AXI UART 16550 v2.0

LogiCORE IP Product Guide

Vivado Design Suite

PG143 October 5, 2016

XILINX
ALL PROGRAMMABLE™

Table of Contents

Introduction

The LogiCORE™ IP AXI Universal Asynchronous Receiver Transmitter (UART) 16550 connects to the Advance Microcontroller Bus Architecture (AMBA®) AXI and provides the controller interface for asynchronous serial data transfer. This soft IP core is designed to connect through an AXI4-Lite interface.

The AXI UART 16550 detailed in this document incorporates features described in the *PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs Data Sheet* [Ref 1].

The PC16550D data sheet is referenced throughout this document and should be used as the authoritative specification. Differences between the PC16550D and the AXI UART 16550 product guide are highlighted in Interrupts in Chapter 2.

Features

- AXI4-Lite interface for register access and data transfers
- Hardware and software register compatible with all standard 16450 and 16550 UARTs
- Supports default core configuration for 9600 baud, 8 bits data length, 1 stop bit and no parity
- Implements all standard serial interface protocols
  - 5, 6, 7 or 8 bits per character
  - Odd, Even or no parity detection and generation
  - 1, 1.5 or 2 stop bit detection and generation
  - Internal baud rate generator and separate receiver clock input
  - Modem control functions
  - Prioritized transmit, receive, line status and modem control interrupts
  - False start bit detection and recover
  - Line break detection and generation
  - Internal loopback diagnostic functionality
  - 16 character transmit and receive FIFOs

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™ UltraScale™ Zynq®-7000, 7 Series
Supported User Interfaces	AXI4-Lite
Resources	See Table 2-2.
Provided with Core	
Design Files	VHDL
Example Design	VHDL
Test Bench	VHDL
Constraints File	Xilinx Design Constraints (XDC) File
Simulation Model	Not Provided
Supported S/W Driver(2)	Standalone and Linux
Tested Design Flows(3)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
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 software development kit (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

The AXI UART 16550 IP core implements the hardware and software functionality of the PC16550D UART, which works in both the 16450 and 16550 UART modes. For complete details, see the *PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs data sheet* [Ref 1].

The AXI UART 16550 core performs parallel-to-serial conversion on characters received from the AXI master and serial-to-parallel conversion on characters received from a modem or serial peripheral. The AXI UART 16550 is capable of transmitting and receiving 8, 7, 6, or 5-bit characters, with 2, 1.5 or 1 stop bits and odd, even or no parity. The AXI UART 16550 can transmit and receive independently.

The AXI UART 16550 core has internal registers to monitor its status in the configured state. The core can signal receiver, transmitter, and modem control interrupts. These interrupts can be masked and prioritized, and they can be identified by reading an internal register. The core contains a 16-bit, programmable, baud-rate generator, and independent, 16-character-length transmit and receive FIFOs. The FIFOs can be enabled or disabled through software.

The top-level block diagram for the AXI UART 16550 core is shown in Figure 1-1.

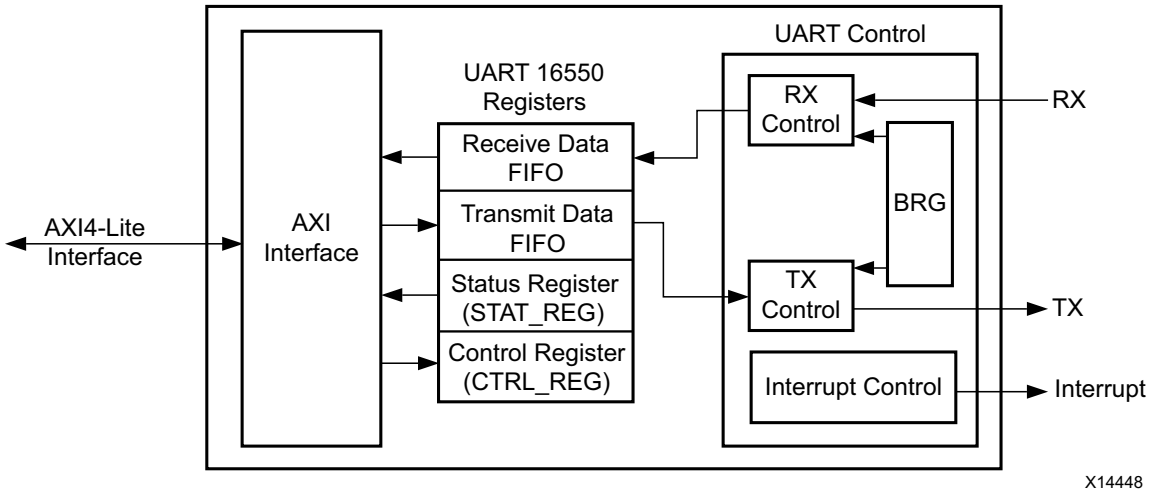

Figure 1-1: **UART 16550 Block Diagram**

The AXI UART 16550 modules are described in these sections:

- **AXI Interface**: This block implements the AXI4-Lite slave interface for register access and data transfer.
- **UART Control**: This block consists of the following.
  - RX Control – This block samples received data with respect to generated baud rate and writes it to Receive Data FIFO.
  - TX Control – This block reads data from Transmit Data FIFO and sends it out on UART TX interface.
  - BRG (Baud Rate Generator) – This block generates various baud rates that are user programmed.
  - Interrupt Control

    The AXI UART 16550 core provides separate interrupt enable and interrupt identification registers. If interrupts are enabled, a level-sensitive interrupt is generated for the following conditions. See Interrupts in Chapter 2 for more details.
    - Receiver line status
    - Received data available
    - Character timeout
    - Transmitter holding register empty
    - Modem status

Feature Summary

- AXI4-Lite interface for register access and data transfers
- Hardware and software register compatible with all standard 16450 and 16550 UARTs
- Supports default core configuration for 9600 baud, 8 bits data length, 1 stop bit and no parity
- Implements all standard serial interface protocols
  - 5, 6, 7 or 8 bits per character
  - Odd, Even or no parity detection and generation
  - 1, 1.5 or 2 stop bit detection and generation
  - Internal baud rate generator and separate receiver clock input
  - Modem control functions
  - Prioritized transmit, receive, line status and modem control interrupts

- False start bit detection and recover
- Line break detection and generation
- Internal loopback diagnostic functionality
- 16 character transmit and receive FIFOs

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.

Product Specification

Performance

The AXI UART 16550 is characterized as per the benchmarking methodology described in Appendix A, IP Characterization and F_{MAX} Margin System Methodology, *Vivado Design Suite User Guide: Designing With IP* (UG896) [Ref 2]. Table 2-1 shows the results of the characterization runs.

Note: Frequency data for UltraScale™ and Zynq®-7000 devices are expected to be similar to 7 series device numbers.

