PpsMasterClock

Reference Manual

Product Info	
Product Manager	Sven Meier
Author(s)	Sven Meier
Reviewer(s)	-
Version	1.6
Date	03.01.2023

Copyright Notice

Disclaimer

The information available to you in this document/code may contain errors and is subject to periods of interruption. While NetTimeLogic GmbH does its best to maintain the information it offers in the document/code, it cannot be held responsible for any errors, defects, lost profits, or other consequential damages arising from the use of this document/code.

NETTIMELOGIC GMBH PROVIDES THE INFORMATION, SERVICES AND PRODUCTS AVAILABLE IN THIS DOCUMENT/CODE "AS IS," WITH NO WARRANTIES WHATSOEVER. ALL EXPRESS WARRANTIES AND ALL IMPLIED WARRANTIES, INCLUDING WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF PROPRIETARY RIGHTS ARE HEREBY DISCLAIMED TO THE FULLEST EXTENT PERMITTED BY LAW. IN NO EVENT SHALL NETTIMELOGIC GMBH BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, SPECIAL AND EXEMPLARY DAMAGES, OR ANY DAMAGES WHATSOEVER, ARISING FROM THE USE OR PERFORMANCE OF THIS DOCUMENT/CODE OR FROM ANY INFORMATION, SERVICES OR PRODUCTS PROVIDED THROUGH THIS DOCUMENT/CODE, EVEN IF NETTIMELOGIC GMBH HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

IF YOU ARE DISSATISFIED WITH THIS DOCUMENT/CODE, OR ANY PORTION THEREOF, YOUR EXCLUSIVE REMEDY SHALL BE TO CEASE USING THE DOCUMENT/CODE.

Overview

NetTimeLogic's PPS Master Clock is a full hardware (FPGA) only implementation of a synchronization core able to synchronize other nodes to a Pulse per Second output. The whole algorithms and calculations are implemented in the core, no CPU is required. This allows running PPS synchronization completely independent and standalone from the user application. The core can be configured either by signals or by an AXI4Lite-Slave Register interface.

Key Features:

- PPS Master Clock
- Output delay compensation
- PPS duty cycle adjustment, width can be set via register
- AXI4Lite register set or static configuration
- PPS resolution with 50 MHz system clock: 10ns
- Optional High Resolution PPS generation with 4ns resolution
- Optional cable delay compensation

Revision History

This table shows the revision history of this document.

Version	Date	Revision
0.1	28.12.2015	First draft
1.0	19.05.2016	First release
1.1	07.06.2016	Added polarity
1.2	20.12.2017	Status interface added
1.3	25.02.2020	HighResolution added
1.4	22.04.2021	Cable Delay added
1.5	03.01.2023	Added Vivado upgrade version description
1.6	26.05.2023	Extended Calble delay range

Table 1: Revision History

Content

Definitions

Definitions	
PPS Master Clock	A clock that can synchronize others to a PPS output
PI Servo Loop	Proportional–integral servo loop, allows for smooth corrections
Offset	Phase difference between clocks
Drift	Frequency difference between clocks

Table 2: Definitions

Abbreviations

Abbreviations	
AXI	AMBA4 Specification (Stream and Memory Mapped)
IRQ	Interrupt, Signaling to e.g. a CPU
PPS	Pulse Per Second
PS	PPS Slave
TS	Timestamp
TB	Testbench
LUT	Look Up Table
FF	Flip Flop
RAM	Random Access Memory
ROM	Read Only Memory
FPGA	Field Programmable Gate Array
VHDL	Hardware description Language for FPGA's

Table 3: Abbreviations

1 Introduction

1.1 Context Overview

The PPS Master Clock is meant as a co-processor handling a PPS output.
It takes a time as reference input and calculates when the next PPS has to be generated at the nanosecond overflow by taking output and optional cable delays into account to get the maximum accuracy.
In parallel the duty cycle can be set via registers to show for e.g. the accuracy of the time against a primary reference.
The PPS Master Clock is designed to work in cooperation with the Counter Clock core from NetTimeLogic (not a requirement). It contains an AXI4Lite slave for configuration and status supervision from a CPU, this is however not required since the PPS Master Clock can also be configured statically via signals/constants directly from the FPGA.
It can be combined with a TOD Master clock to act like for e.g. a GPS receiver. Offset and drift are then distributed via the PPS Master Clock to the next second and the TOD Master Clock will distribute the absolute time on seconds level.

