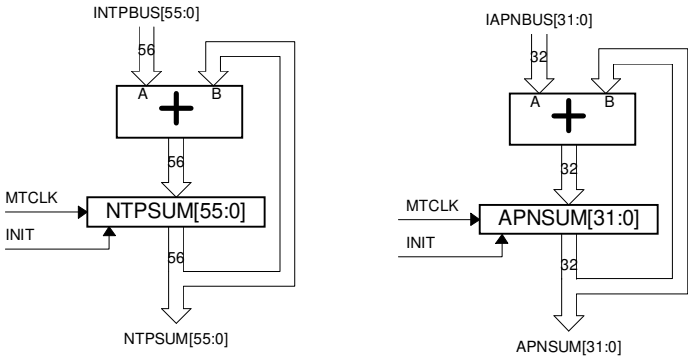


Figure 8-117 Timestamp and Accuracystamp Blocksum Calculation

Observe that blocksum computation is completely independent of the signature computation based on the MISR approach described above.

8.7.3 Signature and Blocksum Computation Timing

The following explanation holds for both signature computation algorithms, MISR and blocksum. Their timing is the same, only the signal and register names differ. Three different sources can start the test time window and two different sources can capture the test results. Signals MSYNCRUNDEL, TSGATE and PUREACTGATE, shown in Figure 8-114 and Figure 8-118 can reset the signature computation blocks, hence start the test window. Either signal TSGATE, for sporadic signature evaluation, or signal PUREACTGATE for periodic signature evaluation at the beginning of every amortization phase, copy the current signature into holding registers MSSIG, TSSIG and APNSIG for MISR-signatures and MSSUM, TSSUM and ACCSUM for blocksum-signatures. Note that both signatures are computed independent by each other, hence start moments for both signatures are independently choosen from the set TSGATE and PUREACTGATE and capture moments are independently choosen from the set MSYNCRUNDEL, TSGATE and PUREACTGATE.

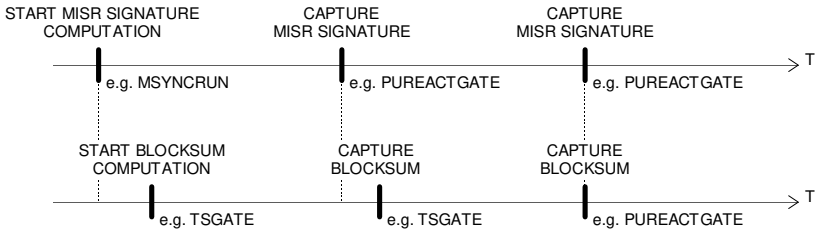


Figure 8-118 Signature Computation Start and Capture Moments

Signal TSGATE is activated on every read from register TSGETL (004h) in LTU. Hence whenever the CPU samples a timestamp, the signature can be captured and evaluated if the corresponding configuration bit CSARSTRTS or CSUMSTRTS is set.

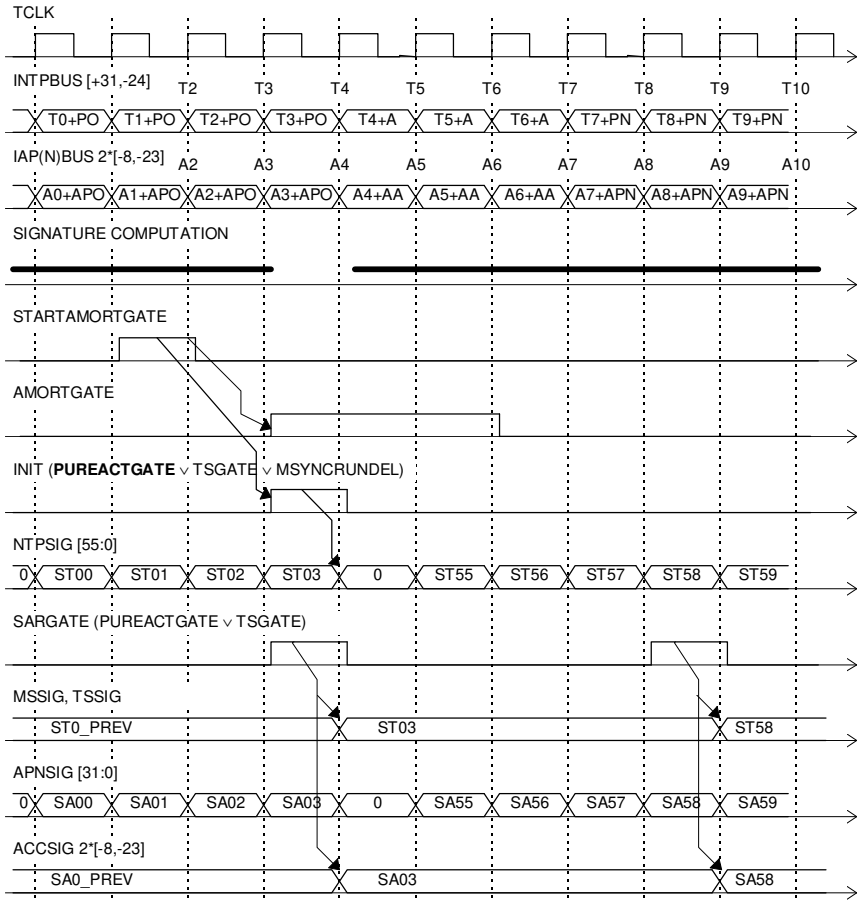


Figure 8-119 Built-In Self-Test: MISR Signature Computation

One possible signature computation timing is shown in Figure 8-119. The abbreviations used are the following. T0 to T9 indicate timestamps sampled at distinct moments T0-T9. PO stands for the old STEPPUREACT register contents in LTU, PN for the new contents, A for the data word stored in STEPAMORT, A0 to A10 are accuracystamps, APO is the old LAMBDAP(N)PUREACT register, APN is the new LAMBDAP(N)PUREACT register contents in the ACU, and AA represents LAMBDAP(N)AMORT. ST00 to ST03 and ST55 to ST59 (ST_{ij}) are the computed signatures between times T0 and T3 resp. T5 and T9 resp. T_i and T_j. Signal MSYNCRUN (see SNU) delayed by one T-clock results in MSYNCRUNDEL.

In Figure 8-119 signal PUREACTGATE opens a test window and either PUREACTGATE or TSGATE strobe the test results. The bold black line (row #4) in Figure 8-119, from now on denoted as test-line, illustrates the test window. Running the signature and blocksum computation in a cyclic fashion along with amortization cycles shows a short interruption in the test-line, which indicates a missed timestamp at the end of a test window.

Signatures and blocksums are sampled into holding registers MSSIG, TSSIG and ACCSIG, triggered by a gate pulse on SARGATE resp. SUMGATE. The CPU can read these registers and evaluate the results. The same flow works with blocksum calculation but for signal and register names SARxxx use SUMxxx and instead of xxxSIG use xxxSUM.

CSAR INIT TSGATE	CSAR INIT PURE	CSAR STROBE TSGATE	CSAR STROBE PURE	Signature Start Action	Signature Capture Action
0	0	0	0	not useful	
1	0	0	0	not useful	
0	1	0	0	not useful	
1	1	0	0	not useful	
0	0	1	0	MSYNCRUNDEL	read TSGETL
1	0	1	0	read TSGETL	read TSGETL
0	1	1	0	start of amortization phase	read TSGETL
1	1	1	0	read TSGETL, start of amort. ph.	read TSGETL
0	0	0	1	MSYNCRUNDEL	start of amortization phase
1	0	0	1	read TSGETL	start of amortization phase
0	1	0	1	start of amortization phase	start of amortization phase
1	1	0	1	read TSGETL, start of amort. ph.	start of amortization phase
0	0	1	1	MSYNCRUNDEL	read TSGETL + start amort
1	0	1	1	read TSGETL	read TSGETL + start amort
0	1	1	1	start of amortization phase	read TSGETL + start amort
1	1	1	1	read TSGETL, start of amort. ph.	read TSGETL + start amort

Table 8-38 Test Configurations for Signature Calculation

Table 8-38 lists all possible configurations how to adjust the start and end times for a signature calculation window. The same table can be used for blocksum calculation if shortcuts SAR are replaced by SUM.

8.7.4 Pure Phase Counter

The length of every amortization phase is determined by the amortization timer, but the length of the pure phase is not exactly known. In order to compute the signatures in a CPU in parallel to the calculations in the UTCSU, the CPU demands the exact length of the pure phase. Hence a pure phase counter counts the number of T-clock ticks while signal AMORTGATE is low (during pure phase). The number is sampled into register PURECOUNTER and the counter is cleared with the rising edge of signal AMORTGATE indicating the end of the pure phase.

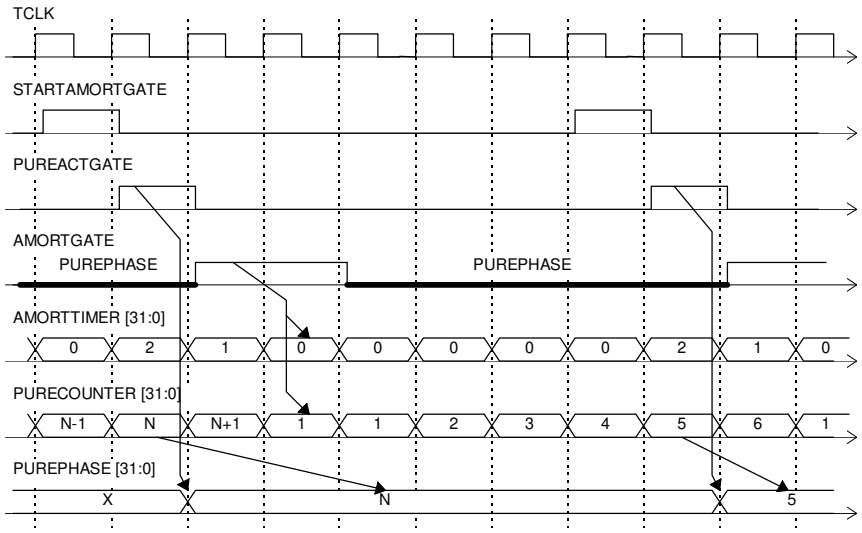


Figure 8-120 Pure Phase Counter Timing Diagram

For timing information see Figure 8-120. If the pure phase lasts longer than 2^32 T-clock ticks which is almost 430 s with a 10 MHz T-clock signal, then the pure phase counter overrun interrupt IPUREPH is raised as shown in the interrupt timing diagram in Figure 8-121.

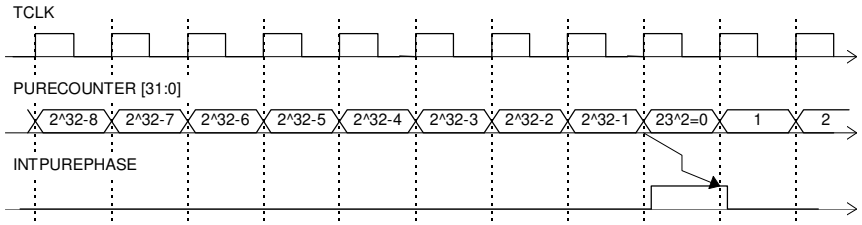


Figure 8-121 Pure Phase Counter Interrupt Timing Diagram

8.7.5 Configuration BTU

Configuration bits are required to select the events for start and end of the checksum calculation window. All configuration bits are listed in Table 8-39.

Name	Signal	CONFIG Reg.	Bit #	Functionality
CSARINTS	CSARINITTSGATE	UTCCONF1	16	1: signature starts with read TSGETL
CSARINPU	CSARINITPURE	UTCCONF1	17	1: signature starts with start amortization
CSARSTRTS	CSARSTROBETSGATE	UTCCONF1	18	1: signature captures with read TSGETL
CSARSTRPU	CSARSTROBEPURE	UTCCONF1	19	1: signature captures with amortization
CSUMINTS	CSUMINITTSGATE	UTCCONF1	20	1: sum starts with read TSGETL
CSUMINPU	CSUMINITPURE	UTCCONF1	21	1: sum starts with start amortization
CSUMSTRTS	CSUMSTROBETSGATE	UTCCONF1	22	1: sum captures with read TSGETL
CSUMSTRPU	CSUMSTROBEPURE	UTCCONF1	23	1: sum captures with start of amortization

Table 8-39 Configuration Bits in the BTU

8.7.6 Interrupt Sources BTU

All interrupt sources due to events within the BTU are summarized in Table 8-40.

Name	Signal	Interrupt Reg.	Bit #	Active	Functionality
IPUREPH	INTPUREPHASE	UTCINT1	12	pulse	overrun in pure phase counter

Table 8-40 Interrupt Source in BTU

Interrupt IPUREPH (Interrupt PURE PHase counter) is activated on every overrun of the 32-bit pure phase counter. The interrupt appears on the interrupt output pin INTA and is cleared by writing bit 12 in register UTCINTCLEAR1 in the interrupt unit.

8.7.7 Register Map

All BTU registers and decoded latch signals are listed in Table 8-41. The table header lists the memory element involved, the number of bits needed for that memory element, the access type (read/write/copy), the memory address, the latch element access width for write operations, the byte enable signals tied to and the multiplexer channel used for read operations. Furthermore the memory element gate signals are listed as well as a field indicating a very dynamic or a more or less static (D/S) memory element, the memory contents after reset and the data bus bit positions joining the register. This is a very agglomerating view of all BTU registers. A more detailed one is given in the next section explaining all programming details.

Memory Element	Width	R/W/C	Addr	Latch	BE [3:0]	Mux Sel [2:0]	Mux #	Memory Element Gate Signal	D/S	Reset	Data Bus Pos.
TESTSEL	16	W	0	8	1:0	-	-	MREGW[0] ∧ MBE[1:0]	S	0	15:0
TESTGATE	-	W	1	-	0	-	-	MREGW[1] ∧ MBE[0]	S	0	-
MSSIG	24	R	0	24	x	000	0	SARGATE	S	0	23:0
TSSIG	32	R	1	32	x	001	1	SARGATE	S	0	31:0
ACCSIG	32	R	2	32	x	010	2	SARGATE	S	0	31:0
MSSUM	24	R	3	24	x	011	3	SUMGATE	S	0	23:0
TSSUM	32	R	4	32	x	100	4	SUMGATE	S	0	31:0
ACCSUM	32	R	5	32	x	101	5	SUMGATE	S	0	31:0
PUREPHASE	32	R	6	32	x	110	6	PUREACTGATE (LTU)	S	0	31:0

Table 8-41 Built-In Test Unit Register Map

8.7.8 Programmers Model

A very detailed view of all BTU registers is given in this section intended for hardware level programmers. The list order is increasing memory address.

8.7.8.1 Unit Test Registers

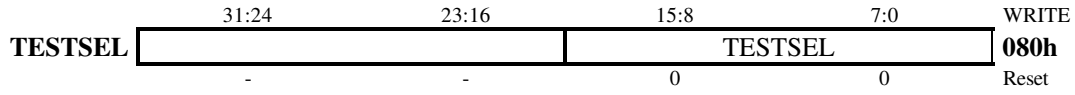


Figure 8-122 TESTSEL - UTCSU Test Select Register

Each bit in this 16-bit register represents a unit in the UTCSU. Bit 0 stands for unit LTU, bit 1 for unit ACU and so forth. It is the same numbering as it is used for the module select lines decoded in the BIU. Setting the bit puts the unit into test mode, clearing the bit disables test mode. Test mode is disabled after reset.

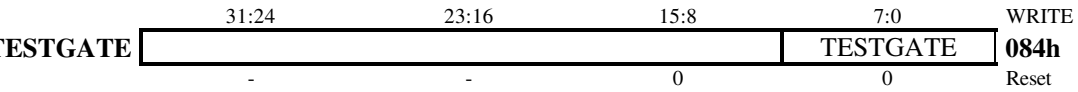


Figure 8-123 TESTGATE - UTCSU Test Gate Register

Writing this register with any number simulates an external event which is distributed to all units. Units in test mode use this pseudo external event to test their data path between the input pin and the bus interface unit. Units not in test mode ignore this simulated event. For more information see the section about self-test in the specific unit descriptions. No memory is used for this address.

8.7.8.2 Signature Analysis Registers

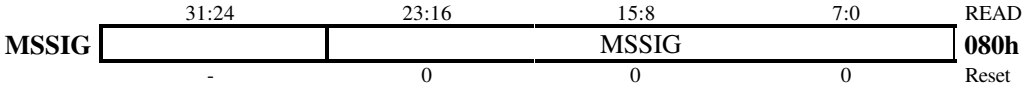


Figure 8-124 MSSIG - Macro Stamp MISR Signature Holding Register

Results from signature calculations in the NTPSAR block are sampled into this register. Sampling is initiated either by reading TSGETL (LTU) or the start of an amortization phase. This depends on the test configuration selected. The upper 24 bits of the signature are stored here.

Figure 8-125 TSSIG - Timestamp MISR Signature Holding Register

Results from signature calculations in the NTPSAR block are sampled into this register. Sampling is initiated either by reading TSGETL (LTU) or the start of an amortization phase. This depends on the test configuration selected. The lower 32 bits of the signature are stored here.

Figure 8-126 ACCSIG - Accuracy Stamp MISR Signature Holding Register

Results from signature calculations in the APNSAR block are sampled into this register. Sampling is initiated either by reading TSGETL (LTU) or the start of an amortization phase. This depends on the test configuration selected. The 16-Bit IAPBUS and the 16-Bit IANBUS are combined into a single 32-Bit signature.

8.7.8.3 Blocksum Registers

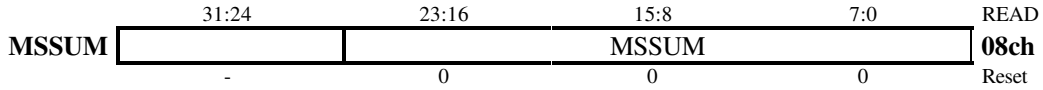


Figure 8-127 MSSUM - Macro Stamp Blocksum Signature Holding Register

Results from blocksum calculation in the NTPSUM-block are captured into this register. Sampling is initiated either by reading TSGETL (LTU) or the start of an amortization phase. This depends on the test configuration selected. The upper 24 bits of the blocksum are stored here.

Figure 8-128 TSSUM - Timestamp Blocksum Signature Holding Register

Results from blocksum calculation in the NTPSUM-block are captured into this register. Sampling is initiated either by reading TSGETL (LTU) or the start of an amortization phase. This depends on the test configuration selected. The lower 32 bits of the blocksum are stored here.

Figure 8-129 ACCSUM - Accuracy Stamp Blocksum Signature Holding Register

Results from blocksum calculation in the APNSUM-block are captured into this register. Sampling is initiated either by reading TSGETL (LTU) or the start of an amortization phase. This de-

pends on the test configuration selected. The 16 bit IAPBUS and the 16-bit IANBUS are combined into a single 32-bit blocksum.

8.7.8.4 Blocksum Support Register

	31:24	23:16	15:8	7:0	READ
PUREPHASE	PUREPHASE				**098h**
	0	0	0	0	Reset

Figure 8-130 PUREPHASE - Pure Phase Counter Holding Register

The length of the latest pure phase in T-clocks is captured into this holding register at the beginning of every amortization phase. Pure phases which last longer than 2^32 ticks raise the interrupt IPUREPH. This register is cleared upon reset.

8.7.9 BTU Test Strategy

The basic test concept was elucidated in Chapter 5.1. In addition, a scan-path is introduced in this unit. Testing through the bus interface unit (BIU) is also accomplished here. Furthermore some of the unit internal control signals are mapped to a unit test port and further to primary output pins. Table 8-42 lists the control signals and the primary output pins they are mapped to. Figure 8-131 illustrates the BTU test port.