Table 2-1: **Maximum Frequencies**

Family	Speed Grade	Fmax (MHz) for AXI4-Lite
Virtex-7	-1	180
Kintex-7		180
Artix-7		120
Virtex-7	-2	200
Kintex-7		200
Artix-7		140
Virtex-7	-3	220
Kintex-7		220
Artix-7		160

Resource Utilization

The AXI UART 16550 resource utilization for various parameter combinations measured with three architectures is shown in Table 2-2.

Table 2-2: **Resource Estimations for 7 Series, Zynq, and UltraScale Devices**

	UART Mode	Use External Clock for Baud Rate	Enable External Receiver Clock	Slices/CLB	Registers	LUTs
UltraScale	16550	1	1	68	318	342
	16550	0	0	65	308	345
	16450	0	0	52	259	262
7 Series and Zynq	16550	1	1	120	318	352
	16550	0	0	118	308	347
	16450	0	0	82	259	252

Port Descriptions

The I/O signals are listed and described in Table 2-3.

Table 2-3: **I/O Signals**

Signal Name	Interface	Signal Type	Initial State	Description
System Signals				
s_axi_aclk	System	I	-	AXI Clock
s_axi_aresetn	System	I	-	AXI Reset signal, active-Low
ip2intc_irpt	System	O	0	Device interrupt output to microprocessor interrupt input or system interrupt controller (active-High)
freeze	System	I	-	This is used to freeze the UART. When pulled High, the interrupts are disabled and the internal state machine goes into idle state.
s_axi_*	S_AXI	-	-	See Appendix A of the *Vivado AXI Reference Guide* (UG1037) [Ref 3] for description of AXI4 Signals.
UART Interface Signals				
baudoutn	Serial	O	1	16 x clock signal from the transmitter section of the UART
rclk	Serial	I	-	Receiver 16x clock (Optional, can be driven externally under control of the **Enable External Receiver CLK** parameter)
sin	Serial	I	-	Serial data input
sout	Serial	O	1	Serial data output
xin	Serial	I	-	Baud rate generator reference clock (Optional, can be driven externally under control of the **Use External CLK for BAUD Rate** parameter)
xout	Serial	O	0	If **Use External CLK for BAUD Rate** = 0, Xout is 0, if **Use External CLK for BAUD Rate** = 1 Xout can be used as reference feedback clock for Baud rate generator
ctsn	Modem	I	-	Clear to send (active-Low). When Low, this indicates that the MODEM or data set is ready to exchange data.
dcdn	Modem	I	-	Data carrier detect (active-Low). When Low, indicates that the data carrier has been detected by the MODEM or data set.
dsrn	Modem	I	-	Data set ready (active-Low). When Low, this indicates that the MODEM or data set is ready to establish the communication link with the UART.

Table 2-3: **I/O Signals *(Cont'd)***

Signal Name	Interface	Signal Type	Initial State	Description
dtrn	Modem	O	1	Data terminal ready (active-Low). When Low, this informs the MODEM or data set that the UART is ready to establish a communication link.
rin	Modem	I	-	Ring indicator (active-Low). When Low, this indicates that a telephone ringing signal has been received by the MODEM or data set.
rtsn	Modem	O	1	Request to send (active-Low). When Low, this informs the MODEM or data set that the UART is ready to exchange data.
ddis	User	O	1	Driver disable. This goes Low when CPU is reading data from UART.
out1n	User	O	1	User controlled output
our2n	User	O	1	User controlled output
rxrdyn	User	O	1	DMA control signal
txrdyn	User	O	0	DMA control signal

Register Space

Some of the internal registers are accessible only when bit 7 of the Line Control register (LCR) is set. The AXI UART 16550 internal register set is described in Table 2-4.

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 a Write, both the AXI Write Address Valid (*_awvalid) and AXI Write Data Valid (*_wvalid) signals should be asserted together.

Table 2-4: **Register Address Map**

LCR(7)	Address Offset	Register Name	Access Type	Description
0	0x1000	RBR	RO	Receiver Buffer Register
0	0x1000	THR	WO	Transmitter Holding Register
0	0x1004	IER	R/W	Interrupt Enable Register
x	0x1008	IIR	RO	Interrupt Identification Register
x	0x1008	FCR	WO	FIFO Control Register
1	0x1008	FCR	RO	FIFO Control Register
x	0x100C	LCR	R/W	Line Control Register
x	0x1010	MCR	R/W	Modem Control Register
x	0x1014	LSR	R/W	Line Status Register
x	0x1018	MSR	R/W	Modem Status Register
x	0x101C	SCR	R/W	Scratch Register
1	0x1000	DLL	R/W	Divisor Latch (Least Significant Byte) Register
1	0x1004	DLM	R/W	Divisor Latch (Most Significant Byte) Register

Receiver Buffer Register

This 32-bit read register is shown in Figure 2-1. The Receiver Buffer register contains the last received character. The bit definitions for the register are shown in Table 2-5. The offset and accessibility of this register value is as shown in Table 2-4.

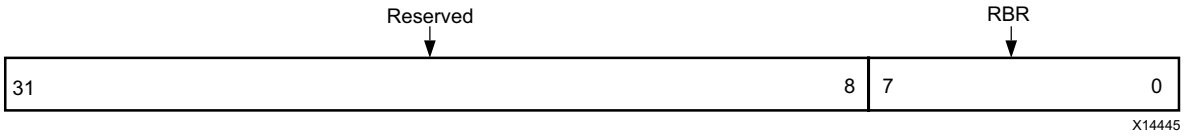

Figure 2-1: **Receiver Buffer Register (RBR)**

Table 2-5: **Receiver Buffer Register Bit Definitions**

Bits	Field Name	Access Type	Reset Value	Description
31-8	Reserved	N/A	N/A	Reserved
7-0	RBR	RO	0x0	Last received character

Transmitter Holding Register

This 32-bit write register is shown in Figure 2-2. The Transmitter Holding register contains the character to be transmitted next. The bit definitions for the register are shown in Table 2-6. The offset and accessibility of this register is shown in Table 2-4.

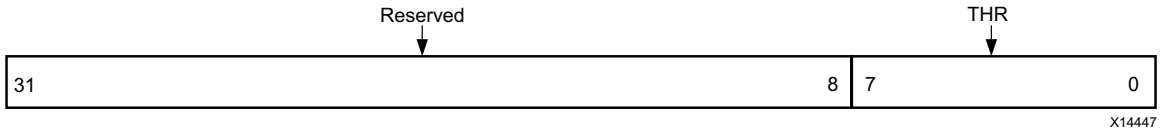

Figure 2-2: **Transmitter Holding Register (THR)**

Table 2-6: **Transmitter Holding Register Bit Definitions**

Bits	Name	Access	Reset Value	Description
31-8	Reserved	N/A	N/A	Reserved
7-0	THR	WO	0xFF	Holds the character to be transmitted next

Interrupt Enable Register

This 32-bit read/write register is shown in Figure 2-3. The Interrupt Enable register contains the bits which enable interrupts. The bit definitions for the register are shown in Table 2-7. The offset and accessibility of this register value is shown in Table 2-4.

