AEROFLEX GAISLER

SpaceWire - Time Distribution Protocol VHDL IP Core User's Manual

July 2014
Version 1.1

Table of contents

1 Introduction

1.1 Scope

This document establishes the Final Report for the High Accuracy Time Synchronization over SpaceWire Networks activity initiated by the European Space Agency, ESA contract 400105931.

1.2 Reference documents

[CCSDS] Time Code Formats, CCSDS 301.0-B-4, www.CCSDS.org

[SPW] Space engineering: SpaceWire - Links, nodes, routers and networks, ECSS-E-ST-50-12C

[RMAP] Space engineering: SpaceWire - Remote memory access protocol, ECSS-E-ST-50-52C

1.3 Overview

The SpaceWire - Time Distribution Protocol (SPWTDP) VHDL IP core is described in this user's manual.

The aim of the Time Distribution Protocol (TDP) is to synchronize time across a SpaceWire network. It does this by an initiator writing a CCSDS Time Code using an RMAP command placed in a SpaceWire packet, transferring it across the SpaceWire network and then extracting the CCSDS Time Code at the target, and by means of SpaceWire Time-Codes the time instant at which CCSDS Time Code becomes valid (synchronization event).

1.4 Implementation

The time distribution specification has been implemented as a synthesizable VHDL model which has been verified by means of simulation and FPGA based rapid prototyping has been used to facilitate early validation of the IP core. This user's manual has been established for the VHDL model (also referred to as VHDL IP core).

The implementation of the jitter and drift mitigation is based on a simple time interval measurement of the incoming SpaceWire Time-Codes using the local clock, gathering statistical information which is then used to calculate an average correction value that is applied to a locally generated signal which is free from jitter and local drift. The variance of the corrected locally generated signal is limited to one period of the local clock.

1.5 Verification

A VHDL test bench was developed to verify the functionality of the VHDL IP core. Emphasis has been placed on the following areas:

- CCSDS Time Code transmission using RMAP commands in SpaceWire packets
- Synchronization via Time-Codes
- Time-stamping of reception and transmission of Distributed Interrupts
- Latency offset adjustment
- Jitter mitigation
- Drift mitigation

During simulation the real-world data collected during independent ESA measurements have been used as stimuli to validate the jitter and drift mitigation technique implemented in the VHDL IP core.

FPGA based rapid prototyping has been used during the development of the VHDL IP core. The VHDL IP core has been integrated with a LEON3 32-bit SPARC processor in a system-on-chip design to facilitate early validation of the IP core with software in the loop.

1.6 Terms, definitions and abbreviated terms

AMBA	Advanced Microcontroller Bus Architecture
CCSDS	Consultative Committee for Space Data Systems
CUC	CCSDS Unsegmented Time Code
LSB	least significant bit
MSB	most significant bit
P-Field	preamble field
RASTA	Reference Avionics System Testbench Activity
RMAP	Remote Memory Access Protocol
SPW	SpaceWire
T-Field	time field
TDP	SpaceWire - Time Distribution Protocol
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

2 SpaceWire - Time Distribution Protocol

2.1 Overview

This interface implements the SpaceWire - Time Distribution Protocol (TDP). The protocol provides capability to transfer time values and synchronise them between onboard users of SpaceWire network. The time values are transferred as CCSDS Time Codes and synchronisation is performed through SpaceWire Time-Codes. The synchronisation is achieved through mitigating several factors like jitter, drift and latency. The core also provides datation service.The core operates in an AMBA APB bus system. The AMBA APB bus is used for configuration, control and status handling. The interface is coupled with a SpaceWire node with AMBA AHB master and RMAP target implementation.

2.2 Protocol

The initiator and target maintain their own time locally. The Time Distribution Protocol provides the means for transferring time of initiator to targets and for providing a synchronization point in time. The time is transferred by means of an RMAP write command carrying a CCSDS Time Code (time message). The synchronization event is signalled by means of transferring a SpaceWire Time-Code. The transfer of the SpaceWire Time-Code is synchronized with time maintained by the initiator. To distinguish which SpaceWire Time-Code is to be used for synchronization, the value of SpaceWire Time-Code is transferred from initiator to target by means of an RMAP write command prior to actual transmission of SpaceWire Time-Code itself. When there is more than one target the CCSDS Time Code need to be transferred to each individual target separately.

2.3 Functionality

The block diagram below explains the complete system.

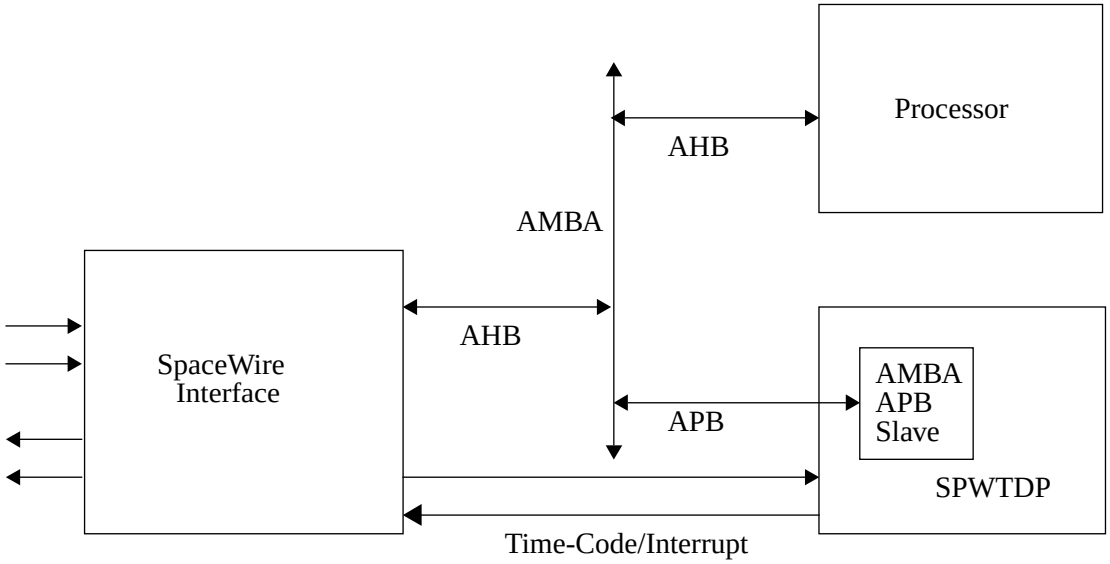


Figure 1. Block diagram

The system can act as initiator (time master) and target being able to send and receive SpaceWire Time-Codes. The initiator requires SpaceWire link interface implements an RMAP initiator. The Target requires SpaceWire link interface implements an RMAP target. The SPWTDP component is a part of this system providing SpaceWire Time-Codes, CCSDS Time Codes, datation, time-stamping of distributed interrupts, support for transmission of CCSDS Time Codes through RMAP and support for latency measurement and correction. In this implementation the CCSDS Time Codes carried

between the SpaceWire network is based on CCSDS Unsegmented Code format (CUC) which is explained below. The table below shows an example Preamble Field (P-Field) which corresponds to 40 bits of coarse time and 24 bits of fine time.

2.3.1 CCSDS Unsegmented Code: Preamble Field (P-Field)

Table 1. CCSDS Unsegmented Code P-Field definition

Bit	Value		Interpretation
0	“1”		Extension flag, P-Field extended with 2nd octet
1-3	“010”	Agency-defined epoch (Level 2)	Time code identification
4 5	“11”	(number of octets of coarse time) + 1	Detail bits for information on the code
6 7	“11”	(number of octets of fine time)	
8	“0”		Extension flag, P-Field not extended with 3rd octet
9-10	“01”	Number of additional octets of the coarse time.	added to octet 1
11-13	“000”	Number of additional octets of the fine time.	added to octet 1
14-15			RESERVED

2.3.2 CCSDS Unsegmented Code: Time Field (T-Field)

For the unsegmented binary time codes described herein, the T-Field consists of a selected number of contiguous time elements, each element being one octet in length. An element represents the state of 8 consecutive bits of a binary counter, cascaded with adjacent counters, which rolls over at a modulo of 256.

