
Designing a High-Speed Timer in SmartFusion Fabric

Table of Contents

Introduction	1
Design Example Overview	1
Description of the Timer Block	2
Interface and Register Description	4
Hardware Implementation	6
Running the Design	7
Conclusion	8
Appendix A – Design Files	9
List of Changes	9

Introduction

This document explains how to implement an independent timer in SmartFusion[®] customizable system-on-chip (cSoC) devices using FPGA fabric. The design example uses a timer with an advanced high-performance bus lite (AHB-Lite) interface. This has been implemented so that the timer can be run with an independent clock that is asynchronous to the microcontroller subsystem (MSS) clock. The independent clock can run at a much higher speed than the MSS clock, allowing applications in which an event must be captured that has a much higher resolution than the MSS clock. This timer can be started with a programmable FPGA I/O. The MSS can initialize the timer, load the timer value, and also start the timer. The fabric logic is used to customize the external interface. The logic that interfaces the timer with the MSS takes care of synchronization so that this timer can act as a standard slave to the MSS; running with an independent clock.

Design Example Overview

The design example shows a 24-bit timer module with an AHB-Lite interface. It acts as a slave to the MSS and can be connected to the MSS via the fabric interface controller (FIC). [Figure 1 on page 2](#) shows how the timer can be connected to the MSS.

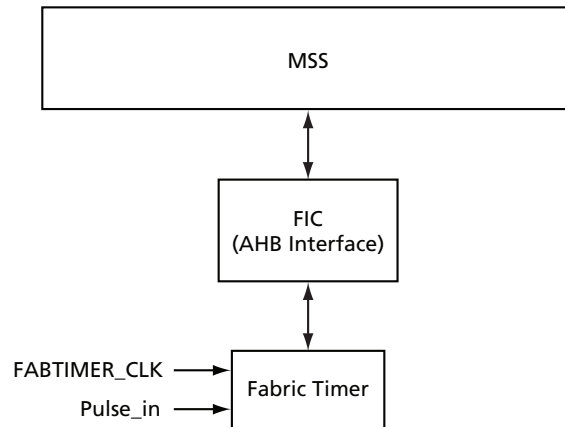


Figure 1 • Fabric Timer Connected to MSS via FIC

Description of the Timer Block

The timer has a decremting counter and its count value is loaded from the MSS. It has several registers that can be accessed through the AHB-Lite interface. The timer can be started by the MSS or the FPGA I/O. The counter uses FABTIMER_CLK, which can run at a separate frequency than the MSS clock. The maximum MSS clock frequency is 100 MHz. However, this design example allows running FABTIMER_CLK at a much higher speed than the MSS clock. [Figure 2](#) shows the top-level block diagram of the timer. It has several unit/components: sync block, prescale, interrupt, sampling, and decremting counter.

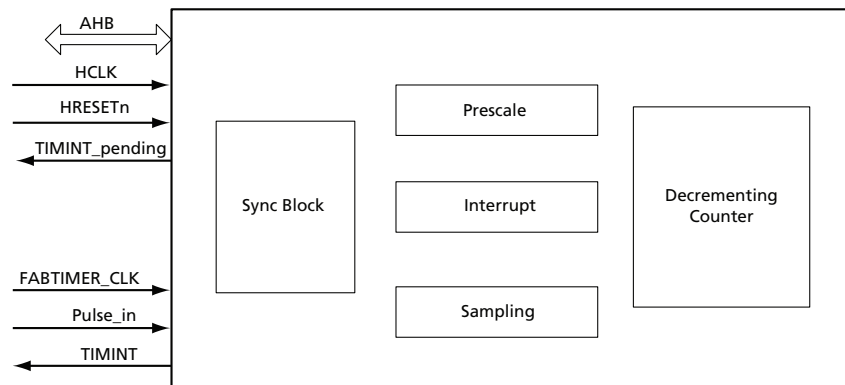


Figure 2 • Fabric Timer Top-Level Block Diagram

Sync Block

This block performs the clock synchronization. It monitors AHB-Lite bus signals and passes various register settings from the MSS clock to the FABTIMER_CLK clock. It also allows reading the register settings from the FABTIMER_CLK clock domain and passing it to the MSS clock domain. This block decodes the AHB-Lite address and depending on the HWRITE signal, performs a read or write operation. The design example assumes that the MSS clock and the FABTIMER_CLK are asynchronous to each other. It uses the handshake protocol to transfer data between these two asynchronous clock domains.