BTU Signal Name	Testbus Pin #	CSUTIME Pin #	Comments
ITESTGATE	0	15	Simulated external event pulse
SARINIT	1	16	Init pulse for signature calculation
SARGATE	2	17	Latch pulse for signature evaluation
SUMINIT	3	18	Init pulse for blocksum calculation
SUMGATE	4	19	Latch pulse for blocksum evaluation
INTPUREPHASE	5	20	Interrupt in pure phase counter

Table 8-42 Various Observed Control Signals in BTU

Here the ITESTSEL signal is different from all other units. Both ITESTSEL and ITESTGATE originate in this unit thus ITESTSEL and ITESTGATE are output signals.

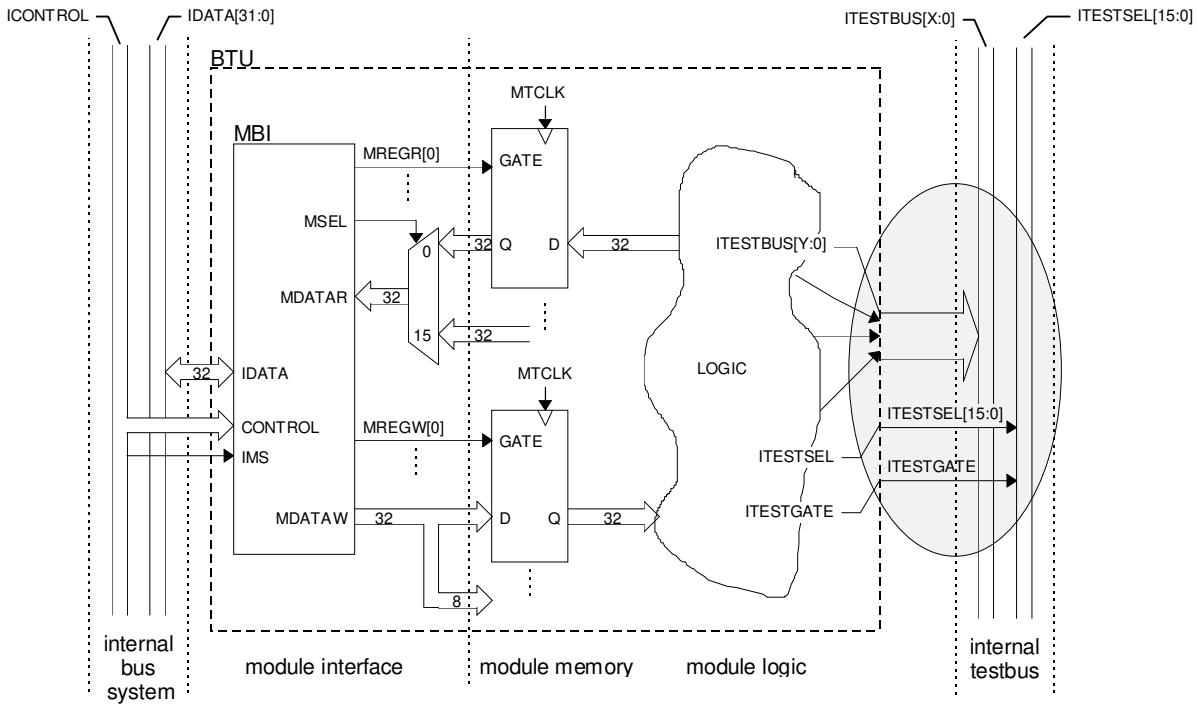


Figure 8-131 BTU Test Port Layout

8.8 INTERRUPT UNIT

Status information, configuration data and interrupt processing are provided in the interrupt unit ITU. 64 status bits, organized in two 32-bit registers UTCSTAT1 and UTCSTAT2, represent the current state of the chip. The relationship between the status bits and the status signals in the different units is given in Table 8-43 and Table 8-44. An identification register holds the current chip version.

In addition 64 configuration bits, also organized in two 32-bit registers UTCCONF1 and UTCCONF2, control above all the polarity of the external input pins of SYU. Table 8-45 resp. Table 8-46 show the bit to pin resp. bit to signal mapping.

Up to 64 interrupts generated in the units are processed here. Programming details can be found in the next subsection. The basic ITU layout is illustrated in Figure 8-132.

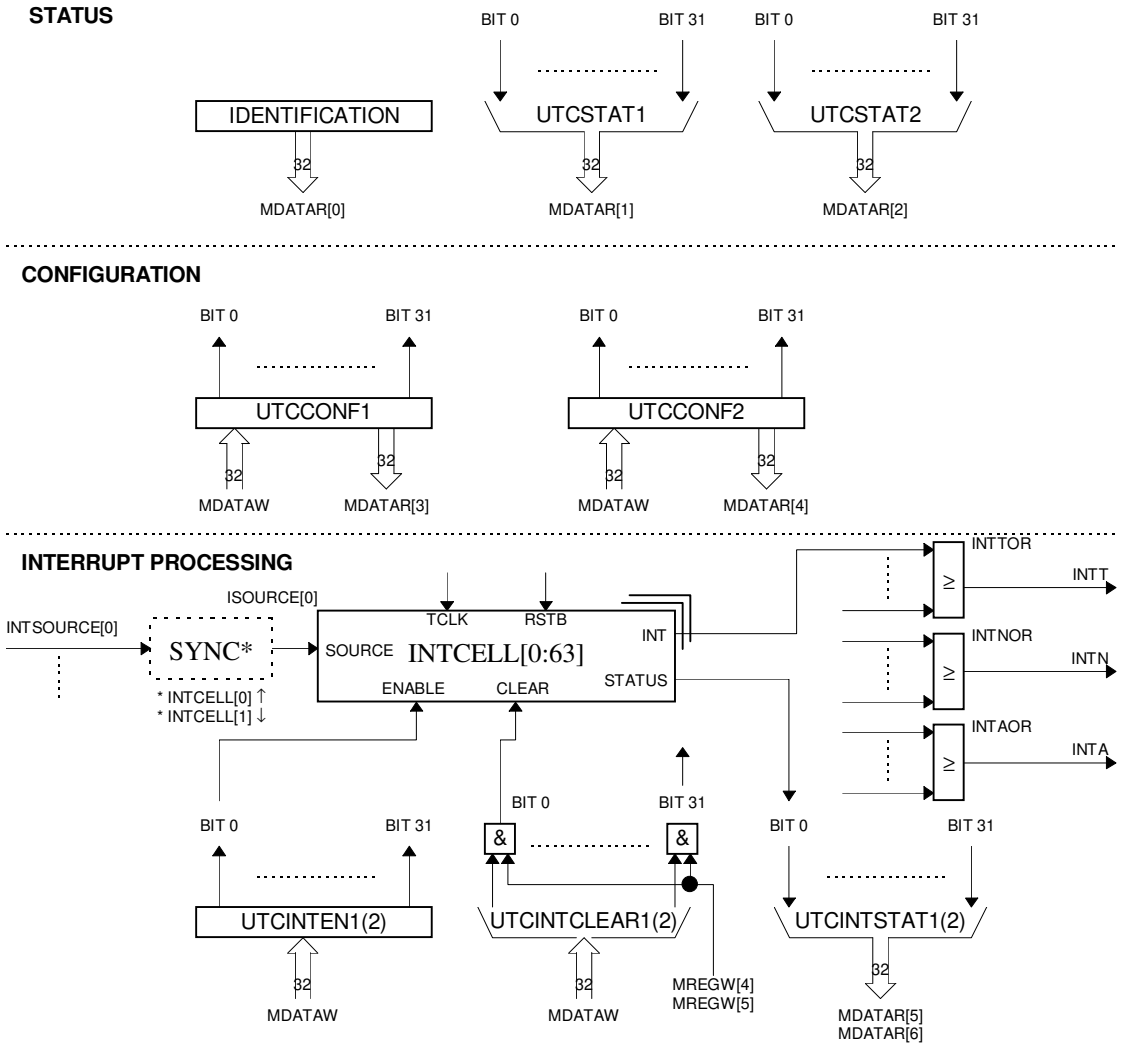


Figure 8-132 Interrupt Unit Data Flow Diagram

8.8.1 Chip Version Identification

A 32-bit read-only chip identification register is provided in the ITU. It is coded as shown in Figure 8-131.

IDENTIFICATION	"I"	"C"	"T"	VERSION	**060h**

Figure 8-133 UTCSU Version Identification Register

8.8.2 Status Information

53 status bits, organized in two 32-bit registers, reflect the current state of the UTCSU. Register UTCSTAT1 is read through address 064h and register UTCSTAT2 through address 068h. Observe that status bits directly reflect the state of the related signals as summarized in Table 8-43 and Table 8-44. Status bits named "-" are reserved for future use.

Bit #	Name	Unit	Signal
0	SCAMA	LTU	AMORTGATE
1	-	LTU	-
2	SAPN	ACU	SALPHAP
3	SANN	ACU	SALPHAN
4	SAPPL	ACU	INTPLALPHAP
5	SAPNL	ACU	INTNLALPHAP
6	SANPL	ACU	INTPLALPHAN
7	SANNL	ACU	INTNLALPHAN
8	SBNDP	ACU	INTBOUNDP
9	SBNDN	ACU	INTBOUNDN
10	SPLLLOCK	CLU	SPLLLOCK
11	-	CLU	-
12	IINTT	ITU	INTT
13	IINTN	ITU	INTN
14	IINTA	ITU	INTA
15-17	-	BTU	-
18	SHWSNAP	SNU	HWSNAPSHOTSYNC
19	SSWDUTY	SNU	SSWDUTY
20	STSAPP	APU	TSAPPSYNC[0]
21-28	STSAPP1-8	APU	TSAPPSYNC[1-8]
29	SAPPDUTY	APU	SAPPDUTY
30	SAPPREF	APU	SAPPREF
31	-	APU	-

Table 8-43 Status Bits in Register UTCSTAT1 (064h)

Bit #	Name	Unit	Signal
0-2	SGPS1PPS1-3	GPU1-3	GPS1PPSSYNC
3-5	SGPSSTAT1-3	GPU1-3	GPSSTATSYNC
6-7	-	GPU1-3	-
8-13	STSRCV1-6	SSU1-6	TSRCVSYNC
14-19	STSXMT1-6	SSU1-6	TSXMTSYNC
20-25	SDUTYA1-6	SSU1-6	SDUTYA
26-31	SDUTYB1-6	SSU1-6	SDUTYB

Table 8-44 Status Bits in Register UTCSTAT2 (068h)

8.8.3 Configuration Data

Configuration bits 0 and 1 in register UTCCONF1 control the leap second insertion and deletion in unit LTU. Bits 16 to 23 control the initialization and the test result evaluation of the different checksum blocks in the built-in test unit (BTU). All other configuration bits adapt the polarity of the external input signals to the internal logic. Table 8-45 and Table 8-46 summarize the configuration signals and bits. UTCCONF1 is written at address 060h and UTCCONF2 at address 064h. Both registers offer a read-back function.

Bit #	Name	Unit	Signal
0	CLEAPSECOND	LTU	CLEAPSECOND
1	CLEAPSINS	LTU	CLEAPSINS
2-3	-	LTU	-
4-7	-	ACU	-
8-11	-	CLU	-
12-15	-	ITU	-
16	CSARINTS	BTU	CSARINITTSGATE
17	CSARINPU	BTU	CSARINITPURE
18	CSARSTRTS	BTU	CSARSTROBETSGATE
19	CSARSTRPU	BTU	CSARSTROBEPURE
20	CSUMINTS	BTU	CSUMINITTSGATE
21	CSUMINPU	BTU	CSUMINITPURE
22	CSUMSTRTS	BTU	CSUMSTROBETSGATE
23	CSUMSTRPU	BTU	CSUMSTROBEPURE
24-27	-	BTU	-
28	CPOLHWSNAP	SNU	POLARITY[0]
29	CPOLSYNCRUN	SNU	POLARITY[1]-
30-31		SNU	-

Table 8-45 Configuration Bits in Register UTCCONF1 (060h)

Bit #	Name	Unit	Signal
0	CPOLAPP	APU	POLARITY[0]
1-8	CPOLAPP1-8	APU	POLARITY[1-8]
9-11	-	APU	-
12-14	CPOLGPS1PPS1-3	GPU1-3	POLARITY[0:2]
15-17	CPOLGPSSTAT1-3	GPU1-3	POLARITY[3:5]
18-19	-	GPU	-
20-25	CPOLRCV1-6	SSU1-6	POLARITY[0:5]
26-31	CPOLXMT1-6	SSU1-6	POLARITY[6:11]

Table 8-46 Configuration Bits in Register UTCCONF2 (064h)

8.8.4 Interrupt Processing

Interrupt processing is based on the interrupt cell illustrated in Figure 8-134. An event, at least one T-clock long, on signal INTSOURCE is stored in the flip-flop and cleared with CLEAR = 1. The interrupt signal INT is driven high as long as the interrupt is not cleared and signal ENABLE = 1. Signal INT drives the according open collector interrupt output pin low. Pending interrupts are marked with signal INTSTATUS being high.

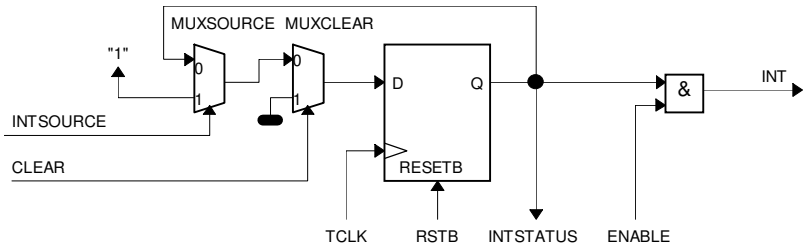


Figure 8-134 Basic Interrupt Cell

Three different registers are required to control the interrupt cell as shown in Figure 8-132. Registers UTCINTEN1 and UTCINTEN2 are connected to signal ENABLE. Writing a one into these register bits enables the interrupt output, whereas writing a zero disables it. Polling register UTCINTSTAT1 or UTCINTSTAT2 returns pending interrupts as the INTSTATUS signal is connected to these registers. An interrupt is cleared if a one is written into the corresponding bit position in registers UTCINTCLEAR1 or UTCINTCLEAR2 while writing a zero has no effect. Interrupt processing is shown in Figure 8-135.

First a CPU has to initialize the interrupt. It clears a pending interrupt and enables the interrupt cell (UTCINTEN1,2[x] = 1). When an interrupt occurs (INT = 1), the CPU has to service the interrupt cell (poll UTCINTSTAT1,2) and clear the pending interrupt in UTCINTCLEAR1,2 as well as execute the interrupt subroutine ISR.

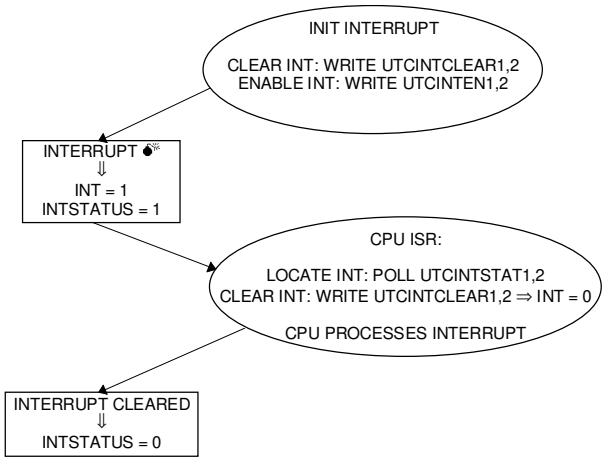


Figure 8-135 CPU Interrupt Processing Data Flow

The following two tables Table 8-47 and Table 8-48 show the relation between the bit positions in the UTCSU interrupt registers and the interrupt sources in the different units. The last column shows the interrupt output pin for the listed interrupt source.

Bit #	Name	Unit	Signal	Signaltype	INT-PIN
0	ICAMS	LTU	AMORTGATE	r-edge	INTT
1	ICAMT	LTU	AMORTGATE	f-edge	INTT
2	INTPO	LTU	INTNTPOVR	pulse	INTT
3	-	LTU	-		-
4	IBNDP	ACU	INTBOUNDP	r-edge	INTT
5	IBNDN	ACU	INTBOUNDN	r-edge	INTT
6	IAPPL	ACU	INTPLALPHAP	r-edge	INTT
7	IAPNL	ACU	INTNLALPHAP	r-edge	INTT
8	IANPL	ACU	INTPLALPHAN	r-edge	INTT
9	IANNL	ACU	INTNLALPHAN	r-edge	INTT
10	IFAILED	CLU	INTTCLKFAILED	perm. low	INTT
11	-	CLU	-		-
12	IPUREPH	BTU	INTPUREPHASE	pulse	INTA
13,14,15	-	BTU	-		-
16	IHWSNAP	SNU	INTHWSNAPSHOT	pulse	INTA
17	ISWSNAP	SNU	ISWSNAPSHOT	pulse	INTA
18	ISYNCRUN	SNU	ISYNCRUN	pulse	INTA
19	-	SNU	-		-
20	ITSAPP	APU	INTTSAPP[0]	pulse	INTA
21	ITSAPP1	APU	INTTSAPP[1]	pulse	INTA
22	ITSAPP2	APU	INTTSAPP[2]	pulse	INTA
23	ITSAPP3	APU	INTTSAPP[3]	pulse	INTA
24	ITSAPP4	APU	INTTSAPP[4]	pulse	INTA
25	ITSAPP5	APU	INTTSAPP[5]	pulse	INTA
26	ITSAPP6	APU	INTTSAPP[6]	pulse	INTA
27	ITSAPP7	APU	INTTSAPP[7]	pulse	INTA
28	ITSAPP8	APU	INTTSAPP[8]	pulse	INTA
29	IAPPDUTY	APU	INTAPPDUTY	pulse	INTA
30, 31	-	APU	-		-

Table 8-47 Interrupt Sources in Register UTCINTEN1 (068h), UTCINTSTAT1 (074h), UTCINTCLEAR1 (070h)

Bit #	Name	Unit	Signal	Signaltype	INT-PIN
0	IGPS1PPS1	GPU1	INTGPS1PPS	pulse	INTN
1	IGPS1PPS2	GPU2	INTGPS1PPS	pulse	INTN
2	IGPS1PPS3	GPU3	INTGPS1PPS	pulse	INTN
3	IGPSSTAT1	GPU1	INTGPSSTAT	pulse	INTN
4	IGPSSTAT2	GPU2	INTGPSSTAT	pulse	INTN
5	IGPSSTAT3	GPU3	INTGPSSTAT	pulse	INTN
6	-	GPU	-	pulse	-
7	-	GPU	-	pulse	-
8-13	ITSRCV1-6	SSU1-6	INTTSRCV	pulse	INTN
14-19	ITSXMT1-6	SSU1-6	INTTSXMT	pulse	INTN
20-25	IDUTYA1-6	SSU1-6	INTDUTYA	pulse	INTT
26-31	IDUTYB1-6	SSU1-6	INTDUTYB	pulse	INTT

Table 8-48 Interrupt Sources in Register UTCINTEN2 (06Ch), UTCINTSTAT2 (078h), UTCINTCLEAR2 (074h)

8.8.5 Register Map

All ITU registers and decoded latch signals are listed in Table 8-49. The table header lists the memory element involved, the number of bits needed for that memory element, the access type (read/write/copy), the memory address, the latch element access width for write operations, the byte enable signals tied to and the multiplexer channel used for read operations. Furthermore the memory element gate signals are listed as well as a field indicating a very dynamic or a more or less static (D/S) memory element, the memory contents after reset and the data bus bit positions joining the register. This is a very agglomerating view of all ITU registers. A more detailed one is given in the next section explaining all programming details.