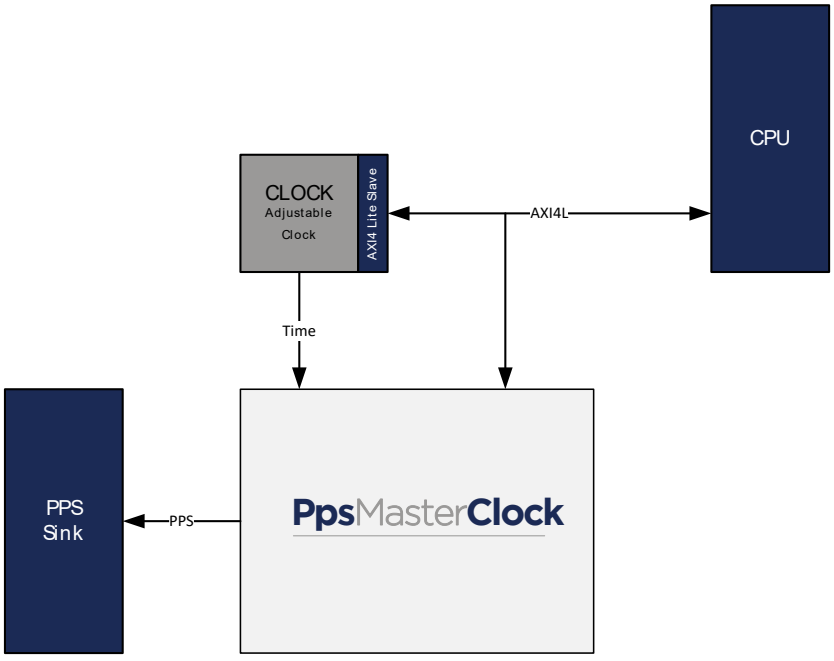

Figure 1: Context Block Diagram

1.2 Function

The PPS Master Clock generates a PPS of configurable polarity at the nanoseconds overflow of the local clock. It takes the duty cycle of the PPS from a register and generates a pulse of that width with an accuracy of +/- 1ms.
If the reference clock makes a jump in time, the PPS generation is skipped for the moment of the time jump and restarted at the next second overflow. This can cause two pulses to be very close to each other, overlapped or missing, this condition is marked as an error condition and provided to a register.

1.3 Architecture

The core is split up into different functional blocks for reduction of the complexity, modularity and maximum reuse of blocks. The interfaces between the functional blocks are kept as small as possible for easier understanding of the core.

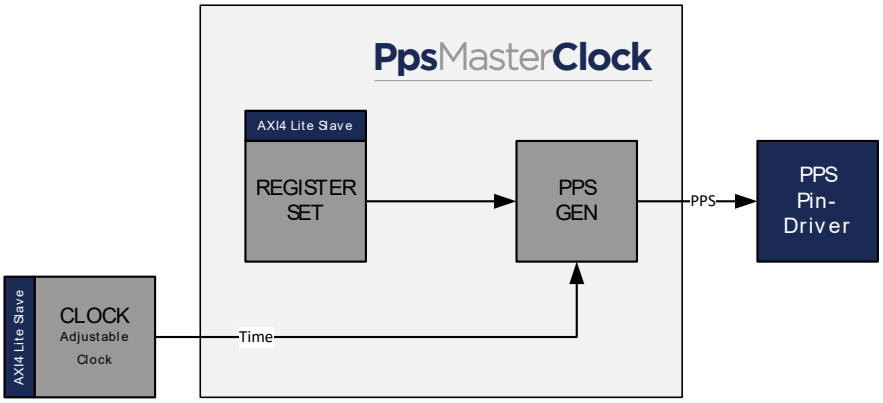

Figure 2: Architecture Block Diagram

Register Set

This block allows reading status values and writing configuration.

PPS Gen

This generates the PPS at the second overflow of the reference time minus the output delay and optional cable delay with a configurable duty cycle.

2 PPS Basics

2.1 Interface

The Pulse per Second is a very simple interface and can be electrical or optical. It can be a single ended, differential, open drain, open collector and therefore also high or low active signal. The signal has a frequency of 1Hz as the name says.
The reference point is the edge to the active level; this shall be at the second overflow of the reference clock. This edge shall be very accurate compared to the edge to the idle level (drive active level, tristate idle level).
For high accuracy synchronization delays have to be compensated for, and the duty cycle of the PPS can be used for accuracy encoding.
A PPS network is normally a one-to-many configuration: one PPS master synchronizes multiple PPS slave of different distance from the master.

2.2 Delays

There are two kinds of delays in a PPS Network. One is the output delay of the PPS to the core; this shall be constant and is compensated for. The second delay is the propagation delay of the signal from the master to the slave. This is dependent on the cable length and medium: 15cm of copper cable are equal to roughly 1ns of propagation delay. As stated earlier in a PPS network is normally one PPS master synchronizes multiple PPS slave of different distance from the master. Because the propagation delay to the master is different for each slave this delay has to be normally compensated for in the slave (supported by NetTimeLogic's PPS Slave core). However if it is a one-to-one connection and the PPS slave can not compensate the cable delay, it is possible that the PPS Master can compensate for the cable delay.

2.3 Accuracy

Some PPS Masters (also this one) are capable of encoding its synchronization accuracy to the duty cycle of the PPS signal. Often a logarithmic scale is used to encode the accuracy to a primary reference. E.g. 100ms of duty cycle = 10ns, 200ms = 100ns, 300ms = 1000ns, 400ms = 10000ns ... However this is not standardized. This core is capable of adjusting the duty cycle with millisecond resolution (dependend on the alignment of the 1ms pulse of the clock). Interpretation/definition of the duty cycle is up to the user.

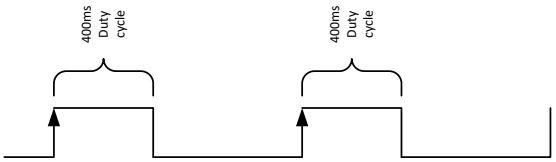

Figure 3: PPS Waveform

3 Register Set

This is the register set of the PPS Master Clock. It is accessible via AXI4Lite Memory Mapped. All registers are 32bit wide, no burst access, no unaligned access, no byte enables, no timeouts are supported. Register address space is not contiguous. Register addresses are only offsets in the memory area where the core is mapped in the AXI inter connects. Non existing register access in the mapped memory area is answered with a slave decoding error.