A handshake protocol circuit uses less logic and guarantees that all bits of a data bus crossing an asynchronous clock domain are registered by the same clock edge in the receiving clock domain. The two domains exchange data via the `reg_load_en` signal, which is the request signal from HCLK to the FABTIMER_CLK domain, and the `reg_load_ack` signal, which is the acknowledge signal from FABTIMER_CLK to the HCLK domain. Figure 3 shows this state simplified. The basic operation sequence has the following steps:

1. The HCLK domain sends a `reg_load_en0` signal to the FABTIMER_CLK domain, which is registered at FABTIMER_CLK domain, to generate `reg_load_en2`.
2. The FABTIMER_CLK domain at this point reads the AHB_lite bus signals from the HCLK domain and updates the appropriate registers.
3. After this read operation has finished, the FABTIMER_CLK domain sends a `reg_load_ack0` signal, which gets registered by HCLK domain, to generate `reg_load_ack2`.
4. The HCLK domain turns off the `reg_load_en0`, which eventually turns off the `reg_load_en2`.
5. Once the FABTIMER_CLK domain sees a low `reg_load_en2` signal, it turns off the `reg_load_ack0` signal to the FABTIMER_CLK domain.
6. Once the HCLK domain sees the low `reg_load_ack2` signal, the data transfer between the two asynchronous domains is complete.