Table 2. Example CCSDS Unsegmented Code T-Field with 32 bit coarse and 24 bit fine time

CCSDS Unsegmented Code								
Preamble Field	Time Field							
	Coarse time				Fine time			
-	2^{31} 2^{24}	2^{23} 2^{16}	2^{15} 2^{8}	2^{7} 2^{0}	2^{-1} 2^{-8}	2^{-9} 2^{-15}	2^{-16} 2^{-24}	
0:15	0			31	32		55	

The basic time unit is the second. The T-Field coarse time (seconds) can be maximum 56 bits and minimum 8 bits. The T-Field fine time (sub seconds) can be maximum 80 bits and minimum of 0 bits. The coarse time code elements are a count of the number of seconds elapsed from the initial time value. This code is not UTC-based and leap second corrections do not apply according to CCSDS.

2.3.3 Time generation

The SPWTDP component consist of local time generator which is the source for time in this system. The initiator and target both have their respective local time generator. The local Elapsed Time (ET) counter is implemented complying with the CUC T-Field. The counter is incremented on the system clock only when enabled by the frequency synthesizer. The binary frequency required to determine the counter increment is derived from the system clock using a frequency synthesizer (FS). The frequency synthesizer is incremented with a pre-calculated increment value, which matches the available system clock frequency. The frequency synthesizer generates a tick every time it wraps around, which makes the ET time counter to step forward with the precalculated increment value. The output of frequency synthesizer is used for enabling the increment of ET counter. The increment rate of the ET counter and frequency synthesizer counter is set according to the system clock frequency. The core provides support for setting the increment rate of the ET counter as well as of the frequency synthesizer counter.

The number of bits representing coarse and fine time in a local time counter implemented in a design can be obtained by reading the DPF bits of Datation Preamble Field register.

2.3.4 Initiator

An initiator is a SpaceWire node distributing CCSDS Time Codes and SpaceWire Time-Codes. It is also an RMAP initiator, capable of transmitting RMAP commands and receiving RMAP replies. There is only one active initiator in a SpaceWire network during a mission phase.

The initiator performs the following tasks

- Transmission of SpaceWire Time-Codes

The SpaceWire Time-Codes are provided by SPWTDP component and transmission of those codes to targets should be performed by a SpaceWire interface.

- Transmission of CCSDS Time Codes through RMAP
- Datation, time-stamping and latency measurement

2.3.5 Target

A target is a SpaceWire node receiving CCSDS Time Codes and SpaceWire Time-Codes. A target is also an RMAP target, capable of receiving RMAP commands and transmitting RMAP replies. There can be one or more targets in a SpaceWire network.

The target performs the following tasks

- Reception of SpaceWire Time-Codes

The SpaceWire Time-Codes sent from initiator are received by SpaceWire interface and provided to SPWTDP component in target.

- Reception of CCSDS Time Codes through RMAP
- Qualification of received time messages (CCSDS Time Codes) using SpaceWire Time-Codes
- Initialization and Synchronisation of received CCSDS Time Codes with local Elapsed Time counter in SPWTDP component
- Datation, time-stamping and latency correction
- Jitter and drift mitigation

2.3.6 Configuring initiator and target

The SPWTDP component is interfaced via an AMBA Advanced Peripheral Bus (APB) slave interface, providing a register view that is compatible with the Time Distribution Protocol (TDP). The SPWTDP component in initiator and target must be configured separately. The target time value is configured with time values available in the initiator. The targets register space must be configured and controlled through RMAP to achieve time synchronisation.

- Initializing initiator

The initiator transmits the SpaceWire Time-Codes only when the SPWTDP component is initialized. Initialization is done by writing a time value into the Command Elapsed Time registers available in the command field, the NC bit in the Control register of command field should be enabled to initialize the time value stored in the Command Elapsed Time registers to be the local time (Transmit Enable TE bit in Configuration 0 register must be enabled). The NC bit in the Control register will disable itself when the time is initialized. The INSYNC bit in Status 0 register will enable when initialization is performed. The MAPPING bits in Configuration 0 register determines the interval between SpaceWire Time-Code transmissions which is explained in detail in the section below.

2.3.7 SpaceWire Time-Code

SpaceWire Time-Codes are continuously transmitted from an initiator node (time master) to all slave nodes. The transmission of the SpaceWire Time-Code is synchronized with the local ET counter in the initiator node. The six bits of the Time-Code time information correspond to six bits of the local ET counter (MAPPING bits in Configuration 0 register determines its exact mapping and interval between SpaceWire Time-Code transmissions). Value of 0b00000 for MAPPING bits in Configuration 0 register will send SpaceWire Time-Code at every Second. When the value is 0b00001 SpaceWire Time-Codes are sent at every 0.5 Seconds interval and so on (maximum value of MAPPING can be 0b11111 but this value cannot be more than the number of bits implemented as fine time). The ET bits with lower weights than the size bits mapped to Time Codes time information bits are all zero at time of SpaceWire Time-Codes transmission. The Table below shows an example Local ET counter and Mapping. If the Coarse time is 32 bits and Fine time is 24 bits and mapping value is 6 then 0 to 31 is coarse(32 bits), 32 to 55 is fine time and mapped SpaceWire Time-Code is 32 to 37.

Table 3. Example Local ET counter with Mapping values

0															25	26	27	28	29	30	31	32	33	34	35	36	37	38				55
Mapping Values																					0	1	2	3	4	5	6	7				24
If the Mapping value is 6 then the mapped SpaceWire Time-Codes is 32 to 37																						32	33	34	35	36	37					
If the Mapping value is 0 then the mapped SpaceWire Time-Codes is 26 to 31																26	27	28	29	30	31											
If the Mapping value is 5 then the mapped SpaceWire Time-Codes is 31 to 36																					31	32	33	34	35	36					
If the Mapping value is 7 then the mapped SpaceWire Time-Codes is 33 to 38																							33	34	35	36	37	38				

2.3.8 Initialization and synchronisation of target through RMAP

The SPWTC in Control register of initiator SPWTDP component should be configured initially with a SpaceWire Time-Code value at which the time message needed to be transferred. When the SpaceWire Time-Code generated internally using the local ET counter matches the SPWTC in Control register a Time Message TM interrupt will be generated (TME bit Time Message Enable should be enabled in the Interrupt Enable register). Based on this interrupt the local time in initiator should be accessed from the Datation registers and used to calculate the time message needed to be transmitted.

- Time message generation

The Time message transmitted using RMAP should be an exact mapping of the Command field. The Time message transmitted should write the Command field available in target. Control register available in Command field specify weather the target should be initialized or synchronized, at which SpaceWire Time-Codes it should happen (synchronization event) and details of coarse and fine time available in the time message. The New code NC bit available in Control register should be enabled and if the target should be initialized then Init Sync IS bit in Control register must be enabled otherwise target will be synchronized.

The Command Elapsed Time in time message are calculated from the local time available in the initiator. The local time can be obtained by reading the Datation Field of SPWTDP component. While reading the Datation registers always the total implemented coarse time and fine time must be read. The DPF of Datation Preamble Field register gives the coarse and fine time implemented which gives the total local ET counter (coarse + fine width). For example if the implementation has 32 bit coarse and 24 bit fine time then it is enough to access the first two Datation Elapsed Time registers (0 and 1). The 32 bits of Datation Elapsed Time 0 and only the most significant 24 bits (31 to 8) of Datation Elapsed Time 1 registers (32 + 24 =56 bits) represents the local time. These 56 bits only be used for Command Elapsed time (time message) calculation.

