Table of Contents

IP Facts

Chapter 1: Overview
  Functional Description .................................................. 5
  Feature Summary .......................................................... 7
  Applications .............................................................. 8
  Licensing and Ordering Information .................................. 8

Chapter 2: Product Specification
  Performance ............................................................... 9
  Resource Utilization .................................................... 9
  Port Descriptions ....................................................... 10
  Register Space .......................................................... 11

Chapter 3: Designing with the Core
  Clocking ................................................................. 18
  Resets ................................................................. 18
  Programming Sequence ................................................ 18

Chapter 4: Design Flow Steps
  Customizing and Generating the Core ................................ 23
  Constraining the Core ................................................ 25
  Simulation ............................................................... 26
  Synthesis and Implementation ......................................... 26

Chapter 5: Example Design
  Overview ............................................................... 27
  Implementing the Example Design .................................... 28
  Example Design Directory Structure ................................ 28
  Simulating the Example Design ....................................... 29
Chapter 6: Test Bench

Appendix A: Debugging
   Finding Help on Xilinx.com ................................................................. 31
   Debug Tools ...................................................................................... 32
   Hardware Debug .............................................................................. 33
   Interface Debug .............................................................................. 33

Appendix B: Migrating and Upgrading
   Migrating to the Vivado Design Suite .............................................. 34
   Upgrading in the Vivado Design Suite ............................................. 34

Appendix C: Additional Resources and Legal Notices
   Xilinx Resources .............................................................................. 35
   References ......................................................................................... 35
   Revision History .............................................................................. 36
   Please Read: Important Legal Notices ........................................... 37
Introduction

The LogiCORE™ IP AXI Timer/Counter is a 32/64-bit timer module that interfaces to the AXI4-Lite interface.

Features

- AXI interface based on the AXI4-Lite specification
- Two programmable interval timers with interrupt, event generation, and event capture capabilities
- Configurable counter width
- One Pulse Width Modulation (PWM) output
- Cascaded operation of timers in generate and capture modes
- Freeze input for halting counters during software debug

<table>
<thead>
<tr>
<th>LogiCORE IP Facts Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogiCORE IP Facts Table</td>
</tr>
<tr>
<td>Core Specifics</td>
</tr>
<tr>
<td>Supported Device Family</td>
</tr>
<tr>
<td>UltraScale™ Families,</td>
</tr>
<tr>
<td>UltraScale™ Architecture,</td>
</tr>
<tr>
<td>Zynq®-7000, 7 Series</td>
</tr>
<tr>
<td>Supported User Interfaces</td>
</tr>
<tr>
<td>AXI4-Lite</td>
</tr>
<tr>
<td>Resources</td>
</tr>
<tr>
<td>See Table 2-2.</td>
</tr>
<tr>
<td>Provided with Core</td>
</tr>
<tr>
<td>Design Files</td>
</tr>
<tr>
<td>VHDL</td>
</tr>
<tr>
<td>Example Design</td>
</tr>
<tr>
<td>VHDL</td>
</tr>
<tr>
<td>Test Bench</td>
</tr>
<tr>
<td>VHDL</td>
</tr>
<tr>
<td>Constraints File</td>
</tr>
<tr>
<td>XDC File</td>
</tr>
<tr>
<td>Simulation Model</td>
</tr>
<tr>
<td>Not Provided</td>
</tr>
<tr>
<td>Supported S/W Driver</td>
</tr>
<tr>
<td>Standalone and Linux</td>
</tr>
<tr>
<td>Tested Design Flows(3)</td>
</tr>
<tr>
<td>Design Entry</td>
</tr>
<tr>
<td>Vivado® Design Suite</td>
</tr>
<tr>
<td>Simulation</td>
</tr>
<tr>
<td>For support simulators, see the Xilinx Design Tools: Release Notes Guide.</td>
</tr>
<tr>
<td>Synthesis</td>
</tr>
<tr>
<td>Vivado Synthesis</td>
</tr>
</tbody>
</table>

Support

Provided by Xilinx at the Xilinx Support web page

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. Standalone driver details can be found in the SDK directory (<install_directory>/SDK/<release>/data/embeddedsw/doc/xilinx_drivers.htm). Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.
Overview

Chapter 1

Functional Description

The AXI Timer is organized as two identical timer modules as shown in Figure 1-1. Each timer module has an associated load register that is used to hold either the initial value for the counter for event generation or a capture value, depending on the mode of the timer.

The *generate value*, which is the value loaded into the load register, is used to generate a single interrupt at the expiration of an interval or a continuous series of interrupts with a programmable interval. The *capture value* is the timer value that has been latched on detection of an external event. The clock rate of the timer modules is s_axi_aclk (no prescaling of the clock is performed). All of the timer/counter interrupts are OR'ed together to generate a single external interrupt signal. The interrupt service routine reads the control/status registers to determine the source of the interrupt.

The block diagram of AXI Timer, also known as AXI Timer/Counter, is shown in Figure 1-1.
Operation Overview

The AXI Timer/Counter modules provides an AXI4-Lite interface to communicate with the host. The timer/counter design has the following key modules:

- **AXI4-Lite Interface**: The AXI4-Lite Interface module implements an AXI4-Lite slave interface for accessing memory mapped Timer registers. For details about the AXI4-Lite slave interface, see the LogiCORE IP AXI4-Lite IPIF Data Sheet (DS765) [Ref 5].

- **Timer Registers**: The Register block implements a set of 32-bit registers for each timer/counter. This set of register contains load register, timer/counter register and control/status register.

- **32-bit Counters**: The Timer/Counter module has two 32-bit counters, each of which can be configured for up/down counts and can be loaded with a value from the load register.