3.1 Register Overview

Registerset Overview			
Name	Description	Offset	Access
Pps MasterControl Reg	PPS Master Enable Control Register	0x00000000	RW
Pps MasterStatus Reg	PPS Master Error Status Register	0x00000004	WC
Pps MasterPolarity Reg	PPS Master Polarity Register	0x00000008	RW
Pps MasterVersion Reg	PPS Master Version Register	0x0000000C	RO
Pps MasterPulseWidth Reg	PPS Master Pulse Width Register	0x00000010	RW
Pps MasterCableDelay Reg	PPS Master Cable Delay Register	0x00000020	RW

Table 4: Register Set Overview

3.2 Register Descriptions

3.2.1 General

3.2.1.1 PPS Master Control Register

Used for general control over the PPS Master Clock, all configurations on the core shall only be done when disabled.

PPS MasterControl Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															ENABLE
RO																															RW
Reset: 0x00000000																															
Offset: 0x0000																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
ENABLE	Enable	Bit: 0	RW

3.2.1.2 PPS Master Status Register

Shows the current status of the PPS Master Clock.

Pps MasterStatus Reg																														
Reg Description																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																														ERROR
RO																														WC
Reset: 0x00000000																														
Offset: 0x0004																														

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
ERROR	Error (sticky)	Bit: 0	WC

3.2.1.3 PPS Master Polarity Register

Used for setting the signal output polarity of the PPS Master Clock, shall only be done when disabled. Default value is set by the OutputPolarity_Gen generic.

PPS MasterPolarity Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															POLARITY
RO																															RW
Reset: 0x0000000X																															
Offset: 0x0008																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
POLARITY	Signal Polarity (1 active high, 0 active low)	Bit: 0	RW

3.2.1.4 PPS Master Version Register

Version of the IP core, even though is seen as a 32bit value, bits 31 down to 24 represent the major, bits 23 down to 16 the minor and bits 15 down to 0 the build numbers.

Pps MasterVersion Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION																															
RO																															
0xXXXXXXXX																															
Offset: 0x000C																															

Name	Description	Bits	Access
VERSION	Version of the core	Bit: 31:0	RO

3.2.1.5 PPS Master Pulse Width Register

Defines the current pulse width in milliseconds of the PPS output generated. This can be useful if the Slave supports accuracy detection on the PPS duty cycle (as NetTimeLogic's Slave is capable of). Changes to this register will have an effect at the next PPS.

Pps MasterPulseWidth Reg																														
Reg Description																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																						PULSE_WIDTH									
RO																						RW									
Reset: 0x000003FF																														
Offset: 0x0010																														

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:10	RO
PULSE_WIDTH	Pulse width of PPS in milliseconds (min 1, max 999) Accuracy (dependent on the synchronization of the clock)	Bit: 9:0	RW

3.2.1.6 PPS Master Cable Delay Register

This register allows to compensate for the propagation delay of the cable between the PPS master and the PPS slave (only for one-to-one connections). To calculate the delay a rule of thumb says around 1ns per 15cm of cable.

Pps MasterCableDelay Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGN	-	CABLE_DELAY																													
RW	RO	RW																													
Reset: 0x00000000																															
Offset: 0x0020																															

Name	Description	Bits	Access
SIGN	Sign of the cable delay, default positive (0)	Bit: 31	RW
-	Reserved, read 0	Bit: 30	RO
CABLE_DELAY	Cable delay of cable to slave in nanoseconds (15cm is around 1ns)	Bit: 29:0	RW

4 Design Description

The following chapters describe the internals of the PPS Master Clock: starting with the Top Level, which is a collection of subcores, followed by the description of all subcores.

4.1 Top Level – Pps Master

4.1.1.1 Parameters

The core must be parametrized at synthesis time. There are a couple of parameters which define the final behavior and resource usage of the core.

Name	Type	Size	Description
PulseWidthDynamic Support_Gen	boolean	1	Support for Pulse width generation: true = pulse width can dynamically set, false = pulse width is fixed to OutputPulseWidth-Millsecond_Gen
CableDelaySup-port_Gen	boolean	1	If the core shall support to compensate for cable delays to the PPS sink
OutputPulseWidth Millsecond_Gen	natural	1	Default Pulse width, minimum pulse width and fixed pulse width if no dynamic support
StaticConfig_Gen	boolean	1	If Static Configuration or AXI is used
ClockClkPeriod Nanosecond_Gen	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
OutputDelay Nanosecond_Gen	natural	1	Output delay of the PPS from the output signal to the connector.
OutputPolarity_Gen	boolean	1	true = high active, false = low active
AxiAddressRange Low_Gen	std_logic_vector	32	AXI Base Address
AxiAddressRange	std_logic_vector	32	AXI Base Address plus Regis-

High_Gen			terset Size Default plus 0xFFFF
HighResSupport_Gen	boolean	1	If a high-resolution clock SysClkNx with alignment to SysClk is used
HighResFreq Multiply_Gen	natural	1	Multiplication factor of the high-resolution clock compared to SysClk
Sim_Gen	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis

Table 5: Parameters

4.1.1.2 Structured Types

4.1.1.2.1 Clk_Time_Type

Defined in Clk_Package.vhd of library ClkLib
Type represents the time used everywhere. For this type overloaded operators + and – with different parameters exist.

