



UART 16550 IP

User Guide

FPGA-IPUG-02100-1.4

April 2024

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language [FAQ 6878](#) for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Contents.....	3
Acronyms in This Document.....	5
1. Introduction.....	6
1.1. Overview of the IP.....	6
1.2. Quick Facts.....	6
1.3. Features.....	6
1.4. Licensing Information.....	7
1.5. IP Validation Summary.....	8
1.6. Minimum Device Requirements.....	8
1.7. Naming Conventions.....	8
1.7.1. Nomenclature.....	8
1.7.2. Signal Names.....	8
1.7.3. Host.....	8
1.7.4. Attribute Names.....	8
2. Functional Description.....	9
2.1. IP Architecture Overview.....	9
2.2. Clocking and Reset.....	10
2.3. User Interfaces.....	10
2.3.1. Selectable Memory-Mapped Interface.....	10
2.4. Programming Flow.....	10
2.4.1. Initialization.....	10
2.4.2. FIFO Interrupt Mode Operation.....	10
2.4.3. Transmit Operation.....	11
2.4.4. Receive Operation.....	11
2.5. Operations Details.....	12
2.5.1. Data Format.....	12
2.5.2. Timing Diagrams.....	12
3. IP Parameter Description.....	13
4. Signal Description.....	16
5. Register Description.....	18
5.1. Receive Buffer Register (RBR).....	18
5.2. Transmitter Holding Register (THR).....	19
5.3. Interrupt Enable Register (IER).....	19
5.4. Interrupt Identification Register (IIR).....	20
5.5. FIFO Control Register (FCR).....	21
5.6. Line Control Register (LCR).....	21
5.7. MODEM Control Register (MCR).....	22
5.8. Line Status Register (LSR).....	23
5.9. MODEM Status Register (MSR).....	25
5.10. Divisor Latch Register (DLR_MSB, DLR_LSB).....	26
6. Designing with the IP.....	28
6.1. Generating and Instantiating the IP.....	28
6.1.1. Generated Files and File Structure.....	30
6.2. Running Functional Simulation.....	30
7. Debugging.....	33
7.1. Debug Methods.....	33
7.1.1. Invalid Data.....	33
7.2. Debug Tools.....	34
7.2.1. Reveal Analyzer.....	34
7.2.2. Siemens EDA ModelSim Lattice FPGA Edition.....	34
8. Design Considerations.....	35
8.1. Compatibility.....	35

8.2. Configuration.....	35
8.3. Interrupts	35
Appendix A. Resource Utilization.....	36
References	37
Technical Support Assistance	38
Revision History	39

Figures

Figure 1.1. Strategies Dialog Box	7
Figure 2.1. Functional Block Diagram	9
Figure 2.2. Tx/Rx Data Format	12
Figure 2.3. Transmit Operation Timing Diagram with Flow Control Signals (1 byte).....	12
Figure 2.4. Transmit Operation Timing Diagram with Flow Control Signals (2 bytes)	12
Figure 2.5. Receive Operation Timing Diagram with Flow Control Signals (1 byte)	12
Figure 2.6. Transmit Operation Timing Diagram with Flow Control Signals (2 bytes)	12
Figure 6.1. Module/IP Block Wizard	28
Figure 6.2. IP Configuration	29
Figure 6.3. Check Generated Result.....	29
Figure 6.4. Simulation Wizard.....	30
Figure 6.5. Add and Reorder Source.....	31
Figure 6.6. Parse HDL Files for Simulation	31
Figure 6.7. Summary Window.....	32
Figure 6.8. Simulation Waveform	32

Tables

Table 1.1. Summary of the UART 16550 IP	6
Table 1.2. IP Validation Level	8
Table 3.1. Attributes List.....	13
Table 3.2. Attributes Descriptions	14
Table 4.1. UART 16550 IP Core Signal Description	16
Table 5.1. Registers Address Map.....	18
Table 5.2. Access Type Definition	18
Table 5.3. Receive Buffer Register	18
Table 5.4. Transmitter Holding Register	19
Table 5.5. Interrupt Enable Register	19
Table 5.6. Interrupt Identification Register	20
Table 5.7. Interrupt Control Function	20
Table 5.8. FIFO Control Register	21
Table 5.9. Line Control Register	21
Table 5.10. MODEM Control Register.....	22
Table 5.11. Line Status Control Register	24
Table 5.12. Line Control Register	25
Table 5.13. Line Control Register	26
Table 5.14. Line Control Register	26
Table 5.15. Standard Baud Rates Grid with DLR Values for 55.296 MHz System Clock	27
Table 6.1. Generated File List	30
Table A.1. Resource Utilization	36

Abbreviations in This Document

A list of abbreviations used in this document.