The SpaceWire Time-Codes at which the Time Message interrupt generated is embedded in the local ET counter. The Command Elapsed time which is transmitted as time message should be an incremented time value of this SpaceWire Time-Code and ET bits with lower weights than the size bits mapped to SpaceWire Time-Code time information bits are all must be zero. The incremented time value is to make the initialization or synchronisation of time message in target will happen after the reception of qualifying SpaceWire Time-Codes. The qualifying SpaceWire Time-Code is embedded in the Command Elapsed time (part of time message) sent from initiator. This qualifying SpaceWire Time-Code value should also be written in the SPWTC in Control section of the time message.

- Time qualification in target

In target, the Command field will contain the time message when it is written by the initiator through RMAP. When the SPWTC of Control register in Command field matches with a received SpaceWire Time-Code then initialization or synchronization will occur (according to NC bit and IS bit in the Control register) to the local ET counter of the target SPWTDP component. When the local ET counter is initialized or synchronized the NC bit in the control register will disable itself. The INSYNC bit in Status 0 register will enable when initialization is performed specifying the target is initialized. Initialization completely writes time message values into the implemented local Elapsed time counter and synchronisation verifies whether the time message Command Elapsed Time and local Elapsed Time counter matches till the mapped SpaceWire Time-Code level (with a tolerance of previous value) and only modifies the local Elapsed Time if their is a mismatch.

2.3.9 Latency measurement using Time-Stamps

The incoming and outgoing SpaceWire Distributed Interrupts are time stamped in initiator and target. The initiator calculates latency based on these time stamp values. The time stamped values in target are accessed from initiator through RMAP. The Latency Enable LE bit in Configuration 0 register must be enabled between the two nodes in the SpaceWire network for which the latency is to be calculated. The distributed interrupt transmission from initiator (which is the origin for latency calculation) is controlled by a mask register STM available in Configuration 3 register and SpaceWire time code register TSTC available in Time-Stamp SpaceWire Time-Code and Preamble Field Tx register, these registers specifies how often and at which time code distributed interrupt is transmitted and time stamping is performed.

The time stamping can be performed in two methods (only Interrupts or Interrupts and Acknowledgement), the DI bit in Configuration 3 register of SPWTDP component in target should be configured to specify which type of method is used. If only distributed interrupts (no acknowledgement) are used then DI bit should be 0. The transmitted and received distributed interrupts INTX and INRX in the Configuration 0 registers of both initiator and target must be configured with the interrupt number which will be used for the latency measurement. For example if the INTX in initiator Configuration 0 is configured with 0b00100 then the target INRX should be configured with the same value. Similarly if the INTX in target Configuration 0 is configured to be 0b00101 then the initiator INRX should be configured with the same value. Initially initiator sends a distributed interrupt when the conditions are matched (STM and TSTC registers match) and when the target received this distributed interrupt it will send another interrupt which will be received by the initiator. At each end transmission and reception is time stamped (current local time is stored in Time Stamp registers) and interrupt transmitted is INTX and received interrupt is checked whether it received INRX.

If both distributed interrupts and acknowledgement method is to be used then DI bit should be 1. The transmitted and received distributed interrupts INTX and INRX in the Configuration 0 registers of both initiator and target can have the same interrupt number (the acknowledgement number for a particular interrupt will be same as interrupt number). Similar to the previous method at each end transmission and reception is time stamped which will be used for latency calculations.

The Latency calculation can be started in initiator based on DIR (distributed interrupt received) interrupt available in Interrupt Status register (the interrupt should be enabled in the Interrupt Enable register). The latency is calculated form the time stamp registers based on the equation explained below

Latency = ((initiator time stamp Rx - initiator time stamp Tx) - (target time stamp Tx - target time stamp Rx)) /2

By calculating the Latency value repeatedly (at least for about 128 times, more number of times provides increased accuracy) and taking an average of it will provide the final latency value. The initiator should transfer the latency correction information to the Latency Field registers in the target by means of RMAP transfer. When the latency values are written it will be adjusted to local time in the target.

2.3.10 Mitigation of jitter and drift

The Jitter and drift mitigation is performed in target when Jitter Enable JE and Mitigation Enable ME bit in Configuration 0 register is enabled. The process will only start when the target is initialized (the target local ET counter should have initialized through a time message from initiator and INSYNC bit in Status 0 register is enabled).

2.4 Data formats

All Elapsed Time (ET) information is compliant with the CCSDS Unsegmented Code defined in [CCSDS] and repeated hereafter.

2.4.1 Numbering and naming conventions

Convention according to the CCSDS recommendations, applying to time structures:

- The most significant bit of an array is located to the left, carrying index number zero.
- An octet comprises eight bits.

Table 4. CCSDS n-bit field definition

CCSDS n-bit field		
most significant		least significant
0	1 to n-2	n-1

Convention according to AMBA specification:

- The least significant bit of an array is located to the right, carrying index number zero.
- Big-endian support.

Table 5. AMBA n-bit field definition

AMBA n-bit field		
most significant		least significant
n-1	n-2 down to 1	0

2.5 Registers

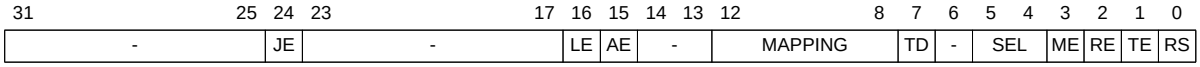
The core is programmed through registers mapped into AMBA APB address space.

Table 6. Registers

APB address offset	Register
0x00-0x0F	Configuration Field
0x00	Configuration 0
0x04	Configuration 1
0x08	Configuration 2
0x0c	Configuration 3
0x10 - 0x1F	Status Field
0x10	Status 0
0x14	Status 1
0x18	RESERVED
0x1c	RESERVED
0x20 - 0x3F	Command Field
0x20	Control
0x24	Command Elapsed Time 0
0x28	Command Elapsed Time 1
0x2C	Command Elapsed Time 2
0x30	Command Elapsed Time 3
0x34	Command Elapsed Time 4
0x38	RESERVED
0x3C	RESERVED
0x40 - 0x5F	Datation Field
0x40	Datation Preamble Field
0x44	Datation Elapsed Time 0
0x48	Datation Elapsed Time 1
0x4C	Datation Elapsed Time 2
0x50	Datation Elapsed Time 3
0x54	Datation Elapsed Time 4
0x58	RESERVED
0x5C	RESERVED
0x60 - 0x9F	Time-Stamp Field
0x60	Time-Stamp Preamble Field Rx
0x64	Time-Stamp Elapsed Time 0 Rx
0x68	Time-Stamp Elapsed Time 1 Rx
0x6C	Time-Stamp Elapsed Time 2 Rx
0x70	Time-Stamp Elapsed Time 3 Rx
0x74	Time-Stamp Elapsed Time 4 Rx
0x78	RESERVED
0x7C	RESERVED
0x80	Time-Stamp SpaceWire Time-Code and Preamble Field Tx
0x84	Time-Stamp Elapsed Time 0 Tx