Field Name	Type	Size	Description
Second	std_logic_vector	32	Seconds of time
Nanosecond	std_logic_vector	32	Nanoseconds of time
Fraction	std_logic_vector	2	Fraction numerator (mostly not used)
Sign	std_logic	1	Positive or negative time, 1 = negative, 0 = positive.
TimeJump	std_logic	1	Marks when the clock makes a time jump (mostly not used)

Table 6: Clk_Time_Type

4.1.1.2.2 Pps_MasterStaticConfig_Type

Defined in Pps_MasterAddrPackage.vhd of library PpsLib
This is the type used for static configuration.

Field Name	Type	Size	Description
Polarity	std_logic	1	'1' = high active, '0' = low active
PulseWidth	std_logic_vector	10	Pulse width in milliseconds 1-999 to generate if dynamic pulse generation is enabled.
CableDelay	std_logic_vector	30	Cable delay in nanoseconds 0 to 64k to compensate id cable delay compensation is enabled

Table 7: Pps_MasterStaticConfig_Type

4.1.1.2.3 Pps_MasterStaticConfigVal_Type

Defined in Pps_MasterAddrPackage.vhd of library PpsLib
This is the type used for valid flags of the static configuration.

Field Name	Type	Size	Description
Enable_Val	std_logic	1	Enables the PPS Master

Table 8: Pps_MasterStaticConfigVal_Type

4.1.1.2.4 Pps_MasterStaticStatus_Type

Defined in Pps_MasterAddrPackage.vhd of library PpsLib
This is the type used for static status supervision.

Field Name	Type	Size	Description
CoreInfo	Clk_CoreInfo_ Type	1	Infor about the Cores state

Table 9: Pps_MasterStaticConfig_Type

4.1.1.2.5 Pps_MasterStaticStatusVal_Type

Defined in Pps_MasterAddrPackage.vhd of library PpsLib
This is the type used for valid flags of the static status supervision.

Field Name	Type	Size	Description

CoreInfo_Val	std_logic	1	Core Info valid

Table 10: Pps_MasterStaticConfigVal_Type

4.1.1.3 Entity Block Diagram

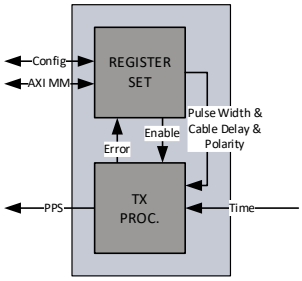

Figure 4: PPS Master Clock

4.1.1.4 Entity Description

Tx Processor

This module generates the PPS signal at the nanoseconds overflow of the reference time. It generates a pulse with a duty cycle which is provided by the Registerset.
See 4.2.1 for more details.

Registerset

This module is an AXI4Lite Memory Mapped Slave. It provides access to all registers and allows configuring the PPS Master Clock. It can be configured to either run in AXI or StaticConfig mode. If in StaticConfig mode, the configuration of the registers is done via signals and can be easily done from within the FPGA without CPU. If in AXI mode, an AXI Master has to configure the registers with AXI writes to the registers, which is typically done by a CPU
See 4.2.2 for more details.

4.1.1.5 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
PulseWidthDynamicSupport_Gen	-	boolean	1	Support for Pulse width analysis
CableDelaySupport_Gen	-	boolean	1	If the core shall support to compensate for cable delays

				to the PPS sink
StaticConfig_Gen	-	boolean	1	If Static Configuration or AXI is used
ClockClkPeriod Nanosecond_Gen	-	natural	1	Integer Clock Period
OutputPulseWidth Millsecond_Gen	-	natural	1	Default Pulse width, minimum pulse width and fixed pulse width if no dynamic support
OutputDelay Nanosecond_Gen	-	natural	1	Output delay of the PPS from the output signal to the connector
OutputPolarity_Gen	-	boolean	1	True: High active, False: Low active
AxiAddressRange Low_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size
HighResSupport_Gen	-	boolean	1	If a high-resolution clock SysClkNx with alignment to SysClk is used
HighResFreq Multiply_Gen	-	natural	1	Multiplication factor of the high-resolution clock compared to SysClk
MaxOffset_Gen	-	natural	1	If Offset is larger than this change into Uncalibrated state if Slave
Sim_Gen	-	boolean	1	If in Testbench simulation mode
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock

SysClkNx_ClkIn	in	std_logic	1	High-resolution clock (multiple of Sys Clock)
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig_DatIn	in	Pps_Master StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Pps_Master StaticConfigVal _Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Pps_Master StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Pps_Master StaticStatusVal _Type	1	Static Status valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted PTP Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted PTP Clock Time valid
AXI4 Lite Slave				
AxiWriteAddrValid _ValIn	in	std_logic	1	Write Address Valid
AxiWriteAddrReady _RdyOut	out	std_logic	1	Write Address Ready
AxiWriteAddrAddress _AdrIn	in	std_logic_vector	32	Write Address
AxiWriteAddrProt _DatIn	in	std_logic_vector	3	Write Address Protocol
AxiWriteDataValid _ValIn	in	std_logic	1	Write Data Valid
AxiWriteDataReady _RdyOut	out	std_logic	1	Write Data Ready
AxiWriteDataData _DatIn	in	std_logic_vector	32	Write Data
AxiWriteDataStrobe _DatIn	in	std_logic_vector	4	Write Data Strobe
AxiWriteRespValid _ValOut	out	std_logic	1	Write Response