- **Interrupt Control**: The Interrupt control module generates a single interrupt depending on the mode of operation.

- **Pulse Width Modulation (PWM)**: The PWM block generates a pulse signal, PWM0, with a specified frequency and duty factor. It uses Timer 0 for PWM0 period, and Timer 1 for PWM0 output width.

Timer Modes

Four modes can be used with the two Timer/Counter modules:

- **Generate Mode**
- **Capture Mode**
- **Pulse Width Modulation Mode**
- **Cascade Mode**

**Generate Mode**

In the Generate mode, the value in the load register is loaded into the counter. The counter, when enabled, begins to count up or down, depending on the selection of the Up/Down Count Timer (UDT) bit in the Timer Control Status Register (TCSR). See Figure 2-2, page 12 and Figure 2-5, page 15. On transition of the carry out of the counter, the counter stops or automatically reloads the generate value from the load register and, after reaching the timeout value, continues counting as selected by the Auto Reload/Hold (ARHT) bit in the TCSR. The Timer Interrupt Status (TINT) bit is set in TCSR and, if enabled, the external GenerateOut signal is driven to 1 for one clock cycle.

If enabled, the interrupt signal for the timer is driven to 1 when reaching the timeout value. Clear the interrupt by writing a 1 to the Timer Interrupt register. Use this mode for generating repetitive interrupts or external signals with a specified interval.
**Capture Mode**

In Capture mode, the value of the counter is stored in the load register when the external capture signal is asserted. The TINT bit is also set in the TCSR on detection of the capture event. The counter can be configured as an up or down counter for this mode as determined by the selection of the UDT bit in TCSR. The Auto Reload/Hold (ARHT) bit controls whether the capture value is overwritten with a new capture value before the previous TINT flag is cleared. Use this mode for time-tagging external events while simultaneously generating an interrupt.

**Pulse Width Modulation Mode**

In Pulse Width Modulation (PWM) mode, two timer/counters are used as a pair to produce an output signal (PWM0) with a specified frequency and duty factor. Timer 0 sets the period and Timer 1 sets the high time for the PWM0 output.

**Cascade Mode**

In the Cascade mode, the two timer/counters are cascaded to operate as a single 64-bit counter/timer. The cascaded counter can work in both generate and capture modes. TCSR0 acts as the control and status register for the cascaded counter. TCSR1 is ignored in this mode.

Use this mode when a timer/counter more than 32-bits wide is required. Cascaded operation requires using Timer 0 and Timer 1 together as a pair. The counting event for Timer 1 is when Timer 0 rolls over from all 1s to all 0s, or vice-versa when counting down.

**Interrupts**

The TC interrupt signals can be enabled or disabled with the Enable Interrupt for Timer (ENIT) bit in the TCSR. The interrupt status bit (TINT) in the TCSR cannot be disabled and always reflects the current state of the timer interrupt. In Generate mode, a timer interrupt is caused by the counter rolling over (the same condition used to reload the counter when ARHT is set to 1). In Capture mode, the interrupt event is the capture event.

**Feature Summary**

The features of the AXI Timer/Counter core are as follows:

- 32-bit slave peripheral with AXI4-Lite Interface
- Two programmable interval timers with interrupt, event generation and capture capabilities
• Configurable counter width. Supports cascaded mode of timers in generate and capture modes.
• Supports programmable auto-reload & hold operation on each timer/counter.
• One Pulse Width Modulation (PWM) output
• Programmable interrupt generation to enable/disable the timer interrupt.
• Freeze input for halting counters during software debug

Applications
Applications of this core include:
• Clock generation with different time periods and duty cycle
• Pulse generation circuits
• Generating time-related interrupts
• Pulse Width Modulation signal for motor control

Licensing and Ordering Information
This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.
Chapter 2

Product Specification

This chapter contains specific details about the performance and resource utilization of the core.

Performance

Performance characterization of this core has been done using the margin system methodology. The details of the margin system characterization methodology is described in the Vivado Design Suite User Guide: Designing With IP (UG896) [Ref 13].

The summary of performance $F_{\text{MAX}}$ with this core performed on margin system is provided in Table 2-1.

*Note:* The performance numbers for UltraScale™ architecture-based and Zynq®-7000 devices are expected to be similar to 7 series device numbers.

**Table 2-1: Maximum Frequencies**

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Speed Grade</th>
<th>$F_{\text{MAX}}$ (MHz) AXI4-Lite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtex-7</td>
<td>-1</td>
<td>180</td>
</tr>
<tr>
<td>Kintex-7</td>
<td>-1</td>
<td>180</td>
</tr>
<tr>
<td>Artix-7</td>
<td>-1</td>
<td>120</td>
</tr>
<tr>
<td>Virtex-7</td>
<td>-2</td>
<td>200</td>
</tr>
<tr>
<td>Kintex-7</td>
<td>-2</td>
<td>200</td>
</tr>
<tr>
<td>Artix-7</td>
<td>-2</td>
<td>140</td>
</tr>
<tr>
<td>Virtex-7</td>
<td>-3</td>
<td>220</td>
</tr>
<tr>
<td>Kintex-7</td>
<td>-3</td>
<td>220</td>
</tr>
<tr>
<td>Artix-7</td>
<td>-3</td>
<td>160</td>
</tr>
</tbody>
</table>

Resource Utilization

Resources required for the AXI Timer core has been estimated for the 7 series FPGAs (Table 2-2). These values were generated using the Vivado® Design Suite.
The AXI Timer resource utilization for various parameter combinations measured on a 7 series device are detailed in Table 2-2.