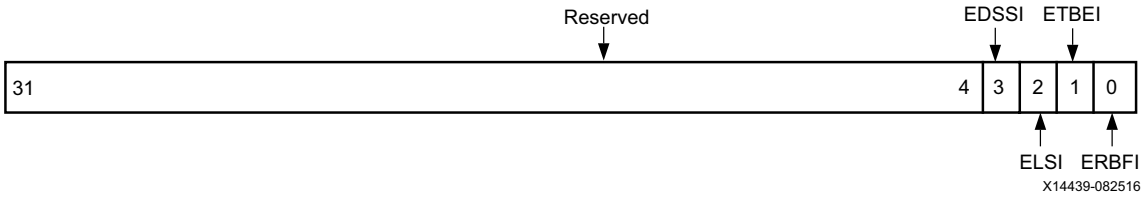

Figure 2-3: **Interrupt Enable Register (IER)**

Table 2-7: **Interrupt Enable Register Bit Definitions**

Bits	Name	Access	Reset Value	Description
31-4	Reserved	N/A	N/A	Reserved
3	EDSSI	R/W	0x0	Enable Modem Status Interrupt. 0 = Disables Modem Status Interrupts. 1 = Enables Modem Status Interrupts.

Table 2-7: **Interrupt Enable Register Bit Definitions *(Cont'd)***

Bits	Name	Access	Reset Value	Description
2	ELSI	R/W	0x0	Enable Receiver Line Status Interrupt. 0 = Disables Receiver Line Status Interrupts. 1 = Enables Receiver Line Status Interrupts.
1	ETBEI	R/W	0x0	Enable Transmitter Holding Register Empty Interrupt. 0 = Disables Transmitter Holding Register Empty Interrupts. 1 = Enables Transmitter Holding Register Interrupts.
0	ERBFI	R/W	0x0	Enable Received Data Available Interrupt. 0 = Disables Received Data Available Interrupts. 1 = Enables Received Data Available Interrupts.

Interrupt Identification Register

This 32-bit read register is shown in Figure 2-4. The Interrupt Identification register contains the priority interrupt identification. The bit definitions for the register are shown in Table 2-8. The offset and accessibility of this register value is shown in Table 2-4.

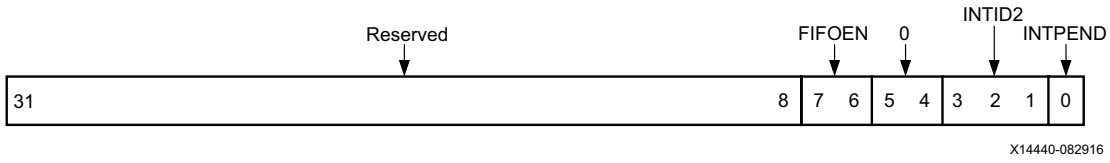

Figure 2-4: **Interrupt Identification Register (IIR)**

Table 2-8: **Interrupt Identification Register Bit Definitions**

Bits	Name	Access	Reset Value	Description
31-8[2]	Reserved	N/A	N/A	Reserved
7-6	FIFOEN[1]	RO	0x0	FIFOs Enabled. Always zero if not in FIFO mode. 0 - 16450 mode 1 - 16550 mode
5-4	Reserved	RO	0	Reserved

Table 2-8: **Interrupt Identification Register Bit Definitions *(Cont'd)***

Bits	Name	Access	Reset Value	Description
3-1	INTID2	RO	0x0	Interrupt ID. 011 = Receiver Line Status (Highest).(4) 010 = Received Data Available (Second). 110 = Character Timeout (Second). 001 = Transmitter Holding Register Empty (Third). 000 = Modem Status (Fourth).
0	INTPEND(3)	RO	0x1	0 - Interrupt is pending 1 - No interrupt is pending

Notes:

1. Bits are always zero in 16450 UART mode.
2. Reading these bits always return 00
3. If INTPEND = 0, interrupt is pending. See the PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs data sheet [Ref 1] for more details.
4. Line status interrupt is generated for framing, parity, overrun error and break condition.

FIFO Control Register

This is 32-bit write/read register is shown in Figure 2-5. The FIFO Control register contains the FIFO configuration bits. The bit definitions for the register are shown in Table 2-9. The offset and accessibility of this register value is shown in Table 2-4. The DMA mode signaling information is given in Table 2-10.

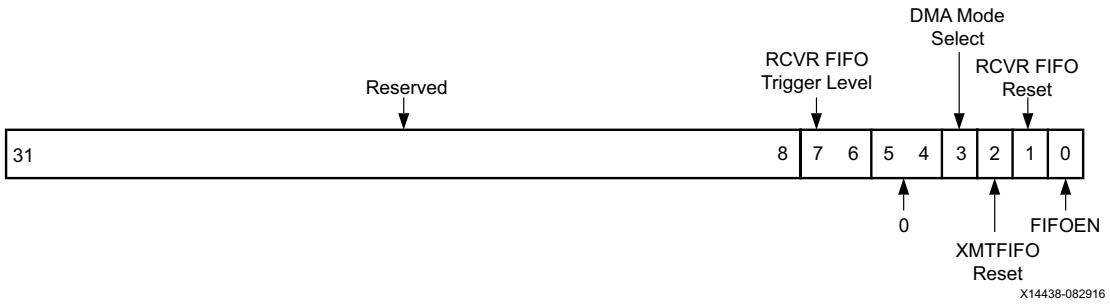

Figure 2-5: **FIFO Control Register (FCR)**

Table 2-9: **FIFO Control Register Bit Definitions** (1)

Bits	Name	Access	Reset Value	Description
31-8	Reserved	N/A	N/A	Reserved
7-6	RCVR FIFO Trigger Level	R/W	0x0	RCVR FIFO Trigger Level. 00 = 1 byte. 01 = 4 bytes. 10 = 8 bytes. 11 = 14 bytes.
5-4	Reserved	RO	0	Reserved
3	DMA Mode Select	R/W	0x0	DMA Mode Select. 0 = Mode 0. 1 = Mode 1.
2	XMIT FIFO Reset	R/W	0x0	Transmitter FIFO Reset. 1 = Resets XMIT FIFO.
1	RCVR FIFO Reset	R/W	0x0	Receiver FIFO Reset. 1 = Resets RCVR FIFO.
0	FIFOEN	R/W	0x0	FIFO Enable. 1 = Enables FIFOs. 0 = Disables FIFOs

Notes:
1. FCR is not included in 16450 UART mode.

Table 2-10: **DMA Modes Signaling** (1)

DMA Mode	txrdyn	RXRDYn
Mode 0	In the 16450 mode or in mode 0, when there are no characters in the THR or Transmitter FIFO, this signal is Low. This signal goes High again after the first character is loaded into the THR or FIFO.	In the 16450 mode or in mode 0, when there is at least one character in the Receiver FIFO or Receiver holding register, this signal is Low. This signal goes High again when there are no characters in FIFO or receiver holding register.
Mode 1	When there are no characters in the Transmitter FIFO, this signal goes Low. This signal goes High again if the FIFO is completely full.	When the trigger level or the timeout has been reached, this signal goes Low. This signal goes High again when there are no characters in the FIFO or receiver holding register.