				Valid
AxiWriteRespReady_RdyIn	in	std_logic	1	Write Response Ready
AxiWriteResp Response_DatOut	out	std_logic_vector	2	Write Response
AxiReadAddrValid_ValIn	in	std_logic	1	Read Address Valid
AxiReadAddrReady_RdyOut	out	std_logic	1	Read Address Ready
AxiReadAddrAddress_AdrIn	in	std_logic_vector	32	Read Address
AxiReadAddrProt_DatIn	in	std_logic_vector	3	Read Address Protocol
AxiReadDataValid_ValOut	out	std_logic	1	Read Data Valid
AxiReadDataReady_RdyIn	in	std_logic	1	Read Data Ready
AxiReadData Response_DatOut	out	std_logic_vector	2	Read Data
AxiReadDataData_DatOut	out	std_logic_vector	32	Read Data Response
Pulse Per Second Output				
Pps_EvtOut	out	std_logic	1	PPS output

Table 11: PPS Master Clock

4.2 Design Parts

The PPS Master Clock core consists of a couple of subcores. Each of the subcores itself consist again of smaller function block. The following chapters describe these subcores and their functionality.

4.2.1 TX Processor

4.2.1.1 Entity Block Diagram

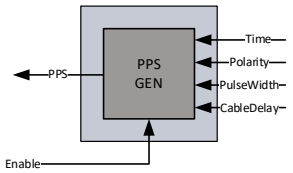

Figure 5: TX Processor

4.2.1.2 Entity Description

PPS Generator

This module generates the PPS with the required polarity at the nanoseconds overflow of the reference clock. It takes the output delay because of e.g. external driver ICs and optionally a configurable cable delay to the PPS sink into account and asserts the internal signal earlier respectively. For maximum accuracy but still minimum frequency requirements the generation can start at both edges of the clock, or it uses the high-resolution functionality with e.g. 250MHz which gives a quantization fault of only 4ns. De-assertion of the PPS is timer driven and has an accuracy of +/- 1ms. The pulse width is generated according to the values received from the Registerset. Once the pulse generation is started it will only be canceled by disabling. The pulse width is evaluated only at the PPS generation; intermediate changes will not have an effect until the next PPS, that way no inconsistency is possible. For a positive polarity the rising edge is accurate, for a negative polarity the falling edge is accurate.

4.2.1.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriod	-	natural	1	Clock Period in

Nanosecond_Gen				Nanosecond
TX Processor				
PulseWidthDynamic Support_Gen	-	boolean	1	Support for Pulse width analysis
CableDelaySup-port_Gen	-	boolean	1	If the core shall support to compen-sate for cable delays to the PPS sink
OutputPulseWidth Millsecond_Gen	-	natural	1	Default Pulse width, minimum pulse width and fixed pulse width if no dynamic support
OutputDelay Nanosecond_Gen	-	natural	1	Output delay of the PPS from the output signal to the con-nector
OutputPolarity_Gen	-	boolean	1	True: High active, False: Low active
HighResSupport_Gen	-	boolean	1	If a high-resolution clock SysClkNx with alignment to SysClk is used
HighResFreq Multiply_Gen	-	natural	1	Multiplication factor of the high-resolution clock compared to SysClk
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysClkNx_ClkIn	in	std_logic	1	High-resolution clock (multiple of Sys Clock)
SysRstN_RstIn	in	std_logic	1	System Reset
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Time Input				

ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted PTP Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted PTP Clock Time valid
Pulse Per Second Polarity				
PpsPolarity_DatIn	in	std_logic	10	'1': High active, '0': Low active
Pulse Per Second Error Output				
Pps_ErrOut	out	std_logic_vector	1	Indicates a time jump
Pulse Per Second Width Input				
PpsPulseWidth_DatIn	in	std_logic_vector	10	0-999 in millisecond marks the duty cycle of the generated PPS
Cable Delay Input				
PpsCableDelay_DatIn	in	Clk_Time_Type	1	Propagation delay in nanoseconds to compensate cable delay if compensation is enabled
Pulse Per Second Output				
Pps_EvtOut	out	std_logic	1	PPS output
Enable Input				
Enable_EnaIn	in	std_logic	1	Enables the correction and supervision

Table 12: TX Processor

4.2.2 Registerset

4.2.2.1 Entity Block Diagram

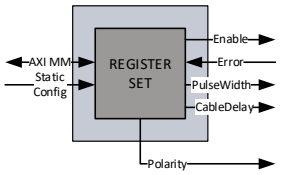

Figure 6: Registerset

4.2.2.2 Entity Description

Register Set

This module is an AXI4Lite Memory Mapped Slave. It provides access to all registers and allows configuring the PPS Master Clock. AXI4Lite only supports 32 bit wide data access, no byte enables, no burst, no simultaneous read and writes and no unaligned access. It can be configured to either run in AXI or StaticConfig mode. If in StaticConfig mode, the configuration of the registers is done via signals and can be easily done from within the FPGA without CPU. For each parameter a valid signal is available, the enable signal shall be set last (or simultaneously). To change configuration parameters the clock has to be disabled and enabled again, the pulse width and cable delay can be changed at runtime. If in AXI mode, an AXI Master has to configure the registers with AXI writes to the registers, which is typically done by a CPU. Parameters can in this case also be changed at runtime.

