DcfMasterClock

Reference Manual

Product Info	
Product Manager	Sven Meier
Author(s)	Sven Meier
Reviewer(s)	-
Version	1.1
Date	03.01.2022

Copyright Notice

Disclaimer

Overview

NetTimeLogic's DCF Master Clock is a full hardware (FPGA) only implementation of a synchronization core able to synchronize other nodes to a DCF output. The whole algorithms and calculations are implemented in the core, no CPU is required. This allows running DCF 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:

- DCF Master Clock
- Supports DCF77 format
- PWM, DCLS encoding
- Output delay compensation
- Additional seconds correction to convert between TAI and UTC time (or any other time base)
- AXI4Lite register set or static configuration
- DCF resolution with 50 MHz system clock: 20ns

Revision History

This table shows the revision history of this document.

Version	Date	Revision
0.1	20.06.2018	First draft
1.0	10.09.2018	First release
1.1	03.01.2022	Added Vivado upgrade version description

Table 1: Revision History

Content

Definitions

Definitions	
DCF-77	German longwave time signal and standard-frequency radio station. The DCF77 station signal carries an amplitude-modulated, pulse-width coded 1-bit/s data signal. The transmitted data signal is repeated every minute.
DCF Master Clock	A clock that can synchronize others to an DCF 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)
BCD	Binary Coded Decimal
PWM	Pulse Width Modulation
DCLS	DC Level Shift
IRQ	Interrupt, Signaling to e.g. a CPU
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 DCF Master Clock is meant as a co-processor handling an DCF output.
It takes a time as reference input converts the time in second/nanosecond format to time of day and converts it to the Binary Coded Decimal (BCD) format used by DCF77. It aligns the second boundary of the local clock with the reference marker symbol on DCF and encodes the BCD formatted time into a Pulse Width Modulated (PWM) continuous DCLS data stream taking output delays into account to get the maximum accuracy.
The symbol period and pulse widths are aligned with the reference time and varies if corrections are done on the reference so a continuous data stream without interruptions can be guaranteed.
The DCF 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 DCF Master Clock can also be configured statically via signals/constants directly from the FPGA.