Notes:
1. See Table 2-9, bit 3.

Line Control Register

This 32-bit write/read register is shown in Figure 2-6. The Line Control register contains the serial communication configuration bits. The bit definitions for the register are shown in Table 2-11. The offset and accessibility of this register value is shown in Table 2-4.

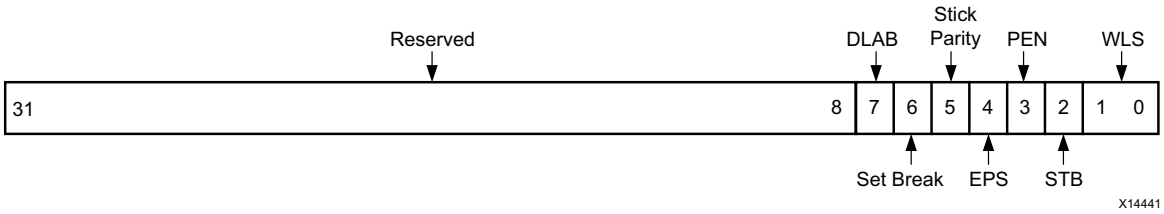

Figure 2-6: **Line Control Register (LCR)**

Table 2-11: **Line Control Register Bit Definitions**

Bits	Name	Access	Reset Value	Description
31-8	Reserved	N/A	N/A	Reserved
7	DLAB	R/W	0x0	Divisor Latch Access Bit. 1 = Allows access to the Divisor Latch Registers and reading of the FIFO Control Register. 0 = Allows access to RBR, THR, IER and IIR registers.
6	Set Break	R/W	0x0	Set Break. 1 = Enables break condition. Sets SOUT to 0 and cause break condition. 0 = Disables break condition.
5	Stick Parity	R/W	0x0	Stick Parity. 1 = When bits 3, 4 are logic 1 the Parity bit is transmitted and checked as a logic 0. If bit 4 is a logic 0 and bit 3 is logic 1 then the Parity bit is transmitted and checked as a logic 1. 0 = Stick Parity is disabled.
4	EPS	R/W	0x0	Even Parity Select. 1 = Selects Even parity. 0 = Selects Odd parity.
3	PEN	R/W	0x0	Parity Enable. 1 = Enables parity. 0 = Disables parity.

Table 2-11: **Line Control Register Bit Definitions *(Cont'd)***

Bits	Name	Access	Reset Value	Description
2	STB	R/W	0x0	Number of Stop Bits. 0 = 1 Stop bit. 1 = 2 Stop bits or 1.5, if 5 bits/character selected. The receiver checks for 1 stop bit only regardless of the number of stop bits selected.
1-0	WLS	R/W	0x0	Word Length Select. 00 = 5 bits/character. 01 = 6 bits/character. 10 = 7 bits/character. 11 = 8 bits/character.

Modem Control Register

This 32-bit write/read register is shown in Figure 2-7. The Modem Control register contains the modem signaling configuration bits. The bit definitions for the register are shown in Table 2-12. The offset and accessibility of this register value is shown in Table 2-4.

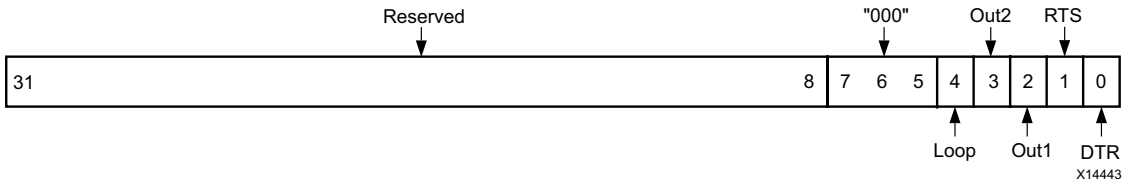

Figure 2-7: **Modem Control Register (MCR)**

Table 2-12: **Modem Control Register Bit Definitions**

Bits	Name	Access	Reset Value	Description
31-8	Reserved	N/A	N/A	Reserved
7-5	N/A	RO	0x0(1)	Always 000
4	Loop	R/W	0x0	Loop Back. 1 = Enables loopback. 0 = Disables loopback.
3	Out2	R/W	0x0	User Output 2. 1 = Drives OUT2N Low. 0 = Drives OUT2N High.
2	Out1	R/W	0x0	User Output 1. 1 = Drives OUT1N Low. 0 = Drives OUT1N High.
1	RTS	R/W	0x0	Request To Send. 1 = Drives RTSN Low. 0 = Drives RTSN High.
0	DTR	R/W	0x0	Data Terminal Ready. 1 = Drives DTRN Low. 0 = Drives DTRN High.

Notes:

1. Reading these bits always returns 000.

Line Status Register

This 32-bit write/read register as shown in Figure 2-8. The Line Status register contains the current status of receiver and transmitter. The bit definitions for the register are shown in Table 2-13. The offset and accessibility of this register value is shown in Table 2-4.

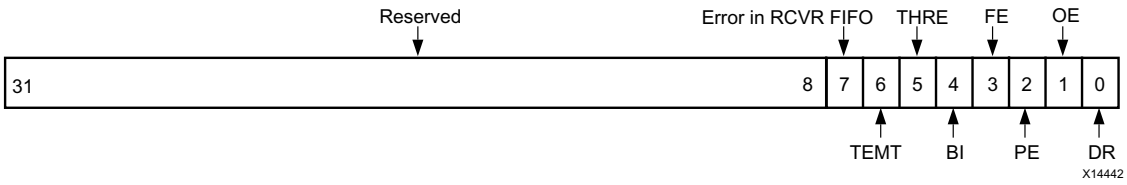

Figure 2-8: **Line Status Register (LSR)**

Table 2-13: **Line Status Register Bit Definitions**

Bits	Name	Access	Reset Value	Description
31-8	Reserved	N/A	N/A	Reserved
7	Error in RCVR FIFO	R/W	0x0	Error in RCVR FIFO(1): RCVR FIFO contains at least one receiver error (Parity, Framing, Break condition).
6	TEMT	R/W	0x1	Transmitter Empty: 0 - THR or Transmitter shift register contains data. 1 - THR and Transmitter shift register empty. In FIFO mode, Transmitter FIFO and shift register are both empty.
5	THRE	R/W	0x1	Transmitter Holding Register Empty. 0 - THR or Transmitter FIFO has data to transmit. 1 - THR is empty. In FIFO mode, Transmitter FIFO is empty.
4	BI	R/W	0x0	Break Interrupt. Set when SIN is held Low for an entire character time. (Start + data bits + Parity + Stop bits). In FIFO mode, this error is associated with a particular character in FIFO. The next character transfer is enabled if the Sin goes to marking state and receives the next valid start bit.
3	FE	R/W	0x0	Framing Error. Character missing a stop bit. In framing error, the UART attempts to re-synchronize by assuming that the framing error was due to next character start bit, so it samples start bit twice and then takes in following data. In FIFO mode, this error is associated with a particular character in the FIFO.