4.2.2.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
PulseWidthDynamicSupport_Gen	-	boolean	1	Support for Pulse width generation
CableDelaySupport_Gen	-	boolean	1	If the core shall support to compensate for cable delays to the PPS sink
OutputPulseWidth		natural	1	Default Pulse width,

Millsecond_Gen				minimum pulse width and fixed pulse width if no dynamic support
Register Set				
StaticConfig_Gen	-	boolean	1	If Static Configuration or AXI is used
AxiAddressRangeLow_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRangeHigh_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig_DatIn	in	Pps_Master StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Pps_Master StaticConfigVal_Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Pps_Master StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Pps_Master StaticStatusVal_Type	1	Static Status valid
AXI4 Lite Slave				
AxiWriteAddrValid_ValIn	in	std_logic	1	Write Address Valid
AxiWriteAddrReady_RdyOut	out	std_logic	1	Write Address Ready
AxiWriteAddrAddress_AdrIn	in	std_logic_vector	32	Write Address
AxiWriteAddrProt_DatIn	in	std_logic_vector	3	Write Address Protocol
AxiWriteDataValid_ValIn	in	std_logic	1	Write Data Valid
AxiWriteDataReady_RdyOut	out	std_logic	1	Write Data Ready
AxiWriteDataData	in	std_logic_vector	32	Write Data

_DatIn				
AxiWriteDataStrobe _DatIn	in	std_logic_vector	4	Write Data Strobe
AxiWriteRespValid _ValOut	out	std_logic	1	Write Response Valid
AxiWriteRespReady _RdyIn	in	std_logic	1	Write Response Ready
AxiWriteResp Response_DatOut	out	std_logic_vector	2	Write Response
AxiReadAddrValid _ValIn	in	std_logic	1	Read Address Valid
AxiReadAddrReady _RdyOut	out	std_logic	1	Read Address Ready
AxiReadAddrAddress _AdrIn	in	std_logic_vector	32	Read Address
AxiReadAddrProt _DatIn	in	std_logic_vector	3	Read Address Protocol
AxiReadDataValid _ValOut	out	std_logic	1	Read Data Valid
AxiReadDataReady _RdyIn	in	std_logic	1	Read Data Ready
AxiReadData Response_DatOut	out	std_logic_vector	2	Read Data
AxiReadDataData _DatOut	out	std_logic_vector	32	Read Data Response
Pulse Per Second Polarity				
PpsPolarity_DatOut	out	std_logic	10	'1': High active, '0': Low active
Pulse Per Second Error Input				
Pps_ErrIn	in	std_logic_vector	1	Indicates a time jump
Pulse Per Second Width Output				
PpsPulseWidth _DatOut	in	std_logic_vector	10	0-999 in millisecond marks the duty cycle of the generated PPS
Cable Delay Output				
PpsCableDelay _DatOut	out	Clk_Time_Type	1	Propagation delay in nanoseconds to compensate cable delay if compensation is enabled
Enable Output				
PpsMaster Enable_DatOut	out	std_logic	1	Enables the correc-

				tion and supervision

Table 13: Registerset

4.3 Configuration example

In both cases the enabling of the core shall be done last, after or together with the configuration.

4.3.1 Static Configuration

```
constant PpsStaticConfigMaster_Con : Pps_MasterStaticConfig_Type := (
  Polarity                    => '1',
  PulseWidth                  => "00" & x"C8", -- 200 ms
  CableDelay                  => std_logic_vector(to_unsigned(128, 30)) - 128 ns
);

constant PpsStaticConfigValMaster_Con : Pps_MasterStaticConfigVal_Type := (
  Enable_Val                  => '1'
);
```

Figure 7: Static Configuration

The pulse width can be changed at runtime. It is always valid.

4.3.2 AXI Configuration

The following code is a simplified pseudocode from the testbench: The base address of the PPS Master Clock is 0x10000000.

```
-- PPS MASTER
-- Config
-- Set polarity to high active
AXI WRITE 10000008 00000001
-- Set pulse width to 200 ms
AXI WRITE 10000010 000000C8
-- Set cable delay 128 ns
AXI WRITE 10000020 00000080
-- enable PPS Slave
AXI WRITE 10000000 00000001
```

Figure 8: AXI Configuration

In the example the pulse width is first set to 200ms, the cable delay set to 128ns, then the core is enabled.

4.4 Clocking and Reset Concept

4.4.1 Clocking

To keep the design as robust and simple as possible, the whole PPS Master Clock, including the Counter Clock and all other cores from NetTimeLogic are run in one clock domain. This is considered to be the system clock. Per default this clock is 50MHz. Where possible also the interfaces are run synchronous to this clock. For clock domain crossing asynchronous fifos with gray counters or message patterns with meta-stability flip-flops are used. Clock domain crossings for the AXI interface is moved from the AXI slave to the AXI interconnect.

Clock	Frequency	Description
System		
System Clock	50MHz (Default)	System clock where the PPS Master runs on as well as the counter clock etc.
AXI Interface		
AXI Clock	50MHz (Default)	Internal AXI bus clock, same as the system clock

Table 14: Clocks

4.4.2Reset

In connection with the clocks, there is a reset signal for each clock domain. All resets are active low. All resets can be asynchronously set and shall be synchronously released with the corresponding clock domain. All resets shall be asserted for the first couple (around 8) clock cycles. All resets shall be set simultaneously and released simultaneously to avoid overflow conditions in the core. See the reference designs top file for an example of how the reset shall be handled.