**Note:** Resources numbers for UltraScale and Zynq-7000 devices are expected to be similar to 7 series device numbers.

**Table 2-2: Performance and Resource Utilization: 7 Series and Zynq-7000 Devices**

<table>
<thead>
<tr>
<th>Parameter Values</th>
<th>Device Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width of Timer/Counter</td>
<td>Enable Timer2</td>
</tr>
<tr>
<td>8</td>
<td>False</td>
</tr>
<tr>
<td>16</td>
<td>False</td>
</tr>
<tr>
<td>32</td>
<td>False</td>
</tr>
<tr>
<td>8</td>
<td>True</td>
</tr>
<tr>
<td>16</td>
<td>True</td>
</tr>
<tr>
<td>32</td>
<td>True</td>
</tr>
</tbody>
</table>

**Port Descriptions**

The AXI Timer input/output (I/O) signals are listed and described in Table 2-3.

**Table 2-3: I/O Signal Descriptions**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Interface</th>
<th>I/O</th>
<th>Initial State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_axi_aclk</td>
<td>Clock</td>
<td>I</td>
<td>NA</td>
<td>AXI Clock</td>
</tr>
<tr>
<td>s_axi_arstn</td>
<td>Reset</td>
<td>I</td>
<td>NA</td>
<td>AXI Reset, active-Low</td>
</tr>
<tr>
<td>s_axi_*</td>
<td>S_AXI</td>
<td>NA</td>
<td>NA</td>
<td>AXI4-Lite Interface signals. See Appendix A of the AXI Reference Guide (UG761) [Ref 2] for AXI4-Lite signals.</td>
</tr>
<tr>
<td>capturetrig0</td>
<td>Timer</td>
<td>I</td>
<td>-</td>
<td>Timer 0 Capture Trigger Input</td>
</tr>
<tr>
<td>capturetrig1</td>
<td>Timer</td>
<td>I</td>
<td>-</td>
<td>Timer 1 Capture Trigger Input</td>
</tr>
<tr>
<td>freeze</td>
<td>Timer</td>
<td>I</td>
<td>-</td>
<td>In cascade mode, this is not used.</td>
</tr>
<tr>
<td>generateout0</td>
<td>Timer</td>
<td>O</td>
<td>0x0</td>
<td>Timer 0 Generate Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Asserts whenever the timer 0 wraps from all 0s to all 1s or vice-versa.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In cascade mode, this signal is asserted when the 64-bit value wraps from</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>all 0s to all 1s or vice-versa.</td>
</tr>
<tr>
<td>generateout1</td>
<td>Timer</td>
<td>O</td>
<td>0x0</td>
<td>Timer 1 Generate Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Asserts when the timer1 wraps from all 0s to all 1s or vice-versa.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In cascaded mode, this signal is asserted when lower 32-bit counter (timer 0) wraps from all 0s to all 1s or vice-versa.</td>
</tr>
</tbody>
</table>
Table 2-3:  I/O Signal Descriptions (Cont’d)

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Interface</th>
<th>I/O</th>
<th>Initial State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interrupt</td>
<td></td>
<td>O</td>
<td>0x0</td>
<td>Indicates that the condition for an interrupt on this timer has occurred. If the timer mode is capture and the timer is enabled, this bit indicates a capture has occurred. If the mode is generate, this bit indicates the counter has rolled over. 0 = No interrupt has occurred 1 = Interrupt has occurred</td>
</tr>
<tr>
<td>pwm0</td>
<td>Timer</td>
<td>O</td>
<td>0x0</td>
<td>Pulse Width Modulation Output 0</td>
</tr>
</tbody>
</table>

Register Space

Timer Counter registers are accessed as one of these types:

- Byte (8 bits)
- Half word (2 bytes)
- Word (4 bytes)

The AXI Timer/Counter registers are organized as little-endian data. The bit and byte labeling for the little-endian data types is shown in the Figure 2-1.

Note: The AXI4-Lite write access register is updated by the 32-bit AXI Write Data (*_wdata) signal, and is not impacted by the AXI Write Data Strobe (*_wstrb) signal. For write access, both the AXI Write Address Valid (*_awvalid) and AXI Write Data Valid (*_wvalid) signals should be asserted together.

Table 2-4 shows all the AXI Timer registers and their addresses. Accesses to addresses inside the core address range other than these registers return an OKAY response, with reads returning 0 data values and writes having no effect.

Table 2-4:  Register Overview

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0h</td>
<td>TCSR0</td>
<td>Timer 0 Control and Status Register</td>
</tr>
<tr>
<td>04h</td>
<td>TLR0</td>
<td>Timer 0 Load Register</td>
</tr>
<tr>
<td>08h</td>
<td>TCR0</td>
<td>Timer 0 Counter Register</td>
</tr>
</tbody>
</table>

Figure 2-1:  Little Endian Data Types
Table 2-4: Register Overview (Cont’d)

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0Ch-0Fh</td>
<td>RSVD</td>
<td>Reserved</td>
</tr>
<tr>
<td>10h</td>
<td>TCSR1</td>
<td>Timer 1 Control and Status Register</td>
</tr>
<tr>
<td>14h</td>
<td>TLR1</td>
<td>Timer 1 Load Register</td>
</tr>
<tr>
<td>18h</td>
<td>TCR1</td>
<td>Timer 1 Counter Register</td>
</tr>
<tr>
<td>1Ch-1Fh</td>
<td>RSVD</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Control/Status Register 0 (TCSR0)

Figure 2-2 and Table 2-5 show the Control/Status register 0. Control/Status Register 0 contains the control and status bits for timer module 0.