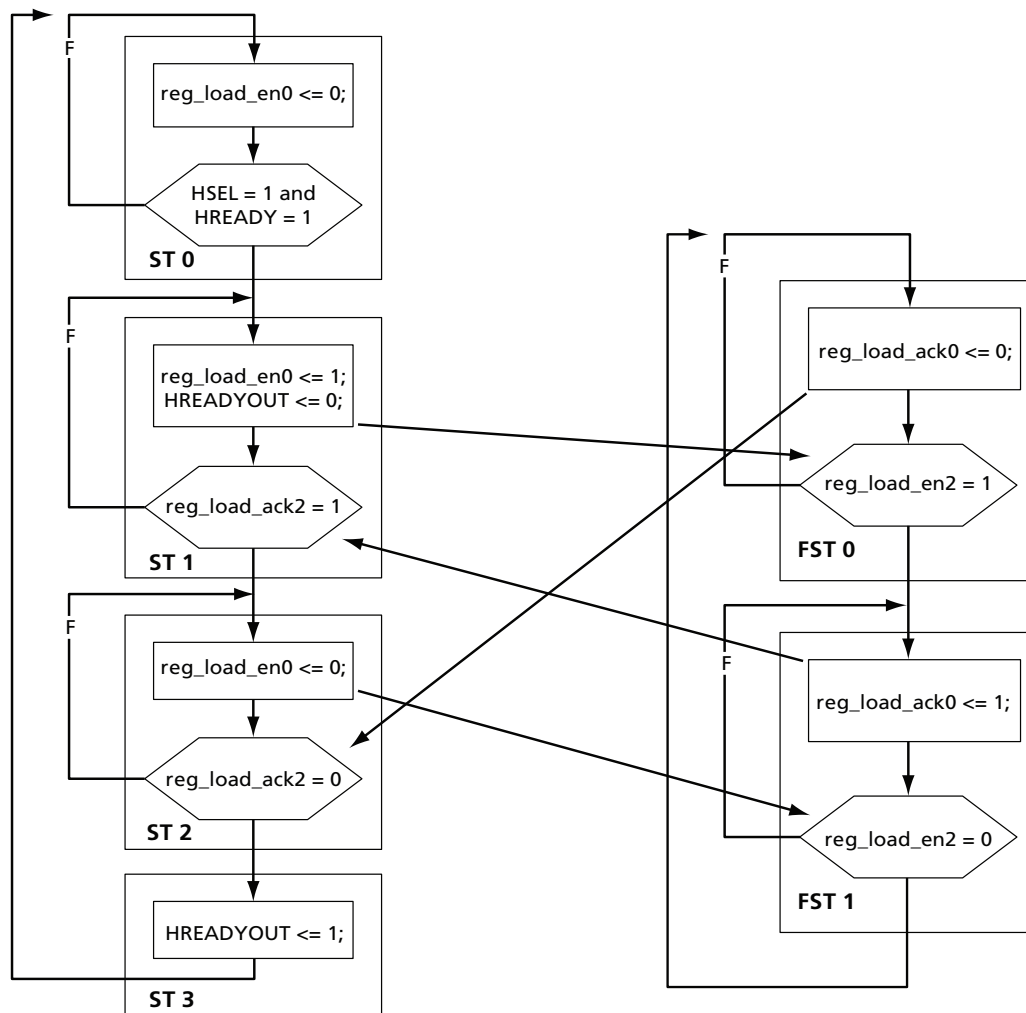


Figure 3 • Handshaking Protocol Between HCLK and FABTIMER_CLK Domain

Prescale

The prescale component is used to provide a clock enable pulse for the decrementing counter. It allows dividing the counter clock (FABTIMER_CLK) by 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1,024. Note that the FABTIMER_CLK can be driven for the clock condition circuit (CCC) or it can be driven by the FPGA I/O.

Decrementing Counter

The decrementing counter is a 24-bit down counter. The counter can operate in two modes: Continuous mode and One-shot Timer mode. In Continuous mode, when the counter reaches zero, the counter is reloaded with the start value, which is stored in a programmable register, and continues to count down. In One-shot Timer mode, the counter decrements from its high value and halts on reaching zero. The timer must be reprogrammed from the MSS to begin counting down again.

Interrupt

The interrupt unit handles the interrupt generation. Two output ports, TIMINT and TIMINT_pending, show two types of interrupts. TIMINT is generated every time the counter reaches zero. TIMINT_pending holds the interrupt and is only cleared when the interrupt clears the register to which TIMERINTCLRA is written. Since the counter is running in the FABTIMER_CLK domain, this block stretches the interrupt pulse so it can be captured by HCLK and also synchronizes the interrupt signals with HCLK.

Sampling

The sampling block samples Pulse_in input and detects the rise and fall condition. The counter starts when Pulse_in has a falling edge.

Interface and Register Description

Table 1 gives the port descriptions for the design example.

Table 1 • Interface Description

Signal	Direction	Description
HCLK	Input	AHB bus clock. This clock times all bus transfers.
HRESETn	Input	Reset. The bus reset signal is active Low and is used to reset the system and the bus. This is the only active Low AHB signal.
HADDR[4:0]	Input	AHB address.
HWRITE	Input	Transfer direction. When High, this signal indicates a write transfer and when Low, a read transfer.
HSIZE[2:0]	Input	Transfer size. Indicates the size of the transfer, which can be byte (8-bit), half-word (16-bit), or word (32-bit). 000 – Byte, 001 – Half-word, and 010 – Word. Only word size transfer is used in the design example.
HWDATA[31:0]	Input	32-bit data from the master
HREADY	Input	Ready signal from all other AHB slaves.
HSEL	Input	Combinatorial decode of HADDR, which indicates that this slave is currently selected.
HRDATA[31:0]	Output	32-bit data written back to the master.
HREADYOUT	Output	Transfer done. When High, the HREADY signal indicates that a transfer has finished on the bus. This signal can be driven Low to extend a transfer.

Table 1 • Interface Description (continued)

HRESP[1:0]	Output	Transfer response, which has the following meanings: 00 = Okay 01 = Error 10 = Retry 11 = Split
HTRANS[1:0]	Input	Types of transfer: 00 = Idle 01 = Busy 10 = Non-sequential 11 = Sequential
TIMINT	Output	Interrupt output for timer. This signal indicates that an interrupt has been generated by the counter having decremented to zero.
TIMINT_pending	Output	Pending interrupt output for timer. This signal indicates that a pending interrupt has been generated by the counter having decremented to zero.
Pulse_in	Input	Input pulse.
FABTIMER_CLK	Input	Fabric clock, can come from Fabric I/O or clock conditioning circuit.
FAB_RESETn	Input	Fabric reset, can be driven from the MSS.