APB address offset	Register
0x88	Time-Stamp Elapsed Time 1 Tx
0x8C	Time-Stamp Elapsed Time 2 Tx
0x90	Time-Stamp Elapsed Time 3 Tx
0x94	Time-Stamp Elapsed Time 4 Tx
0x98	RESERVED
0x9C	RESERVED
0xA0-0xBF	Latency Field
0xA0	Latency Preamble Field
0xA4	Latency Elapsed Time 0
0xA8	Latency Elapsed Time 1
0xAC	Latency Elapsed Time 2
0xB0	Latency Elapsed Time 3
0xB4	Latency Elapsed Time 4
0xB8	RESERVED
0xBC	RESERVED
0xC0	Interrupt Enable
0xC4	Interrupt Status

Table 7. Configuration 0



31: 25	RESERVED	
24:	AE	Jitter Correction Enable (only for target) The jitter correction process in target will start when this bit is enabled. (Mitigation Enable bit should also be enabled). Reset value: ‘0’.
23: 17	RESERVED	
16:	LE	Latency Enable. To calculate latency between an initiator and target this bit must be enabled in both of them. Reset value: ‘0’.
15:	AE	AMBA Interrupt Enable The interrupts (explained in interrupt registers) in this core will generate an AMBA interrupt only when this bit is enabled. Reset value: ‘0’
14 13	RESERVED	
12: 8	MAPPING	Defines mapping of SpaceWire Time-Codes versus CCSDS Time-code. Value 0b00000 will send SpaceWire Time-Codes every Second, Value 0b00001 will send SpaceWire Time-Codes every 0.5 Second, Value 0b00010 will send SpaceWire Time-Codes every 0.25 Second, Value 0b00011 will send SpaceWire Time-Codes every 0.125 Second The maximum value it can take is 0b11111 but this value cannot be more than the number of bits implemented as fine time. Reset value: Implementation dependent.
7:	TD	Enable TDP protocol when set, else CUCTP protocol. Reset value: ‘0’.
6:	RESERVED	
5: 4	SEL	Select for SpaceWire Time-Codes and Distributed Interrupt transmission and reception, one of 0 through 3. Reset value: 0b00
3:	ME	Mitigation Enable (only for target) The drift correction process in target will start when this bit is enabled. Reset value: ‘0’.
2:	RE	Receiver Enable (only for target) Reset value: ‘0’.
1	TE	Transmit Enable (only for initiator) Reset value: ‘0’.
0	RS	Reset core. Makes complete reset when enabled. Reset value: ‘0’.

All implemented registers are writable and readable.

Table 8. Configuration 1

31: 30	RESERVED	
29: 0	FSINC	Increment value of the Frequency Synthesizer which is added to the counter every system clock cycle. It defines the frequency of the synthesized reference time. Reset value: Implementation dependent

All implemented registers are writable and readable.

Table 9. Configuration 2

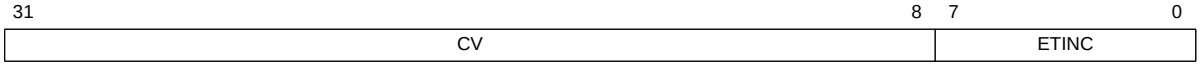
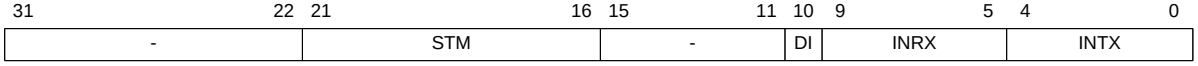


Table 9. Configuration 2

31: 8	CV	Compensation Value Value added to FSINC for variations of drift of the target clock.(only for target) Reset value: Implementation Dependent
7: 0	ETINC	Value of the Elapsed Time counter is to be incremented each time when the Frequency Synthesizer wraps around. Reset value: Implementation dependent

All registers are writable and readable.

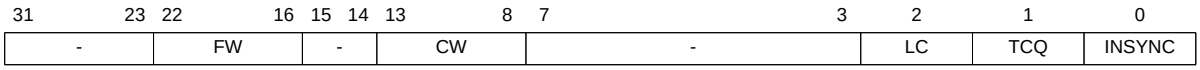
Table 10. Configuration 3



31: 22	RESERVED	
21: 16	STM	SpaceWire Time-Code Mask Mask For TSTC register available at Time-Stamp SpaceWire Time-Code and Preamble Field Tx register. Value all bits zero will send Distributed interrupts at all SpaceWire Time-Codes irrespective of any values in TSTC register. Value all ones will send Distributed interrupts at complete match of SpaceWire Time-Code with TSTC register. (only for initiator) Reset value: 0b000000
15: 11	RESERVED	
10:	DI	Distributed Interrupt method, when set interrupt and acknowledge mode else only interrupt mode. (only for target) Reset value: ‘0’
9: 5	INRX	Interrupt Received.(Distributed) The distributed interrupt number received by initiator or target. Reset value: 0b000000
4: 0	INTX	Interrupt Transmitted.(Distributed) The distributed interrupt number transmitted by initiator or target. Reset value: 0b000000

All implemented registers are writable and readable.

Table 11. Status Register 0



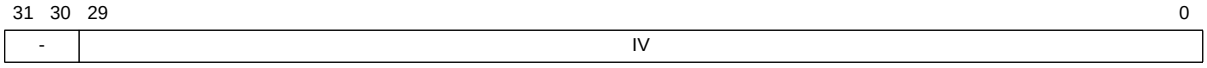
31 23:	RESERVED	
22: 16	FW	Fine width of command CCSDS Time Code received. Calculated from Preamble field of Command Register. Reset value: 8
15: 14	RESERVED	
13: 8	CW	Coarse width of command CCSDS Time Code received, calculated from Preamble field of Command Register. Reset value: 0
7: 3	RESERVED	
2	LC	Latency Corrected (only for target) Reset value: ‘0’
1	TCQ	Time message is qualified by SpaceWire Time-Codes Reset value: ‘0’

Table 11. Status Register 0

0	INSYNC	In Sync at Time code level, enabled when time values are Initialized or Synchronized Reset value: ‘0’

All implemented registers are only readable.

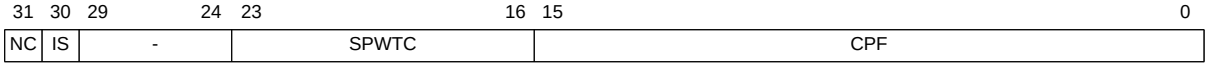
Table 12. Status Register 1



31: 30	RESERVED	
29: 0	IV	Increment Variation. The variation in FSINC while achieving the time synchronisation (only for target) Reset value: Implementation dependent

All implemented registers are only readable.

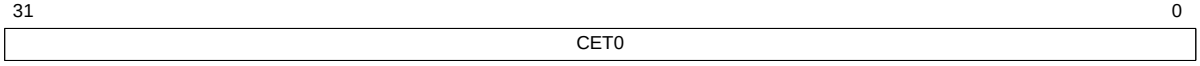
Table 13. Control



31:	NC	New Command Reset value: ‘0’
30:	IS	Init or Sync 1 Initialization of received time message 0 Synchronisation of received time message (only for target) Reset value: ‘0’
29: 24	RESERVED	
23: 16	SPWTC	Spacewire Time-code value used for initialization and synchronisation In initiator the SpaceWire Time-Codes generated internally using the local ET counter matches this register a Time Message TM interrupt will be generated which is used to send Time message over the SpaceWire network. In target this register should match the received SpaceWire Time-code for time qualification. Reset value: 0
15: 0	CPF	Command Preamble Field. The number of coarse and fine time available in Command Elapsed Time registers should be mentioned in this field. Based on this preamble field the target will initialize or synchronise the local ET counter.(only for target) Reset value:0

All implemented registers are writable and readable.

Table 14. Command Elapsed Time 0



31: 0	CET0	Command Elapsed Time 0 Initialize or Synchronise local ET counter value (0 to 31).Reset value: 0

All registers are writable and readable.