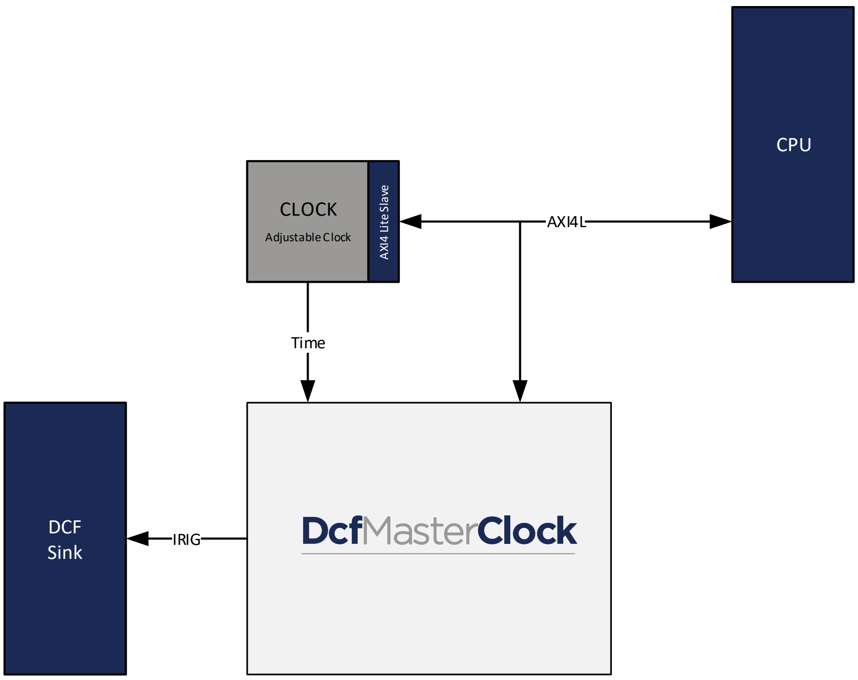

Figure 1: Context Block Diagram

1.2 Function

The DCF Master Clock generates an DCF PWM DCLS stream aligned with the local clock, compensating the output delay and taking the correction value between the two time-domains into account. It uses the local clocks frequency for the DCF encoding to achieve a continuous data stream.
If the reference clock makes a jump in time, the DCF generation is skipped for the moment of the time jump and restarted at the next minute overflow. This can cause that two marker symbols are 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.

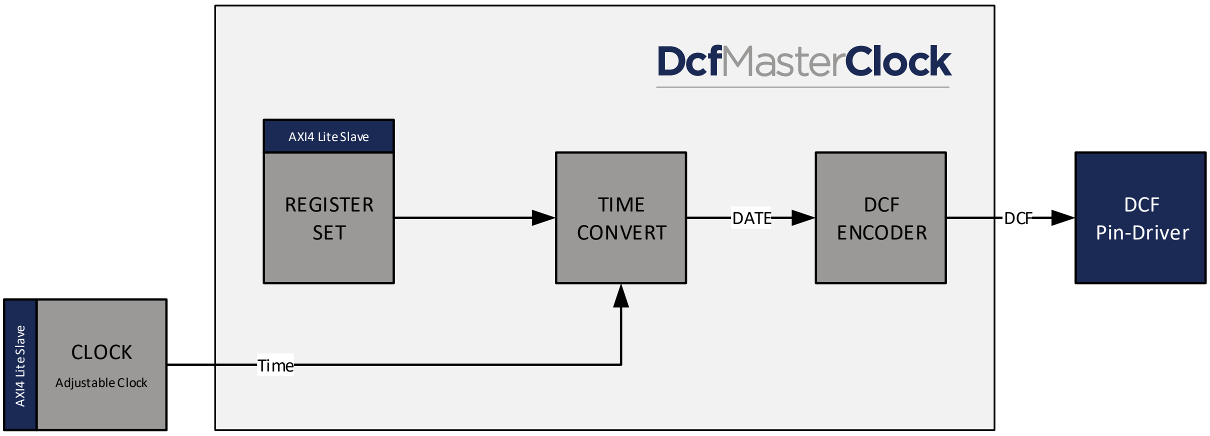

Figure 2: Architecture Block Diagram

Register Set

This block allows reading status values and writing configuration.

Time Converter

This block converts the time from seconds/nanoseconds format to time of day.

DCF Encoder

This block converts the time of day into Binary Coded Decimal (BCD) and generates the DCF Pulse Width (PWM) DC Level Shift (DCLS) modulated aligned with the second overflow of the reference time minus the output delay with a configurable duty cycle.

2 DCF Basics

2.1 Interface

DCF is a very simple wireless interface amplitude modulated on a base frequency of 77,5 kHz. The sender is a 50kW sender near Frankfurt, Germany and reaches a distance of around 2000km. A DCF antenna and encoder which has a very narrowband receiver at 77,5kHz create a Pulse Width Modulated (PWM) symbol. The period of a symbol is 1000 ms. A duty cycle of 100 ms marks a “Zero”, a duty cycle of 200 ms marks a “One” and no duty cycle at all marks a “Mark” symbol. It is a continuous data stream of “One”, “Zero” and “Mark” symbols with symbol patterns to mark the beginning of a time frame. The reference point is the edge to the active level of bit 0; this shall be at the minute overflow of the reference clock. This edge shall be very accurate compared to the other edges of the symbols of a time frame. A time frame is repeated once every minute, therefor the number of symbols in a time frame is fixed to 60.