Memory Element	Width	R/W/C	Addr	Latch	BE [3:0]	Mux Sel [2:0]	Mux #	Memory Element Gate Signal	D/S	Res et	Data Bus Pos.
UTCCONF1	32	W	0	8	3:0	-	-	MREGW[0] ∧ MBE[3:0]	S	0	31:0
UTCCONF2	32	W	1	8	3:0	-	-	MREGW[1] ∧ MBE[3:0]	S	0	31:0
UTCINTEN1	32	W	2	8	3:0	-	-	MREGW[2] ∧ MBE[3:0]	S	0	31:0
UTCINTEN2	32	W	3	8	3:0	-	-	MREGW[3] ∧ MBE[3:0]	S	0	31:0
UTCINTCLEAR1	32	W	4	8	3:0	-	-	MREGW[4] ∧ MBE[3:0]	S	0	31:0
UTCINTCLEAR2	32	W	5	8	3:0	-	-	MREGW[5] ∧ MBE[3:0]	S	0	31:0
IDENTIFICATION	32	R	0	32	x	000	0	MREGR[0]	S	0	31:0
UTCSTAT1	32	R	1	32	x	001	1	MREGR[1]	S	0	31:0
UTCSTAT2	32	R	2	32	x	010	2	MREGR[2]	S	0	31:0
UTCCONF1	32	R	3	32	x	011	3	MREGR[3]	S	0	31:0
UTCCONF2	32	R	4	32	x	100	4	MREGR[4]	S	0	31:0
UTCINTSTAT1	32	R	5	32	x	101	5	MREGR[5]	S	0	31:0
UTCINTSTAT2	32	R	6	32	x	110	6	MREGR[6]	S	0	31:0

Table 8-49 Interrupt Unit Register Map

8.8.6 Programmers Model

A very detailed view of all Interrupt Unit registers is given in this section intended for hardware level programmers. The list order is increasing memory address.

8.8.6.1 Status Registers

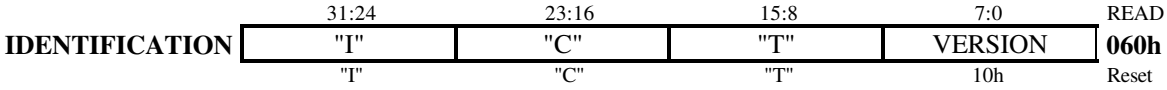


Figure 8-136 IDENTIFICATION - UTCSU Identification Register

This 32-bit register returns a chip identification and a version number. It is hardwired to the value given in the reset state. The upper 24 bits return the string "ICT" while the lower 8 bits are split into two four bit fields. The upper four bits hold the major version number (1 in this implementation) and the lower four bits the minor version number (0 in this implementation).

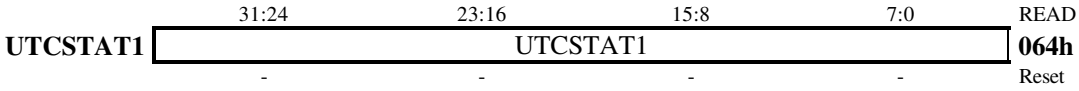


Figure 8-137 UTCSTAT1 - UTCSU Status Register 1

This 32-bit register provides status information on the current state of the chip. For the meaning of these bits see Table 8-43.

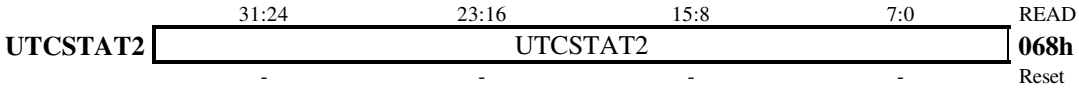


Figure 8-138 UTCSTAT2 - UTCSU Status Register 2

This 32-bit register provides status information on the current state of the chip. For the meaning of these bits see Table 8-44.

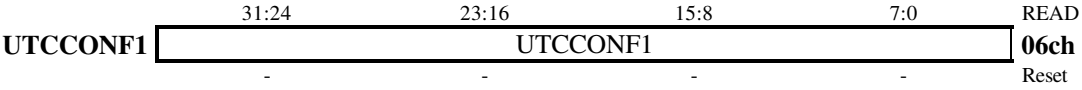


Figure 8-139 UTCCONF1 - UTCSU Configuration Register 1 Read-back

This 32-bit register is a read-back feature for the configuration register UTCCONF1.

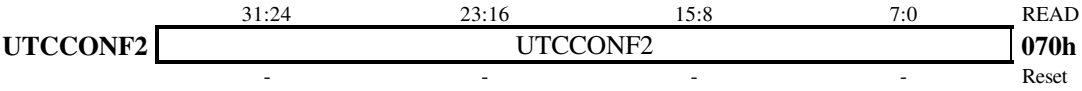


Figure 8-140 UTCCONF2 - UTCSU Configuration Register 2 Read-back

This 32-bit register is a read-back feature for the configuration register UTCCONF2.

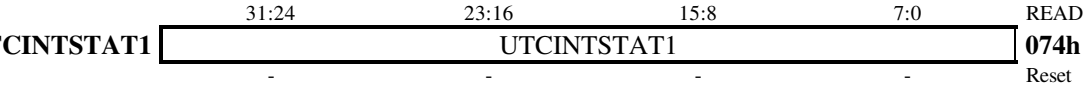


Figure 8-141 UTCINTSTAT1 - UTCSU Interrupt Status Register 1

This 32-bit register provides status information on pending interrupts. For the meaning of these bits see Table 8-47. A one indicates a pending interrupt.

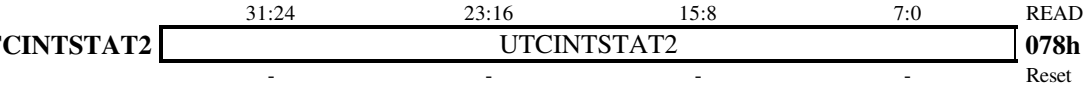


Figure 8-142 UTCINTSTAT2 - UTCSU Interrupt Status Register 2

This 32-bit register provides status information on pending interrupts. For the meaning of these bits see Table 8-48. A one indicates a pending interrupt.

8.8.6.2 Configuration Registers

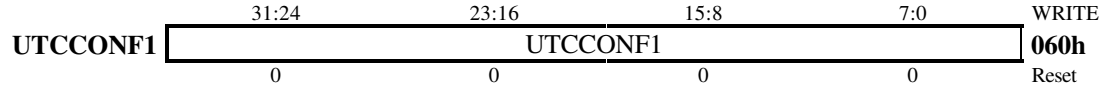


Figure 8-143 UTCCONF1 - UTCSU Configuration Register 1

This 32-bit register holds configuration information. The meaning of these bits is listed in Table 8-45. Bit 0 and bit 1 configure leap second insertion or deletion in LTU according Table 8-50.

Bit 1 (CLEAPSINS) / Bit 0 (CLEAPSECOND)	Action
0 / 0	Normal LTU operation
0 / 1	Leap Second Deletion
1 / 0	Normal LTU operation
1 / 1	Leap Second Insertion

Table 8-50 Leap Second Management in LTU

Bits 16 to 23 control the built-in self-test configuration as specified in the BTU. The remaining bits configure the active edge of the polarity programmable external input pins as listed in Table 8-45. A one in the appropriate bit position sets the rising edge as the active edge and a zero sets the falling edge as the active edge. This register is cleared upon reset.

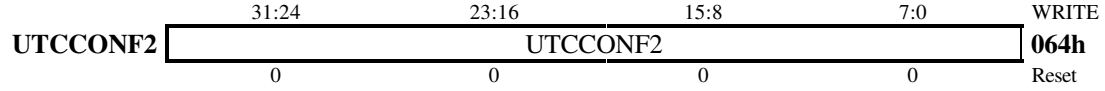


Figure 8-144 UTCCONF2 - UTCSU Configuration Register 2

This 32-bit register holds configuration information. The meaning of these bits is listed in Table 8-46. A one in the appropriate bit position configures the rising edge as the active edge and a zero sets the falling edge as the active edge. This register is cleared upon reset.

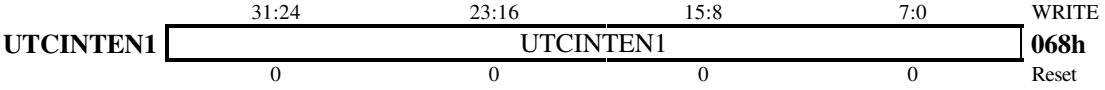


Figure 8-145 UTCINTEN1 - UTCSU Interrupt Enable Register 1

This 32-bit register holds interrupt enable/disable information. The meaning of these bits is listed in Table 8-47. A one in the appropriate bit position enables the corresponding interrupt cell to activate the interrupt output pin. A zero disables the interrupt. This register is cleared during reset.

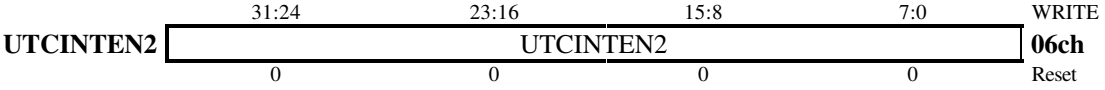


Figure 8-146 UTCINTEN2 - UTCSU Interrupt Enable Register 2

This 32-bit register holds interrupt enable/disable information. The meaning of these bits is listed in Table 8-48. A one in the appropriate bit position enables the corresponding interrupt cell to activate the interrupt output pin. A zero disables the interrupt. This register is cleared during reset.

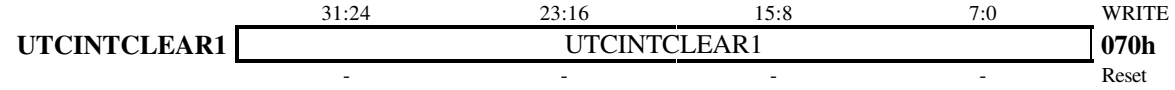


Figure 8-147 UTCINTCLEAR1 - UTCSU Interrupt Clear Register 1

Writing a one to the corresponding bit position at this memory address clears a pending interrupt. Writing a zero has no effect. The meaning of the bits is listed in Table 8-47.

	31:24	23:16	15:8	7:0	WRITE
UTCINTCLEAR2	UTCINTCLEAR2				**074h**
	-	-	-	-	Reset

Figure 8-148 UTCINTCLEAR2 - UTCSU Interrupt Clear Register 2

Writing a one to the corresponding bit position at this memory address clears a pending interrupt. Writing a zero has no effect. The meaning of the bits is listed in Table 8-48.

8.8.7 ITU Test Strategy

The basic test concept was elucidated in Chapter 5.1. Testing through the bus interface unit (BIU) is accomplished here. Furthermore some of the unit internal control signals are mapped to a unit test port and further to primary output pins. Table 8-51 lists the observed signals and the primary output pins they are mapped to. Figure illustrates the interrupt unit test port.

ITU Signal Name	Testbus Pin #	CSUTIME Pin #	Comments
UTCINTEN1[2]	0	10	1: interrupt cell [2] is enabled
UTCINTCLEAR1[2]	1	11	pulse: Clear pending interrupt in cell [2]
STATUS[2]	2	12	1: show pending interrupt in cell [2] (INTPOVR)
INT[2]	3	13	1: interrupt pending in interrupt cell [2]
MREGW[0] $\wedge$ MBE[3:0]	4	14	pulse: write-gate signal for register UTCCONF1

Table 8-51 Various Observed Control Signals in ITU

One input port bit ITESTSEL and one output port bus ITESTBUS[y:0] is required for the test interface. Activating ITESTSEL puts the interrupt unit in test mode. Observed internal signals are propagated to primary output pins through the ITU ITESTBUS port and the ITESTBUS bus system.

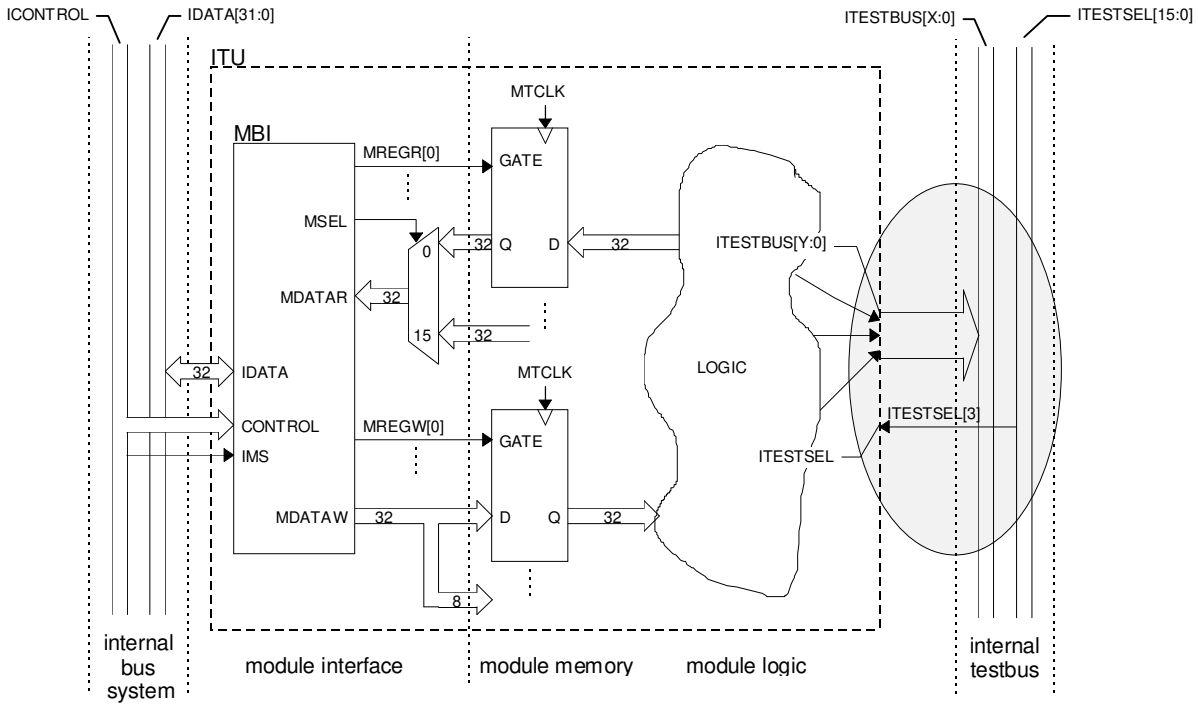


Figure 8-149 ITU Test Port Layout

8.9 NTP-INTERFACE UNIT

The number of obtainable timestamps from the UTCSU is limited by the throughput of the bus interface unit and the CPU connected to this bus interface. Even the fastest CPU needs 3 read accesses, which is equivalent to 6 T-clocks, to read out a complete timestamp (64 bits) together with accuracy information (32 bits).

Hence the UTCSU provides an NTP-interface with 64-bit RENTP timestamp and 32-bit accuracystamp to the UTCSU external world. This allows external processing units to capture and process timestamps independent by a possible UTCSU data bus access.

A significant increase in the total number of processed timestamps can be seen if a simple external logic is connected to the NTP interface to capture timestamps externally. One possible capture logic is shown in Figure 8-150. Two 48-bit edge sensitive registers and two gates are required. On the rising edge of signal CSUTIMEPHASE, the P2 part of the timestamp as defined in Figure 8-151 is latched into P2LATCH and on the falling edge the P1 part is latched into P1LATCH. Here two external blocks EXT1 and EXT2 are shown in Figure 8-150 but any number is possible. To synchronize the external latches to the internal capture registers two D-type Flip-flops are required.

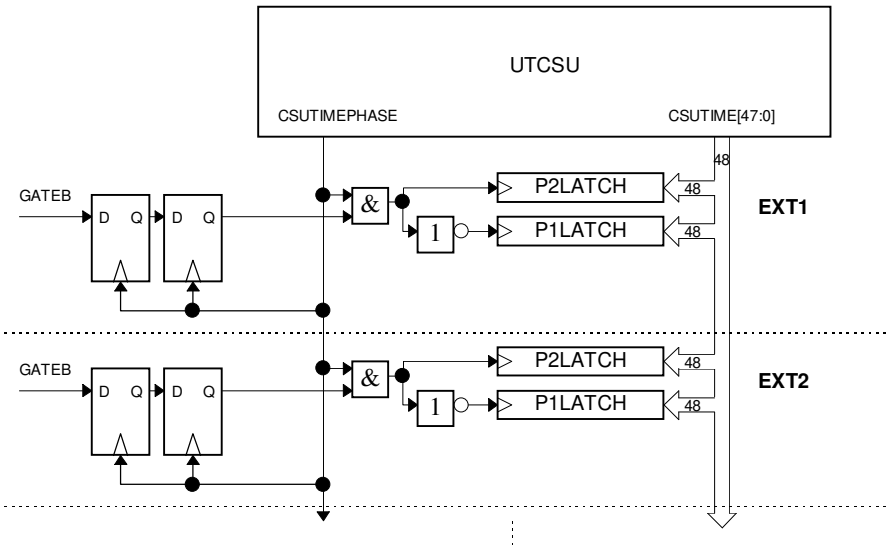


Figure 8-150 NTP-Time Interface Capture Logic

For post chip fabrication test purposes and during the debug phase, the NTP-interface provides a parallel test bus ITESTBUS[47:0] to the outside world. Critical UTCSU internal signals can be observed on this test bus when signal TESTMUX is set high. All observed signals are summarized in Table 8-52. Figure 8-151 shows the data flow in the NTP-interface unit (NTU).

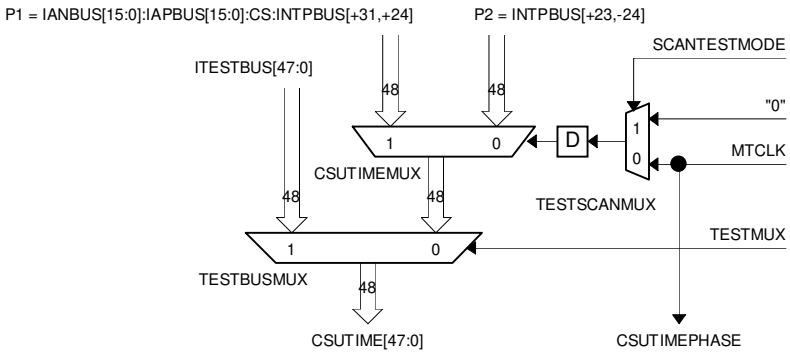


Figure 8-151 NTP-Interface Data Flow Diagram

Timing information for the multiplexed timestamp/accuracystamp/testbus interface is provided in Figure 8-152. Whenever the input pin TESTMUX is high, the internal testbus ITESTBUS[47:0] is

connected to the 48 output pins CSUTIME[47:0] and as soon as TESTMUX goes low, INTPBUS and IAPBUS/IANBUS are connected to these 48 pins. Subsequently right after the rising edge till shortly after the falling edge of signal CSUTIMEPHASE, the 48 CSUTIME pins are connected to P1, while in the remaining phase these pins are connected to P2.

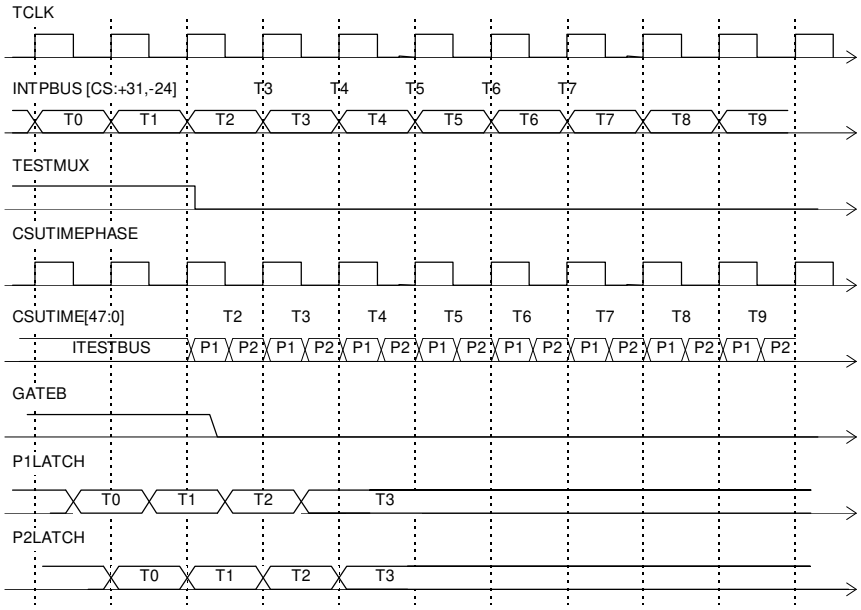


Figure 8-152 NTP-Interface Timing Diagram

Note! CSUTIME output is delayed by one T-clock compared to NTPTIME in LTU.