Table 15. Command Elapsed Time 1

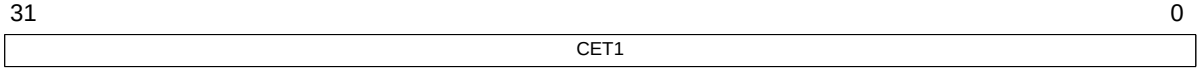


Table 15. Command Elapsed Time 1

31: 0	CET1	Command Elapsed Time 1 Initialize or Synchronise local ET counter value (32 to 63) Reset value: 0

All registers are writable and readable.

Table 16. Command Elapsed Time 2

31	0
CET2	

31: 0	CET2	Command Elapsed Time 2 Initialize or Synchronise local ET counter value (64 to 95). Reset value: 0

All registers are writable and readable.

Table 17. Command Elapsed Time 3

31	0
CET3	

31: 0	CET3	Command Elapsed Time 3 Initialize or Synchronise local ET counter value (96 to 127). Reset value: 0

All registers are writable and readable.

Table 18. Command Elapsed Time 4

31	24	23	0
CET4		-	

31: 24	CET4	Command Elapsed Time 4 Initialize or Synchronise local ET counter value (128 to 135). Reset value: 0
23: 0	RESERVED	

All registers are writable and readable.

Table 19. Datation Preamble Field

31	16	15	0
-		DPF	

31: 16	RESERVED	
15: 0	DPF	Datation Preamble Field The number of coarse and fine time implemented can be obtained from this Preamble Field.Reset value: 0x2f00

All registers are only readable.

Table 20. Datation Elapsed Time 0

31	0
DET0	

31: 0	DET0	Datation Elapsed Time 0 CCSDS Time Code value (0 to 31) of local ET counter value. Reset value: 0

All registers are only readable.

Table 21. Datation Elapsed Time 1

31	0
DET1	

Table 21. Datation Elapsed Time 1

31: 0	DET1	Datation Elapsed Time 1 CCSDS Time Code value (32 to 63) of local ET counter value. Reset value: 0

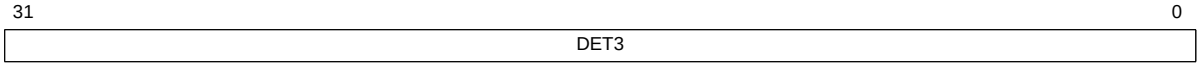
All registers are only readable.

Table 22. Datation Elapsed Time 2

31: 0	DET2	Datation Elapsed Time 2 CCSDS Time Code value (64 to 95) of local ET counter value. Reset value: 0

All registers are only readable.

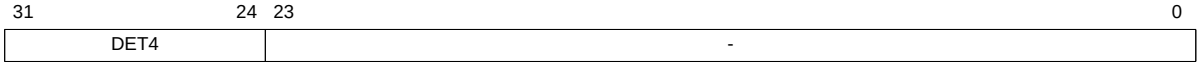
Table 23. Datation Elapsed Time 3



31: 0	DET3	Datation Elapsed Time 3 CCSDS Time Code value (96 to 127) of local ET counter value. Reset value: 0

All registers are only readable.

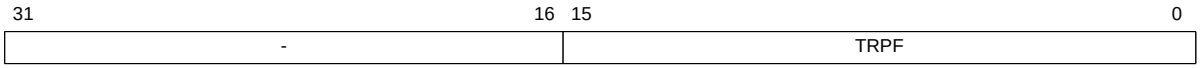
Table 24. Datation Elapsed Time 4



31: 24	DET4	Datation Elapsed Time 4 CCSDS Time Code value (128 to 135) of local ET counter value. Reset value: 0
23: 0	RESERVED	

All implemented registers are only readable.

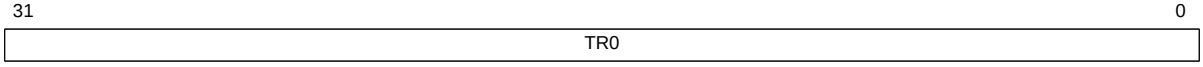
Table 25. Time-Stamp Preamble Field Rx



31: 16	RESERVED	
15: 0	TRPF	Time stamp Preamble Field The number of coarse and fine time implemented can be obtained from this Preamble Field.Reset value: 0x2f00

All implemented registers are only readable.

Table 26. Time Stamp Elapsed Time 0 Rx



31: 0	TR0	Time stamped local ET value (0 To 31) when distributed interrupt received. Reset value: 0

All registers are only readable.

Table 27. Time Stamp Elapsed Time 1 Rx

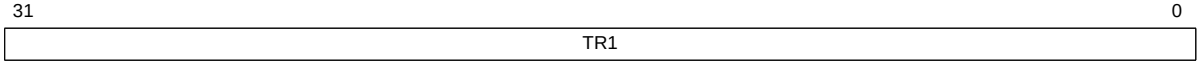
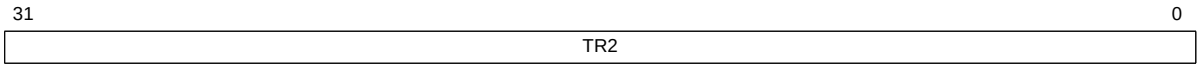


Table 27. Time Stamp Elapsed Time 1 Rx

31: 0	TR1	Time stamped local ET value (32 to 63) when distributed interrupt received. Reset value: 0

All registers are only readable.

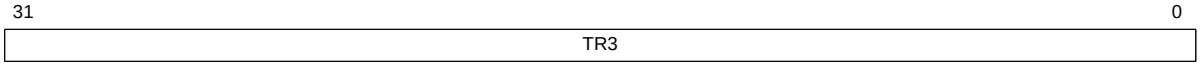
Table 28. Time Stamp Elapsed Time 2 Rx



31: 0	TR2	Time stamped local ET value (64 to 95) when distributed interrupt received. Reset value: 0

All registers are only readable.

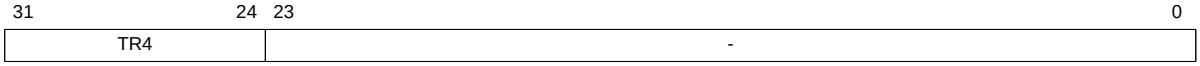
Table 29. Time Stamp Elapsed Time 3 Rx



31: 0	TR3	Time stamped local ET value (96 to 127) when distributed interrupt received. Reset value: 0

All registers are only readable.

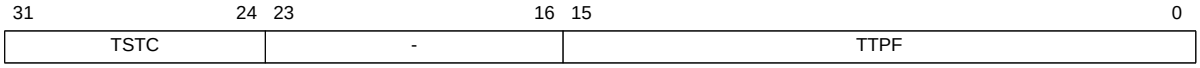
Table 30. Time Stamp Elapsed Time 4 Rx



31: 24	TR4	Time stamped local ET value (128 to 135) when distributed interrupt received. Reset value: 0
23: 0	RESERVED	

All registers are only readable.

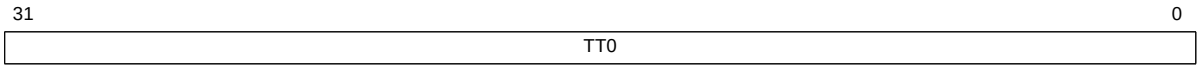
Table 31. Time-Stamp SpaceWire Time-Code and Preamble Field Tx



31: 24	TSTC	Time stamp time code Time stamp on this time-code value, used for time stamping when this register matched with SpaceWire Time-Codes. The mask for this matching is available in configuration register 3. Reset value: 0 (only for initiator)
23: 16	RESERVED	
15: 0	TTPF	Time stamp Preamble Field The number of coarse and fine time implemented can be obtained from this Preamble Field. Reset value: 0x2f00

TSTC is writable and readable, PFIELD is only readable

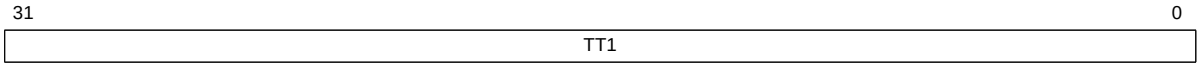
Table 32. Time Stamp Elapsed Time 0 Tx



31: 0	TT0	Time stamped local ET value (0 to 31) when distributed interrupt transmitted. Reset value: 0

All registers are only readable.