A DCF-77 time frame contains the following:

- Weather Info
- Summer/Winter Time announcement (A1)
- Summer/Winter Time (Z1 & Z2)
- Leap Second announcement (A2).
- Time Marker (S)
- Minute (BCD encoded)
- Even Parity Minute (P1)
- Hour (BCD encoded)
- Even Parity Hour (P2)
- Day (BCD encoded)
- Day of Week (BCD encoded, unused)
- Minute (BCD encoded)
- Year (BCD encoded)
- Even Parity Date (P3)
- Minute Marker

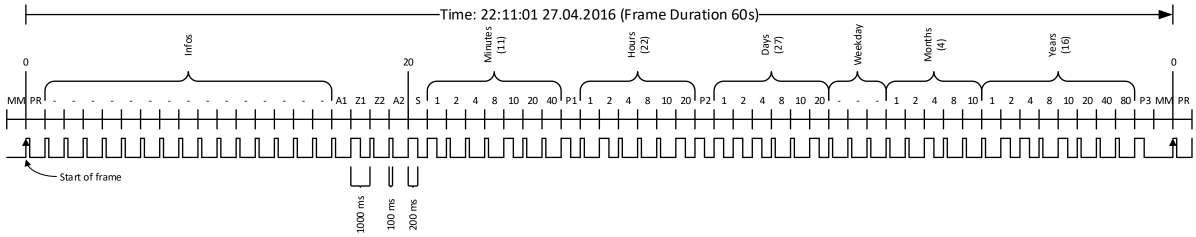

Figure 3: DCF-77 time frame

A DCF network is a one-to-many configuration: one DCF sender synchronizes multiple DCF slaves of different distance from the master.

DCF can also be generated on a wire similar to IRIG and transferred as PWM encoded DCLS signal, this is what this core is made for.

2.2 Delays

There are two kinds of delays in a DCF System. One is the input delay of the DCF signal from the antenna 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 distance from the sender in Mainflingen, Germany or the wire length in case of wired DCF: 15cm are equal to roughly 1ns of propagation delay. This delay can be set in the core.

2.3 UTC vs TAI time bases

DCF time frames contain the time of day on UTC base. UTC has an offset to TAI which is the time base normally used for the Counter Clock. This time offset can be set in the core so the local clock can still run on a TAI base. UTC in comparison to TAI or GPS time has so called leap seconds. A leap second is an additional second which is either added or subtracted from the current time to adjust for the earth rotation variation over time. Until 2016 UTC had additional 36 leap seconds, therefore TAI time is currently 36 seconds ahead of UTC. The issue with UTC time is, that the time makes jumps with the leap seconds which may cause that synchronized nodes go out of sync for a couple of seconds. Leap seconds are normally introduced at midnight of either the 30 of June or 31 of December. For an additional leap second the seconds counter of the UTC time will count to 60 before wrapping around to zero, for one fewer leap second the UTC second counter will wrap directly from 58 to 0 by skipping 59 (this has not happened yet).

Be aware that this core takes no additional precautions to handle leap seconds, so it will make a time jump at a UTC leap second and will lose synchronization since it thinks that it has an offset of one second at tries to readjust this offset. A way to avoid this is to disable the adjustment at the two dates right before midnight (e.g. one minute earlier), wait for the leap second to happen, fetch some time server to get the new offset between TAI and UTC, set this offset to the core and enable the core again. This way the local clock on TAI base makes no jump since the new offset is already taken into account. The only issue with this is that for the time around midnight the clock is free running without a reference.

3 Register Set

This is the register set of the DCF 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
Dcf MasterControl Reg	DCF Master Enable Control Register	0x00000000	RW
Dcf MasterStatus Reg	DCF Master Error Status Register	0x00000004	WC
Dcf MasterVersion Reg	DCF Master Version Register	0x0000000C	RO
Dcf MasterCorrection Reg	DCF Slave Second Corrections Register	0x00000010	RW

Table 4: Register Set Overview

3.2 Register Descriptions

3.2.1 General

3.2.1.1 DCF Master Control Register

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

DCF MasterControl Reg	
Reg Description	
31 … 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 DCF Master Status Register

Shows the current status of the DCF Master Clock.