Some of the critical UTCSU internal signals can be observed through the NTP interface pins. Therefore these signals are mapped to the internal testbus ITESTBUS[47:0] according Table 8-52. Signals denoted with "-" are spare.

ITESTBUS[x]	UNIT	SIGNAL	ITESTBUS[x]	UNIT	SIGNAL
0	LTU	AMORTGATE	24	SNU	ISWSNAPSHOT
1	LTU	PUREACTGATE	25	SNU	ISYNCRUN
2	LTU	TSGATE	26	SNU	E
3	LTU	NTPSET_MUX	27	APU	TSAPPGATE
4	ACU	SALPHAP	28	APU	TSAPPSYNC
5	ACU	SALPHAN	29	APU	SAPPDUTY
6	ACU	SLAMP	30	APU	INTAPPDUTY
7	ACU	SLAMN	31	APU	E
8	ACU	ALPHAPNSET_MUX	32	GPU1	GPS1PPSGATE
9	CLU	IMRSTB	33	GPU1	GPS1PPSSYNC
10	ITU	UTCINTEN1[2]	34	GPU1	GPSSTATSYNC
11	ITU	UTCINTCLEAR1[2]	35	SSU1	TSRCVGATE
12	ITU	STATUS[2]	36	SSU1	TSRCVSYNC
13	ITU	INT[2]	37	SSU1	TSXMTGATE
14	ITU	MREGW[0] $\wedge$ MBE[3:0]	38	SSU1	TSXMTSYNC
15	BTU	ITESTGATE	39	SSU1	SDUTYA
16	BTU	SARINIT	40	SSU1	SDUTYB
17	BTU	SARGATE	41	SSU1	EA
18	BTU	SUMINIT	42	SSU1	EB
19	BTU	SUMGATE	43	CLU	SPLLLOCK
20	BTU	INTPUREPHASE	44	-	-
21	SNU	HWSNAPGATE	45	-	-
22	SNU	HWSNAPSYNC	46	-	-
23	SNU	SSWDUTY	47	-	-

Table 8-52 Critical Signals mapped to ITESTBUS[47:0]

8.10 BUS INTERFACE UNIT

Interfacing a CPU to the UTCSU is a two-step process as illustrated in Figure 8-153. A bus interface unit (BIU) connects the CPU-Bus with an internal bus system (IDATA, ICONTROL) and the module bus interface (MBI) connects the units with the internal bus system.

The BIU controls the interaction between UTCSU internal units and one or several external CPUs and communication coprocessors. Data port width of 8, 16 and 32 bits as well as big endian and little endian CPUs are supported. Different bus sizes are selected via control signals BUS8 and BUS16. In a dedicated reliable mode (RELIABLEHOST = 1), metastability problems on the asynchronous input signal chip select bar (CSB) are improved in a high reliable input synchronizer (SYNC). Microcontrollers with a slow bus interface can slow down bus communication with WAITCYCLE set high.

Observe that the UTCSU consumes 512 bytes address space, equally shared under the 16 UTCSU units, each consuming 32 bytes organized in 8 times 32-bit words. The lower two address bits ADDR[0] and ADDR[1] appear as four byte enable signals BE[3:0] and each of the four signals indicates a valid data byte on the data input bus DATA_INPUT[31:0]. A block diagram is shown in Figure 8-153.

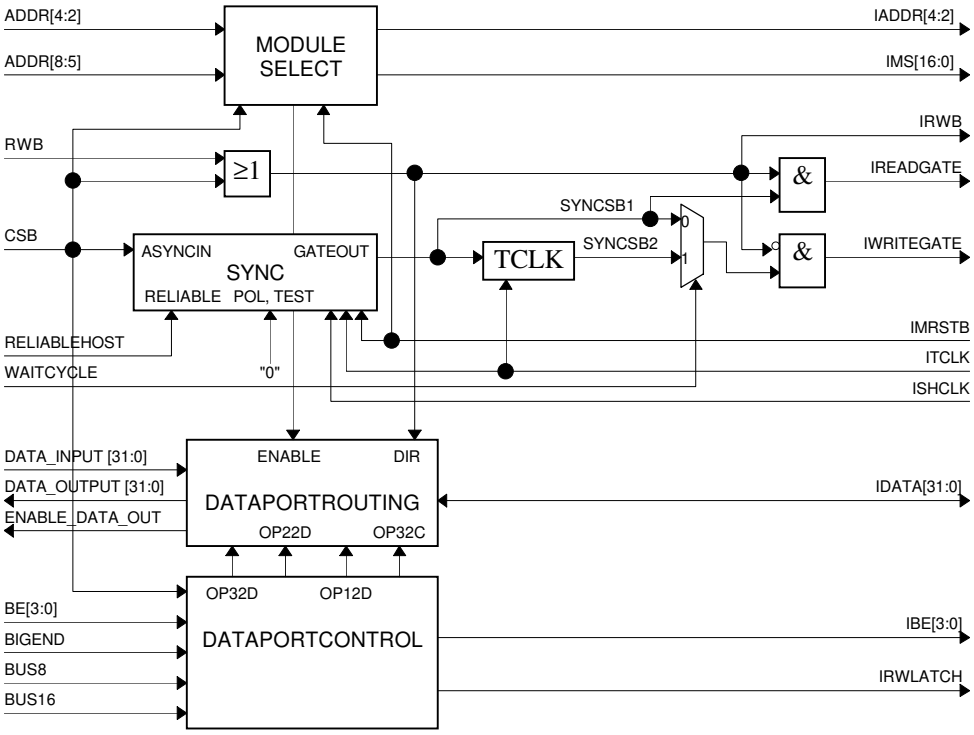


Figure 8-153 Bus Interface Unit (BIU) Block Diagram

The Module Select block in Figure 8-153 decodes address bits 8 to 5 and CSB and selects one out of 17 UTCSU units with the module select bus IMS[16:0]. If CSB = 1, the chip is deselected, signal IMS[16] activates a bus termination block and terminates the chip internal tristate data bus. All three remaining address bits ADDR[4:2] address 8 different 32-bit registers within a unit for read or write operations.

Interfacing to CPUs with data bus sizes other than 32-bit wide requires data port routing in the block DATAPORTROUTING. For detailed information see Section 8.10.4. Different bus sizes and big endian - little endian CPUs are also considered in the data port control block. Details are revealed in Section 8.10.5.

8.10.1 Write Timing

Figure 8-154 shows the write timing for a CPU ⇒ UTCSU access. Reliable mode is deactivated and the fast access mode (WAITCYCLE = 0) is selected. Of course, the timing is the same for different bus sizes.

Signal CSB must satisfy the chip select setup time t_{SCS} which is measured from the falling edge of CSB until the next rising edge of TCLK. If t_{SCS} is not met, then the bus interface enters a wait state for one T-clock and the next upcoming rising edge on TCLK will be the reference timeline. Before chip select CSB can go low, the address signals ADDR[8:2], the read/write signal RWB and the byte enable signals BE[3:0] must be stable for t_{SA} seconds. RWB is low for a write and high during a read operation. As soon as CSB goes low, one out of the 16 internal units is selected with signal IMS[16:0] which is stable after t_{SM}.

The next rising edge on TCLK is the active edge where data is latched. Hence the data input bus DATA_INPUT[31:0] must be stable for the data setup time t_{SD} before the rising edge of TCLK. The data lines can be released after the data hold time t_{HD}. CSB can be deactivated after the chip select hold time t_{HCS}. The address bus and the byte enable signals may be released after the address hold time t_{HA}. Table 8-53 lists the alternating current (AC) timing requirements in tabular form.

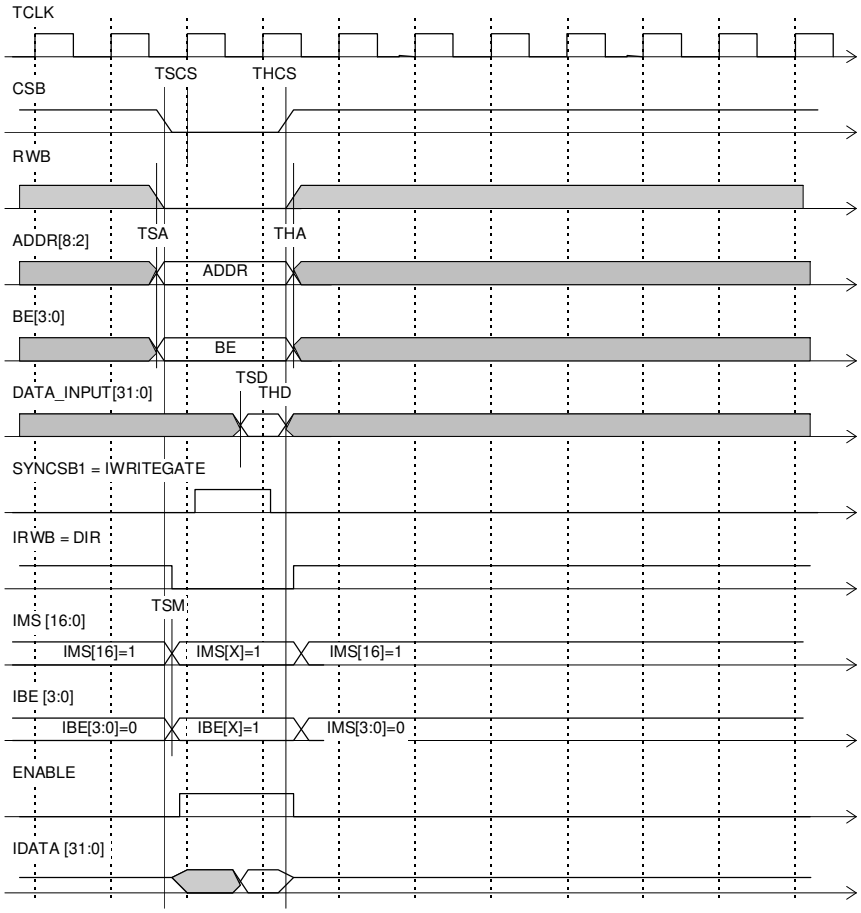


Figure 8-154 BIU Write-Timing

A write pulse IWRITEGATE is derived from the chip select input signal and is used to latch data from the data bus into UTCSU internal registers. The direction signal DIR together with the ENABLE signal control the internal tristate data bus output drivers. If ENABLE is high and DIR is low, as shown in Figure 8-154, then the tristate data bus drivers are enabled towards the internal tristate data.

Description	Symbol	Min.	Max.	Units
System time clock	TCLK	0.5	25	MHz
Chip select input setup time	t_{SCS}	TBC	TBC	ns
Chip select input hold time	t_{HCS}	TBC	TBC	ns
Address input setup time	t_{SA}	TBC	TBC	ns
Address input hold time	t_{HA}	TBC	TBC	ns
Data input setup time	t_{SD}	TBC	TBC	ns
Data input hold time	t_{HD}	TBC	TBC	ns
Module select bus setup time	t_{SM}	TBC	TBC	ns

Table 8-53 AC Timing Requirements for the BIU Write Access

8.10.2 Read Timing

Figure 8-155 shows the read timing for CPU ⇐ UTCSU access. Reliable mode is deactivated and the fast access mode (WAITCYCLE) does not influence the read operation. The timing is the same for different bus-sizes.

Signal CSB must satisfy the chip select setup time t_{SCS}. This time is measured from the falling edge of CSB until the next rising edge of TCLK (T0). If this time is not met then the bus interface enters a wait state for one T-clock and the next upcoming rising edge (T1) on TCLK will be the reference point. Some time t_{SA} before chip select CSB goes low, the address signals ADDR[8:2], the read/write signal RWB and the byte enable signals BE[3:0] must be stable. RWB is high for a read access. As soon as CSB goes low, one out of the 16 internal units is selected with signal IMS[16:0]. IMS is stable after t_{SM}.

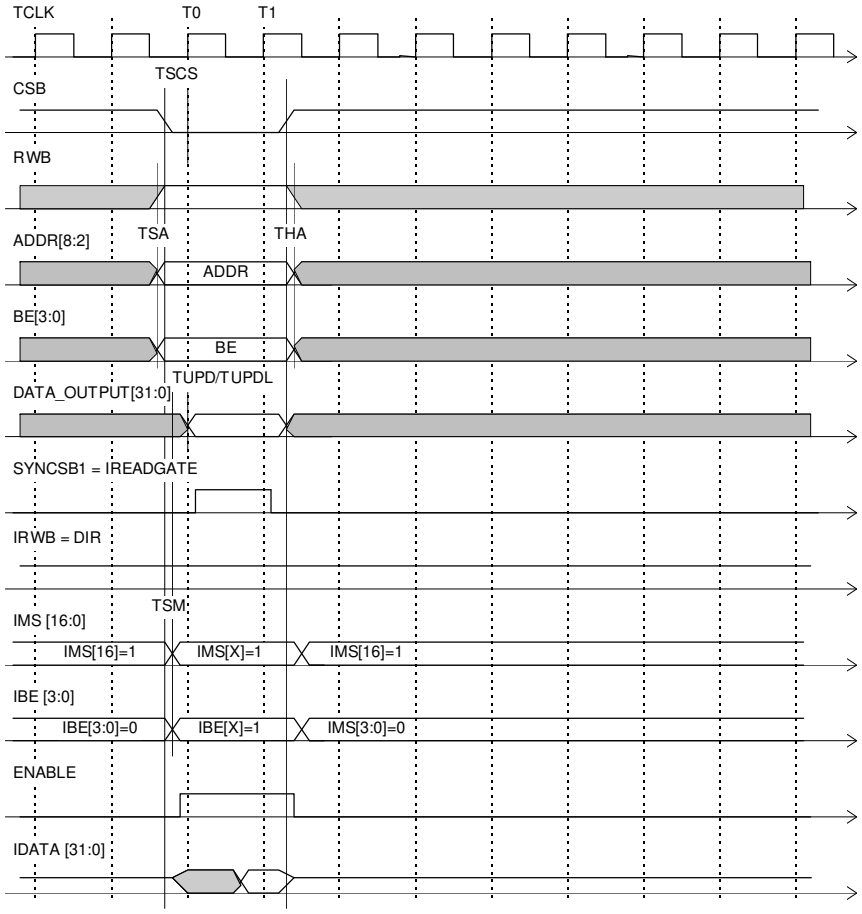


Figure 8-155 BIU Read-Timing

If data are already available in the read register, eg data were latched in a previous cycle, then these data appear, after a short unit propagation delay t_{UPD}, at the data bus output

DATA_OUTPUT[31:0]. If new data have to be latched during this read operation, then the next rising edge (T1) on TCLK is the active edge where data is latched and data appear at the data bus output after a longer delay t_{UPDL} (unit propagation delay latch). The data bus output is stable as long as CSB is low. CSB can be deactivated any time after the CPU has accepted the data. The address bus and the byte enable signals can be released after the address hold time t_{HA}. Table 8-54 lists the AC timing requirements in tabular form.

Description	Symbol	Min.	Max.	Units
System time clock	TCLK	0.5	25	MHz
Chip select input setup time	t_{SCS}	TBC	TBC	ns
Address input setup time	t_{SA}	TBC	TBC	ns
Address input hold time	t_{HA}	TBC	TBC	ns
Data output unit propagation delay	t_{UPD}	TBC	TBC	ns
Data output unit propagation delay latched	t_{UPDL}	TBC	TBC	ns
Module select bus setup time	t_{SM}	TBC	TBC	ns

Table 8-54 AC Timing Requirements for BIU Read Access

A read pulse IREADGATE is derived from the chip select input signal and is used to latch internal data into holding registers. The direction signal DIR together with the ENABLE signal control the tristate data bus output drivers. If ENABLE is high and DIR is high, as shown in Figure 8-154, then the bi-directional data bus pins are set to output mode.

8.10.3 Module Select

The module select block decodes the internal module select signals IMS[16:0] out of address bits 5 to 8, the chip select signal (CSB) and the master reset signal (IMRSTB) according the following equations, where $\langle \ | \ \rangle$ denotes the one out of n decoding property.

$$IMS[16] = CSB \vee \overline{IMRSTB}$$

$$IMS[15:0] = \overline{CSB} \wedge IMRSTB \wedge \left\langle ADDR[8] \middle| ADDR[7] \middle| ADDR[6] \middle| ADDR[5] \right\rangle$$

IMS[16] enables the UTCSU internal bus termination block if no unit is selected and during reset. In addition, the internal address bits 2 to 4 are set low while CSB is high.

$$IADDR[4:2] = \overline{CSB} \wedge ADDR[4:2]$$

To enable the data bus drivers, an ENABLE signal is generated.

$$ENABLE = \overline{CSB} \wedge IMRSTB$$

8.10.4 Data Port Routing

Data port routing is accomplished according to Figure 7-3. The basic routing block is given in Figure 8-156. Four control signals OP12D, OP22D, OP32C and OP32D control the path from the internal data bus to the external data bus pins. Two select signals SEL0 and SEL1 are derived from these signals. The data flow is shown in Figure 8-156.

$$SEL0 = OP12D \vee OP32D$$

$$SEL1 = OP22D \vee OP32D$$

Furthermore two control signals are required to enable and disable the bi-directional I/O pins and the internal tristate driver.

$$ENABLE_DATA_OUT = ENABLE \wedge DIR$$

$$ENABLE_DATA_IN = ENABLE \wedge \overline{DIR}$$

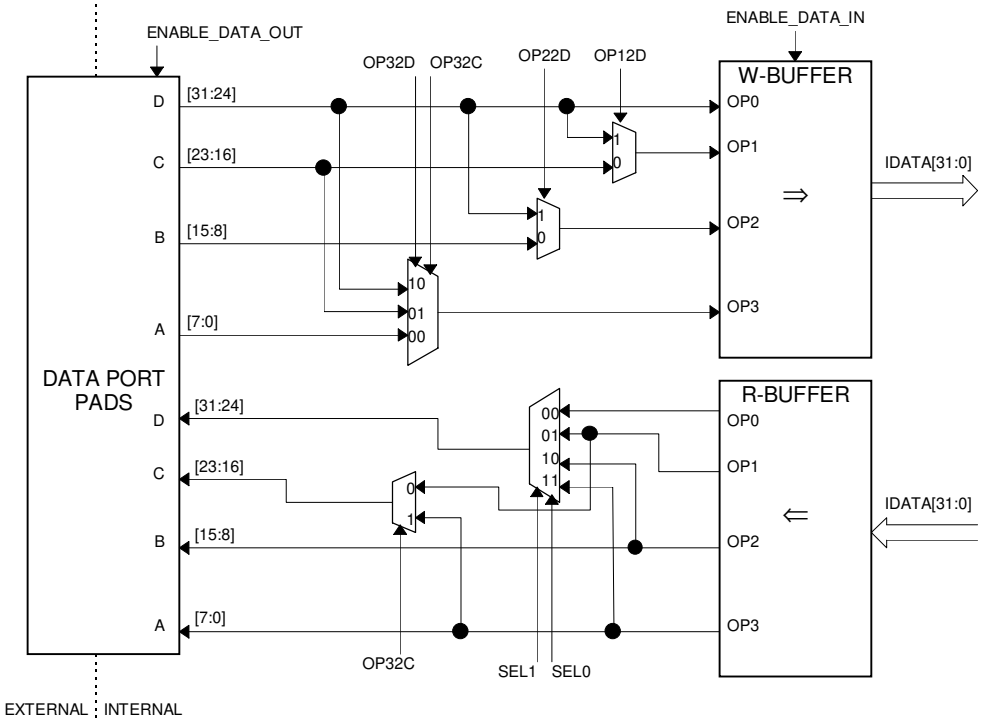


Figure 8-156 Data Port Routing Block

8.10.5 Data Port Control

The data port control block provides the control signals to direct the incoming data stream to the proper internal data bus bits and vice versa. This allows interfacing to CPUs with 8-, 16-, or 32-bit data bus width. Priority logic encodes the incoming bus enable signal (BE[3:0]) to clean out all redundant bit combinations.