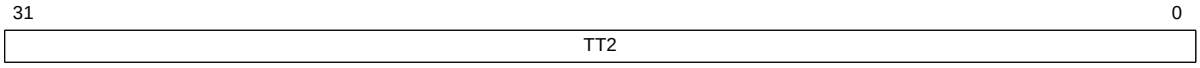
Table 33. Time Stamp Elapsed Time 1 Tx



31: 0 TT1 Time stamped local ET value (32 to 63) when distributed interrupt transmitted. Reset value: 0

All registers are only readable.

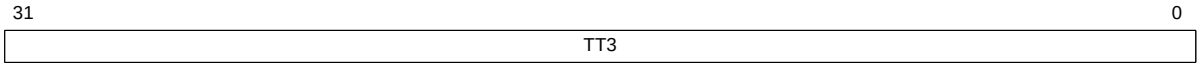
Table 34. Time Stamp Elapsed Time 2 Tx



31: 0 TT2 Time stamped local ET value (64 to 95) when distributed interrupt transmitted. Reset value: 0

All registers are only readable.

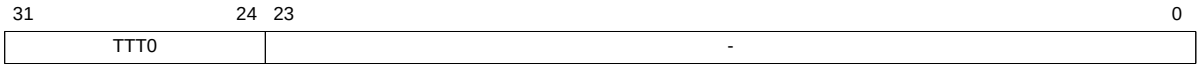
Table 35. Time Stamp Elapsed Time 3 Tx



31: 0 TT3 Time stamped local ET value (96 to 127) when distributed interrupt transmitted. Reset value: 0

All registers are only readable.

Table 36. Time Stamp Elapsed Time 4 Tx

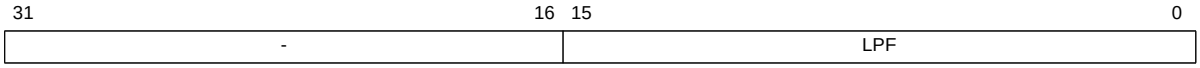


31: 24 TT4 Time stamped local ET value (128 to 135) when distributed interrupt transmitted. Reset value: 0

23: 0 RESERVED

All implemented registers are only readable.

Table 37. Latency Preamble Field



31: 16 RESERVED

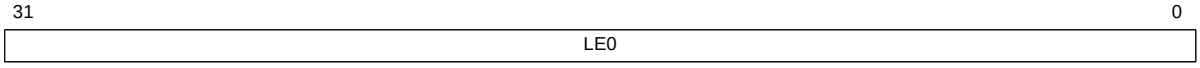
15: 0 LPF Latency Preamble Field

The number of coarse and fine time implemented can be obtained from this Preamble Field.Reset value: 0x2f00

(only for target)

All implemented registers are only readable.

Table 38. Latency Elapsed Time 0

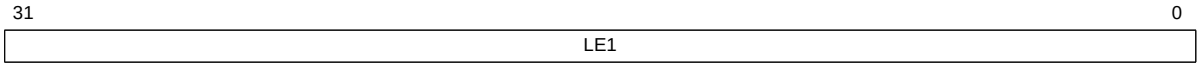


31: 0 LE0 Latency Value (0 to 31) written by initiator. Reset value: 0

(only for target)

All registers are writable and readable.Reset value: 0

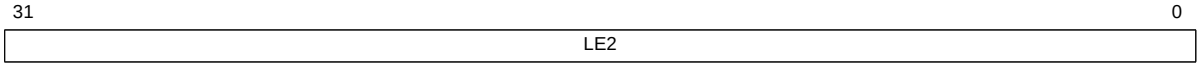
Table 39. Latency Elapsed Time 1



31: 0 LE1 Latency Value (32 to 63) written by initiator. Reset value: 0
(only for target)

All registers are writable and readable.Reset value: 0

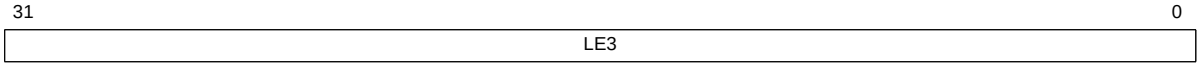
Table 40. Latency Elapsed Time 2



31: 0 LE2 Latency Value (64 to 95) written by initiator. Reset value: 0
(only for target)

All registers are writable and readable.Reset value: 0

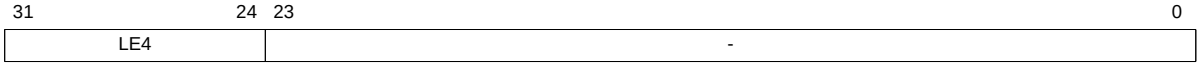
Table 41. Latency Elapsed Time 3



31: 0 LE3 Latency Value (96 to 127) written by initiator. Reset value: 0
(only for target)

All registers are writable and readable.Reset value: 0

Table 42. Latency Elapsed Time 4

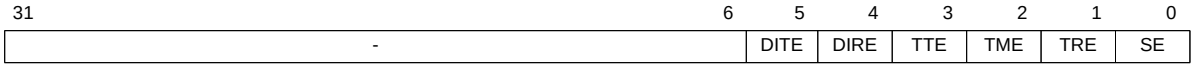


31: 24 LE4 Latency Value (128 to 135) written by initiator. Reset value: 0
(only for target)

23: 0 RESERVED

All implemented registers are writable and readable. Reset value: 0

Table 43. Interrupt Enable



31: 6	RESERVED	
5	DITE	Distributed Interrupt Transmitted Interrupt Enable
4	DIRE	Distributed Interrupt Received Interrupt Enable
3	TTE	SpaceWire Time-Code Transmitted Interrupt Enable (only for initiator)
2	TME	Time Message transmit Interrupt Enable (only for initiator)
1	TRE	SpaceWire Time-Code Received Interrupt Enable (only for target)
0	SE	Sync Interrupt Enable (only for target)

All implemented bits are writable and readable.

All bits are Reset to 0b after Reset,

Table 44. Interrupt Status

31	6	5	4	3	2	1	0
-		DIT	DIR	TT	TM	TR	S

31: 6	RESERVED	
5	DIT	Generated when distributed interrupt is transmitted (Latency calculation should be enabled)
4	DIR	Generated when distributed interrupt is Received (Latency calculation should be enabled)
3	TT	Generated when SpaceWire Time-Codes is transmitted (only for initiator)
2	TM	Generated when the conditions for transmitting time message occurred, based on this time message should be transmitted from initiator (only for initiator)
1	TR	Generated when SpaceWire Time-Code is received (only for target)
0	S	Generated when the target is initialized or synchronized with initiator (only for target)

All bits are Reset to 0b after Reset,

The interrupts are cleared by writing value 1 on respective bits. All implemented bits are writable and readable.

Note: The registers which are not mentioned either as only for initiator or target are used in both initiator and target.

2.6 Configuration options

Table below shows the configuration options of the core (VHDL generics).