Table 2-5: Control/Status Register 0 Bit Definitions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 - 12</td>
<td>Reserved</td>
<td>N/A</td>
<td>-</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
| 11   | CASC  | Read/Write  | 0           | Enable cascade mode of timers  
      |       |             |             | 0 = Disable cascaded operation  
      |       |             |             | 1 = Enable cascaded operation  
      |       |             |             | Cascaded operation requires using Timer 0 and Timer 1 together as a pair. The counting event for the Timer 1 is when the Timer 0 rolls over from all 1s to all 0s or vice-versa when counting down.  
      |       |             |             | TLR0 and TLR1 are used for lower 32-bit and higher 32-bit respectively. Similarly, TCR0 contains lower 32-bits for the 64-bit counter and TCR1 contains the higher 32-bits.  
      |       |             |             | Only TCSR0 is valid for both the timer/counters in this mode.  
      |       |             |             | This CASC bit must be set before enabling the timer/counter. |
### Table 2-5: Control/Status Register 0 Bit Definitions (Cont’d)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 10  | ENALL  | Read/Write  | 0           | Enable All Timers  
0 = No effect on timers  
1 = Enable all timers (counters run)  
This bit is mirrored in all control/status registers and is used to enable all counters simultaneously. Writing a 1 to this bit sets ENALL, ENT0, and ENT1. Writing a 0 to this register clears ENALL but has no effect on ENT0 and ENT1. |
| 9   | PWMA0  | Read/Write  | 0           | Enable Pulse Width Modulation for Timer 0  
0 = Disable pulse width modulation  
1 = Enable pulse width modulation  
PWM requires using Timer 0 and Timer 1 together as a pair. Timer 0 sets the period of the PWM output, and Timer 1 sets the high time for the PWM output. For PWM mode, MDT0 and MDT1 must be 0 and C_GEN0_ASSERT and C_GEN1_ASSERT must be 1. |
| 8   | T0INT  | Read/Write  | 0           | Timer 0 Interrupt  
Indicates that the condition for an interrupt on this timer has occurred. If the timer mode is capture and the timer is enabled, this bit indicates a capture has occurred. If the mode is generate, this bit indicates the counter has rolled over. Must be cleared by writing a 1.  
*Read:*  
0 = No interrupt has occurred  
1 = Interrupt has occurred  
*Write:*  
0 = No change in state of T0INT  
1 = Clear T0INT (clear to 0) |
| 7   | ENT0   | Read/Write  | 0           | Enable Timer 0  
0 = Disable timer (counter halts)  
1 = Enable timer (counter runs) |
| 6   | ENIT0  | Read/Write  | 0           | Enable Interrupt for Timer 0  
Enables the assertion of the interrupt signal for this timer. Has no effect on the interrupt flag (T0INT) in TCSR0.  
0 = Disable interrupt signal  
1 = Enable interrupt signal |
| 5   | LOAD0  | Read/Write  | 0           | Load Timer 0  
0 = No load  
1 = Loads timer with value in TLR0  
Setting this bit loads timer/counter register (TCR0) with a specified value in the timer/counter load register (TLR0). This bit prevents the running of the timer/counter; hence, this should be cleared alongside setting Enable Timer/Counter (ENT0) bit in TCSR0. |
Load Register (TLR0 and TLR1)

When the counter width has been configured as less than 32 bits, the load register value is right-justified in TLR0 and TLR1. The least-significant counter bit is always mapped to load register bit 0.

In cascade mode, TLR0 has the least significant 32-bits of the generate value and TLR1 should have the most significant bits of the generate value in generate mode. Similarly, in cascade mode TLR0 has the captured value from TCR0 and TLR1 from TCR1.

Figure 2-3 and Table 2-6 show the load register.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 4   | ARHT0  | Read/Write  | 0           | Auto Reload/Hold Timer 0  
When the timer is in Generate mode, this bit determines whether the counter reloads the generate value and continues running or holds at the termination value. In Capture mode, this bit determines whether a new capture trigger overwrites the previous captured value or if the previous value is held.  
0 = Hold counter or capture value. The TLR must be read before providing the external capture.  
1 = Reload generate value or overwrite capture value |
| 3   | CAPT0  | Read/Write  | 0           | Enable External Capture Trigger Timer 0  
0 = Disables external capture trigger  
1 = Enables external capture trigger |
| 2   | GENT0  | Read/Write  | 0           | Enable External Generate Signal Timer 0  
0 = Disables external generate signal  
1 = Enables external generate signal |
| 1   | UDT0   | Read/Write  | 0           | Up/Down Count Timer 0  
0 = Timer functions as up counter  
1 = Timer functions as down counter |
| 0   | MDT0   | Read/Write  | 0           | Timer 0 Mode  
See Timer Modes.  
0 = Timer mode is generate  
1 = Timer mode is capture |
Chapter 2: Product Specification

Timer/Counter Register (TCR0 and TCR1)

When the counter width has been configured as less than 32 bits, the count value is right-justified in TCR0 and TCR1. The least-significant counter bit is always mapped to Timer/Counter Register bit 0. Figure 2-4 and Table 2-7 show the Timer/counter register. In cascade mode, TCR0 has the least significant 32-bits of the 64-bit counter, and TCR1 has the most significant bits.

Table 2-6: Load Register Bit Definitions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>Timer/Counter Load Register</td>
<td>Read/Write</td>
<td>0x0</td>
<td>Timer/Counter Load register</td>
</tr>
</tbody>
</table>

![Timer/Counter Register Image](PG079_c2_04_082412)

**Figure 2-4: Timer/Counter Register**

Table 2-7: Timer/Counter Register Bit Definitions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>Timer/Counter Register</td>
<td>Read</td>
<td>0x0</td>
<td>Timer/Counter register</td>
</tr>
</tbody>
</table>

Control/Status Register 1 (TCSR1)

Figure 2-5 and Table 2-8 show the Control/Status register 1. Control/Status Register 1 contains the control and status bits for timer module 1. This register is used only for loading the TLR1 register in cascade mode.