Additional control logic handles big endian - little endian CPU read accesses. Therefore control signal IRWLATCH indicates the first phase in a multiple-phase CPU read operation. It is only defined for 32-bit registers and specified as

$$\text{IRWLATCH} = \left(\text{BUS8} \wedge \text{BUS16} \vee \overline{\text{BUS8}} \wedge \overline{\text{BUS16}}\right) \vee$$
$$\left(\text{BUS16} \wedge \text{BE3} \wedge \text{BIGEND} \vee \text{BUS16} \wedge \text{BE0} \wedge \overline{\text{BIGEND}}\right) \vee$$
$$\left(\text{BUS8} \wedge \text{BE3} \wedge \text{BIGEND} \vee \text{BUS8} \wedge \text{BE0} \wedge \overline{\text{BIGEND}}\right)$$

State tables for the priority logic for the bus enable signals and for the control logic for the data port routing control signals are summarized in Table 8-55.

BUS8	BUS16	BE[3:0]	IBE[3:0]	OP32D	OP22D	OP12D	OP32C
1	0	1XXX	1000	0	0	0	0
1	0	01XX	0100	0	0	1	0
1	0	001X	0010	0	1	0	0
1	0	0001	0001	1	0	0	0
0	1	1XXX	1100	0	0	0	0
0	1	X1XX	1100	0	0	0	0
0	1	001X	0011	0	1	0	1
0	1	00X1	0011	0	1	0	1
0	0	XXXX	1111	0	0	0	0
1	1	XXXX	1111	0	0	0	0

Table 8-55 Data Port Routing Control Signals

8.11 CLOCK UNIT

The clock unit (CLU) provides the chip-wide system clock (T-clock) as well as the global reset signal IMRSTB. Either an external TTL-compatible clock signal is fed into the CLU or a cheap quartz crystal, hooked to an on-chip clock oscillator circuitry, sources the internal logic. As far as the physic is concerned, only an upper limit is defined for the external clock, so any clock signal in the range of DC to 25 MHz is accepted. If the on-chip clock oscillator is used, the lowest usable frequency (LUF) is 3 MHz and the maximum usable frequency (MUF) is 20 MHz, cf. Table 8-56.

Clock Source	LUF	MUF
external oscillator	DC	25 MHz
internal oscillator	3 MHz	20 MHz

Table 8-56 Lowest Usable and Maximum Usable Clock Frequencies

The built-in reset circuitry makes a distinction between three reset sources. Of course, their effects are the same and affect the global reset signal IMRSTB. These are:

- Power-On Reset (POR),
- External Reset (EXR) and
- Software Reset (SWR).

The entire core logic is synthesized for an asynchronous reset. Hence only modest pretensions are raised on the signal quality. Power-On reset (POR) brings the chip into reset state after power is coming up. The implementation is ASIC-foundry specific. External reset (EXR) makes use of the primary input pin (RESET) directly fed to the global IMRSTB reset line, cf. UTCSU - Block Diagram (color print).

Software reset (SWR) is raised when the CPU writes byte 0 at memory address 040h. This drives the global reset line IMRSTB low for a single T-clock. Observe that sequential logic is involved with this software reset circuitry which affects the global chip behaviour. In case of scan-path testing, all Flip-flops are connected in a test chain, as elucidated in Section 5.3.1, unpredictable reset conditions could occur. Hence test control logic must be added to disable the software reset during the scan chain shift.

The CLU also hosts the primary input pin RELIABLE which is directly passed to the SYNC unit. RELIABLE enables a highly resistive metastability protection logic on all asynchronous primary input pins serving the SSUx, GPUy, APU and SNU.

8.12 SYNCHRONIZATION UNIT

The Synchronization Unit (SYU) physically comprises all SYNC-cells from the SSUx, GPUy, APU and SNU. A SYNC-cell synchronizes the external asynchronous input signal to the system clock (ITCLK).

8.12.1 Metastability Firmness

The synchronization of asynchronous signals is always subject to the metastability phenomenon. Only a ‘probability’ on the correct behaviour of the synchronizer can be quoted. Numbers are estimated in Section 5.5.6, but these numbers depend on the process technology whose parameters are usually not published.

One can optimize the synchronizer design by comparing different structures, hence only relative numbers have to be rated and the better structure can be chosen. Metastability firmness is a tradeoff between allowed latency and desired meantime between failures (MTBF). The SYNC-cell offers two operating modes. A high reliable mode with increased latency (dual stage design) and a low latency mode with relaxed MTBF (single stage design). Both modes run with the system clock.

8.12.2 Single/Dual Stage Synchronizer

The basic building block is the SYNC-cell as shown in Figure 8-157.

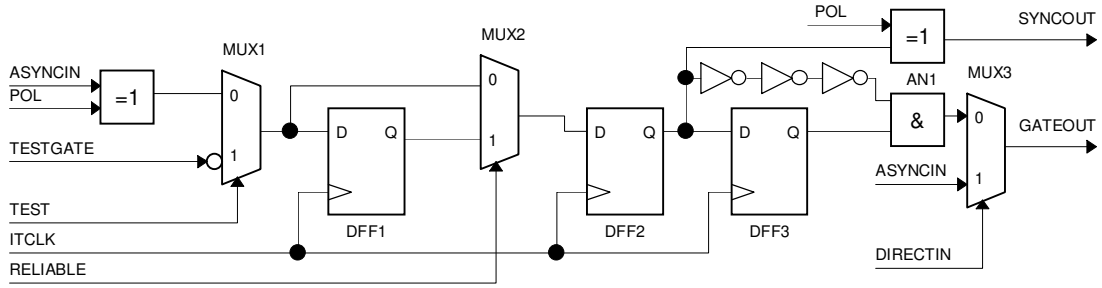


Figure 8-157 SYNC-Cell Schematic View

Signal ASYNCIN, connected to a primary input pin, is synchronized either in a single stage synchronizer (DFF2) or if signal RELIABLE is high, in a dual stage synchronizer (DFF1 + DFF2). Multiplexer MUX2, controlled by signal RELIABLE, selects one of the two paths. The synchronized signal is fed into a pulse generation logic made up of DFF3 and AN1 and is available at the output port GATEOUT with a gate pulse compatible timing. Output SYNCOUT represents the asynchronous input synchronized to the system clock and is used to generate chip internal status information.

The ASYNCIN input is polarity programmable. Multiplexer MUX1 feeds the test signal TESTGATE into the synchronizer logic, if channel 1 is selected. Further information on this test feature can be found in the passage on self-test in the unit descriptions.

8.12.3 Uncertainty Assessment

Timestamps and accuracystamps are latched into their capture registers on the rising edge of the system clock (TCLK), cf. sections on SSU, GPU, SNU, APU and BTU. We use Figure 8-158 to determine upper and lower bounds on the latency in the SYNC-cell specified above.

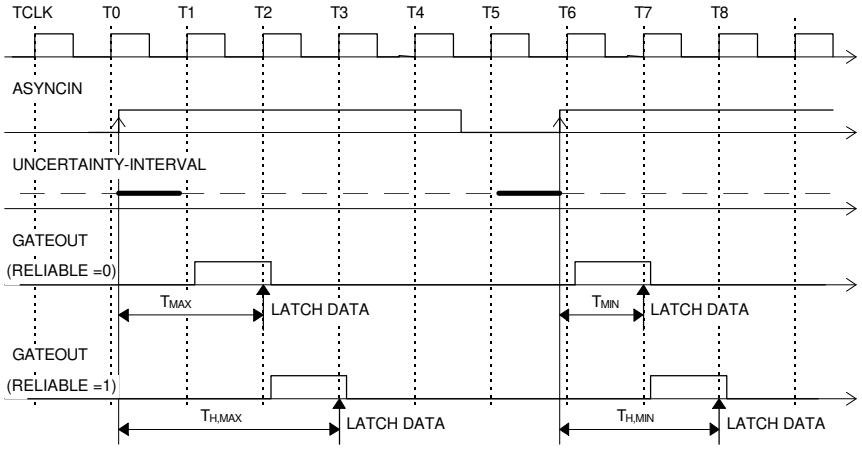


Figure 8-158 Synchronization Uncertainty Assessment

One obtains for the standard reliability mode with RELIABLE = 0

$$t_{\max} = t_{su} + 2 * t_{TCLK} \quad \text{and} \quad t_{\min} = t_{su} + t_{TCLK}, \qquad \text{Equ. 8-13}$$

where t_{max} represents the upper bound, t_{min} the lower bound and t_{su} the flip-flop setup-time. For the *H*igh-reliable mode, where RELIABLE = 1, one obtains

$$t_{H,\max} = t_{su} + 3 * t_{TCLK} \quad \text{and} \quad t_{H,\min} = t_{su} + 2 * t_{TCLK}. \qquad \text{Equ. 8-14}$$

The maximum uncertainty for recording asynchronous external events comes to

$$\max.\text{uncertainty} = t_{\max} - t_{\min} = t_{H,\max} - t_{H,\min} = t_{TCLK}. \qquad \text{Equ. 8-15}$$

CHAPTER 9

9 DESIGN FLOW

Wir sind einzig und allein das, wozu uns die uns umgebenden Dinge machen.
Helvetius, De l'esprit

The UTCSU strictly follows a top-down design flow. It started all out with a functional specification of the UTCSU [SS95] followed by a refined document focusing on the critical blocks in the design (eg high speed adder, synchronizer block, validation of register length, ...). A behavioral VHDL model of the UTCSU paved the way from the written specification to the VHDL RTL description of the UTCSU. After simulation and logic synthesis, test logic was inserted and the back end design was carried out. Currently foundry specific test programs are run on the design, which will be shipped to European Silicon Structure (ES2) for manufacturing.

9.1 FRONT END DESIGN

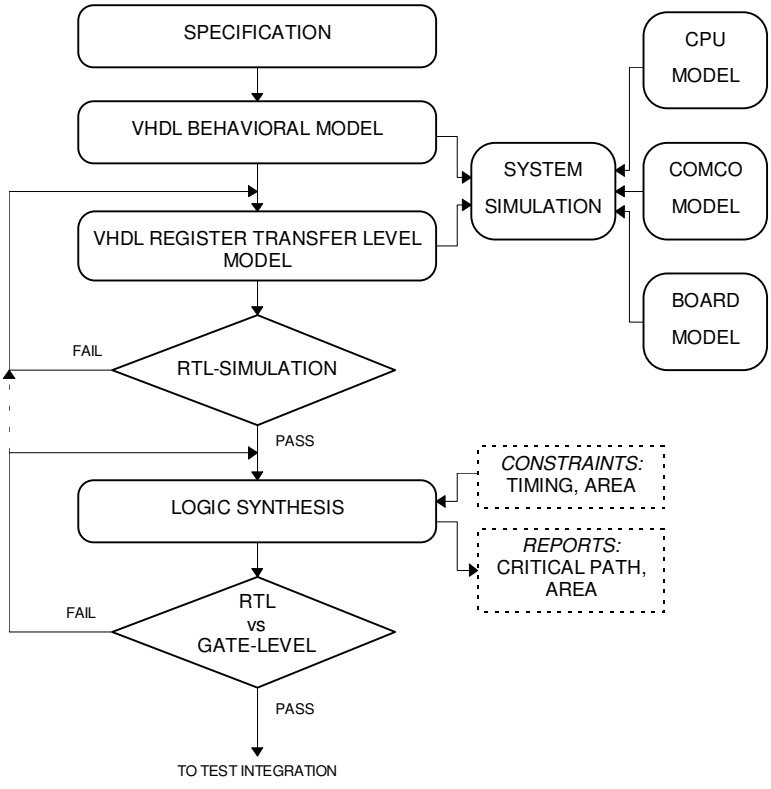


Figure 9-1 Front-End Design Flow

The UTCSU front-end design flow is illustrated in Figure 9-1. Although we don't have a behavioral design compiler installed, a VHDL behavioral model was written to get familiar with the functional specification and to build a bridge between the specification and the register transfer level (RTL) model. Furthermore some sort of system or board level simulation comprising behavioral VHDL models of the CPU, the local area network controller (LANC), the board glue logic, and of course the UTCSU was prepared.

Design compilers (logic synthesis tools) only support a subset of the VHDL constructs, hence transferring the behavioral model into a synthesizeable RTL model has to be done manually. RTL simulation reuses stimuli patterns from the behavioral simulation, but also new patterns. If RTL simulation fails, rework on the design solves the logical failures, cf. Section 9.1.2.

Automatic logic synthesis starts at the register transfer level. Timing and/or area constraints have to be defined in order to meet the system specification in terms of speed and die size. Reports on critical paths and area consumption are generated. Finally a RTL vs. gate-level compare verifies the synthesis results. If this comparison fails, maybe because of too weak constraints or misinterpretation of the VHDL synthesis subset, modifying the RTL description or tighten the constraints sometimes solve the problem, however, occasionally a rework on the specification is inevitable.

9.1.1 System Simulation

At the beginning of the UTCSU design phase, a test bench was created suitable to embed the forthcoming UTCSU behavioral model. Valuable preparatory work was done with the NCSC designed by [HOR94]. However, the Motorola 68030 CPU, the Intel 82596 LAN controller (LANC), the LANC local memory (LANC-MEMORY) as well as the board glue logic had to be modeled in a behavioral VHDL description. For complexity and simulation time reasons, the CPU and LANC functionality were stripped down to support services for board level simulation only [LH95]. Hence it is not possible within this test environment to run a complete clock synchronization algorithm, but the interaction with the UTCSU can be simulated. All main players and the data path are shown in Figure 9-2. Note that a CPU-transceiver decouples the CPU from the local LANC ⇔ UTCSU traffic and a LANC-transceiver decouples the LANC ⇔ LANC-memory traffic from the CPU ⇔ UTCSU traffic to allow parallel operation. This speeds up CPU processing and guarantees the LANC one hundred percent LANC-memory accessibility.

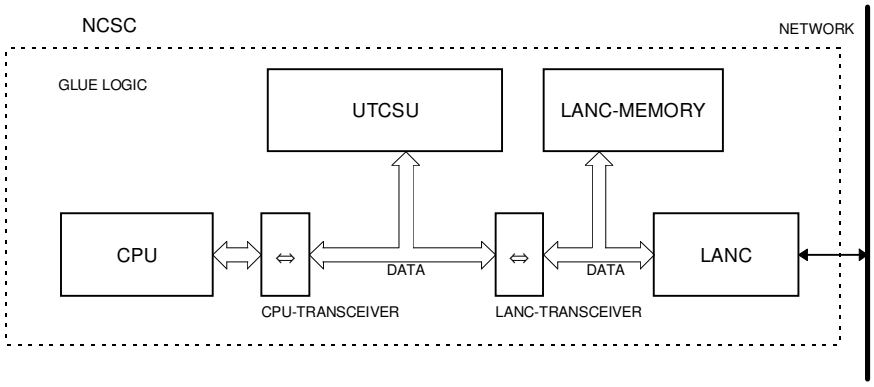


Figure 9-2 Board Level System Simulation

The UTCSU behavioral model is tested within this test bed. CPU resp. LANC read and write operations from and to the UTCSU can be simulated and verified. Interrupt conditions are checked and timing problems on the data bus to the CPU resp. LANC can be detected.

9.1.2 RTL Simulation

As soon as a register transfer level description of a UTCSU unit is available, simulation can be refined to cover all functional details. Martin Horauer implemented a VHDL test bench as shown in Figure 9-3. VHDL processes controlled by STIMULI text files generate signals to interface to the bus interface unit (BIU) and also unit specific control signals (LOCAL SIGNALS). Hence test stimuli are applied to the BIU and passed to the module bus interface (MBI) of the unit under test (UUT). Local signals interconnect the UUT with other units. However, during simulation of a single UUT, local sig-

nals are driven by the VHDL stimuli processes (LOCAL: PROCESS). Test responses are observed in a waveform display and stored on disc for postprocessing (RESPONSE).

Important unit internal control signals are mapped onto a test bus which leads to primary output pins. RTL simulation has to justify the proper selection of test signals mapped onto the test bus.

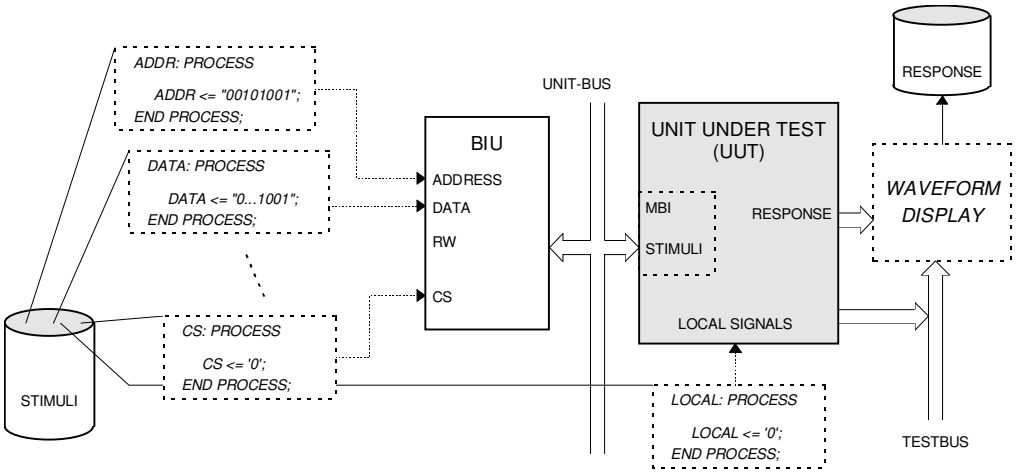


Figure 9-3 RTL Simulation Test Bench

9.1.3 Logic Synthesis

Logic synthesis is the process of automatically transferring a RTL or behavioral description of a logic design into a gate-level netlist. Sometimes logic synthesis is also called logic compile or simply compile. A one step compile of the complete design ends up in hundreds of Megabyte main memory, several CPU-days processing time and bad timing and area behavior. Hence strategies must be developed to limit system resources and to improve the performance of the synthesized chip. The magic formula is 'Divide and Conquer!'. It is pretty straightforward to follow the natural hierarchy inherent to the UTCSU with its 19 more or less separate units. Do a unit compile by unit compile with proper design constraints put at the unit's inputs and outputs leads to a computing resource saving synthesis process. Design constraints put on inputs and outputs are depicted in Figure 9-4 and have the following meaning:

- *Input delay* t_{input} models the timing behavior of the logic sitting in front of the unit's input. More precisely, t_{input} represents the delay between the output of the previous flip-flop (unit *i*) and the input of the unit (*j*) to synthesize. It covers cell delays, routing delays, and unit interconnect delays.
- *Drive strength* specifies the load (capacitance), the unit's input loads the net connected to the input.
- *Output delay* t_{output} models the timing behavior of the logic connected to the unit's output. t_{output} represents the delay between the output of the unit to synthesize and the input of the next flip-flop (unit *k*). It covers cell delays, routing delays, and unit interconnect delays.
- *Load* characterizes the load (capacitance) of the net connected to the unit's output.