Several registers are available to dynamically control the operation of the timer. [Table 2](#) shows the register settings.

Table 2 • Register Map

Offsets	Type	Reset Values	Name	Description
0x00	W	0x0000	TIMERLOADA	Load address
0x04	R	0x0000	TIMERVALUEA	Timer load value
0x08	R/W	0x0000	TIMERCONTROLA	Control register Bit 2: Timer operation mode 0 = Continuous operation 1 = One-shot count Bit 1: Interrupt enable 0 = Interrupt disabled 1 = Interrupt enabled Bit 0: Enable bit for timer 0 = Timer disabled 1 = Timer enabled
0x0C	R/W	0x0000	TIMERPRESCALEA	Prescale setting Bit 3: 0 Prescale field, based on FAB_CLK 0000 = Divide by 1 (default) 0001 = Divide by 2
0x10	R/W	0x0000	TIMERINTCLRA	Interrupt clear register. Any write to this register will clear (deassert) the TIMINT_pending interrupt output.
0x14	R	0x0000	TIMERTIMINT1	Input pulse start value
0x18	R	0x0000	TIMERTIMINT2	Input pulse stop value

Hardware Implementation

The design example is available as a VHDL source file (refer to "Appendix A – Design Files" on page 9). This HDL file can be imported to any existing SmartFusion cSoC project or any new project. Here are the important steps for using the HDL file in a Libero[®] System-on-Chip (SoC) software:

1. Import the HDL files (AHB_timer.vhd and Decrementor.vhd) into the Libero SoC project.
2. Configure the fabric interface inside the MSS configurator by double-clicking the **Fabric Interface** sub-configurator.
3. Configure the FIC with an AHB-Lite interface (Figure 4), and check **Use Master Interface**.

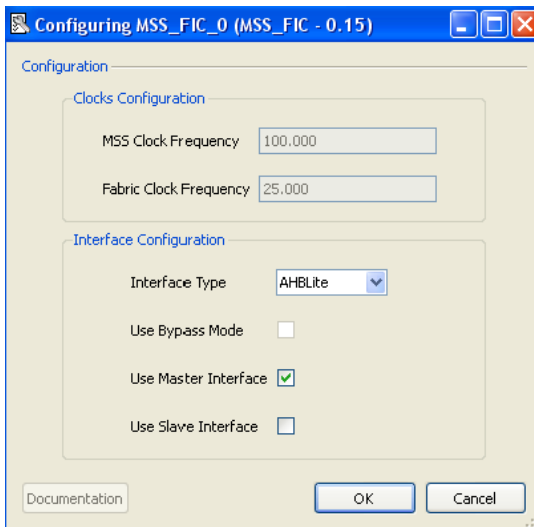


Figure 4 • Configuring the Fabric Interface Controller

4. Select **MSS_Master_AHB** bus interface on the fabric interface and promote to top level.
5. Configure the other block inside the MSS configurator as needed and generate the MSS.

- Add a bus interface to AHB_timer logic. Right-click on the AHB timer HDL file in the Design Hierarchy and select **Create Core from HDL**. Click on **Add Bus Interface** and select **AHB slave**. The **Edit Core Definition** window is displayed as shown in [Figure 5](#). Click **Map by Name** to map the signals automatically and the configurator attempts to map any similar signal names between the bus definition and pin names on the instance. Map other signals manually that are not mapped by **Map by Name** and click **OK**.