Acronym	Definition
APB	Advanced Peripheral Bus
EIA	Electronic Industries Association
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
LMMI	Lattice Memory Mapped Interface
MODEM	Modulator-Demodulator
RTL	Register Transfer Language
UART	Universal Asynchronous Receiver/Transmitter

1. Introduction

1.1. Overview of the IP

The Lattice Semiconductor UART (Universal Asynchronous Receiver/Transmitter) 16550 IP is designed for use in serial communication, supporting the RS-232, RS-422, RS-485, and Electronic Industries Association (EIA) standards, among others. The design features a receiver (serial to parallel converter) and a transmitter (parallel to serial converter), each controlled separately. The register set, data transfer protocol, and interrupt generation of this IP is compatible with the National Semiconductor PC16550D UART with integrated transmit and receive FIFOs which relieves the Host of excessive overhead.

1.2. Quick Facts

Table 1.1. Summary of the UART 16550 IP

IP Requirements	Supported FPGA Family	All
	IP Version	1.3.0
Resource Utilization	Targeted Devices	All
	Supported User Interface	LMMI (Lattice Memory Mapped Interface)*, APB (Advanced Peripheral Bus)
	Resources	Refer to Appendix A. Resource Utilization .
Design Tool Support	Lattice Implementation	IP Core v1.x.x - Lattice Radiant™ Software 2.1 or later
	Synthesis	Lattice Synthesis Engine
		Synopsys Synplify Pro® for Lattice
Simulation	For a list of supported simulators, see the Lattice Radiant Software User Guide .	

1.3. Features

Key features of the UART 16550 IP include:

- Compatible with the National Semiconductor PC16550D UART (NS-PC16550D):
 - Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data.
 - Independently controlled transmit, receive, line status, and data set interrupts.
 - MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD).
 - False start bit detection.
 - Complete status reporting capabilities.
 - Line break generation and detection.
 - Full prioritized interrupt system controls.
- FIFO mode transmitter and receiver, each are buffered with the 16-bytes FIFO to reduce the CPU overhead.
- Configurable data widths of 5, 6, 7, or 8 bits.
- Configurable stop bits – 1 or 2 bits for transmit operations.
- Configurable Baud Rate support for Standard and Custom modes:
 - In Standard mode, programmable divisor latch for baud rates provides fixed pre-defined values.
 - In Custom mode, Baud Rate accepts any value from range 1-999999.
- Verilog RTL test bench.

1.4. Licensing Information

The UART 16550 IP is provided at no additional cost with the Lattice Radiant software. However, an IP-specific license string is required to enable full use of the UART 16550 IP Core in a complete, top-level design. For more information, contact your local [Lattice Sales Office](#).

When the IP is used in the Lattice iCE40 UltraPlus devices, bit stream file cannot be generated without the license string. When the IP is used in FPGA devices built on the Lattice Nexus™ and Avant™ platforms, you can fully evaluate the IP through functional simulation and implementation (synthesis, map, place, and route) without an IP license string.

This IP supports Lattice’s IP hardware evaluation capability which makes it possible to create versions of the IP that operate in hardware for a limited period (approximately four hours) without requiring a license string. It may also be used to evaluate the IP in hardware in the user-defined designs.

The IP evaluation setting is disabled by default and can be enabled/disabled within the Lattice Radiant software by performing the following steps:

1. Launch the Lattice Radiant software and select **Project > Active Strategy > Bitstream Settings**. This will open the Strategies dialog box as shown in [Figure 1.1](#).
2. Enable bitstream generation for IP evaluation purposes by setting the IP Evaluation to True (checked) or disable this feature by setting the IP Evaluation to False (unchecked).