Dcf 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
ENABLE	Error (sticky)	Bit: 0	WC

3.2.1.3 DCF 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.

Dcf 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

3.2.1.4 DCF Master Correction Register

Correction register to compensate for leap seconds between the different time domains. DCF is UTC time, all other time in the system is TAI, this leads to a correction of 36 seconds by 2016.

Dcf MasterCorrection 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
COR_SIGN	COR_S																														
RW	RW																														
Reset: 0x00000000																															
Offset: 0x0010																															

Name	Description	Bits	Access
COR_SIGN	Correction sign	Bit: 31	RW
COR_S	Correction in seconds to the time extracted from the DCF => used to convert between TAI, UTC and GPS (leap seconds)	Bit: 30:0	RW

4 Design Description

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

4.1 Top Level – Dcf 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
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 DCF from the output signal to the connector.
AxiAddressRange Low_Gen	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	std_logic_vector	32	AXI Base Address plus Registerset Size Default plus 0xFFFF
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 Dcf_MasterStaticConfig_Type

Defined in Dcf_MasterAddrPackage.vhd of library DcfLib
This is the type used for static configuration.

Field Name	Type	Size	Description
Correction	Clk_Time_Type	1	Time to correct TAI to UTC or another base.

Table 7: Dcf_MasterStaticConfig_Type

4.1.1.2.3 Dcf_MasterStaticConfigVal_Type

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

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

Table 8: Dcf_MasterStaticConfigVal_Type

4.1.1.2.4 Dcf_MasterStaticStatus_Type

Defined in Dcf_MasterAddrPackage.vhd of library DcfLib
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: Dcf_MasterStaticStatus_Type

4.1.1.2.5 Dcf_MasterStaticStatusVal_Type

Defined in Dcf_MasterAddrPackage.vhd of library DcfLib

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: Dcf_MasterStaticStatusVal_Type

4.1.1.3 Entity Block Diagram

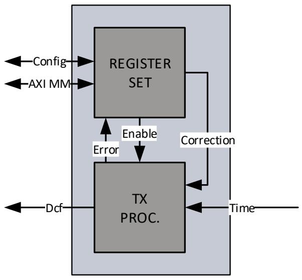

Figure 4: DCF Master Clock

4.1.1.4 Entity Description

Tx Processor

This module handles the generation of the DCF signal. It adds the Correction and converts the local clock time in TAI format in seconds since 1.1.1970 without leap seconds into UTC time in time of day format and then into a BCD encoded time and encodes this then to a DCLS encoded continuous PWM signal. It aligns the second boundary exactly to the reference Mark symbol when generating the DCF data stream. It takes the output delay because of e.g. external driver ICs into account and asserts the internal signal earlier respectively.
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 DCF 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				
StaticConfig_Gen	-	boolean	1	If Static Configuration or AXI is used
ClockClkPeriod Nanosecond_Gen	-	natural	1	Integer Clock Period
OutputDelay Nanosecond_Gen	-	natural	1	Output delay of the DCF from the output signal to the connector
AxiAddressRange Low_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size
Sim_Gen	-	boolean	1	If in Testbench simulation mode
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig_DatIn	in	Dcf_Master StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Dcf_Master StaticConfigVal _Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Dcf_Master StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Dcf_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 Re-sponse
DCF Output				

Dcf_DatOut	out	std_logic	1	DCF output

Table 11: DCF Master Clock

4.2 Design Parts

The DCF 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

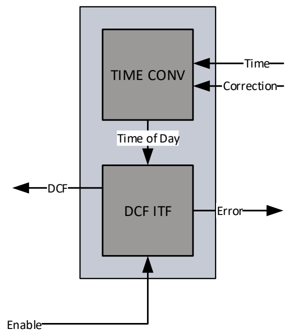

Figure 5: TX Processor

4.2.1.2 Entity Description

DCF Interface Adapter

This module converts the UTC time in time of day format to a BCD encoded time and encodes this then to a DCLS encoded continuous PWM signal. It aligns the minute boundary exactly to the reference Mark symbol when generating the DCF data stream. It takes the output delay because of e.g. external driver ICs into account and asserts the internal signal earlier respectively. When disabled the DCF signal will be constantly '0'. If the time makes a time jump it will stop generating the DCF stream immediately and will pull the DCF signal to '0', it will continue with generation at the next minute boundary.