How can one determine input and output delays? Let's denote $t_{input,j}$ as the input delay passed as a design constraint for unit *j* to the synthesis tool and let $t_{output,j}$ be the output delay of unit *j*. Furthermore let $t_{in,b}$ denote the delay between the unit *b* input and the first flip-flop input, $t_{out,b}$ the delay between the last flip-flop output and the unit *b* output and t_{ab} the net interconnect delay between units *a* and *b*. $t_{pd,b}$ is the clock to Q propagation delay of the flip-flop in unit *b*. $t_{su,b}$ is the D to clock setup time for the flip-flop in unit *b*. Hence one observes from Figure 9-4 for the input and output delays of unit *j*

$$t_{input,j} = t_{ij} + t_{out,i} + t_{pd,i} \qquad \text{Equ. 9-1}$$

$$t_{output,j} = t_{jk} + t_{in,k} + t_{su,k}. \qquad \text{Equ. 9-2}$$

Figure 9-4 illustrates the terms affecting the input delay of unit *j* and also all terms affecting the output delay of unit *j*. The arrows at the picture lower border depict the timing behaviour between flip-flops. TCLK represents a single clock-tick while N*TCLK represents N clock-ticks. In a single phase signal path, propagation delay between two consecutive flip-flops is always less than one TCLK whereas in a multi-phase signal path propagation delay can exceed TCLK but is bound with M*TCLK. Multi-phase signal paths must be specified as a design constraint in the synthesis tool.

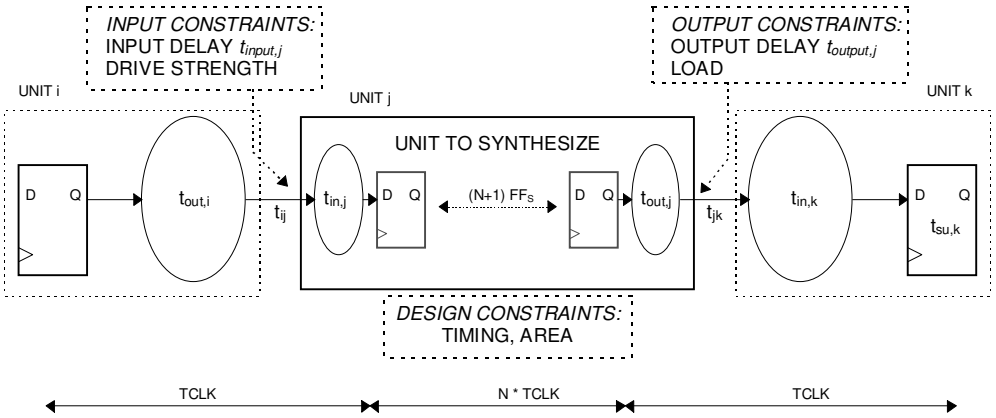


Figure 9-4 Design Synthesis Constraints Put on Inputs and Outputs

Before starting the logic synthesis process, test strategies as the ones shown in Section 9.2 must be deployed. To allow full-scan path insertion later on in the design process, the logic synthesis tool has to be told not to use scan flip-flops during logic synthesis. This technique is sometimes called *smart compile*.

The clock network which feeds all flip-flop clock inputs (4720 in the UTCSU) must not be touched by the synthesis process. Clock distribution and clock tree buffering is organized during place and route in the back end design phase.

A cell area report split up in combinatorial area and noncombinatorial area for each unit is given in Table 9-1. Cell routing area is not considered here. The numbers represent μm^2 for the ES2 0.7 µm CMOS (ECPD07) process as the synthesis target process. LTU and ACU together make up one fourth of the total cell are (25.96 %). The SSU cell area must be multiplied by six for the six SSUs (SSU1-SSU6) and the GPU cell area must be multiplied by three to account for the three GPUs (GPU1-GPU3). A total cell area of approximately 20 mm^2 is reported. Units with dominating sequential logic are the GPUs and the CLU and with dominating combinatorial logic the BIU.

Unit	Combinatorial Area	% Comb. in Unit	Noncombinatorial Area	% Noncomb. in Unit	Total Unit Cell Area	% Unit
LTU	1252947	50.47	1229490	49.53	2482437	12.73
ACU	1397313	54.21	1180158	45.79	2577471	13.23
SSU	515226	47.70	564892	52.30	1080118	5.54
GPU	84679	36.15	149568	63.85	234247	1.20
APU	1399760	41.87	1943016	58.13	3342776	17.15
SNU	294716	45.45	323668	54.55	648384	3.33
BTU	997161	50.97	959044	49.03	1956205	10.04
NTU	66940	100.00	0	0.00	66940	0.34
BIU	98948	80.66	23712	19.34	122660	0.63
ITU	332663	43.58	430646	56.42	763309	3.92
SYU	146566	44.54	182491	55.46	329057	1.69
CLU	8804	31.86	18832	68.14	27636	0.14
BUSTERM	821	5.09	15322	94.91	16143	0.08
TOTAL	*9342032*	*47.94*	*10144435*	*52.06*	*19486467*	*100.00*

Table 9-1 Area Report without Routing Area in μm^2 with 6 SSUs and 3 GPUs for ECPD07

Note! *The BUSTERM block uses simple tristate buffers to terminate the internal tristate data bus in case it is floating. The area report, however, reports 94.91 % non-combinational logic. Hence one observes that tristate buffers are reported as noncombinational logic.*

Table 9-2 includes routing area estimated by the Synopsys *design compiler*. Numbers for the routing area are based on a linear fan-out/wire length model considering the total cell area. The total synthesized core size is reported with 91.28 mm^2.

Unit	Cell Area	% Cells in Unit	Routing Area	% Routing in Unit	Total Unit Area	% Unit
LTU	2482437	18.02	11295193	81.98	13777630	15.09
ACU	2577471	16.31	13220391	83.69	15797862	17.31
SSU	1080118	25.47	3161422	74.53	4241542	4.65
GPU	234247	39.11	364621	60.89	598868	0.66
APU	3342776	18.43	14794529	81.57	18137306	19.87
SNU	648384	30.33	1519460	69.67	2137846	2.34
BTU	1956205	20.13	7762979	79.87	9719186	10.65
NTU	66940	30.98	149164	69.02	216105	0.24
BIU	122660	34.69	230919	65.31	353579	0.39
ITU	763309	25.50	2230231	74.50	2993541	3.28
SYU	329057	40.27	488032	59.73	817090	0.86
CLU	27636	51.12	26416	48.88	54054	0.06
BUSTERM	16143	55.16	13124	44.84	29266	0.03
TOTAL	*19486467*	*21.35*	*71792854*	*78.65*	*91279321*	*100.00*

Table 9-2 Area Report with Estimated Routing Area in μm^2 for ECPD07 (6 SSUs and 3 GPUs)

9.2 TEST INTEGRATION

Test integration ensures proper device testing and operation after chip manufacture and assembly on a PCB. In the design flow, test integration is placed between logic synthesis and place & route. A fault-free gate level netlist is the input into *test compiler*.

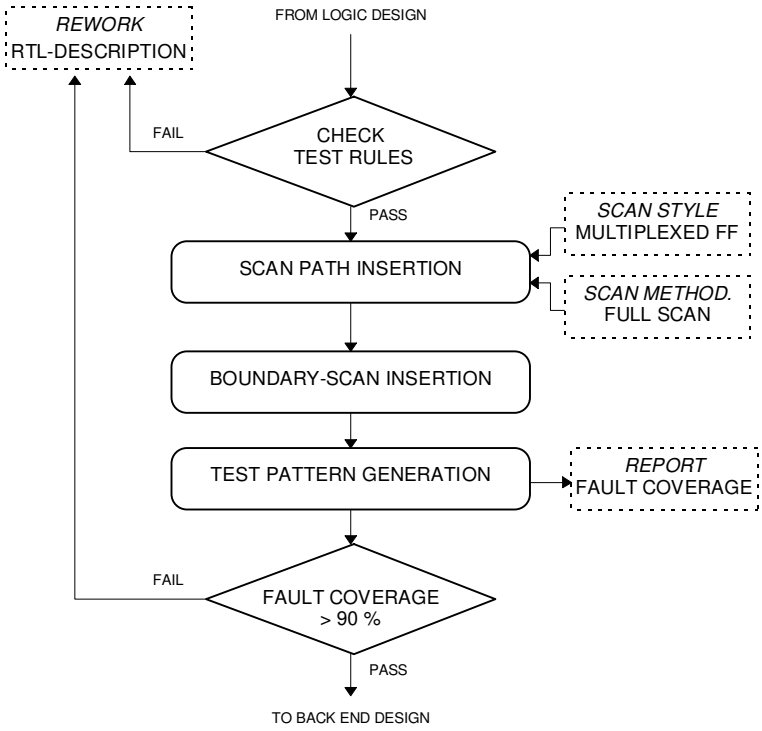


Figure 9-5 Test Integration Flow

First of all testability violations are identified with structural and logical checks. Structural checks are solely based on the topology of the design, while logical checks are based solely on the simulation of the test protocol. Structural checks test for

- *combinational feedback loops*, which introduce internal states to the design that cannot be synchronously controlled. To perform test pattern generation, these feedback loops must be broken.
- *unknown blocks* that are black boxes which cannot propagate fault effects, so these cells, and cells in their local area, are not testable.
- *generic logic* that is logic which is not currently synthesized into the final gate level netlist.
- *multiple-driver nets* which must have a resolution function involved.
- *tristate nets*, to guarantee that tristate nets are never floating during test and only one driver is active. Tristate disabling logic is automatically inserted during test insertion.

Logical checks test for

- *validity of the test clock network*. The test clock network is considered valid if it is controllable from an input port and an active edge on the network can only be generated by applying a pulse to the clock input port.
- *asynchronous pin networks*. Networks driving an asynchronous pin are considered valid if it can be combinationally disabled from an input port, or a combination of input ports (eg a global reset line feeding asynchronous set or reset inputs of flip-flops is considered as an asynchronous pin network).
- *ability to load data* into a scan-chain.
- *ability to capture data* into scan cells.

If the netlist passes ‘check test rules’ then scan-path is inserted, boundary-scan logic is inserted, test pattern are generated and the fault coverage is computed. If fault coverage is below 90 %, rework on the RTL description is required.

9.2.1 Scan-Path Insertion

One way to test a chip after production is the integration of scan-path logic. Test patterns are shifted in at the SCANIN input and shifted out at the SCANOUT output if signal SCANEN (scan-enable) is active, otherwise the scan flip-flops behave as usual.

Note that for the UTCSU partial scan would be best, because most units can be tested through out the 32-bit wide data bus interface. But unfortunately our Testcompiler only supports full scan. The predicted increase in die size, applying the full scan methodology for the UTCSU with its 4720 flip-flops, is about 10.5 % in a 0.7 µm CMOS technology from ES2.

Standard IC tester can handle around 500 flip-flops in one scan-chain. Hence the 4720 flip-flops in the UTCSU are split up into 10 chains with about 470 flip-flops each. The arrangement of the flip-flops in the chains is fixed by the automatic Testcompiler.

9.2.2 Boundary-Scan Insertion

Boundary-scan logic is implemented according Section 5.4.

9.2.3 Fault Simulation

Test patterns are automatically created by the Synopsys *test compiler*. The achieved fault coverage based on a stuck at 0 (SA0), stuck at 1 (SA1) model is 95.6 %. CPU time to create the test patterns was 2158 s on a SUN-Hypersparc workstation with 143 MHz CPU speed and 128 Mbytes main memory.

9.3 BACK END DESIGN

Back end design finalizes the chip design process. After test logic is inserted and fault-grading is carried out successfully, pad instances are added and the cells are placed and routed. Parasitics (viz delay and load) are extracted from the layout and back annotated into the gate-level simulator to check for timing and fan-out violations, cf. Figure 9-6.

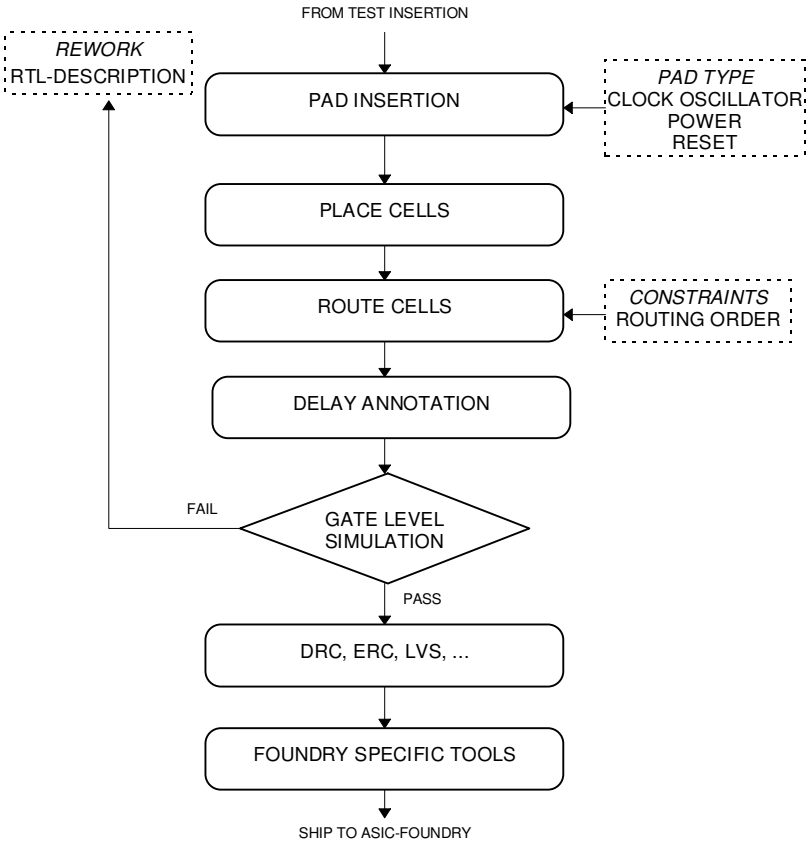


Figure 9-6 Back End Design Flow

Design rule checking (DRC), electrical rule checking (ERC) and layout versus schematic (LVS) verification is performed on the chip. Silicon foundry specific pre-shipment tools for die packaging, test pattern conversion, bonding diagram generation, pad rules checking, clock skew checking and simulation rules checking are run on the design. Finally the customer sign-off form is filled out and the database transmitted to the ASIC-foundry.

CHAPTER 10

10 FUTURE ASPECTS

Alles zu seiner Zeit.
Georg Friedrich von Brandenburg

A wise proverb says 'Never change a running system!'. But things are moving on, new technologies emerge, new requirements are set. To keep track with this steady evolution one must start thinking of new aspects at a time when the old thoughts are still seething. During the design cycle of the UTCSU new ideas came to daylight providing material for the next generation UTCSU. These ideas are summarized shortly without any claim for correctness.

10.1 ACCURACY AND PRECISION

These issues will turn out to be a never ending story. Currently the UTCSU is designed to operate up to 25 MHz clock frequency which results in a total accuracy of about 1 µs. Higher accuracy and better precision is established with a higher system clock frequency. Speeding up the whole part (eg by reducing the feature size) is one way, to tighten the timing constraints for the synthesis process is another way. It turns out very quickly that the huge adder in the LTU and the ACU are the limiting speed factors. Changing the architecture of these two units is another way to speed up the part. Next generation, even more accurate UTCSUs may employ the interleaving adder and counter approach from Section 3.6.4.

Another way to go is to use a voltage controlled oscillator (VCO) instead of the adder based approach. Data have to be gathered and analyzed first, before any decisions can be made in that direction.

10.2 SECURITY

Some applications may require even more security aspects as currently implemented in the UTCSU. Similar to modern CPUs, UTCSU access may be restricted for different operating modes and different type users. A four step access authorization could be structured into a

- *supervisor mode* that gains access to the complete chip while
- a *clock management mode* has access to all registers in the LTU, ACU, SSU and GPU,
- a *monitor mode* has access to debug facilities in the SNU and BTU, and
- a *user mode* has access to registers in the APU.

Access rights for the different operating modes must be implemented in hardware. A *lock state* on all currently not accessible registers prohibits erroneous or malevolent manipulation of their contents (eg clock deterioration values, ...).

10.3 TIMESTAMPING DATA PACKETS

Several methods how to pull-out timestamps of received data packets from the UTCSU are described in Section 2.6.2.4. A pretty elegant way is to implement a first-in first-out (FIFO) memory to hold a number (depends on the maximum network load) of timestamps within the UTCSU and raise interrupts with every new timestamp captured. Currently the synchronization subnet units (SSUs) cannot sustain this operating mode under heavy network load. Adding FIFO memory to the SSUs could support this method even under heavy network load.

10.4 UTCSU FAMILY

It is planned to create a whole family of UTCSUs to better serve the different application and environmental needs. Devices will vary in performance, pin-count, package size, power consumption, price, functionality, offered interfaces, reliability and security aspects. Dedicated *P-, S-, C-, and G-nodes* could emerge. Subsets of the functionality have to be defined for the dedicated parts.

10.5 APPLICATION SUPPORT

Right now the UTCSU offers very little application support. Some new ideas are listed.

- Future designs may increase the number of output channels to trigger external events.
- Another idea is a built-in alarm clock, that works off a list of programmed events and produces interrupts or external events.
- The highly accurate UTCSU time can be used to measure pulse-width or count pulses within a time window.
- Maybe some other time notions need to be supported in future.

10.6 REDUCE DIE SIZE

The biggest concern of the current design is the large chip area of nearly 100 mm^2. Even so this design is currently an academic affair which hosts all the functionality one can think of, saving chip area is the most important concern for next generation parts. Hopefully we will find redundant or unused logic in the UTCSU during the implementation of the clock synchronization layer. First die size minimizing steps are suggested below.

- Use partial-scan instead of full-scan as the selected test strategy. This can probably save 2-3 % of the total cell area.
- Get rid of asymmetric accuracy intervals and introduce symmetric intervals. This can save up to 5 %.
- Don't support leap second correction in the LTU. The NTP adder gets smaller, savings are hard to estimate.
- Use only MISRs or Blocksums in the BTU. Completely dropping the BTU saves 10 %.
- Implement only one SSU instead of six. This saves about 27 % from total chip size.
- Implement only one GPU instead of three. This saves about 2.4 %.
- Strip down the complete chip for one dedicated clock synchronization algorithm and one dedicated application.
- Restructuring the timestamp capture logic as explained in the next paragraph can approximately save another 6 %.

Currently 56-bit wide timestamps are sampled into the SSUs, GPUs, SNU and APU on external events. The 56 bits are split into a 24-bit macrostamp and a 32-bit timestamp portion. Bit 0, the least significant bit of the macrostamp, represents NTP-bit [+8] which is equivalent to 4.27 min. Think, die size can be reduced if the following condition can be fulfilled.

Note! Timestamps in the UTCSU older than 4.27 min are automatically overwritten with new timestamps and are no longer available.

Figure 10-1 shows the concept. The macrostamp portion of the timestamp is no longer captured in the attached units but in a globally accessible macrostamp register. Actually there are two global macrostamp registers MACROSTAMP0 and MACROSTAMP1 which alternately capture the upper 24 bits in 4.27 min. intervals. Only the lower 32 bits plus the 8 bit checksum plus the macrostamp register select signal MS is captured in the attached units. Hence a total of 23 memory bits is saved in every timestamp capture block. With six SSUs, three GPUs, the APU and SNU on a chip, a total of 25 * 23 bits = 575 memory elements is saved which represents a chip area of 1.2 mm^2 or 6.18 % in the 0.7 µm ES2 process.

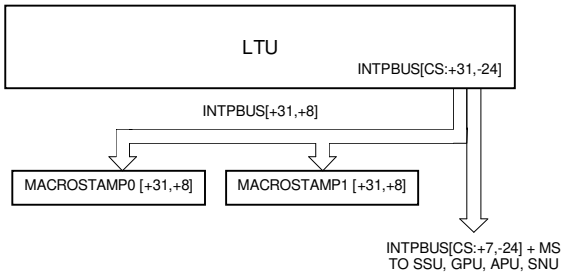


Figure 10-1 To Reduce Die Size, Capture the Macrostamp Portion in two Global Registers

10.7 UTCSU MACROCELL

Once die size shrinks to a reasonable number, the UTCSU should be offered as a macrocell for integration in any other chip design. Like a CPU-core a UTCSU-core has to be created. The VHDL simulation model already exists and can be reused for new designs.