Table 2-13: **Line Status Register Bit Definitions *(Cont'd)***

Bits	Name	Access	Reset Value	Description
2	PE	R/W	0x0	Parity Error. Indicates that the received data character does not have correct even or odd parity as selected by the Even parity select bit. In FIFO mode, this error is associated with a particular character in the FIFO.
1	OE	R/W	0x0	Overrun Error. RBR not read before next character is received, thereby destroying the previous character. In FIFO mode, data continues to fill the FIFO beyond the trigger level, an overrun error occurs only after the FIFO is full and the next character has been completely received in the shift register. The character in the shift register is overwritten but it is not transferred to the FIFO.
0	DR	R/W	0x0	Data Ready. 0 - All the data in RBR or FIFO is read. 1 - Complete incoming character has been received and transferred into the RBR of FIFO.

Notes:
1. The error is reported until the last character containing an error in the FIFO is read out of the FIFO.

Modem Status Register

This 32-bit write/read register is shown in Figure 2-9. The Modem Status register contains the current state of the Modem Interface. The bit definitions for the register are shown in Table 2-14. The offset and accessibility of this register value is shown in Table 2-4.

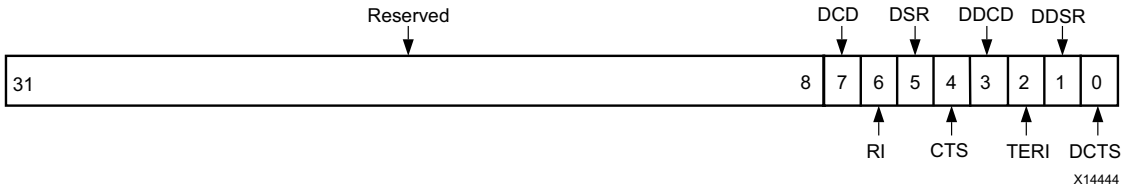

Figure 2-9: **Modem Status Register (MSR)**

Table 2-14: **Modem Status Register Bit Definitions**

Bits	Name	Access	Reset Value	Description
31-8	Reserved	N/A	N/A	Reserved
7	DCD	R/W	'X'(1)	Data Carrier Detect. Complement of DCDN input.
6	RI	R/W	'X'(1)	Ring Indicator. Complement of RIN input.
5	DSR	R/W	'X'(1)	Data Set Ready. Complement of DSRN input.
4	CTS	R/W	'X'(1)	Clear To Send. Complement of CTSN input.
3	DDCD	R/W	0x0	Delta Data Carrier Detect. Change in DCDN after last MSR read.
2	TERI	R/W	0x0	Trailing Edge Ring Indicator. RIN has changed from a Low to a High.
1	DDSR	R/W	0x0	Delta Data Set Ready. Change in DSRN after last MSR read.
0	DCTS	R/W	0x0	Delta Clear To Send. Change in CTSN after last MSR read.

Notes:
1. X represents the value of the driven input.

Scratch Register

This 32-bit write/read register is shown in Figure 2-10. The Scratch register can be used to hold user data. The bit definitions for the register are shown in Table 2-15. The offset and accessibility of this register value is shown in Table 2-4.

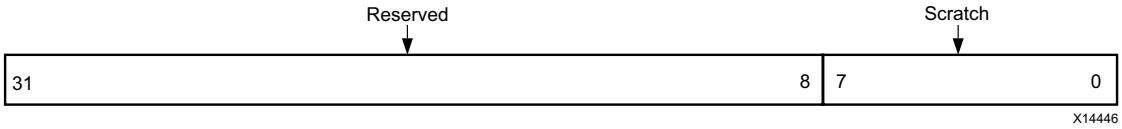

Figure 2-10: **Scratch Register (SCR)**

Table 2-15: **Scratch Register Bit Definitions**

Bits	Name	Access	Reset Value	Description
31-8	Reserved	N/A	N/A	Reserved.
7-0	Scratch	R/W	0x00	Hold user data.

Divisor Latch (Least Significant Byte) Register

This 32-bit write/read register is shown in Figure 2-11. The Divisor Latch (Least Significant Byte) register holds the least significant byte of the Baud rate generator counter. The bit definitions for the register are shown in Table 2-16. The offset and accessibility of this register value is shown in Table 2-4.

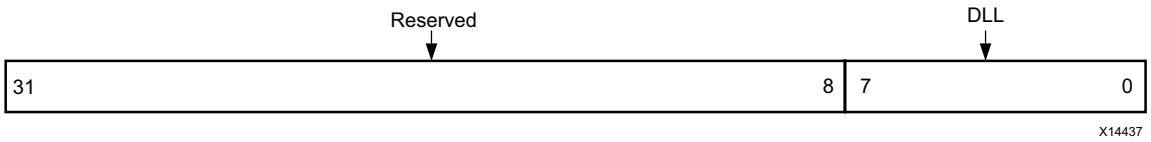

Figure 2-11: **Divisor Latch (Least Significant Byte) Register**

Table 2-16: **Divisor Latch (Least Significant Byte) Register Bit Definitions**

Bits	Name	Access	Reset Value	Description
31-8	Reserved	N/A	N/A	Reserved.
7-0	DLL	R/W	See Note(1)	Divisor Latch Least Significant Byte.

Notes:

1. On reset, the DLL gets configured for 9600 Baud. The DLL reset value, [LSB(divisor)] is calculated from the formula, divisor = AXI CLK frequency/(16 x 9600)).

Divisor Latch (Most Significant Byte) Register

This 32-bit write/read register is shown in Figure 2-12. The Divisor Latch (Most Significant Byte) register holds the most significant byte of the baud rate generator counter. The bit definitions for the register are shown in Table 2-17. The offset and accessibility of this register value is shown in Table 2-4.

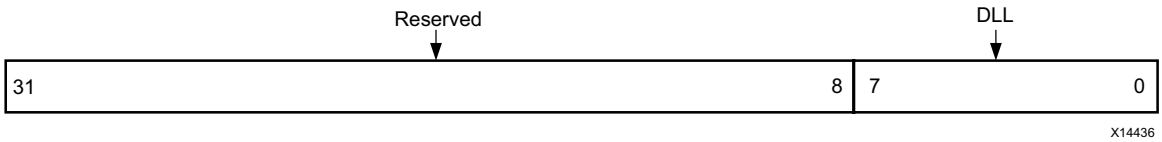

Figure 2-12: **Divisor Latch (Most Significant Byte) Register**

Table 2-17: **Divisor (Most Significant Byte) Register Bit Definitions**

Bits	Name	Access	Reset Value	Description
31-8	Reserved	N/A	N/A	Reserved
7-0	DLM	R/W	unknown(1)	Divisor Latch Most Significant Byte

Notes:

1. On reset, the DLM gets configured for 9600 Baud. The DLM reset value, [MSB(divisor)] is calculated from the formula, divisor = (AXI CLK frequency/(16 x 9600)).