![Control/Status Register Image](PG079_c2_05_082412)

**Figure 2-5: Control/Status Register 1**
### Table 2-8: Control/Status Register 1 Bit Definitions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 - 11</td>
<td>Reserved</td>
<td>N/A</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>ENALL</td>
<td>Read/Write</td>
<td>0</td>
<td>Enable All Timers&lt;br&gt;0 = No effect on timers&lt;br&gt;1 = Enable all timers (counters run)&lt;br&gt;This bit is mirrored in all control/status registers and is used to enable all counters simultaneously. Writing a 1 to this bit sets ENALL, ENT0, and ENT1. Writing a 0 to this register clears ENALL but has no effect on ENT0 and ENT1.</td>
</tr>
<tr>
<td>9</td>
<td>PWMA0</td>
<td>Read/Write</td>
<td>0</td>
<td>Enable Pulse Width Modulation for Timer1&lt;br&gt;0 = Disable pulse width modulation&lt;br&gt;1 = Enable pulse width modulation&lt;br&gt;PWM requires using Timer 0 and Timer 1 together as a pair. Timer 0 sets the period of the PWM output, and Timer 1 sets the high time for the PWM output. For PWM mode, MDT0 and MDT1 must be 0.</td>
</tr>
<tr>
<td>8</td>
<td>T1INT</td>
<td>Read/Write</td>
<td>0</td>
<td>Timer1 Interrupt&lt;br&gt;Indicates that the condition for an interrupt on this timer has occurred. If the timer mode is capture and the timer is enabled, this bit indicates a capture has occurred. If the mode is generate, this bit indicates the counter has rolled over. Must be cleared by writing a 1.&lt;br&gt;Read:&lt;br&gt;0 = No interrupt has occurred&lt;br&gt;1 = Interrupt has occurred&lt;br&gt;Write:&lt;br&gt;0 = No change in state of T1INT&lt;br&gt;1 = Clear T1INT (clear to 0)</td>
</tr>
<tr>
<td>7</td>
<td>ENT1</td>
<td>Read/Write</td>
<td>0</td>
<td>Enable Timer1&lt;br&gt;0 = Disable timer (counter halts)&lt;br&gt;1 = Enable timer (counter runs)</td>
</tr>
<tr>
<td>6</td>
<td>ENIT1</td>
<td>Read/Write</td>
<td>0</td>
<td>Enable Interrupt for Timer1&lt;br&gt;Enables the assertion of the interrupt signal for this timer. Has no effect on the interrupt flag (T1INT) in TCSR1.&lt;br&gt;0 = Disable interrupt signal&lt;br&gt;1 = Enable interrupt signal</td>
</tr>
<tr>
<td>5</td>
<td>LOAD1</td>
<td>Read/Write</td>
<td>0</td>
<td>Load Timer1&lt;br&gt;0 = No load&lt;br&gt;1 = Loads timer with value in TLR1&lt;br&gt;Setting this bit loads the timer/counter register (TCR1) with a specified value in the timer/counter load register (TLR1). This bit prevents running of timer/counter; hence, this should be cleared alongside setting Enable Timer/Counter (ENT1) bit in TCSR1.</td>
</tr>
</tbody>
</table>
### Table 2-8: Control/Status Register 1 Bit Definitions (Cont’d)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 4   | ARHT1| Read/Write  | 0           | Auto Reload/Hold Timer1  
When the timer is in generate mode, this bit determines whether the counter reloads the generate value and continues running or holds at the termination value. In capture mode, this bit determines whether a new capture trigger overwrites the previous captured value or if the previous value is held until it is read.  
0 = Hold counter or capture value  
1 = Reload generate value or overwrite capture value |
| 3   | CAPT1| Read/Write  | 0           | Enable External Capture Trigger Timer1  
0 = Disables external capture trigger  
1 = Enables external capture trigger |
| 2   | GENT1| Read/Write  | 0           | Enable External Generate Signal Timer1  
0 = Disables external generate signal  
1 = Enables external generate signal |
| 1   | UDT1 | Read/Write  | 0           | Up/Down Count Timer1  
0 = Timer functions as up counter  
1 = Timer functions as down counter |
| 0   | MDT1 | Read/Write  | 0           | Timer1 Mode  
See Timer Modes, page 6.  
0 = Timer mode is generate  
1 = Timer mode is capture |
Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

Clocking

The AXI timer operates on s_axi_aclk.

Resets

The AXI timer resets when s_axi_aresetn is asserted. This is an active low signal and should be synchronous to s_axi_aclk.

Programming Sequence

Generate Mode

The Generate mode has the following programming sequence:

- On start-up, the generate value in the load register is loaded into the counter by setting the Load bit in the TCSR. This applies whether the counter is set up to Auto Reload or Hold when the interval has expired. Setting the Load bit to 1 loads the counter with the value in the load register. For proper operation, the Load bit must be cleared before the counter is enabled or along with setting the enable bit. The timer/counter starts ticking when Enable is set (ENT).

- When the ARHT bit (Auto Reload/Hold) is set to 1 and the counter rolls over from all 1s to all 0s when counting up, or conversely from all 0s to all 1s when counting down, the generate value in the load register is automatically reloaded into the counter and the counter continues to count. If the GenerateOut signal is enabled (bit GENT in the TCSR), an output pulse is generated (one clock period in width). This is useful for generating a repetitive pulse train with a specified period.
• When the Auto Reload/Hold (ARHT) bit is set to 0 and the counter rolls over from all 1s to all 0s when counting up, or conversely, from all 0s to all 1s when counting down, the counter holds at the current value and does not reload the generate value. If the \texttt{GenerateOut} signal is enabled (bit GENT in the TCSR), an output pulse of one clock period in width is generated. This is useful for a one-shot pulse that is to be generated after a specified period of time.