Table 45. Configuration options

Generic	Description	Allowed range	Default
mitigation	Jitter and drift correction enable	0-1	0
tech	unused	0	0
gCoarse	Number of CUC coarse bits	8-56	32
gFine	Number of CUC fine bits	0-80	24
gFrequency	Frequency Synthesizer	2-30	30
gFSIncrement	Increment of FS counter	0-0x3FFFFFFF	360287970
gETIncrement	Increment of ET counter	0-255	0
gComp	Compensation value for jitter and drift variations	0-0xFFFFFF	461
gPField	P-Field	0-0xAF7C	0x2F00
gMapping	Initial mapping value	0-31	6
gMaster	Initiator implementation enable	0-1	0
gSlave	Target implementation enable	0-1	0
delay	Delay induced between SpaceWire Time-Codes and Distributed Interrupts transmission in system clock units. The delay introduced is (2^delay) multiplied by system clock time duration.	2-15	9
mul	unused	0-1	0
pindex	APB slave index	0-15	0

2.7 Signal descriptions

Table below shows the interface signals of the core (VHDL ports).

Table 46. Signal descriptions

Signal name	Field	Type	Function	Active
RSTN	N/A	Input	Reset	Low
CLK	N/A	Input	Clock	-
APBI	*		APB slave input signals	-
	psel	Input	Slave select	-
	penable	Input	Strobe	High
	paddr	Input	Address bus (byte)	-
	pwrite	Input	Write	High
	pwdata	Input	Write data bus	-
APBO	*		APB slave output signals	-
	prdata	Output	Read data bus	-
TDPI	tickindone	Input	SpaceWire Time-Code input processed	High
	tickoutraw	Input	SpaceWire Time-Code/Distributed interrupt output event	High
	timeout	Input	SpaceWire Time-Code/Distributed interrupt output	-
TDPO	tickinraw	Output	SpaceWire Time-Code/Distributed interrupt input request	High
	timein	Output	SpaceWire Time-Code/Distributed interrupt input	-
	elapsedtime	Output	Elapsed time	-
	irq	Output	Interrupt event	High
	enable	Output	TDP enable	High
diag_ctick	N/A	Output	This tick is generated when SpaceWire Time-Code is transmitted in initiator. This tick is generated when a diagnostic SpaceWire Time-Code is generated from targets fully corrected time.	High
diag_jtick	N/A	Output	The incoming SpaceWire Time-Code tick, to visualize the jitter (only for target)	High

3 Implementation

3.1 Overview:

The Local time available in an initiator is transmitted to synchronize time across a SpaceWire network. It is done by an initiator writing a CCSDS Time Code (time message) using an RMAP command placed in a SpaceWire packet, transferring it across the SpaceWire network and then extracting the CCSDS Time Code at the target, and by means of SpaceWire Time-Codes used to convey the time instant at which the CCSDS Time Code becomes valid (synchronization event). The source for the CCSDS Time Codes and SpaceWire Time-Codes are local time generator available in the SPWTDP component. The SpaceWire Time-Code is driven from local time in the originating node (initiator) and received SpaceWire Time-Code is to be correlated with local time available in the terminating node (target). The transmission and reception of these time codes suffer from time distribution delay (or latency) and jitter in a system. Further the local time is generated using the local clock available in initiator and target, the oscillator or crystal used to generate the local clock may not only have an incorrect frequency, it may also experience frequency variations over time (drift), which will directly affect the time keeping in a system. The SPWTDP component implements latency, jitter and drift mitigation in target for achieving accuracy in time maintained between initiator and target.

3.2 Operation

The sections below explains the mitigation process in the target.

3.2.1 Generation of local time from local clock

The local Elapsed Time (ET) is incremented with a pre-calculated increment value (ETINC). The frequency synthesizer (FS) is incremented with a pre-calculated increment value (FSINC), which matches the available system clock frequency. The FS simply generates a tick every time it wraps around, which makes the ET to step forward with the pre-calculated increment value. The output of FS is used for enabling the increment of ET. The ETINC and FSINC values needed for a particular system clock frequency can be obtained from the Spreadsheet provided along with this document.

$$FSINC = ((2 \wedge FSW) * (2 \wedge FTW)) / F \quad .....(1)$$

FTW - Fine time width

FSW - Frequency Synthesizer width

FSINC - pre-calculated increment value

F - Frequency of the local node

3.2.2 Compensation value calculations

Frequency synthesizer clocked by the local clock drives the local time at a given rate. By changing the frequency synthesizer settings one can adjust the local time. The coupling between local clock and the local time (frequency synthesizer increment value FSINC) is adjusted to the amount of variations seen in the target due to drift or jitter. The variations are obtained in local clock count and multiplied with a fixed value (compensation value CV) and added or subtracted to frequency synthesizer according to sign. The compensation value is obtained from equation 2 which is derived from equation 1 taking time code interval into account.

$$CV = (1-(1/ (1+(D*FSINC)/(2\wedge FSW)*((2\wedge FTW)/ N)))) * FSINC \quad ...(2)$$

CV - value added or subtracted to the frequency synthesizer

D - variations in local clock count

FSINC - pre-calculated increment value

FSW - Frequency Synthesizer width

FTW - Fine time width

N - Number of SpaceWire Time codes per second.

The CV, ETINC and FSINC values needed for a particular system can be obtained from the Spreadsheet provided along with this document. These values should be provided for a particular design as VHDL generics (gComp, gETIncrement, gFSIncrement) and the registers implemented in the design for these values can be programmable through AMBA APB bus. The VHDL generics are the default value for these registers.

3.2.3 Calculating variations in terms of local clock count

The correction needed to be performed for time synchronization are initial offset difference in local oscillator (incorrect frequency), jitter and drift variations. The variations are calculated as differences in local ticks and external ticks. The external ticks are provided by the SpaceWire interface to the SPWTDP component when SpaceWire Time Codes are received from a remote initiator. The local ticks in target are provided internally in the SPWTDP component, it happens when a local SpaceWire Time Code is generated (generated from the local time according to the mapping value) which is only used for internal calculations.

- Initial offset difference

The number of local clock counts between two local ticks is obtained, similarly number of local clock counts between two external ticks are obtained. These two values are subtracted and the difference is collected and averaged to get a variation value in local clock count. The variation obtained is based on the local clock of the target node. The variation is multiplied with compensation value and provided to frequency synthesizer to get the initial offset difference corrected.

- Phase correction and initial jitter correction

The phase difference resulted from initial offset difference and initial jitter corrections are combined together. The number of local clock difference between a local tick and external tick is measured and multiplied with compensation value to obtain the correction value. The correction value is added or subtracted to the frequency synthesizer according to the sign of correction needed. This results in removal of phase variation resulted from initial offset difference and partially gets the local ticks near the center of external ticks (jitter center).

- Jitter correction and drift mitigation

The number of clock ticks between local tick and external tick is obtained and averaged over a period. The averaged value is multiplied with compensation value to obtain the correction value and fed into frequency synthesizer and added or subtracted according to sign. The main aspect of jitter correction is to keep the local tick in the center of arriving external ticks (or jitter free), the correction value is immediately applied and the local tick is got back to the center, further the correction term is equally distributed to the entire jitter correction interval. The variation in local clock drift is seen as local ticks movement from the center which is caught by the averaging process and correction values are fed back as explained above. This will keep the local ticks jitter and drift free.

- Latency correction

Finally the Latency correction information provided by the initiator is added with the local time to get the time synchronized. The latency correction do not perform any variations to the frequency synthesizer.

3.3 Time keeping complete process

The initiator initiates the target node through a time message transfer, calculates latency using distributed interrupts and provides latency correction value and starts the jitter and drift mitigation process in the target. The initiator can also send synchronization time messages at regular intervals and the target checks the local time with the received synchronization time messages and adjust the target time if any variations. The transmission of SpaceWire Time-Codes at regular intervals helps to correct any clock drift in the target.