Interrupts

The AXI UART 16550 core provides interrupts for the following conditions.

- Receiver line status
- Received data available
- Character timeout
- Transmitter holding register empty
- Modem status

Receiver Line Status

- Overrun error — An interrupt is generated when the Receive buffer is not read by the master and the next character is transferred to the Receive buffer register. In FIFO mode, an overrun interrupt is generated only when the FIFO is full and the next character is completely received in the shift register.
- Parity error — This interrupt is generated when the receive character has an invalid parity bit.
- Framing error — This interrupt is generated if the received character has an invalid stop bit. These interrupts are cleared when the register is read.

Received Data Available

The Received Data available interrupt is generated when the Receiver FIFO trigger level is reached. This interrupt is cleared when the Receiver FIFO drops below the trigger level.

Character Timeout

The Character Timeout interrupt is generated when no character has been removed from, or input to, the receiver FIFO during the last four character times and there is at least one character in the FIFO during this time. The character time considered for timeout (Start + 8 data bits + 1 Parity bit + 2 Stop bits) is constant for all configurations. This interrupt is cleared by reading the Receiver Buffer register.

Transmitter Holding Register Empty

The Transmitter Holding register empty interrupt is generated when the character is transferred from the Transmitter holding register to the Transmitter shift register. In FIFO mode, this interrupt is generated when the Transmitter FIFO becomes empty. This interrupt is cleared by reading the Interrupt identification register (IIR) or writing into the Transmitter holding register.

Modem Status

This interrupt is generated for these modem status conditions:

- Clear to Send
- Data Set Ready
- Ring Indicator
- Data Carrier Detect

This interrupt is cleared by reading the Modem Status register.

Designing with the Core

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

Clocking

System Clock

The asynchronous microprocessor interface of the PC16550D is synchronized to the system clock input of the UART.

XIN Clock

If the `xin` input is driven externally, the xin clock must be less than or equal to half of the system clock (that is, xin ≤ (S_AXI_ACLK/2)). This is mandatory for the proper functioning of the core.

Resets

The AXI UART 16550 core works on the `s_axi_aresetn` signal, which is active-Low.

Programming Sequence

Follow these general steps to use the AXI UART 16550 core in 16550 mode:

1. Specify the format of the asynchronous data communications exchange. For example: data bits (5, 6, 7 or 8), set parity ON and select even or odd parity, set the number of stop bits for the transmission, and set the Divisor latch access bit by programming the Line Control Register.
2. Write to the Interrupt Enable register to activate the individual interrupts.

3. Write to the FIFO Control register to enable the FIFOs, clear the FIFOs, and set the RCVR FIFO trigger level.
4. Write to the Divisor Latch, least significant byte first followed by most significant byte, for proper setting of the UART Baud rate.
5. Service the interrupts whenever an interrupt is triggered by the AXI UART 16550.

Example 1 — 16550 Mode

Settings

- Baud rate: 56 Kbps
- System clock: 100 MHz
- Enabled threshold settings for the FIFO receive buffer
- Format of asynchronous data exchange, 8 data bits, Even parity, and 2 stop bits

Procedure

1. Write `0x0000_0080` to the Line Control Register. This configures the DLAB bit, which allows writing into the Divisor Latch least significant and most significant bytes.
2. Write `0x0000_006F` to the Divisor Latch least significant byte and write `0x0000_0000` to the Divisor Latch most significant byte in that order. This configures the Baud rate setup of the UART to 56 Kbps operation. The divisor value is calculated by using the formula:

   divisor = (AXI CLK frequency/(16 × Baud Rate))

3. Write `0x0000_001F` to Line Control register. This configures word length to 8 bits, number of stop bits to 2, parity is enabled and set to even parity and the DLAB bit is set to 0 to enable the use of the Transmitter Holding register and Receiver Buffer register data for transmission and reception.
4. Write `0x0000_0011` to the Interrupt Enable register. This enables the Transmitter Holding register empty interrupt and the receive data available interrupt.
5. Write the data to the Transmitter Holding register and read the data received from Receiver Buffer register by servicing the interrupts generated.

Example 2 - 16550 Mode with External XIN Clock

Settings

- Baud rate: 56 Kb/s
- System clock: 100 MHz
- External xin clock: 1.8432 MHz
- Enabled threshold settings for the FIFO receive buffer
- Format of asynchronous data exchange, 8 data bits, Even parity, and 2 stop bits

Procedure

1. Write `0x0000_0080` to the Line Control register. This configures the DLAB bit which allows writing into the Divisor Latch least significant and most significant bytes.
2. Write `0x0000_0002` to the Divisor Latch least significant byte and write `0x0000_0000` to the Divisor Latch most significant byte in that order. This configures the Baud rate setup of the UART to 56 Kbps operation. Other steps remain the same as shown in the previous example.
3. Write `0x0000_001F` to the Line Control register. This configures word length to 8 bits, number of stop bits to 2, parity is enabled and set to even, and the DLAB bit is set to 0 to enable the use of the Transmitter Holding register and Receiver Buffer register data for transmission and reception.
4. Write `0x0000_0011` to the Interrupt Enable register. This enables the Transmitter Holding register empty interrupt and the receive data available interrupt.
5. Write the data to the Transmitter Holding register and read the data received from Receiver Buffer register by servicing the interrupts generated.

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 IP Subsystems using IP Integrator* (UG994) [Ref 4]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6]

Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado Design Suite.

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 2] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5].

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 4] 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.