• The counter can be set up to count either up or down as determined by the selection of the UDT bit in the TCSR. If the counter is set up as a down counter, the generate value is the number of clocks in the timing interval. The period of the \texttt{GenerateOut} signal is the generate value (value in load register TLRx) multiplied by the clock period.

• When the counter is set to count down,
\[
\text{TIMING\_INTERVAL} = (\text{TLRx} + 2) \times \text{AXI\_CLOCK\_PERIOD}
\]
• When the counter is set to count up,
\[
\text{TIMING\_INTERVAL} = (\text{MAX\_COUNT} - \text{TLRx} + 2) \times \text{AXI\_CLOCK\_PERIOD}
\]

where \texttt{MAX\_COUNT} is the maximum count value of the counter, such as \texttt{0xFFFFFFFF} for a 32-bit counter.

**Capture Mode**

Capture mode has the following programming sequence:

• The capture signal can be configured to be low-true or high-true.

• When the capture event occurs, the counter value is written to the load register. This value is called the capture value.

• When the ARHT bit is set to 0 and the capture event occurs, the capture value is written to the load register, which holds the capture value until the load register is read. If the load register is not read, subsequent capture events do not update the load register and are lost.

• When the ARHT bit is set to 1 and the capture event occurs, the capture value is always written to the load register. Subsequent capture events update the load register and overwrite the previous value, whether it has been read or not.

• The counter can be set up to count either up or down as determined by the selection of the UDT bit in the TCSR.

**Pulse Width Modulation Mode**

Pulse Width Modulation (PWM) mode has the following programming sequence:

• The mode for both Timer 0 and Timer 1 must be set to Generate mode (bit MDT in the TCSR set to 0).
• The PWMA0 bit in TCSR0 and PWMB0 bit in TCSR1 must be set to 1 to enable PWM mode.

• The GenerateOut signals must be enabled in the TCSR (bit GENT set to 1). The PWM0 signal is generated from the GenerateOut signals of Timer 0 and Timer 1, so these signals must be enabled in both timer/counters.

• The assertion level of the GenerateOut signals for both timers in the pair must be set to Active High.

• The counter can be set to count up or down.

**Setting the PWM Period and Duty Factor**

The PWM period is determined by the generate value in the Timer 0 load register (TLR0). The PWM high time is determined by the generate value in the Timer 1 load register (TLR1). The period and duty factor are calculated as follows:

When counters are configured to count up (UDT = 0):

\[
PWM_{\text{PERIOD}} = (MAX\_\text{COUNT} - TLR0 + 2) \times AXI\_\text{CLOCK\_PERIOD} \\
PWM_{\text{HIGH\_TIME}} = (MAX\_\text{COUNT} - TLR1 + 2) \times AXI\_\text{CLOCK\_PERIOD}
\]

When counters are configured to count down (UDT = 1):

\[
PWM_{\text{PERIOD}} = (TLR0 + 2) \times AXI\_\text{CLOCK\_PERIOD} \\
PWM_{\text{HIGH\_TIME}} = (TLR1 + 2) \times AXI\_\text{CLOCK\_PERIOD}
\]

where \(MAX\_\text{COUNT}\) is the maximum count value for the counter, such as 0xFFFFFFFF for a 32-bit counter.

**Cascade Mode**

In Cascade mode, the two timer/counters are cascaded to operate as a single 64-bit counter/timer. The cascaded counter can work in both generate and capture modes. TCSR) acts as the control and status register for the cascaded counter. TCSR1 is ignored in this mode.

This mode is used when there is a requirement for a timer/counter more than 32 bits wide. Cascaded operation requires using Timer 0 and Timer 1 together as a pair. The counting event for the Timer 1 is when the Timer 0 rolls over from all 1s to all 0s or vice-versa when counting down.

The cascade mode has the following programming sequence:

• Parameter enable_timer2 (Enable Timer 2) should be set to 0 because both timers are required for a cascaded operation.

• The width of the AXI Timer should be 32 because it represents the width of each timer/counter in the core.
• Load Registers of both timer/counters are used (TLR0 and TLR1 - TLR1 for higher 32-bit and TLR0 for lower 32-bit). The value loaded into the load registers is called the generate value in generate mode. And the capture value is captured in these load registers in capture mode.

• Timer/counter 0 control register TCSR0, GenerateOut0, Capture event 0 are valid in this mode. Timer 1 related signals are invalid, that is, TCSR1, GenerateOut1 and CaptureEvent1 are not used. TCSR1 is used only for loading the TLR1 register.

• CASC bit in Timer Control Status Register 0 (TCSR0) must be set for the counters to be in cascade mode. This bit must be set before enabling the timer/counter.

• The sequence of accesses for generate and capture modes are as mentioned in previous sections.

• In generate mode, when the counter is set to count down,

\[
TIMING\_INTERVAL = (TLR + 4) \times AXI\_CLOCK\_PERIOD
\]

where TLR is the concatenated value of TLR1 and TLR0 (TLR = {TLR1, TLR0}).

• In generate mode, when the counter is set to count up,

\[
TIMING\_INTERVAL = (MAX\_COUNT - TLR + 4) \times AXI\_CLOCK\_PERIOD
\]

where MAX_COUNT is the maximum count value of the counter, such as 0xFFFFFFFF for a 64-bit counter, and TLR is the concatenated value of TLR1 and TLR0 (TLR = {TLR1, TLR0}).