Reset	Polarity	Description
System		
System Reset	Active low	Asynchronous set, synchronous release with the system clock
AXI Interface		
AXI Reset	Active low	Asynchronous set, synchronous release with the AXI clock, which is the same as

		the system clock

Table 15: Resets

5 Resource Usage

Since the FPGA Architecture between vendors and FPGA families differ there is a split up into the two major FPGA vendors.

5.1 Intel/Altera (Cyclone V)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (No Dynamic pulse width support)	15	67	0	0
Maximal (Dynamic pulse width support)	116	180	0	0

Table 16: Resource Usage Intel/Altera

5.2 AMD/Xilinx (Artix 7)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (No Dynamic pulse width support)	15	105	0	0
Maximal (Dynamic pulse width support)	116	184	0	0

Table 17: Resource Usage AMD/Xilinx

6 Delivery Structure

```
AXI                                 -- AXI library folder
 |-Library                          -- AXI library component sources
 |-Package                          -- AXI library package sources

CLK                                 -- CLK library folder
 |-Library                          -- CLK library component sources
 |-Package                          -- CLK library package sources

COMMON                              -- COMMON library folder
 |-Library                          -- COMMON library component sources
 |-Package                          -- COMMON library package sources

PPS                                 -- PPS library folder
|-Core                              -- PPS library cores
 |-Doc                              -- PPS library cores documentations
 |-Library                          -- PPS library component sources
 |-Package                          -- PPS library package sources
 |-Refdesign                        -- PPS library cores reference designs
 |-Testbench                        -- PPS library cores testbench sources and sim/log

SIM                                 -- SIM library folder
 |-Doc                              -- SIM library command documentation
 |-Package                          -- SIM library package sources
 |-Testbench                        -- SIM library testbench template sources
 |-Tools                            -- SIM simulation tools
```

7 Testbench

The PPS Master testbench consist of 2 parse/port types: AXI and SIG.
The SIG input port takes the time of the Clock instance as reference and the PPS signal from the DUT port. The SIG input checks the waveform if the pulse is asserted at the correct point in time and if the pulse width is ok. In addition for configuration and result checks an AXI read and write port is used in the testbench and for accessing more than one AXI slave also an AXI interconnect is required.

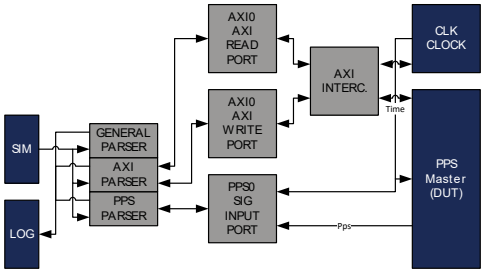

Figure 9: Testbench Framework

For more information on the testbench framework check the Sim_ReferenceManual documentation.

With the Sim parameter set the time base for timeouts are divided by 1000 to 100000 to speed up simulation time.

7.1 Run Testbench

1. Run the general script first

```
source XXX/SIM/Tools/source_with_args.tcl
```

2. Start the testbench with all test cases

```
src XXX/PPS/Testbench/Core/PpsMaster/Script/run_Pps_Master_Tb.tcl
```

3. Check the log file LogFile1.txt in the XXX/PPS/Testbench/Core/PpsMaster/Log/ folder for simulation results.

8 Reference Designs

The PPS Master reference design contains a PLL to generate all necessary clocks (cores are run at 50 MHz), an instance of the PPS Master Clock IP core and an instance of the Adjustable Counter Clock IP core (needs to be purchased separately). The Reference Design is intended to be connected to any PPS Slave which can handle a logic high single ended PPS signal with pulse widths of 100ms +. Via dipswitches the pulse width can be set in 50ms steps It is a free running PPS with 50ppm accuracy and no lock to any primary reference.
All generics can be adapted to the specific needs.

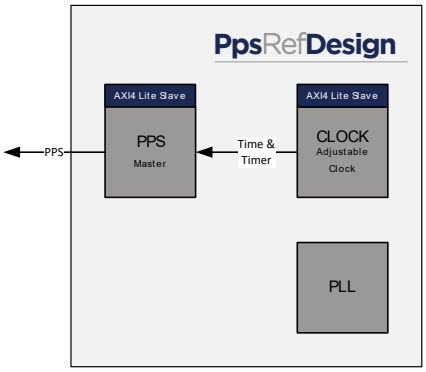

Figure 10: Reference Design

8.1 Intel/Altera: Terasic SocKit

The SocKit board is an FPGA board from Terasic Inc. with a Cyclone V SoC FPGA from Intel/Altera. (http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=205&No=816)

1. Open Quartus 16.x
2. Open Project /PPS/Refdesign/Altera/SocKit/PpsMaster/PpsMaster.qpf
3. Rerun implementation
4. Download to FPGA via JTAG