4 Verification

4.1 Overview

A VHDL test bench was developed to verify the functionality of the VHDL IP core (SPWTDP). The Jitter and drift correction unit a part of the SPWTDP is verified separately using a VHDL test bench before integrating into the system. FPGA based rapid prototyping has been used during the development of the VHDL IP core. The VHDL IP core has been integrated with a LEON3 32-bit SPARC processor in a system-on-chip design to facilitate early validation of the IP core with software in the loop.

4.2 Verification of functionality

A VHDL test bench was developed to verify the functionality of the VHDL IP core. Emphasis has been placed on the following areas:

- CCSDS Time Code transmission using RMAP commands in SpaceWire packets

- Synchronization via Time-Codes

- Time-stamping of reception and transmission of Distributed Interrupts

- Latency offset adjustment

- Jitter mitigation

- Drift mitigation

The Test bench consist of initiator and target each with GRSPW2 SpaceWire Link, SPWTDP, AMBA controllers and other components needed for verification. The time synchronization achieved between initiator and target is verified using this test bench. The Time messages from initiator are transferred to target using RMAP writes through SpaceWire Link and qualification of these time messages is performed by the SpaceWire Time-Codes transmitted from initiator to target. The target local time is initialized and synchronized (using time messages). The latency is calculated based on the values obtained by time-stamping of received and transmitted distributed interrupts and calculated value is transmitted using RMAP writes to the target. The Local time maintained in both initiator and target are nearly equal, only a single difference (local time least significant bit) between the initiator and target local time was noticed. The number of bits to represent coarse time is 32 and fine time is 24, System clock used is 50 MHz and verification is performed for 10 Mbit/s and 200 Mbit/s transmission data rate, in both cases only a single difference (local time least significant bit) between the initiator and target local time was noticed. The simulation is performed between two SpaceWire nodes without any routers and no additional data traffic in the network other than NULL control codes and Time-Codes.

4.3 Verification of jitter and drift correction unit

The Jitter and drift correction unit is verified in simulation using a VHDL test bench. The local ticks and external ticks are generated from different ET counters (local time) and external ticks are provided to correction unit with delay (latency) and variations in delay (jitter), similar to latency and jitter experienced by a SpaceWire Time Code passing through the network. The clock provided to one of ET counters (the one which acts as target) is also modified slowly to simulate drift in the local clock of target. The correction unit must perform the following, move the local tick in the center of arriving external ticks (or jitter free) and adapt the frequency synthesizer according to the drift introduced in the local clock. The correction unit performed both the needed corrections. The registers in the correction unit jitter positive and jitter negative (which gives an averaged difference between local tick and external tick in local clock count) are monitored whether they have equal value and the values are in correspondence with the amount of jitter introduced externally, the conditions are met proving that the local tick is at center of arriving external ticks (or jitter free). The FSINC provided to the Frequency synthesizer is also monitored whether it varies according to the amount of drift induced in the local clock, we know the local clock frequency variation and by using the equation below (which is used to

generate Elapsed time) we can calculate what the FSINC value should be for this frequency variation, the FSINC value varied accordingly and local time remained stable and the influence of drift from local clock is nullified.

FSINC = ((2 ^ FSW) * (2 ^ FTW)) / F

FTW - Fine time width

FSW - Frequency Synthesizer width

FSINC - pre-calculated increment value

F - Frequency of the local node

The local clock frequency is increased and decreased to simulate the drift in either direction and verified for adaptation in frequency synthesizer. Also the real-world data collected during the independent ESA measurements have been used as stimuli to validate the jitter and drift mitigation technique implemented in the jitter correction unit.

4.4 FPGA based prototyping

The SPWTDP is integrated into RASTA testbed. The RASTA RTEMS software drivers are updated to support the SPWTDP. A test application is developed to validate the functionality. The test application performs time message transfer, time qualification using SpaceWire Time-codes, time stamping, latency measurement and correction. The setup consists of GR-RASTA-105 acting as an initiator consist of GRSPW2 SpaceWire interface integrated with the newly developed SPWTDP IP core, the GR-RASTA-TMTC act as target which also consist of GRSPW2 SpaceWire interface and SPWTDP IP core. Two GR718 SpaceWire 18x routers are connected in between the initiator and target. The system clock used in all the hardware is 50 MHz except the target system which used 33 MHz system clock. The time synchronization functionality is tested without any routers in the middle and also with 1 and 2 routers in the middle, the functionality is tested for varying link data rate 2, 10, 50, 100 Mbps. The number of time codes transmitted per second is 64. The target was also replaced with GR-RASTA-105 with system clock 50 MHz and tested similar to the previous set up with 2, 10, 50, 100 and 200 Mbps link data rate.

The tick generated in initiator during SpaceWire Time-Code transmission and similar diagnostic tick from target (SpaceWire Time-Code and tick is generated just for diagnostics in target) is pulled out and monitored using an oscilloscope. When a tick occurred the local time with lower weights than the size bits mapped to SpaceWire Time-Code time information bits are all zero, so comparing the instance at which ticks generated provides the accuracy in time maintained between the initiator and target.

The Initial oscillator frequency offset in the target is nullified for all the cases mentioned in the previous section and a stable time is maintained between the initiator and target. When the mitigation is disabled (correction unit disabled) in target due to the differences in the local oscillators of initiator and target the tick moved away and time maintained between the system is incorrect, but when the mitigation is enabled the target ticks does not move away from the initiator ticks and maintain a stable time difference. This proves that the effect of oscillator frequency offset is nullified by the mitigation unit.

The time in initiator and target was monitored directly by freezing them to a register by an external trigger which occurs at same instance to initiator and target. The contents of the registers are read out using two debug monitor (GRMON) and the values of local time are compared. The local time differences seen are in correspondence to the time differences seen between the ticks of initiator and target.

Initially the accuracy was measured between the initiator and target without any additional traffic (other than NULL control codes and SpaceWire Time-Codes) the time maintained in both initiator and target have only a single difference between the initiator and target time was noticed and this corresponds to an accuracy of 60 ns, i.e. the difference seen in the 24th fine time bit which represents 2^-24 (~60 ns). The same level of accuracy was not able to achieve with data traffic in the network. The variation in jitter because of data traffic influences this accuracy. The data characters are 10 bits

length whereas the NULL's are 8 bits, the jitter varies from 8 to 10 bits of transmission period. The jitter mitigation technique implemented in this design tries to nullify the jitter by being in the center of the incoming Time-Codes, the variation in jitter from 8 to 10 bits due to variation in traffic results in an inaccuracy of single transmission bit period per link. During these measurements separate clock crystals used for generating the system clock and SpaceWire transmission clock. While using the same crystals for system clock and SpaceWire transmission clock the same level of accuracy was achieved but not maintained for long time, the reason for this is due to the behavior of jitter which varies from using different crystals and should be further studied.

The implementation successfully mitigates effects of the oscillator in the target and maintains a stable time between initiator and target i.e. the time in the target and time in the initiator is maintained at a constant rate. The implementation also nullifies the impact of drift (local clock oscillator) in local time in the target. A methodology to calculate latency using distributed interrupts is defined, implemented and verified. The inaccuracy resulting from the jitter variation have a significant impact for low link data rate like 2 Mbps (500 ns) the effect of jitter variations will have less impact for higher data rate like 200 Mbps (5 ns). Even with a defect in correction principle the jitters impact on time keeping is reduced by a factor of 10 for any number of links.

Aeroflex Gaisler AB
Kungsgatan 12
411 19 Göteborg
Sweden

tel +46 31 7758650
fax +46 31 421407
sales@gaisler.com
www.aeroflex.com/gaisler