The following are the steps for running the 64-bit counter/timer in generate mode:

1. Clear the timer enable bits in control registers (TCSR0 and TCSR1).
2. Write the lower 32-bit timer/counter load register (TLR0).
3. Write the higher 32-bit timer/counter load register (TLR1).
4. Set the CASC bit in Control register TCSR0.
5. Set other mode control bits in control register (TCSR0) as needed.
6. Enable the timer in Control register (TCSR0).

The following are the steps for reading the 64-bit counter/timer:

1. Read the upper 32-bit timer/counter register (TCR1).
2. Read the lower 32-bit timer/counter register (TCR0).
3. Read the upper 32-bit timer/counter register (TCR1) again. If the value is different from the 32-bit upper value read previously, go back to previous step (reading TCR0). Otherwise 64-bit timer counter value is correct.
Interrupts

The programming requirements to enable/disable interrupts and the programming sequence are as follows:

• Interrupt events can only occur when the timer is enabled. In Capture mode, this prevents interrupts from occurring before the timer is enabled.

• The interrupt signal goes high when the interrupt condition is met and the interrupt is enabled in the TCSR. The interrupt is asserted when the interrupt signal is High.

• A single interrupt signal is provided. The interrupt signal is the OR of the interrupts from the two counters. The interrupt service routine must poll the TCSRs to determine the source or sources of the interrupt.

• The interrupt status bit (TINT in the TCSR) can only be cleared by writing a 1 to it. Writing a 0 to it has no effect on the bit. Because the interrupt condition is an edge (the counter rollover or the capture event), it can be cleared at any time and does not indicate an interrupt condition until the next interrupt event.

• In cascade mode, only Timer 0 interrupt events occur. There are no interrupts from Timer 1.
Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 13]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 12]

Customizing and Generating the Core

This chapter includes information about using the Vivado Design Suite to customize and generate the core.

Vivado Integrated Design Environment

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 13] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 12].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.

If you are customizing and generating the core in the Vivado IP Integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 6] for detailed information. IP Integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the
description of the parameter in this chapter. To view the parameter value you can run the `validate_bd_design` command in the tcl console.

**Configuring the Core Parameters**

The Vivado IDE for the AXI Timer for the AXI4-Lite interface is shown in Figure 4-1.

![Figure 4-1: AXI Timer Configuration](image)

The following are the core parameters:

- **Enable 64-bit Mode**: Check this box if an AXI Timer with a 64-bit width is needed. This option disables the Timer 2 option because a 64-bit AXI Timer is created by cascading 2 timers inside the IP core.

- **Width of the Timer/Counter (bits)**: Select 8, 16, 32 bit count widths based on user requirements.

- Timer 1
  - **Capture Trigger**: Select to configure this signal as active-High or active-Low. Default is active-High
• **Generate Out**: Select to configure this signal as active-High or active-Low. Default is active-High.

• **Enable Timer 2**: Select if Timer 2 is needed.

• **Timer 2**
  • **Capture Trigger**: Select to configure this signal as active-High or active-Low. Default is active-High
  • **Generate Out**: Select to configure this signal as active-High or active-Low. Default is active-High.

**Output Generation**

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 13].

---

**Constraining the Core**

Any necessary constraints are delivered when the IP core is generated.

**Required Constraints**

This section is not applicable for this IP core.

**Device, Package, and Speed Grade Selections**

This section is not applicable for this IP core.

**Clock Frequencies**

This section is not applicable for this IP core.

**Clock Management**

This section is not applicable for this IP core.

**Clock Placement**

This section is not applicable for this IP core.

**Banking**

This section is not applicable for this IP core.
Transceiver Placement
This section is not applicable for this IP core.

I/O Standard and Placement
This section is not applicable for this IP core.

Simulation
For details, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7].

Synthesis and Implementation
For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 13].
Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.

Overview

The top module instantiates all components of the core and example design that are needed to implement the design in hardware, as shown in Figure 5-1. This includes the clock generator (MMCME2), register configuration, data generator and data checker modules.

This example design includes two modules:

- **Clock Generator**: MMCME2 is used to generate the clocks for the example design. MMCME2 is used to generate 100 MHz clock for s_axi_aclk. The DUT and other modules of the example design are kept under reset until MMCME2 is locked.
Chapter 5: Example Design

- **AXI Traffic Generator (ATG):** This module (IP) is configured in the System Test mode. All the AXI Timer-related AXI4-Lite transactions are stored in the COE file. For more information on the AXI Traffic Generator, see the AXI Traffic Generator Product Guide (PG125) [Ref 9]. The ATG automatically starts the AXI4-Lite transaction after coming out of reset.

## Implementing the Example Design

After following the steps described in Customizing and Generating the Core, implement the example design as follows:

1. Right-click the core in the Hierarchy window, and select **Open IP Example Design**.
2. A new window pops up, asking you to specify a directory for the example design. Select a new directory, or keep the default directory.
3. A new project is automatically created in the selected directory and opened in a new Vivado window.
4. In the Flow Navigator (left side pane), click **Run Implementation** and follow the directions.

The ATG writes value 0x45 to 0x00 register. This starts the timer counters. When the timer value is reached certain count, the glowLed output drives 1. The glowLed output of the example design is connected to the GPIO_LED 7 of the KC705 evaluation board and indicates the status of the example design. On successful completion of ATG write transactions to the timer registers and counters initiation, the eighth LED is lit. If an error occurs, the GPIO_LED7 is not lit.

## Example Design Directory Structure

In the current project directory, a new project with name "<component_name>_example" is created and the files are delivered to <component_name>_example/<component_name>_example.srcs. This directory and its subdirectories contain all the source files required to create the AXI Timer controller example design.