10.8 CONCLUSION

A Universal Time Coordinated Clock Synchronization Unit (UTCSU) manufactured as an Application Specific Integrated Circuit (ASIC) was presented. Time granularity targets the 40 ns range. The state and rate programmable local clock is based on the Network Time Protocol (NTP) format to represent local time and uses a novel Adder-Based-Clock (ABC) to compensate for oscillator drift. Clock accuracy information is provided as well. Packet timestamping, application timestamping and an interface to Global Positioning System (GPS) timing-receivers are supported. Comprehensive test and self-test services guarantee reliable operation.

The top-down design flow starting from a written specification led to the behavioral UTCSU description, Register Transfer Level (RTL) logic synthesis, automatic test logic insertion and final place and route for a standard cell CMOS process. New aspects for next generation UTCSU families were presented.

LIST OF FIGURES

Dies Bildnis ist bezaubernd schön!
Emanuel Schikaneder, Zauberflöte

Figure 1-1	A Centralized Clock Triggers and Records Events E_1, E_2, E_3, ...	21
Figure 1-2	Computer Integrated Manufacturing (CIM) Pyramid [SCHN94]	22
Figure 1-3	Decentralized Process Control based on the CIM-Pyramid	22
Figure 1-4	GPS-Based Time Distribution requires Direct Satellite View	24
Figure 1-5	Some Timecodes like the IRIG-H are Based on Dedicated Time-Channels	24
Figure 1-6	Clock Synchronization Unit of [KO87], AMI S65C60	26
Figure 2-1	System Components of a Synchronization Subnet (SSN)	30
Figure 2-2	UTCSU in a Profibus Environment	30
Figure 2-3	UTCSUs in an Ethernet Based Versatile Timing Analyzer	31
Figure 2-4	Entities in the Triple Layer Clock Synchronization Approach	31
Figure 2-5	Clock Correction happens in Rounds	32
Figure 2-6	Sequence of Virtual Clocks	32
Figure 2-7	Local Clock with Continuous Amortization	33
Figure 2-8	State Quantities of a Virtual Clock C_i	34
Figure 2-9	Upper and Lower Envelopes Surround UTC	34
Figure 2-10	Packet Format and Timestamping in LAN and WAN	37
Figure 2-11	Packet Format and Timestamping in FAN	38
Figure 2-12	Entities for Timestamping Data Packets in the Clock Hardware Layer	38
Figure 2-13	Simplified UTCSU Block Diagram	40
Figure 2-14	The UTCSU is Prepared to Serve as a Gateway-Node in a Triple Redundant Network	41
Figure 2-15	Network Clock Synchronization Coprocessor (NCSC) based on the UTCSU	41
Figure 3-1	Leap Second Insertion in UTC and NTP on Dec. 31st, 1995	45
Figure 3-2	NTP Data Format [+31, -32]	46
Figure 3-3	Extended NTP Data Format [+31, -59]	47
Figure 3-4	Recorded ENTP Format [+31, -24]	47
Figure 3-5	Quartz Crystal Oscillator Characteristics	48
Figure 3-6	a) Clock Adjusted by Step Width Δt b) Clock Adjusted by Step Height s[k]	51
Figure 3-7	This Digital System Solves Equation 4-7	51
Figure 3-8	Single Counter Approach	52
Figure 3-9	Adder Based Clock	53
Figure 3-10	Hardware Clock (NTPTIME) with Rate and State Correction	54
Figure 4-1	GPS Receiver Time Interface	56
Figure 4-2	Typical Time-Based GPS Receiver Design	58
Figure 4-3	GPS Clock Receiver Design	58
Figure 4-4	Interfacing the 10 MHz Output to the UTCSU	59
Figure 4-5	Interfacing the 1PPS Output to the UTCSU	59
Figure 4-6	GPS Receiver Evaluation	60
Figure 5-1	Test Concept Overview	62
Figure 5-2	Multiplexed Flip-Flop Scan-Path Design	63
Figure 5-3	Boundary-Scan Test Logic According IEEE 1149.1	64
Figure 5-4	IDCODE Register	65
Figure 5-5	MISR, $P^{56}(x) = x^{56} + x^{22} + x^{21} + x^1 + 1$	67
Figure 5-6	Blocksum Signatures for Timestamps and Accuracystamps	68
Figure 5-7	Redundant Operation of Multiple UTCSUs	69
Figure 5-8	Signal Path for On-Line Event Testing	70
Figure 5-9	Dual Stage Metastability Design	70

Figure 5-10	Single Stage Metastability Design	70
Figure 5-11	IDENTIFICATION - UTCSU Identification Register	72
Figure 7-1	A CPU Interfaces with UTCSU-Units through the BIU and MBI	78
Figure 7-2	Internal Operand Representation	79
Figure 7-3	UTCSU Interface to Various Port Sizes	79
Figure 7-4	Multiple UTCSU Register Access	81
Figure 8-1	Register Components to Maintain Local Time and Accuracy	83
Figure 8-2	Local Time Unit (LTU) Architecture	84
Figure 8-3	Local Time Initialization Data Path	86
Figure 8-4	Data Path during Continuous Amortization Phase	87
Figure 8-5	Amortization Phase Timing Diagram	88
Figure 8-6	Data Path during Pure Phase for the Time Step Data	88
Figure 8-7	Leap Second Insertion and Deletion	89
Figure 8-8	Synchronous Start in Debug Mode and for Redundant Operation	90
Figure 8-9	NTPTIME Snapshot Mode Timing Diagram	90
Figure 8-10	Amortization Timer Snapshot Mode Timing Diagram	91
Figure 8-11	Eight Control Bits protect 56 Data Bits	91
Figure 8-12	The Optimized Matrix requires 116 Two-Input XOR-Gates	92
Figure 8-13	Boolean Equations for the Optimized EDC Matrix (⊕...XOR-Gate)	93
Figure 8-14	Five Levels of Hierarchy in the EDCU	93
Figure 8-15	MSSET - Macro Stamp Set Register	96
Figure 8-16	MSGET - Macro Stamp Get Register	96
Figure 8-17	TSSET - Timestamp Set Register	97
Figure 8-18	TSGET - Timestamp Get Register	97
Figure 8-19	USSET - Micro Stamp Set Register	97
Figure 8-20	USGET - Micro Stamp Get Register	97
Figure 8-21	NSGET - Nano Stamp Get Register	97
Figure 8-22	NTPSET - NTPTIME Set Register	97
Figure 8-23	STEPLOW - Step Low Register	98
Figure 8-24	STEPAMORT - Step Register for Continuous Amortization Phase	98
Figure 8-25	STEPPURE - Step Register for Pure Phase	98
Figure 8-26	AMORTTIMESET - Preload Value for Amortization Timer	98
Figure 8-27	AMORTTIMEGET - Read Current Amortization Timer	98
Figure 8-28	STARTAMORT - Start the Amortization Timer	99
Figure 8-29	Initialization Procedure for STEPPURE Register	99
Figure 8-30	Initialization of NTPTIME Register	99
Figure 8-31	NTPTIME-Initialization Timing Diagram	99
Figure 8-32	Local Time Unit Test Port Layout	100
Figure 8-33	Accuracy Unit (ACU) Architecture	101
Figure 8-34	Number Scale applied to Accuracy Envelopes ALPHAP(N)	102
Figure 8-35	Realigning Envelopes at Start of Amortization Phase	103
Figure 8-36	Accuracy Tracking during Continuous Amortization	104
Figure 8-37	Envelope Evolution during Amortization and Pure Phase	105
Figure 8-38	Envelope Maintenance during Pure Phase	105
Figure 8-39	Synchronous Start of the ACU	106
Figure 8-40	ACU Snapshot Mode Timing Diagram	106
Figure 8-41	LAMBDAPAMORT - Deterioration during Amortization for Upper Envelope	109
Figure 8-42	LAMBDAPPURE - Deterioration during Pure Phase for Upper Envelope	110
Figure 8-43	LAMBDANAMORT - Deterioration during Amortization for Lower Envelope	110
Figure 8-44	LAMBDANPURE - Deterioration during Pure Phase for Lower Envelope	110
Figure 8-45	STATEPSET - Upper Envelope Adjustment at Start of Amortization Phase	110
Figure 8-46	STATENSET - Lower Envelope Adjustment at Start of Amortization Phase	110
Figure 8-47	STATEP(N)GETL - Upper and Lower Envelope Values with Latch Operation	110
Figure 8-48	STATEP(N)GET - Upper and Lower Envelope Values; No Latch Operation	111
Figure 8-49	BOUNDP(N) - Bounds for Upper and Lower Envelopes	111
Figure 8-50	ALPHASET - Preload Data for Envelope Registers ALPHAP(N)	111
Figure 8-51	ALPHAPNSET - Load ALPHAP(N) with Data from ALPHASET	111
Figure 8-52	ALPHAPGETL - Read Envelope Register ALPHAP with Latch	111
Figure 8-53	ALPHANGETL - Read Envelope Register ALPHAN with Latch	112
Figure 8-54	ALPHAPGET - Read Envelope Register ALPHAP	112
Figure 8-55	ALPHANGET - Read Envelope Register ALPHAN	112

Figure 8-56	NALPHAPGET - Read Envelope Register NALPHAP	112
Figure 8-57	NALPHANGET - Read Envelope Register NALPHAN	112
Figure 8-58	Initialization Procedure for LAMBDAP(N)PURE Register	113
Figure 8-59	Initialization of ALPHAP(N) Registers	113
Figure 8-60	ALPHAP(N) Initialization Timing Diagram	113
Figure 8-61	Accuracy Unit Test Port Layout	114
Figure 8-62	Synchronization Subnet Unit (SSU) Data Flow Diagram	115
Figure 8-63	Receive Timestamp Timing Diagram	116
Figure 8-64	Transmit Timestamp Timing Diagram	117
Figure 8-65	Duty-Timer Operation Timing Diagram	117
Figure 8-66	Duty-Timer B in SSU1 Starts Continuous Amortization, SSU2 Triggers T-Duties	118
Figure 8-67	MSRCV - Macro Stamp Receive Register	120
Figure 8-68	TSRCV - Timestamp Receive Register	120
Figure 8-69	MSXMT - Macro Stamp Transmit Register	120
Figure 8-70	TSXMT - Timestamp Transmit Register	120
Figure 8-71	ACCNPXMT - Accuracy N and P Register	121
Figure 8-72	ACCNPNPLXMT - Accuracy N and P Register, Lower 8 bits	121
Figure 8-73	ACCNPNPHXMT - Accuracy N and P Register, Higher 8 bits	121
Figure 8-74	DUTYAH - Duty-Timer A Higher Preload Register	121
Figure 8-75	DUTYAL - Duty-Timer A Lower Preload Register	121
Figure 8-76	DUTYBH - Duty-Timer B Higher Preload Register	121
Figure 8-77	DUTYAL - Duty-Timer B Lower Preload Register	122
Figure 8-78	SSU Test Port Layout	122
Figure 8-79	Test Sequence for the SSU Self-Test	123
Figure 8-80	Global Positioning Unit Data Flow Diagram	124
Figure 8-81	GPS Timestamp Timing Diagram	125
Figure 8-82	MSGPS - Macro Stamp GPS Register	126
Figure 8-83	TSGPS - Time Stamp GPS Register	127
Figure 8-84	GPU Test Port Layout	127
Figure 8-85	Test Sequence for the GPU Self-Test	128
Figure 8-86	Application Unit Data Flow Diagram	129
Figure 8-87	CPU-Access Procedure for the Application Recording Register Set 5	130
Figure 8-88	Application Event Recording	130
Figure 8-89	Application Event Generation	131
Figure 8-90	MSAPP - Macro Stamp Application Register	133
Figure 8-91	TSAPP - Timestamp Application Register	133
Figure 8-92	ACCAPP - Accuracy Application Register	133
Figure 8-93	MSAPPX - Macro Stamp Application Register X	133
Figure 8-94	TSAPPX - Timestamp Application Register X	133
Figure 8-95	ACCAPPX - Accuracy Application Register X	133
Figure 8-96	DUTYH - Duty-Timer Higher Preload Register	134
Figure 8-97	DUTYL - Duty-Timer Lower Preload Register	134
Figure 8-98	APPSEL - Application Register Set Select Register	134
Figure 8-99	APPCLEAR - Clear Application Referenced Status Bit	134
Figure 8-100	APU Test Port Layout	135
Figure 8-101	Test Sequence for the APU Self-Test	135
Figure 8-102	Snapshot Unit Data Flow Diagram	136
Figure 8-103	Hardware Snapshot Timing Diagram	137
Figure 8-104	Programmed Software-Snapshot Timing Diagram	137
Figure 8-105	Synchronous Operation of Several UTCSU Chips	138
Figure 8-106	MSSNU - Macro Stamp Snapshot Register	139
Figure 8-107	TSSNU - Time Stamp Snapshot Register	140
Figure 8-108	ACCNPSNU - Accuracy N and P Register	140
Figure 8-109	SWSNAP - Fire Software Snapshot	140
Figure 8-110	DUTYH - Duty-Timer Higher Preload Register	140
Figure 8-111	DUTYL - Duty-Timer Lower Preload Register	140
Figure 8-112	SNU Test Port Layout	141
Figure 8-113	Test Sequence for the SNU Self-Test	141
Figure 8-114	Built-In Self-Test Unit Data Flow Diagram	142
Figure 8-115	Timestamp Signature Analysis $P_{56}(x) = x^{56} + x^{22} + x^{21} + x^{1} + 1$ [YAR93]	143
Figure 8-116	Accuracystamp Signature Analysis $P_{32}(x) = x^{32} + x^{28} + x^{27} + x^{1} + 1$ [YAR93]	143
Figure 8-117	Timestamp and Accuracystamp Blocksum Calculation	143

Figure 8-118	Signature Computation Start and Capture Moments	144
Figure 8-119	Built-In Self-Test: MISR Signature Computation	144
Figure 8-120	Pure Phase Counter Timing Diagram	146
Figure 8-121	Pure Phase Counter Interrupt Timing Diagram	146
Figure 8-122	TESTSEL - UTCSU Test Select Register	147
Figure 8-123	TESTGATE - UTCSU Test Gate Register	147
Figure 8-124	MSSIG - Macro Stamp MISR Signature Holding Register	148
Figure 8-125	TSSIG - Timestamp MISR Signature Holding Register	148
Figure 8-126	ACCSIG - Accuracy Stamp MISR Signature Holding Register	148
Figure 8-127	MSSUM - Macro Stamp Blocksum Signature Holding Register	148
Figure 8-128	TSSUM - Timestamp Blocksum Signature Holding Register	148
Figure 8-129	ACCSUM - Accuracy Stamp Blocksum Signature Holding Register	148
Figure 8-130	PUREPHASE - Pure Phase Counter Holding Register	149
Figure 8-131	BTU Test Port Layout	149
Figure 8-132	Interrupt Unit Data Flow Diagram	150
Figure 8-133	UTCSU Version Identification Register	150
Figure 8-134	Basic Interrupt Cell	152
Figure 8-135	CPU Interrupt Processing Data Flow	153
Figure 8-136	IDENTIFICATION - UTCSU Identification Register	155
Figure 8-137	UTCSTAT1 - UTCSU Status Register 1	155
Figure 8-138	UTCSTAT2 - UTCSU Status Register 2	155
Figure 8-139	UTCCONF1 - UTCSU Configuration Register 1 Read-back	155
Figure 8-140	UTCCONF2 - UTCSU Configuration Register 2 Read-back	155
Figure 8-141	UTCINTSTAT1 - UTCSU Interrupt Status Register 1	155
Figure 8-142	UTCINTSTAT2 - UTCSU Interrupt Status Register 2	155
Figure 8-143	UTCCONF1 - UTCSU Configuration Register 1	156
Figure 8-144	UTCCONF2 - UTCSU Configuration Register 2	156
Figure 8-145	UTCINTEN1 - UTCSU Interrupt Enable Register 1	156
Figure 8-146	UTCINTEN2 - UTCSU Interrupt Enable Register 2	156
Figure 8-147	UTCINTCLEAR1 - UTCSU Interrupt Clear Register 1	156
Figure 8-148	UTCINTCLEAR2 - UTCSU Interrupt Clear Register 2	157
Figure 8-149	ITU Test Port Layout	157
Figure 8-150	NTP-Time Interface Capture Logic	158
Figure 8-151	NTP-Interface Data Flow Diagram	158
Figure 8-152	NTP-Interface Timing Diagram	159
Figure 8-153	Bus Interface Unit (BIU) Block Diagram	160
Figure 8-154	BIU Write-Timing	161
Figure 8-155	BIU Read-Timing	162
Figure 8-156	Data Port Routing Block	164
Figure 8-157	SYNC-Cell Schematic View	166
Figure 8-158	Synchronization Uncertainty Assessment	166
Figure 9-1	Front-End Design Flow	167
Figure 9-2	Board Level System Simulation	168
Figure 9-3	RTL Simulation Test Bench	169
Figure 9-4	Design Synthesis Constraints Put on Inputs and Outputs	170
Figure 9-5	Test Integration Flow	171
Figure 9-6	Back End Design Flow	173
Figure 10-1	To Reduce Die Size, Capture the Macrostamp Portion in two Global Registers	177

LIST OF TABLES

Tabula rasa
Aristoteles De anima 3, 4

Table 1-1	System Model Characteristics [SIM90]	25
Table 3-1	Conversion Aid between Gregorian Calendar and JD [ZEM90, p. 138]	44
Table 3-2	Dates of Leap Second Insertion as of Jan 30th, 1996	44
Table 3-3	Time Notions Overview	45
Table 3-4	Influence of Leap Seconds in NTP and GPS Timestamps	46
Table 3-5	ENTP Format	47
Table 3-6	Performance of Analog and Digital TCXOs [GAG90]	49
Table 3-7	Characteristics of Standard Oscillators	50
Table 3-8	Oscillator Stratum Classification	50
Table 4-1	GPS NMEA Output Message Examples	57
Table 4-2	Industry Standard GPS Receivers (Factory Numbers)	60
Table 5-1	Boundary Scan Instruction Codes	66
Table 6-1	UTCSU Memory Map and Unit Summary	73
Table 6-2	UTCSU Register Map	75
Table 7-1	Size Signal Encoding	78
Table 7-2	Internal to External Data Bus Multiplexer	80
Table 8-1	STEPLOW contents for decimal oscillator frequencies	86
Table 8-2	Initialization Sequence for NTPTIME Register	86
Table 8-3	Configuration Bits (CLEAPSECOND, CLEAPINS) Control Leap Second Correction	89
Table 8-4	The Syndrome Table lists Single (#), Double (D) and Triple (T) Faults	94
Table 8-5	Configuration Bits in Unit Local Time	95
Table 8-6	Status Bits in Local Time Unit	95
Table 8-7	Interrupt Sources in Local Time Unit	95
Table 8-8	Local Time Unit Register Map	96
Table 8-9	Various Observed Control Signals in Unit Local Time	100
Table 8-10	Limits and Interrupts for Adder ACCP and ACCN	102
Table 8-11	Initialization Sequence for ALPHAP and ALPHAN Registers	103
Table 8-12	Status Bits in Accuracy Unit	107
Table 8-13	Interrupt Sources in Accuracy Unit	108
Table 8-14	Accuracy Unit Register Map	109
Table 8-15	Various Observed Control Signals in Unit Accuracy	114
Table 8-16	Configuration Bits in Unit Synchronization Subnet	118
Table 8-17	Status Bits in Unit Synchronization Subnet	118
Table 8-18	Interrupt Sources in SSU	119
Table 8-19	Synchronization Subnet Unit Register Map	119
Table 8-20	SSUx Memory Address Locations	120
Table 8-21	Various Observed Control Signals in SSU	122
Table 8-22	Configuration Bits in Unit GPS	125
Table 8-23	Status Bits in Unit GPS	125
Table 8-24	Interrupt Sources in GPU	126
Table 8-25	Global Positioning Unit Register Map	126
Table 8-26	GPUx Memory Address Locations	126
Table 8-27	Various Observed Control Signals in GPU	127
Table 8-28	Configuration Bits in the Application Unit	131
Table 8-29	Status Bits in Application Unit	131
Table 8-30	Interrupt Sources in APU	132