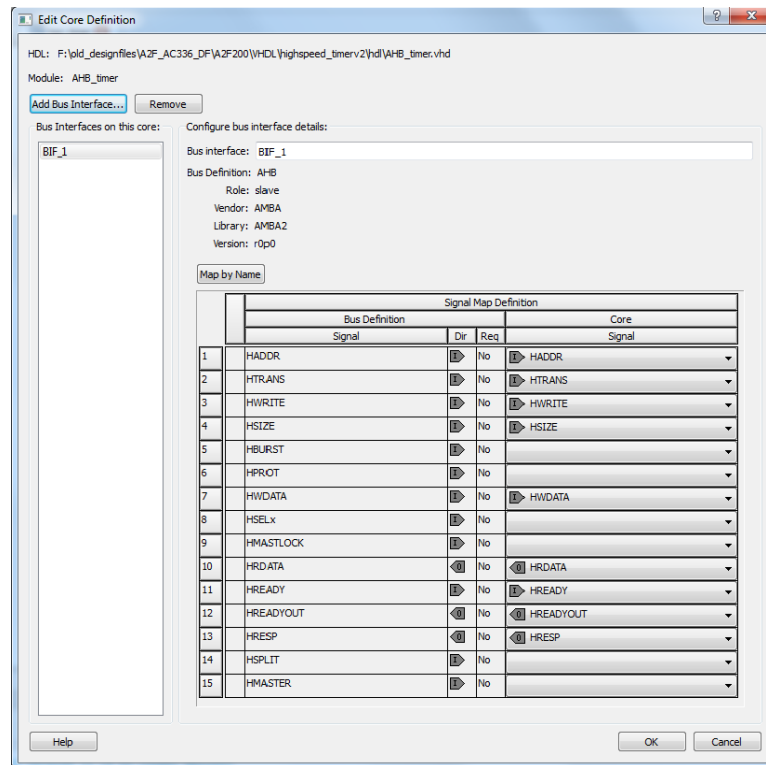


Figure 5 • Add Bus Interface Dialog Box

- Instantiate the AHB_timer.vhd in SmartDesign.
- Connect all other subsystem signals as needed and generate a top-level HDL file.
- Run **BFM simulation**. Use the user.bfm to add BFM to exercise the AHB timer block.
- Run **synthesis** with appropriate Synopsys Design Constraint (*.sdc) file during synthesis.
- Run place-and-route with appropriate pin assignment.

Running the Design

Board Settings

The design example works on the SmartFusion Development Kit Board and the SmartFusion Evaluation Kit Board with default board settings. Refer to the following user's guides for default board settings:

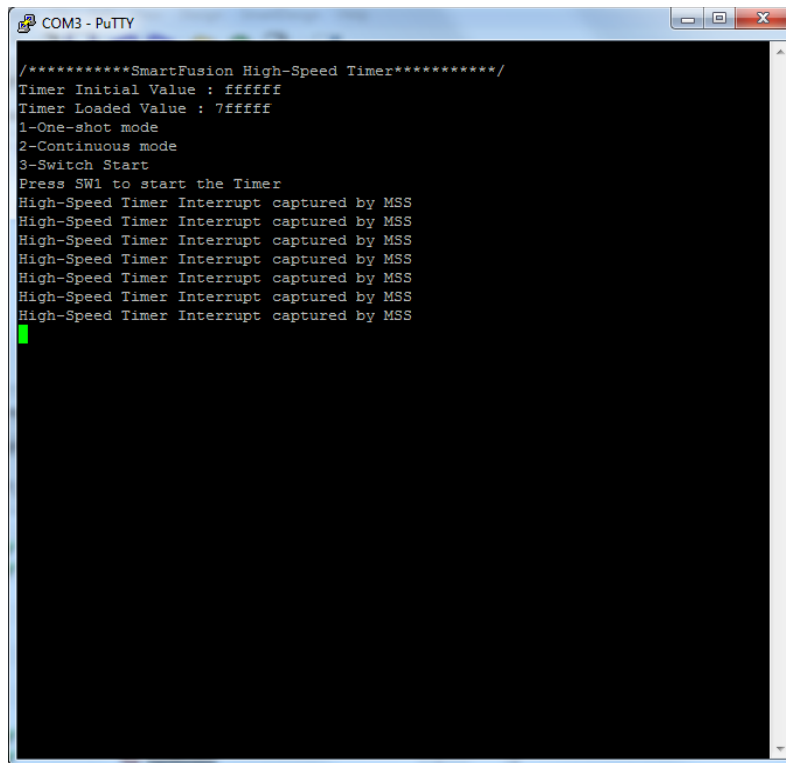
- [SmartFusion Development Kit User's Guide](#)
- [SmartFusion Evaluation Kit User's Guide](#)

Program the Design and Running the Application

Program the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board with the generated/provided STP file (refer to "[Appendix A – Design Files](#)" section on page 9) using FlashPro and then power cycle the board. Invoke the SoftConsole IDE from the Libero SoC project (refer to "[Appendix A – Design Files](#)" section on page 9) and launch the debugger.