Table 5-1 lists the files delivered in <component_name>_example/<component_name>_example.srcs/sources_1/imports/example_design.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;component_name&gt;_exdes.vhd</td>
<td>Top-level HDL file for the example design.</td>
</tr>
<tr>
<td>clock_gen.vhd</td>
<td>Clock generation module for example design.</td>
</tr>
</tbody>
</table>
Chapter 5: Example Design

Table 5-1: Example Design Directory (Cont’d)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>atg_addr.coe</td>
<td>COE file of address. This file contains the AXI Timer register address.</td>
</tr>
<tr>
<td>atg_data.coe</td>
<td>COE file of data. This file contains the data to be written/read from the AXI Timer registers.</td>
</tr>
<tr>
<td>atg_mask.coe</td>
<td>COE file to mask certain reads.</td>
</tr>
<tr>
<td>atg_ctrl.coe</td>
<td>COE file that contains control information of ATG.</td>
</tr>
</tbody>
</table>

The `<component_name>_example/<component_name>_example.srcs/sources_1/sim_1/imports/simulation directory contains the test bench file for the example design: `<component_name>_exdes_tb.vhd."

The `<component_name>_example/<component_name>_example.srcs/sources_1/constrs_1/imports/example_design directory contains the top-level constraints file for the example design: `<component_name>_exdes.xdc."

The XDC has all the necessary constraints needed to run the example design on the KC705 board. All the IO constraints are commented in the XDC file. Uncomment these constraints before implementing the design on the KC705 board.

Simulating the Example Design

The AXI Timer example design quickly simulates and demonstrates the behavior of the AXI Timer.

Simulation Results

The simulation script compiles the AXI Timer example design, and supporting simulation files. It then runs the simulation and checks to ensure that it completed successfully.

If test passes, the following message is displayed:

Test Completed Successfully

If the test fails or does not end, the following message is displayed:

Test Failed !! Test Timed Out
Chapter 6

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite. Figure 6-1 shows the test bench for AXI Timer example design. The top-level test bench generates a 200 MHz clock and drives the initial reset to the example design.

Figure 6-1: Test Bench Block Diagram
This appendix includes details about resources available on the Xilinx® Support website and debugging tools.

---

**Finding Help on Xilinx.com**

To help in the design and debug process when using the AXI Timer, the Xilinx Support web page (Xilinx Support web page) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

**Documentation**

This product guide is the main document associated with the AXI Timer. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

**Answer Records**

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core are listed below, and can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool messages
- Summary of the issue encountered
A filter search is available after results are returned to further target the results.

**Master Answer Record for the AXI Timer**

AR: 54438

**Technical Support**

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

1. Open a WebCase by clicking the Open a WebCase link located under Additional Resources.
   - Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which files to include with the WebCase.

**Debug Tools**

There is one tool available to address AXI Timer design issues.

**Vivado Design Suite Debug Feature**

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See Vivado Design Suite User Guide: Programming and Debugging (UG908) [Ref 11].
Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado Lab Tools is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the Vivado Lab Tools for debugging the specific problems.

Many of these common issues can also be applied to debugging design simulations. Details are provided on:

- General Checks

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the locked port.
- If your outputs go to 0, check your licensing.

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. Output s_axi_arready asserts when the read address is valid, and output s_axi_rvalid asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The s_axi_aclk input is connected and toggling.
- The interface is not being held in reset, and s_axi_arresetn is an active-Low reset.
- If the simulation has been run, verify in simulation to capture that the waveform is correct for accessing the AXI4-Lite interface.
Migrating and Upgrading

This appendix contains information about migrating a design from the ISE® Design Suite to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information about migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 10].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

Parameter Changes

There are no parameter changes.

Port Changes

There are no port changes.
Appendix C

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx® Support.

References

These documents provide supplemental material useful with this product guide:

1. ARM® AMBA® AXI and ACE Specification (ARM IHI 0022D)
2. AXI Reference Guide (UG761)
3. LogiCORE™ IP AXI Interconnect Product Guide (PG059)
4. LogiCORE IP AXI Lite IPIF Product Guide (PG155)
5. LogiCORE IP AXI4-Lite IPIF Data Sheet (DS765)
9. AXI Traffic Generator Product Guide (PG125)
10. ISE® to Vivado Design Suite Migration Methodology Guide (UG911)
Revision History

The following table shows the revision history for this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
</table>
| 10/05/2016 | 2.0     | • Added a note in Register Space section under Chapter 2, Product Specification.  
|            |         | • Updated Automotive Applications Disclaimer in Please Read: Important Legal Notices section. |
| 11/18/2015 | 2.0     | Added support for UltraScale+ families.                                    |
| 04/02/2014 | 2.0     | • General document updates.                                               |
| 12/18/2013 | 2.0     | • Added support for UltraScale™ architecture.                              |
| 10/02/2013 | 2.0     | • Added Chapter 8, Example Design and Chapter 9, Test Bench.              |
|            |         | • Added IP Integrator support.                                             |
|            |         | • Changed all signals/ports to lower case.                                |
| 03/20/2013 | 2.0     | • Updated for core v2.0, and for Vivado Design Suite-only support.         |
|            |         | • Added list of modules, and updated the I/O signal descriptions, and register overview. |
|            |         | • Major revisions made to the Debugging Appendix.                         |
| 10/16/2012 | 1.0     | Initial Xilinx release as a product guide. Based on DS764, LogiCORE IP AXI Timer. Updated core to v1.03a. |
Please Read: Important Legal Notices

The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available “AS IS” and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx’s limited warranty, please refer to Xilinx’s Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx’s Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS “XA” IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE (“SAFETY APPLICATION”) UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD (“SAFETY DESIGN”). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

© Copyright 2012-2016 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.