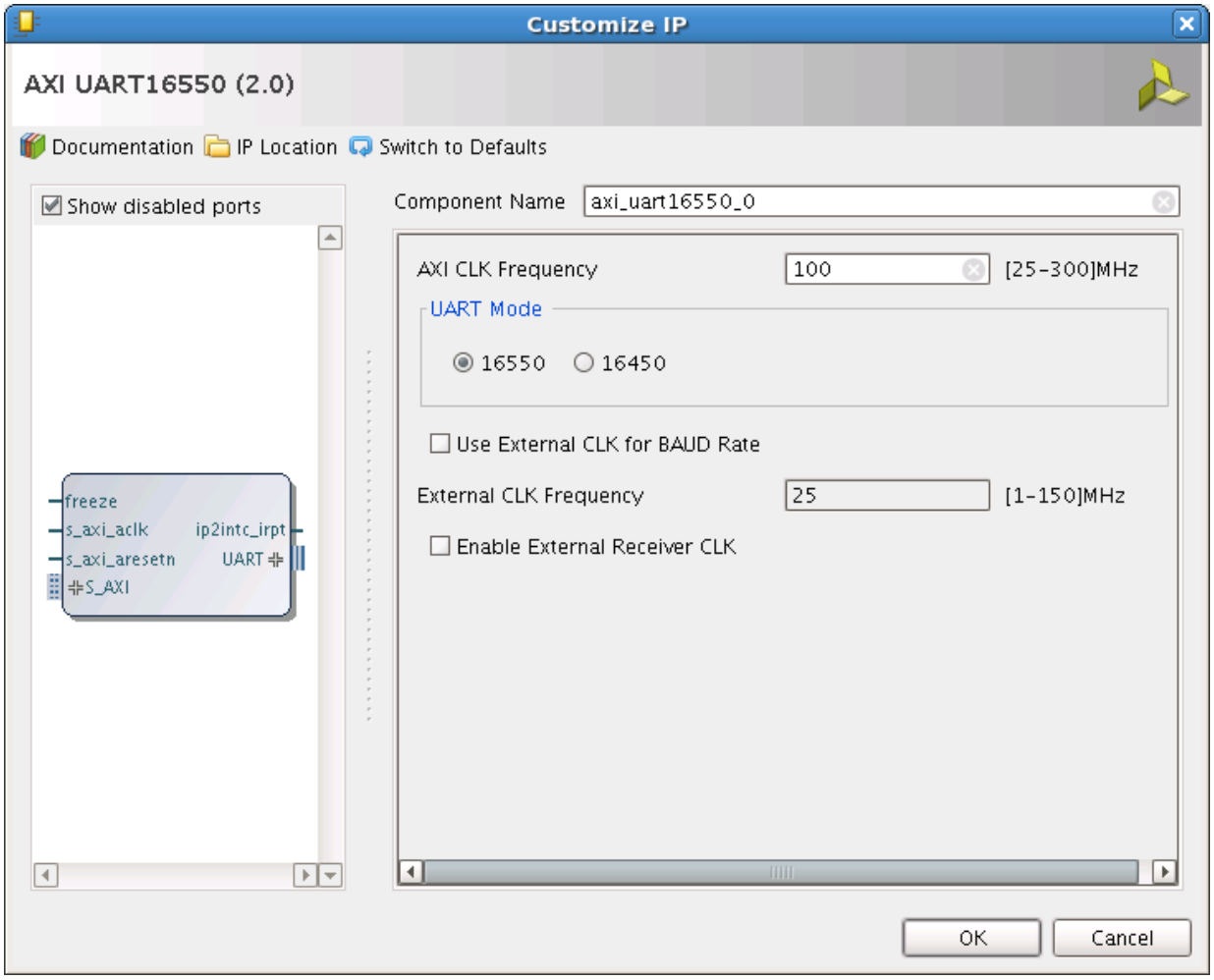

Figure 4-1: **Vivado IDE**

The following describe the default options available in AXI UART 16550 Vivado IDE.

- **AXI CLK Frequency**: System clock frequency (in MHz) driving the AXI UART 16550 peripheral.

  Note: This parameter is auto-updated when the IP is used with IP Integrator.

- **UART Mode**: Select 16550 or 16450 UART.
- **Use External CLK for BAUD Rate**: Checking this box enables AXI UART 16550 to use an external clock for BAUD rate calculation. The `xin` port is externally driven.
- **External CLK Frequency in MHz**: External clock frequency to be expressed in MHz.
- **Enable External Receiver CLK**: `rclk` is externally driven when this option is checked.

User Parameters

Table 4-1 shows the relationship between the fields in the Vivado IDE and the user parameters (which can be viewed in the Tcl console).

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE parameter	User Parameter	Default Value
AXI Clock Frequency	C_S_AXI_ACLK_FREQ_HZ_d	100
UART Mode	C_IS_A_16550	16550
Use External Clock for Baud rate	C_HAS_EXTERNAL_XIN	0
External Clock Frequency	C_EXTERNAL_XIN_CLK_HZ_d	25
Enable External Receiver Clock	C_HAS_EXTERNAL_RCLK	0

Output Generation

For details, see "Generating IP Output Products" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Constraining the Core

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

This section contains information about constraining the core in the Vivado Design Suite.

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 simulation details, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6].

Synthesis and Implementation

For details about synthesis and implementation, see "Synthesizing IP" and "Implementing IP" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

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.

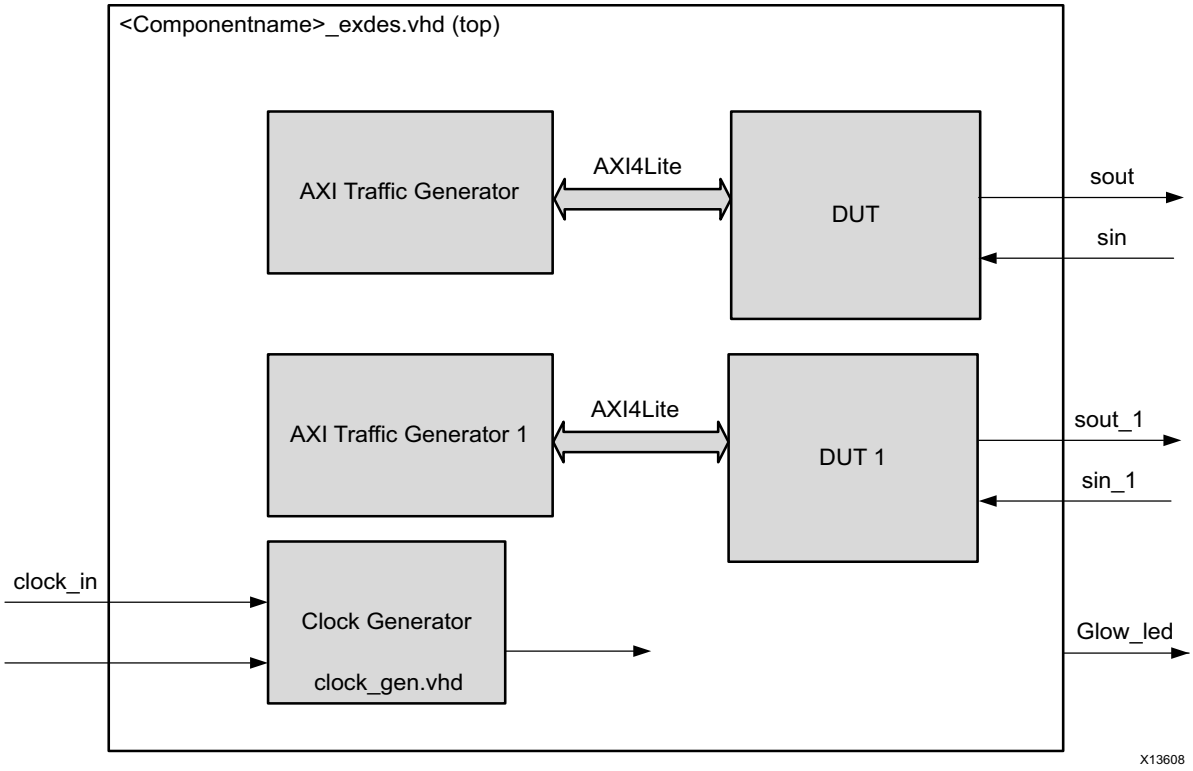

Figure 5-1: **Example Design Block Diagram**

Note: DUT and DUT1 are generated by the same configuration.