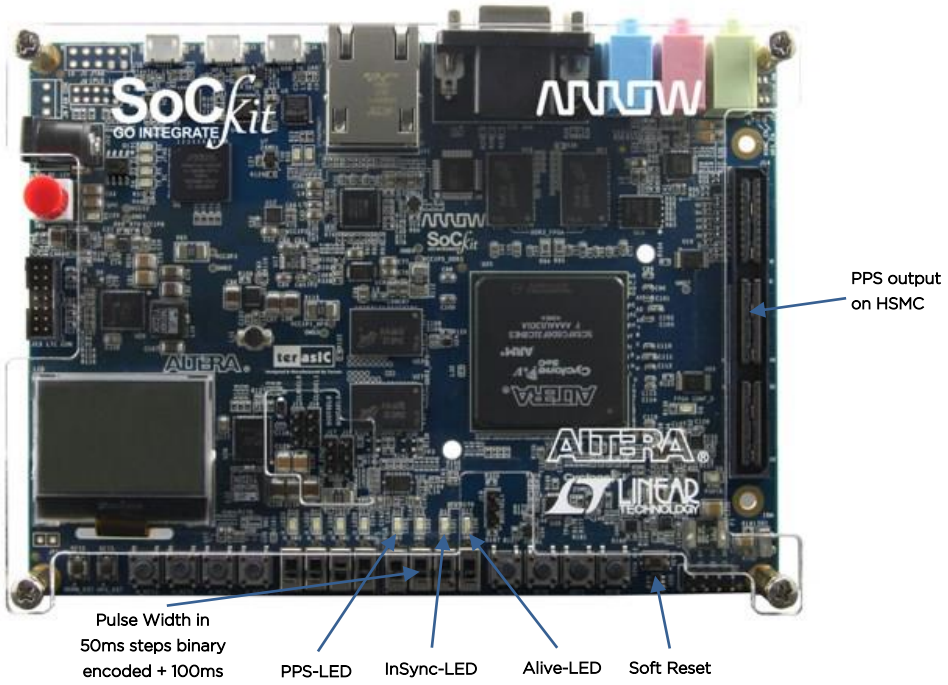

Figure 11: SocKit (source Terasic Inc)

For the ports on the HSMC connector the GPIO to HSMC adapter from Terasic Inc. was used.

8.2 AMD/Xilinx: Digilent Arty

The Arty board is an FPGA board from Digilent Inc. with an Artix7 FPGA from AMD/Xilinx. (http://store.digilentinc.com/arty-board-artix-7-fpga-development-board-for-makers-and-hobbyists/)

1. Open Vivado 2019.1.
   Note: If a different Vivado version is used, see chapter 8.3.
2. Run TCL script /PPS/Refdesign/Xilinx/Arty/PpsMaster/PpsMaster.tcl
   a. This has to be run only the first time and will create a new Vivado Project
3. If the project has been created before open the project and do not rerun the project TCL
4. Rerun implementation
5. Download to FPGA via JTAG

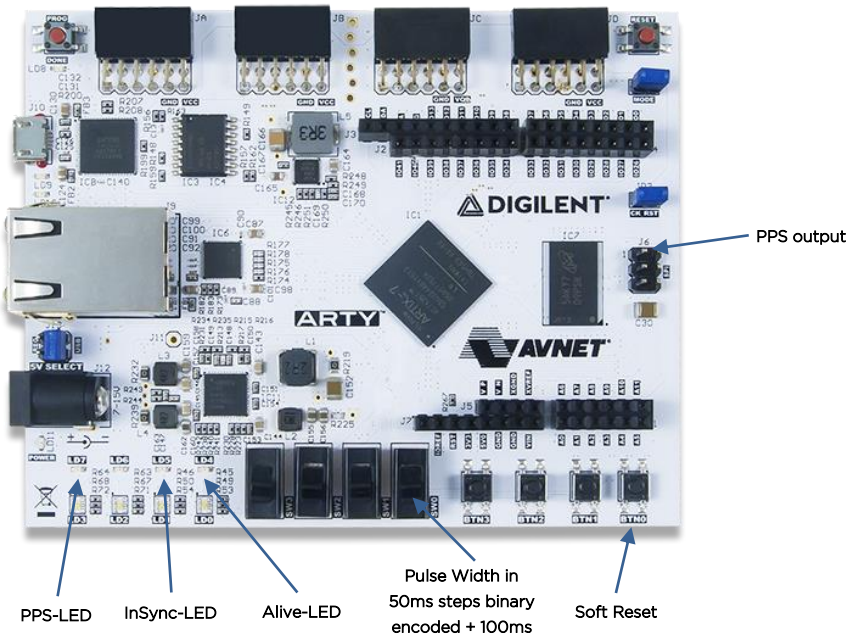

Figure 12: Arty (source Digilent Inc)

8.3 AMD/Xilinx: Vivado version

The provided TCL script for creation of the reference-design project is targeting AMD/Xilinx Vivado 2019.1.

If a lower Vivado version is used, it is recommended to upgrade to Vivado 2019.1 or higher.

If a higher Vivado version is used, the following steps are recommended:

- Before executing the project creation TCL script, the script's references of Vivado 2019 should be manually replaced to the current Vivado version. For example, if version Vivado 2022 is used, then:
  - The statement occurrences:

    ```
    set_property flow "Vivado Synthesis 2019" $obj
    ```

    shall be replaced by:

    ```
    set_property flow "Vivado Synthesis 2022 $obj
    ```

  - The statement occurrences:

    ```
    set_property flow "Vivado Implementation 2019" $obj
    ```

    shall be replaced by:

    ```
    set_property flow "Vivado Implementation 2022" $obj
    ```

- After executing the project creation TCL script, the AMD/Xilinx IP cores, such as the Clocking Wizard core, might be locked and a version upgrade might be required. To do so:

1. At "Reports" menu, select "Report IP Status".
2. At the opened "IP Status" window, select "Upgrade Selected". The tool will upgrade the version of the selected IP cores.

A List of tables

B List of figures