Table 8-31	Application Unit Register Map	132
Table 8-32	Various Observed Control Signals in APU	135
Table 8-33	Configuration Bits in Unit Snapshot	138
Table 8-34	Status Bits in Snapshot Unit	138
Table 8-35	Interrupt Sources in SNU	139
Table 8-36	Snapshot Unit Register Map	139
Table 8-37	Various Observed Control Signals in SNU	141
Table 8-38	Test Configurations for Signature Calculation	145
Table 8-39	Configuration Bits in the BTU	146
Table 8-40	Interrupt Source in BTU	146
Table 8-41	Built-In Test Unit Register Map	147
Table 8-42	Various Observed Control Signals in BTU	149
Table 8-43	Status Bits in Register UTCSTAT1 (064h)	151
Table 8-44	Status Bits in Register UTCSTAT2 (068h)	151
Table 8-45	Configuration Bits in Register UTCCONF1 (060h)	152
Table 8-46	Configuration Bits in Register UTCCONF2 (064h)	152
Table 8-47	Interrupt Sources in Register UTCINTEN1 (068h), UTCINTSTAT1 (074h), ...	153
Table 8-48	Interrupt Sources in Register UTCINTEN2 (06Ch), UTCINTSTAT2 (078h)	154
Table 8-49	Interrupt Unit Register Map	154
Table 8-50	Leap Second Management in LTU	156
Table 8-51	Various Observed Control Signals in ITU	157
Table 8-52	Critical Signals mapped to ITESTBUS[47:0]	159
Table 8-53	AC Timing Requirements for the BIU Write Access	162
Table 8-54	AC Timing Requirements for BIU Read Access	163
Table 8-55	Data Port Routing Control Signals	164
Table 8-56	Lowest Usable and Maximum Usable Clock Frequencies	165
Table 9-1	Area Report without Routing Area in μm^2 with 6 SSUs and 3 GPUs for ECPD07	170
Table 9-2	Area Report with Estimated Routing Area in μm^2 for ECPD07 (6 SSUs and 3 GPUs)	171

LITERATURE

Litterae non erubescunt
Cicero

[ABF90] Abramovici, M., Breuer, M. A., Friedman, A. D., 'Digital Systems Testing and Testable Design', Computer Science Press, 1990

[AMI86] Austria Micro Systems International AMI, 'Clock Synchronization Unit (CSU) S65C60', Data Sheet, AMI, Austria, 1986

[AUD90] Audoin, C., 'Classical Frequency Standards', EFTF 90, Neuchatel, Switzerland

[BELL86] Bell Communications Research, 'Digital Network Synchronization Plan', Technical Advisory TA-NPL-000436, 1 November 1986

[BEN92] Bender, K., 'Profibus - Der Feldbus für die Automation', Hanser Verlag, 1992

[BID90] Bidari, L., Girardet,E., HUET, J., Renoult, P., 'Low Phase Noise, High Frequency TCXO', EFTF 90, Switzerland

[CAS86] Christian, F., Aghili, H., Strong, R., 'Clock Synchronization on the Presence of Omission and Perfomance Faults and Processor Joins', Proc. 16th International Symposium on Fault Tolerant Computing Systems, Vienna, Austria, July 1986, p. 218-223

[DAN92] Danuser, G., 'Zeitübertragung und Synchronisation mit GPS, Anwendung in der Überwachung verteilter Systeme', ETH Zürich, Institut für Geodäsie, April 1992

[DANA95] Dana, P., H., 'Global Positioning System (GPS) Time Dissemination for Real-Time Applications', Department of Geography, University of Texas at Austin, TX 78712, 1995

[DB93] Drummond, R., Babaoglu, Ö., 'Low-cost Clock Synchronization', Distributed Computing, 6, 1993, p. 193-203

[DHS86] Dolev, D., Halpern, J., Strong, H. R., 'On the Possibility and Impossibility of Achieving Clock Synchronization', Journal of Computer and System Science, vol. 32, no. 2, pp.230-250, 1986

[E01A94] ES2, 'ES2 Process-Independent Library Handbook', ES2 France, 1994

[EFTF90] ACTES, 'Fourth European Frequency and Time Forum', EFTF 90, Neuchatel, Switzerland, March 1990

[EK73] Ellingson, C. E., Kulpinsky, R. J., 'Dissemination of System Time', IEEE Transactions on Communications, Vol. COM-21, No. 5, May 1973

[FELD95] Feldmann, K., Solvie, M., 'Die Uhr im Feldbus, OLCHFA', Elektronik, p. 112, 1995

[FUJ90] Fujiwara, H., 'Logic Testing and Design for Testability', The MIT Press, 1990

[GAG90] Gagnepain, J., 'Quartz Crystal Oscillators from their Design to their Performances', EFTF 1990, Switzerland

[GOL82] Golomb, S., W., 'Shift Register Sequences', Aegean Park Press, Laguna Hills, California, 1982

[HALA94] Halang, W. A., Wannemacher, M., ‘A High-Precision Time Processor for Distributed Real-Time Systems’, FernUniversität, Faculty of Electrical Engineering, Hagen, Germany, 1994

[HL95] Horauer, M., Loy, D., 'Adder Synthesis', Austrochip 1995, TU-Wien 1995

[HOR94] Horauer, M., 'Entwicklung einer Network Timestamp Unit für einen Versatile Timing Analyzer zum Monitoring von verteilten Echtzeitsystemen', Diploma Thesis, Institute of Computer Technology, TU Wien, 1994 (in German)

[HS91] Halpern, J., Suzuki, I., 'Clock Synchronization and the Power of Broadcasting', Distributed Computing, 5(2), September 1991, p. 73-82

[HSSD84] Halpern, J., Simons, B., Stong, R., Dolev, D., 'Fault-Tolerant Clock Synchronization', Proc. 3rd Annual ACM Symposium on Principles of Distributed Computing, August 1984, p. 89-102

[KC87] Kleeman, L., Cantoni, A., 'Metastable Behavior in Digital Systems', IEEE Design and Test of Computers, December 1987, pp.4-19

[KES84] Kessels, J.L.W., 'Two Designs of a Fault-Tolerant Clocking System', IEEE Transactions on Computers 33(10), Oct. 1984, p. 912-919

[KL95] Klima, R., Loy, D., 'NTP - EDCU', Computertechniklabor SS1995, Institut für Computertechnik, TU Wien, Wien 1995

[KO87] Kopetz, H., Ochsenreiter, W., 'Clock Synchronization in Distributed Real-Time Systems', IEEE Transactions on Computers, C-36(8), August 1987, p. 933-940

[LACR90] Lacroix, J.P., ‘Oscillators for Real-Time On-Board Applications of Navstar-GPS, EFTF 90, Switzerland

[LAM84] Lamport L., 'Using Time Instead of Timeout for Fault-Tolerant Distributed Systems', ACM Transactions on Programming Languages and Systems 6(2), April 1984, p. 254-280

[LANGE92] Lange, W. R., ‘Synchronisation verteilter Rechner mit Hilfe von standardisierten Zeitcodes’, Lange electronic, Germany 1992

[LANN87] LeLann, G., ‘A variation on the IEEE 802.3 standard for real time LANs’, INRIA-BP 105, France, Oct. 1987

[LEE88] Lee, E. A., Messerschmitt, D. G., 'Digital Communication', Kluwer Academic Publishers, 1988

[LH81] Logsdon, T. S., Helms, C. W., 'Computer Simulation and the Performance Capabilities of the NAVSTAR Space-Based Navigation System', Rockwell International, 1981

[LH95] Loy, D., Horauer, M., ‘NTSU Board Level Simulation’, Institute for Computertechnology, University of Technology Vienna, 1995

[LIS93] Liskov, B., 'Practical uses of synchronized clocks in distributed systems', Distributed Computing 6, 1993, p.211-219

[LK93] Loy, D., Kittenberger, T., ‘Home System Node including Boundary-Scan’, Proceedings EUROCHIP Conference 1993, Toledo Spain

[LL84] Lundelius-Welch, J., Lynch, N., 'An Upper and Lower Bound for Clock Synchronization', Information and Control 62(2-3), August-September 1984, p. 190-204

[LL88] Lundelius-Welch, J., Lynch, N., 'A New Fault-Tolerant Algorithm for Clock Synchronization', Information and Computation 77(1), 1988, p. 1-36

[LM85] Lamport, L., Melliar-Smith, P.M., 'Synchronizing Clocks in the Presence of Faults', Journal of the ACM, 32(1), January 1985, p.52-78

[LO94] Loy, D., et. al., ‘High Level Design of a Parallel Processing RISC Microcontroller’, Proceedings EUROCHIP Conference 1994, Dresden Germany

[LO95] Loy, D. ‘Investigating OSI-Layers 2 and 3 in the Lontalk Protocol’, Technical Report, Echelon Corp., Palo Alto California, 1995

[LS95] Loy, D., Schmalek, R., ‘Thougths about Redundancy in Fieldbus Systems Anchored in OSI Layer-4 and Applied to the Lontalk Protocol on Neuron Based Network Nodes’, IEEE International Workshop on Factory Communication Systems, Switzerland, 1995

[LSP82] Lamport, L., Shostak, R., Pease, M., 'The Byzantine General Problem', ACM Transactions on Programming Languages and Systems, 4(1), July 1982, p. 382-401

[MA92] Mavronicolas, M., 'An Upper and a Lower Bound for Tick Synchronization', Proc. Real-Time Systems Symposium, Phoenix, Arizona, December 1992, p. 246-255

[MAG93] 'User Guide for the Magellan OEM GPS Modules', Rev. 3.0, Magellan Systems Corporation, 1993

[MAR83] Marzullo, K., 'Loosely Coupled Distributed Services - A Distributed Time Service’, Ph.D. Thesis, Stanford University, 1983

[ME96] Mehl, H., ‘GPS - Global Positioning System’, Informatik Spektrum 19:33-34, Springer Verlag 1996, p. 33-34

[MILLS91] Mills, D. L., ‘Internet Time Synchronization: The Network Time Protocol’, IEEE Transactions on Communications, Vol. 39, No. 10, Oct. 1991

[MOT6890] Motorola, MC68030 Enhanced 32-Bit Microprocessor User's Manual, Prentice Hall, 1990

[MOT94] 'Oncore User's Guide', Motorola, 1994

[MS85] Mahaney, S., Schneider, F., 'Inexact Agreement: Accuracy, Precision and Graceful Degradation', Proc. 4th Annual ACM Symposium on Principles of Distributed Computing, August 1985, p. 237-249

[NAV95] User Manual, ‘Installation and Operation Guide NTFR-s’, Issue 2.0, Navstar Systems Limited, 1995

[NBS88] 'Automated Computer Time Services (ACTS)', NBS Research Material 8101, U.S. Department of Commerce, 1988

[NOVAK93] Novak, F., Sutaqnovac, N., Trobec, R. ´Built-in self testing of communications systems using ASIC technology´Microprocessors and Microsystems Volume 17, pp 475, 1993

[NOVI94] Novick, A. N., ‘Atomic clock accuracy needed’, IEEE Potentials, 1994s

[OCO92] O'Conner, P., D., T., 'Practical Reliabilty Engineering', John Wiley & Sons, 1992

[OK93] Osterdock, T., N., Kusters, J., A., 'Using a New GPS Frequency Reference in Frequency Calibration Operations", Proceedings of IEEE International Frequency Control Symposium, June 2-4, 1993, pp. 33-39

[PB91] Pfluegl, M.J., Blough, D.M., 'Evaluation of a New Algorithm for Fault-Tolerant Clock Synchronization', Proc. Pacific Rim International Symposium on Fault Tolerant Systems, Kawasaki, Japan, September 1991, p. 38-43

[PRAD86] Pradhan, D. K., 'Fault-Tolerant Computing', Volume 1, Prentice-Hall, 1986

[PW72] Peterson, W., W., Weldon, E. J., 'Error Correcting Codes', 2nd ed., M.I.T. Press, Cambridge, Massachusetts, 1972

[RFC1119] Mills, D. L., ‘ Network Time Protocol (Version 2) Specification and Implementation, University of Delaware, RFC 1119, Sept. 1989

[RFC1129] Mills, D. L., ‘Internet Time Synchronization: the Network Time Protocol’, University of Delaware, RFC 1129, Oct. 1989

[RG90] Rutman, J., Groslambert, J., ‘Characterization and Measurement of Frequency Stability: Twenty Five Years of Progress’, EFTF90

[RO93] 'NavCore Designer's Guide', Rockwell, 1993

[RSB90] Ramanathan, P., Shin, K.G., Butler, R.W., ‘Fault-Tolerant Clock Synchronization in Distributed Systems’, IEEE Transactions on Computers, 23(10), Oct. 1990, p. 33-42

[SC90] Schmuck, F., Christian, F., 'Continuous Amortization need not affect the Precision of a Clock Synchronization Algorithm', 9th ACM Symposium on Principles of Distributed Computing, 1990, p. 133-143

[SCHI77] Schildt, G.-H., 'Zeitsynchrones Kommunikationssystem mit zentraler Synchronisation durch einen Normalfrequenzsender für ein bedarfsgesteuertes Verkehrssystem', Schriftenreihe Institut für Verkehr, Eisenbahnwesen und Verkehrssicherung, TU Braunschweig, Heft 13, 1977

[SCHM941] Schmid, U., ‘An Annotated Bibliography on Clock Synchronization in Distributed Systems’, Technical Report No. 183/1-45, Department of Automation, TU Vienna, Austria, 1994

[SCHM942] Schmid, U., ‘Synchronized UTC for Distributed Real-Time Systems’, TU Vienna, Technical Report 183/1-39, Department of Automation, Vienna 1994

[SCHM943] Schmid, U., 'Monitoring Distributed Real-Time Systems', Journal of Real-Time Systems 7, 1994, p. 33-56

[SCHM95] Schmid, U., 'Synchronized Universal Time Coordinated for Distributed Real-Time Systems', Control Engineering Practice 3(6), 1995, p. 877-884

[SCHN94] Schnell, G., 'Bussysteme in der Automatisierungstechnik', Vieweg Verlag, 1994

[SIM90] Simons, B., Welch, J. L., Lynch, N. ‘An Overview of Clock Synchronization“, B. Simons, A. spector, editors: Fault-Tolerant Distributed Computing. Lecture Notes on Computer Science 448, 1990, p. 84-96

[SL93] Schmid, U., Loy, D., ‘Project Proposal SynUTC’, TU Wien, Department of Automation and Institute of Computertechnology, Vienna 1993

[SL96] Schossmaier, K., Loy, D., 'An ASIC supporting External Clock Synchronization for Distributed Real-Time Systems’, 8th Euromicro Workshop on Real-Time Systems, 1996, Italy

[SMI81] Smith, T.B., 'Fault-tolerant clocking system', Digest of Papers International Symposium on Fault-Tolerant Computing FTCS-11, June 1981, p. 262-264

[SR88] Shin, F.G., Rmanathan, P., 'Transmission Delays in Hardware Clock Synchronization', IEEE Transactions on Computers 37(11), November 1988, p. 1465-1467

[SS95] Schossmaier, K., Schmid, U., ‘UTCSU Functional Specification’, TU Wien, Department for Automation, TR183 (1-56), Wien 1995

[SS96] Schmid, U., Schossmaier, K., 'Interval-based Clock Synchronization', Special Issue Journal of Real-Time Systems, Kluwer Academic Publishers, 1996

[SSHL] Schossmaier, K., Schmid, U., Horauer, M., Loy, D., ‘Specification and Implementation of the Universal Time Coordinated Synchronization Unit (UTCSU)’, Special Issue Journal of Real-Time Systems, Kluwer Academic Publishers, 1996, (submitted)

[STEL95] User Manual, ‘Model 100 A/B GPS-Clock’, Absolute Time Incorporation, USA; 1995

[TRI94] 'SVeeSix-CM2 Embedded GPS Core Module System Designer Reference', Trimble Navigation, 1994

[VEEN80] Veendrick, H., J., M., 'The Behavior of Flip-Flops Used as Synchronizers and Prediction of Their Failure Rates', IEEE J. Solid-State Circuits, Apr. 1980, pp. 168-176

[VILL911] Villemeur, A., 'Reliability, Availability, Maintainability and Safety Assessment', John Wiley & Sons, Volume 1, 1991

[VILL912] Villemeur, A., 'Reliability, Availability, Maintainability and Safety Assessment', John Wiley & Sons, Volume 2, 1991

[VM88] Vasanthavada, N., Marinos, P.N., 'Synchronization of Fault-Tolerant Clocks in the Presence of Malicious Failures', IEEE Transactions on Computers, C-34(4), April 1988, p.440-448

[VR92] Veríssimo, P., Rodrigues, L., 'A posteriori Agreement for Fault-Tolerant Clock Synchronization on Broadcast Networks', Proc. 22nd Symposium on Fault-Tolerant Computing, Boston, July 1992, p. 527-535

[WANNE] Wannemacher, M., Halang, W.A. ‘A High-Precision Time Processor For Distributed Real-Time Systems’, Fern Universität, Faculty of Electrical Engineering, Hagen, Germany 1994

[WEISS92] Weiss, M. A., Allan, D. W., Davis, D. D., Levine, J., ‘Smart Clock: A New Time’, IEEE Transactions ons Instrumentation and Measurement, Vol. 41, No. 6, Dec. 1992

[YAR93] Yarmolik, V. N., Kachan, I. V., 'Self-Testing VLSI Design', Elsevier Science Publisher, 1993

[ZEM90] Zemanek, H., 'Kalender und Chronologie', 5. Auflage, Oldenbourg Verlag, München Wien, 1990

Nullus est liber tam malus, ut non aliqua parte prosit!

Plinius (23-79 AD)

Dipl.-Ing. DIETMAR LOY
Jörgerstraße 3/51
A-1170 Wien

CURRICULUM VITAE

Staatsbürgerschaft:	Österreich
Familienstand:	ledig
31.3.1967	geboren in Grieskirchen, Oberösterreich (JD 2439581)
1973-1977	Volksschule in Waizenkirchen
1977-1979	Hauptschule in Waizenkirchen
1979-1981	Gymnasium in Dachsberg
1981-1986	Höhere Technische Bundeslehranstalt für Nachrichtentechnik in Steyr
17.6.1986	Reifeprüfung mit ausgezeichnetem Erfolg
1986-1992	Studium der Nachrichtentechnik an der TU Wien
1.1.1992-31.3.1992	Studienassistent am Institut für Datenverarbeitung
17.3.1992	Sponsion zum Diplomingenieur für Elektrotechnik mit ausgezeichnetem Erfolg
seit 1.4.1992	Universitätsassistent am Institut für Computertechnik an der TU Wien
1993	Oxford Summer School: Introduction to VHDL, Oxford GB
1994/95	Forschungsaufenthalt bei MOTOROLA Inc., Austin Texas, USA LONWORKS Research Lab, Al Mouton, 10.7.1994-10.2.1995
1995	Forschungsaufenthalt bei ECHELON Corp., Palo Alto CA, USA
1.10.1995-31.5.1996	Präsenzdienst beim Österreichischen Bundesheer in der Fernmeldetruppenschule, Verwendung als Lehrer

.....

(Dietmar Loy)