Start a HyperTerminal with a baud rate of 57600, 8 data bits, 1 stop bit, no parity, and no flow control. If your computer does not have the HyperTerminal program, use any free serial terminal emulation program such as PuTTY or Tera Term. Refer to the [Configuring Serial Terminal Emulation Programs](#) tutorial for configuring the HyperTerminal, Tera Term, and PuTTY.

When you run the debugger in SoftConsole, the PuTTY window provides the user interface to select Timer mode. [Figure 6](#) shows the screenshot of the PuTTY.



```
COM3 - PuTTY
/*****SmartFusion High-Speed Timer*****/
Timer Initial Value : ffffff
Timer Loaded Value : 7fffff
1-One-shot mode
2-Continuous mode
3-Switch Start
Press SW1 to start the Timer
High-Speed Timer Interrupt captured by MSS
High-Speed Timer Interrupt captured by MSS
High-Speed Timer Interrupt captured by MSS
High-Speed Timer Interrupt captured by MSS
High-Speed Timer Interrupt captured by MSS
High-Speed Timer Interrupt captured by MSS
High-Speed Timer Interrupt captured by MSS
High-Speed Timer Interrupt captured by MSS
```

Figure 6 • PuTTY Window

Release mode

The release mode programming file (STAPL) is also provided. Refer to the Readme.txt file included in the programming zip file for more information.

Refer to [Building Executable Image in Release Mode and Loading into eNVM tutorial](#) for more information on building an application in release mode.

Conclusion

SmartFusion cSoC FPGA devices have a programmable high-performance analog block, FPGA fabric, and a hardened ARM[®] Cortex[™]-M3 processor microcontroller block. The microcontroller block is composed of a 100 MHz Cortex-M3 processor and standard microcontroller peripherals, including two timers. Similar to a standard microcontroller, these timers are clocked with the PCLK0 that is generated from a microcontroller clock, also called MSS clock. This maximum PCLK0 frequency is 100 MHz.

However, the FPGA fabric allows creating a custom timer with an independent clock that can be asynchronous to the MSS clock. This clock can run at a much higher speed than the MSS clock and help applications, where an event with higher resolution than the MSS clock is needed, to be captured. This application note shows the design example for that timer.

Appendix A – Design Files

You can download the design files from the Microsemi SoC Products Group website: www.microsemi.com/soc/download/rsc/?f=A2F_AC336_DF. The design zip file consists of Libero SoC projects, programming file (*.stp) for A2F500, and A2F200. Refer to the Readme.txt file included in the design file for directory structure and description.

You can download the programming files (*.stp) in release mode from the Microsemi SoC Products Group website: www.microsemi.com/soc/download/rsc/?f=A2F_AC336_PF. The programming zip file consists of STAPL programming files (*.stp) for A2F500, A2F200, and a Readme.txt file.

List of Changes

The following table lists critical changes that were made in each revision of the document.

Revision*	Changes	Page
Revision 4 (January 2013)	Added "Board Settings" section and modified "Running the Design" section (SAR 43469).	7
Revision 3 (February 2012)	Removed ".zip" extension in the links (SAR 36763).	9
Revision 2 (January 2012)	Modified point 5 and 6 listed under "Hardware Implementation" section (SAR 35817).	6
	Added new Figure 5 and Figure 6 (SAR 35817).	7, 8
	Added new sections - "Running the Design" and "Release mode" (SAR 35817).	7, 8
	Modified "Appendix A – Design Files" section (SAR 35817).	9
Revision 1 (September 2010)	Modified Table 2 (SAR 27468).	5

*Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.*



Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at www.microsemi.com.

© 2013 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.