ClkConvFreq

Reference Manual

Product Info	
Product Manager	Sven Meier
Author(s)	Kevin Schärer
Reviewer(s)	Sven Meier
Version	0.4
Date	21.01.2025

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 Clock (CLK) Precision Frequency Converter Core (ConvFreq) is a versatile solution designed for precise frequency drift adjustment in applications where ultra-high accuracy is essential. This documentation outlines the core's capabilities, functions, and architecture, providing a comprehensive guide for integrating ConvFreq into your system.

Key Features:

- Precise and Real-Time Frequency Drift Adjustment
- Adaptive Pull Range
- Manual Access to I^2C registers of Oscillator over AXI
- Oscillator Interface via I^2C
- Integration-Friendly
- Support for the SiTime SIT5356 Oscillators (Stratum 3 Super-TCXO)
- Support for the SiTime SiT5811 Oscillators via Generics
- Optional fractional adjustments (fractions of nanoseconds)

Revision History

This table shows the revision history of this document.

Version	Date	Revision
0.1	08.08.2023	First draft
0.2	17.08.2023	Review
0.3	12.07.2024	Review
0.4	21.01.2025	Add support for SiT5811 via Generics

Table 1: Revision History

Content

Definitions

Definitions	

Table 2: Definitions

Abbreviations

Abbreviations	
AXI	AMBA4 Specification (Stream and Memory Mapped)
IRQ	Interrupt, Signaling to e.g. a CPU
PPB	Parts Per Billion
TS	Timestamp
CLK	Clock
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 ConvFreq IP core plays a pivotal role in applications that demand ultra-precise frequency adjustments. It acts as an intermediary between a reference signal source and an oscillator, ensuring the oscillator's output frequency matches the desired value with astounding accuracy. The context block diagram (Figure 1) illustrates how ConvFreq fits within the larger system.

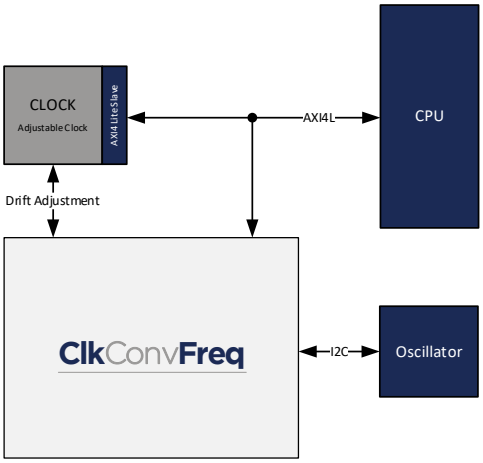

Figure 1: Context Block Diagram

1.2 Function

ConvFreq operates by receiving frequency drift adjustments in parts per billion (PPB) as input. It then processes this input to produce a precise control signal that is transmitted via the I2C protocol to an oscillator. The oscillator uses this control signal to adjust its output frequency accordingly, ensuring it aligns precisely with the desired value, even at extreme levels of accuracy.

2 Design Description

2.1 Top Level – CLK ConvFreq

2.1.1.1 Entity Block Diagram

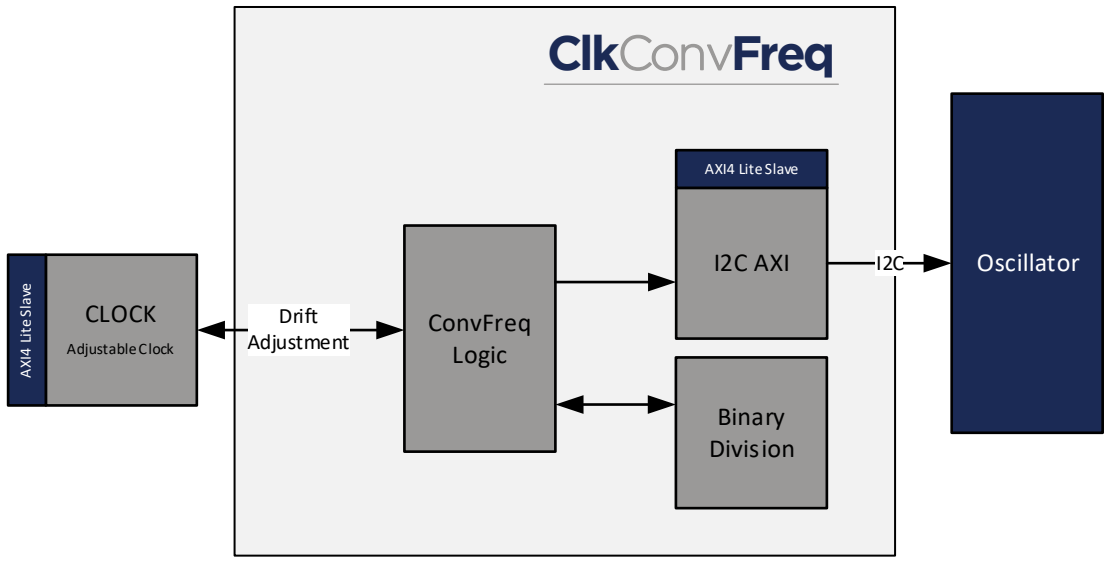

Figure 2: ConvFreq Block Diagram

2.1.1.2 Entity Description

At the highest level, the ConvFreq IP core manages the entire process of frequency adjustment. It receives PPB (ns/s) drift adjustment values (and fractions if available, IPI does not use fractions), processes them (converts it to the format required by the oscillator, which is a step value depending on the pull range which requires a divison by the pull range) with optional adaptive pull range adjustments and interfaces with the oscillator via I2C. The AXI interface serves as an alternative communication link, enabling configuration of registers in the oscillator independently.
At reset the Oscillator is reset to 0 and the default pull range.

2.1.1.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
Clk_OscType_Gen	-	boolean	1	Selects Oscillator Type which one wants to control -

				SiT5356_E or SiT5811_E are currently supported
Adaptive-PullRange_Gen	-	boolean	1	Whether frequency pull range should be adapted, depending on the input PPB
Sim_Gen	-	boolean	1	Set to true when in a simulation environment, otherwise false
ClockClkPeriodNanosecond_Gen	-	natural	1	Integer Clock Period
I2cClkPeriodNanosecond_Gen	-	natural	1	I2C Clock Period
I2cReadWaitTimeNanosecond_Gen	-	natural	1	I2C Read Wait Time in nanoseconds
I2cWriteWaitTimeNanosecond_Gen	-	natural	1	I2C Write Wait Time in nanoseconds
I2cChipAddressWidth_Gen	-	natural	1	I2C chip address bit width
I2cRegAddressWidth_Gen	-	natural	1	I2C register address bit width
I2cRegDataWidth_Gen	-	natural	1	I2C register data bit width
OscI2c-ChipAddress_Gen	-	natural	1	Oscillator I2C Chip address
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
Status				
Clk_StaticStatus_DatI	in	Clk_ClockStatic	1	Static Status (holds

n		Status_Type		drift adjustment)
Clk_StaticStatus_Valln	in	Clk_ClockStaticStatusVal_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
I2C				
I2cClk_ClkOut	out	std_logic	1	I2C Clock
I2cSda_DatInOut	i/o	std_logic	1	I2C Data

Table 4: ConvFreq Entity Declaration

2.2 Clocking and Reset Concept

2.2.1 Clocking

To keep the design as robust and simple as possible, the whole ConvFreq 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. 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 OC 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 5: Clocks

2.2.2 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.

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 6: Resets

A List of tables

B List of figures