Time Converter

This module adds the Correction and converts the time from seconds since midnight 1.1.1970 into the Time of Day format: hh:mm:ss ddd:yyyy with weekdays. It loops over the years and days taking the leap years into account and finally extracts the hours, minutes and seconds. After this conversion, a final correction is done to adjust the second to the next second. Then this time is passed to the DCF Interface Adapter module.

4.2.1.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond
TX Processor				
OutputDelay Nanosecond_Gen	-	natural	1	Output delay of the DCF from the out-put signal to the connector
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System 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
DCF Error Output				
Dcf_ErrOut	out	std_logic_vector	1	Indicates a time jump
DCF Output				
Dcf_DatOut	out	std_logic	1	DCF output
DCF Correction Input				
DcfCorrection_DatIn	in	Clk_Time_Type	1	Additional correc-tion to the transmit-ted TAI time
Enable Input				
Enable_EnaIn	in	std_logic	1	Enables the correc-tion and supervision

Table 12: TX Processor

4.2.2 Registerset

4.2.2.1 Entity Block Diagram

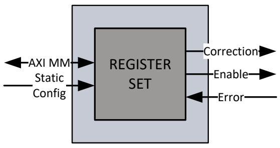

Figure 6: Registerset

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 DCF 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. 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				
Register Set				
StaticConfig_Gen	-	boolean	1	If Static Configuration or AXI is used
AxiAddressRange Low_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRange High_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	Dcf_Master StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Dcf_Master StaticConfigVal _Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Dcf_Master StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Dcf_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 _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
DCF Error Input				
Dcf_ErrIn	in	std_logic_vector	1	Indicates a time jump or other Error conditions
DCF Correction Output				
DcfCorrection_DatOut	out	Clk_Time_Type	1	Additional correction to the transmitted TAI time
Enable Output				
DcfMaster Enable_DatOut	out	std_logic	1	Enables the correction 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 DcfStaticConfigMaster_Con : Dcf_MasterStaticConfig_Type := (
  Correction           => (
    Second               => x"00000024", -- UTC 36 leap seconds
    Nanosecond           => (others => '0'), -- no nanoseconds
    Fraction             => (others => '0'), -- no fractions
    Sign                 => '1', -- UTC correct in negative
    TimeJump             => '0')
);

constant DcfStaticConfigValMaster_Con : Dcf_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 DCF Master Clock is 0x10000000.

```
-- DCF MASTER
-- Config
-- correction of minus 36 second to convert TAI to UTC
AXI WRITE 10000010 80000024
-- enable DCF Master
AXI WRITE 10000000 00000001
```

Figure 8: AXI Configuration

In the example the Correction is first set to minus 36 seconds 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 DCF 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 DCF Master runs on as well as the counter clock etc.
DCF Interface		
DCF	1 Hz	No clock, asynchronous data signal.
AXI Interface		
AXI Clock	50MHz (Default)	Internal AXI bus clock, same as the system clock

Table 14: Clocks

4.4.2 Reset

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 (Static onfiguration)	657	2167	0	0
Maximal (AXI configuration)	701	2222	0	0

Table 16: Resource Usage Intel/Altera

5.2 AMD/Xilinx (Artix 7)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (Static onfiguration)	590	1715	0	0
Maximal (AXI configuration)	632	1784	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

DCF                                    -- DCF library folder
|-Core                                 -- DCF library cores
 |-Doc                                 -- DCF library cores documentations
 |-Library                             -- DCF library component sources
 |-Package                             -- DCF library package sources
 |-Refdesign                           -- DCF library cores reference designs
 |-Testbench                           -- DCF 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 Dcf Master testbench consist of 2 parse/port types: AXI and DCF.
The DCF RX port takes the time of the Clock instance as reference and the DCF stream from the DUT port. The DCF RX port checks the waveform if the stream is encoded correctly and if the second boundary is asserted at the correct point in time. 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.