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`. DUT and other modules of the example design are kept under reset until MMCME2 is locked.
- **AXI Traffic Generator (ATG)**: This module (IP) is configured in the System Test Mode. All the AXI UART 16550-related AXI4-Lite transactions are stored in the COE/MIF file. For more information on the AXI Traffic Generator, see the *AXI Traffic Generator Product Guide* (PG125) [Ref 7]. The ATG automatically starts the AXI4-Lite transaction after coming out of reset.

Implementing the Example Design

After following the steps described in Chapter 4, 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 will display, asking you to specify a directory for the example design. Select a new directory, or keep the default directory.
3. A new project will be automatically created in the selected directory and it will be opened in a new Vivado window.
4. In the Flow Navigator (left side pane), click **Run Implementation** and follow the directions.

AXI UART 16550 is tested by a feedback loop. The ATG writes x"AA" to the tx_fifo register to the DUT. The DUT1 receives the data through the out pin of DUT0. The ATG then reads the DUT1 rx_fifo and matches the received data with x"AA". The glow_led output of the example design of the UART 16550 can be connected to the LED to know the status of the example design.

On successful completion of ATG read transactions from the DUT1 rx_fifo register, the LED is lit. If there is an error, the LED is not lit.

Example Design Directory Structure

In the current project directory, a new project with the 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 UART 16550 controller example design, as shown in Table 5-1.

Table 5-1: **Example Design Directory**

Name	Description
<component_name>_exdes.vhd	Top-level HDL file for the example design.
clock_gen.vhd	Clock generation module for example design.
atg_addr.coe	COE file of address. This file contains the AXI UART 16550 register address.
atg_data.coe	COE file of data. This file contains the data to be written/read from the AXI UART 16550 registers.
atg_mask.coe	COE file to mask certain reads.
atg_ctrl.coe	COE file that contains control information of ATG.

The `Simulation` directory contains the test bench file for the example design: `<component_name>_exdes_tb.vhd`.

The `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 I/O constraints are commented in the XDC file. Uncomment these constraints before implementing the design on the KC705 board.

Simulating the Example Design

The AXI UART 16550 example design quickly simulates and demonstrates the behavior of the AXI Timer.

Simulation Results

The simulation script compiles the AXI UART 16550 example design, and supporting simulation files. It then runs the simulation and checks to ensure that it completed successfully.

If the test passes, the following message is displayed:

```
Test Completed Successfully
```

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

```
Test Failed !! Test Timed Out !!
```

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 UART 16550 example design. The top-level test bench generates a 200 MHz clock and drives the initial reset to the example design.

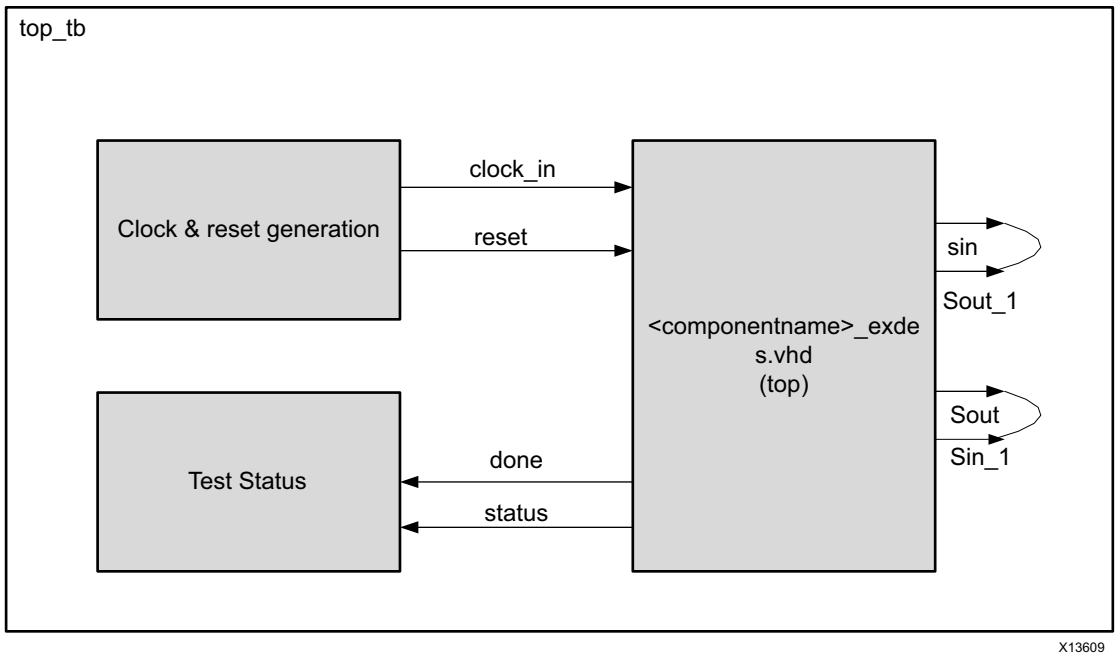

Figure 6-1: **Test Bench Block Diagram**

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® 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 8].

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.

Debugging

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 UART 16550, the 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 UART 16550. 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 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 message(s)
- 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 UART 16550**

AR: 54436

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.

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 represents the functionality in the Vivado IDE that is used for logic debugging and validation of a design running in Xilinx devices in hardware.

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 9].

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 Design Suite debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature for debugging the specific problems.

Additional Resources and Legal Notices

Xilinx Resources

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

References

Unless otherwise noted, IP references are for the product documentation page. These documents provide supplemental material useful with this product guide:

1. *PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs data sheet* (June, 1995)
   www.ti.com/lit/ds/symlink/pc16550d.pdf
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado AXI Reference Guide* (UG1037)
4. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
5. *Vivado Design Suite User Guide: Getting Started* (UG910)
6. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
7. *AXI Traffic Generator LogiCORE IP Product Guide* (PG125)
8. *ISE to Vivado Design Suite Migration Methodology Guide* (UG911)
9. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
10. *ARM® AMBA® Protocol Version 2.0 Specification*
    infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0024b/index.html
11. *LogiCORE IP AXI4-Lite IPIF* (DS765)
12. *AXI Interconnect LogiCORE IP Product Guide* (PG059)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/05/2016	2.0	• Added a note about the AXI4-Lite write access register to the beginning of the Register Space section. • Added the Automotive Applications Disclaimer. • Updated Figures 2-4 and 2-5 and associated Tables 2-8 and 2-9.
11/18/2015	2.0	Added support for UltraScale+ families.
11/19/2014	2.0	Updated resource numbers for 7 series devices and added numbers for UltraScale™ and Zynq® architectures. Added Vivado® IDE Parameter to the User Parameter Relationship table.
04/02/2014	2.0	Updated and corrected register information.
12/18/2013	2.0	Added UltraScale Architecture support.
10/02/2013	2.0	Added Chapter 8, Example Design and Chapter 9, Test Bench. Added support for IP Integrator.
03/20/2013	1.0	Initial release as a product guide. This document is derived from DS748.

Please Read: Important Legal Notices