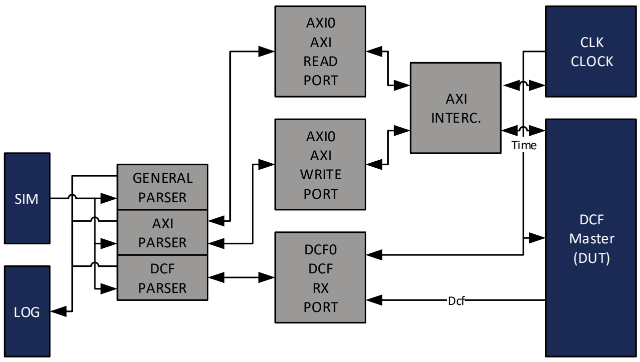

Figure 9: Testbench Framework

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/DCF/Testbench/Core/DcfMaster/Script/run_Dcf_Master_Tb.tcl
```

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

8 Reference Designs

The DCF Master reference design contains a PLL to generate all necessary clocks (cores are run at 50 MHz), an instance of the DCF Master Clock IP core and an instance of the Adjustable Counter Clock IP core (needs to be purchased separately). Optionally it also contains an instance of an PPS Master Clock IP core (has to be purchased separately). To instantiate the optional IP core, change the corresponding generic (PpsMasterAvailable_Gen) to true via the tool specific wizards.
The Reference Design is intended to be connected to any DCF Slave which can handle a logic high single ended DCF signal. The PPS Master Clock is used to create a PPS output which is compensated for the output delay and has a configurable duty cycle, if not available an uncompensated PPS is directly generated out of the MSB of the Time.
All generics can be adapted to the specific needs.

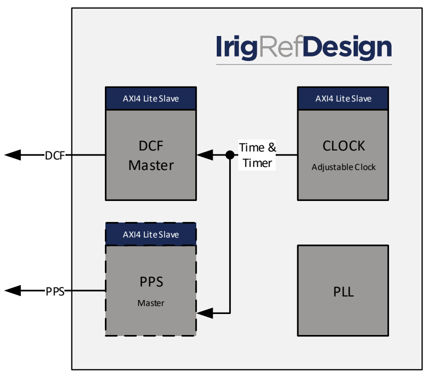

Figure 10: Reference Design

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 /DCF/Refdesign/Altera/SocKit/DcfMaster/DcfMaster.qpf
3. If the optional core PPS Master Clock is available add the files from the corresponding folders (PPS/Core, PPS/Library and PPS/Package)
4. Change the generics (PpsMasterAvailable_Gen) in Quartus (in the settings menu, not in VHDL) to true for the optional cores that are available.
5. Rerun implementation

6. Download to FPGA via JTAG

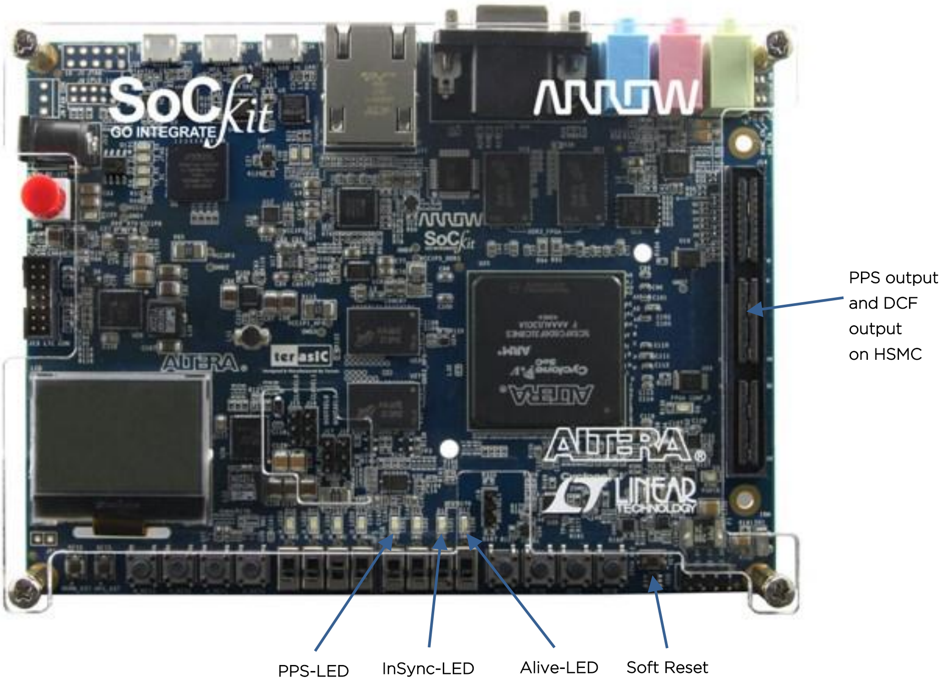

Figure 11: SocKit (source Terasic Inc)

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 /DCF/Refdesign/Xilinx/Arty/DcfMaster/DcfMaster.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. If the optional core PPS Master Clock is available add the files from the corresponding folders (PPS/Core, PPS/Library and PPS/Package) to the corresponding Library (PpsLib).
5. Change the generics (PpsMasterAvailable_Gen) in Vivado (in the settings menu, not in VHDL) to true for the optional cores that are available.
6. Rerun implementation
7. Download to FPGA via JTAG

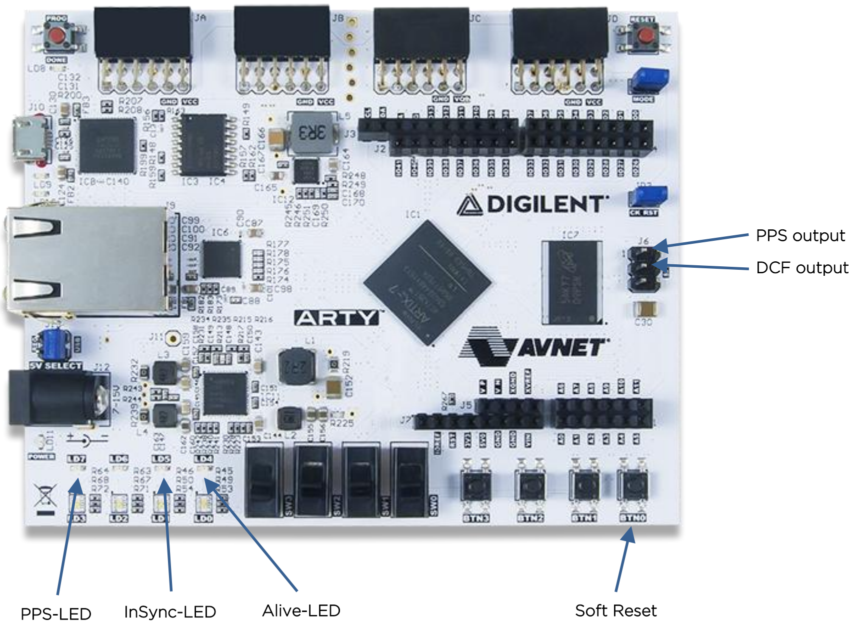

Figure 12: Arty
(source Digilent Inc)

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