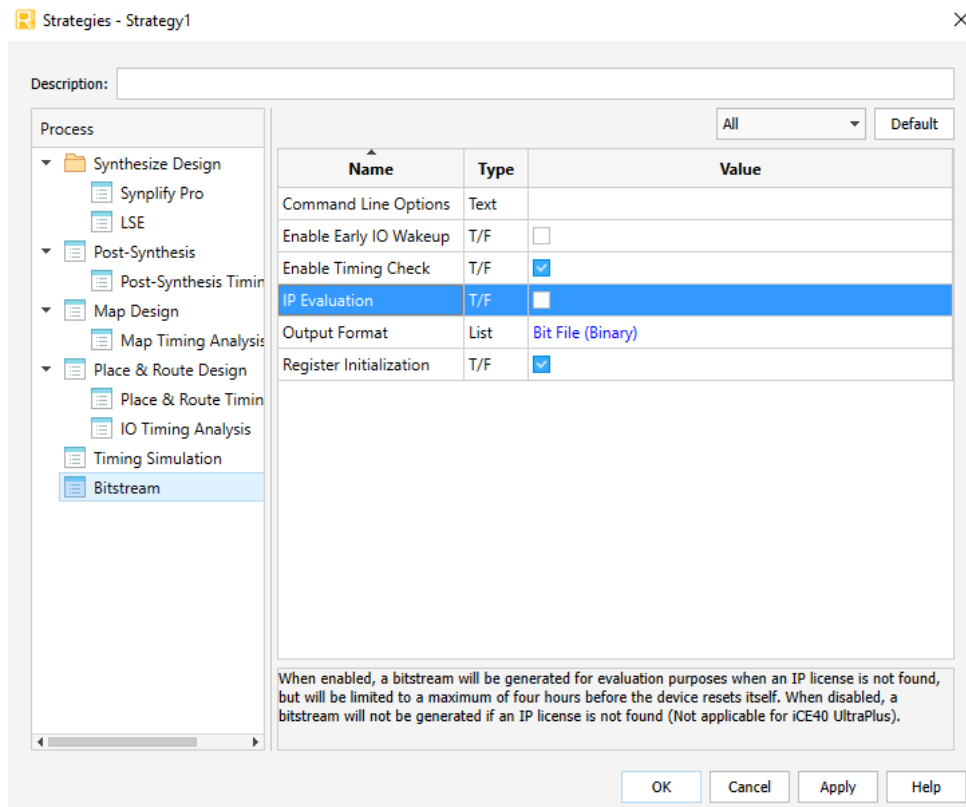


Figure 1.1. Strategies Dialog Box

1.5. IP Validation Summary

The IP is validated with the following synthesis tool: Synopsys Synplify Pro for Lattice or Lattice Synthesis Engine tool.

Table 1.2 shows the validation status for the UART 16550 IP. The ✓ mark indicates whether the IP has been validated for Simulation, Timing, or with Hardware and the ✗ mark indicates otherwise.

Table 1.2. IP Validation Level

Device Family	IP Version	Validation Level			
		Simulation	Compilation	Timing	Hardware
Lattice iCE40 UltraPlus	1.3.0	✓	✓	✓	✗
Lattice Nexus	1.3.0	✓	✓	✓	✓*
Lattice Avant	1.3.0	✓	✓	✓	✓*

*Note: Validated through hardware use cases, no isolated hardware validation was done.

1.6. Minimum Device Requirements

Refer to [Appendix A. Resource Utilization](#) for the minimum required resources to instantiate this IP.

1.7. Naming Conventions

1.7.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.7.2. Signal Names

Signal names that end with:

- *_n* are active low (asserted when value is logic 0)
- *_i* are input signals
- *_o* are output signals

1.7.3. Host

The logic unit inside the FPGA interacts with the UART 16550 IP Core through either LMMI or APB.

1.7.4. Attribute Names

Attribute names in this document are formatted in title case and italicized (*Attribute Name*).

2. Functional Description

This section provides a detailed functional description of UART 16550 IP Core which includes information regarding clock and reset handling, and user interfaces.

2.1. IP Architecture Overview

The UART 16550 IP Core performs two main functions:

- Serial-to-parallel conversion on data characters received from an external UART device or a MODEM; and
- Parallel-to-serial conversion on data characters received from the Host located in the FPGA.

The Host can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART 16550 IP Core, as well as any error conditions (parity, overrun, framing, or break interrupt).

The UART 16550 IP has a complete MODEM-control capability, and a processor-interrupt system. Interrupts can be programmed to your requirements, minimizing the computing required to handle the communications link.

The registers of UART 16550 IP Core are accessed by the Host (FPGA internal components) through either Lattice Memory Mapped Interface (LMMI) or AMBA APB as selected by *Enable APB* attribute. The functional block diagram of UART 16550 IP Core is shown in Figure 2.1. The dashed lines in the figure are optional components/signals, which means they may not be available in the IP when disabled in the attribute.

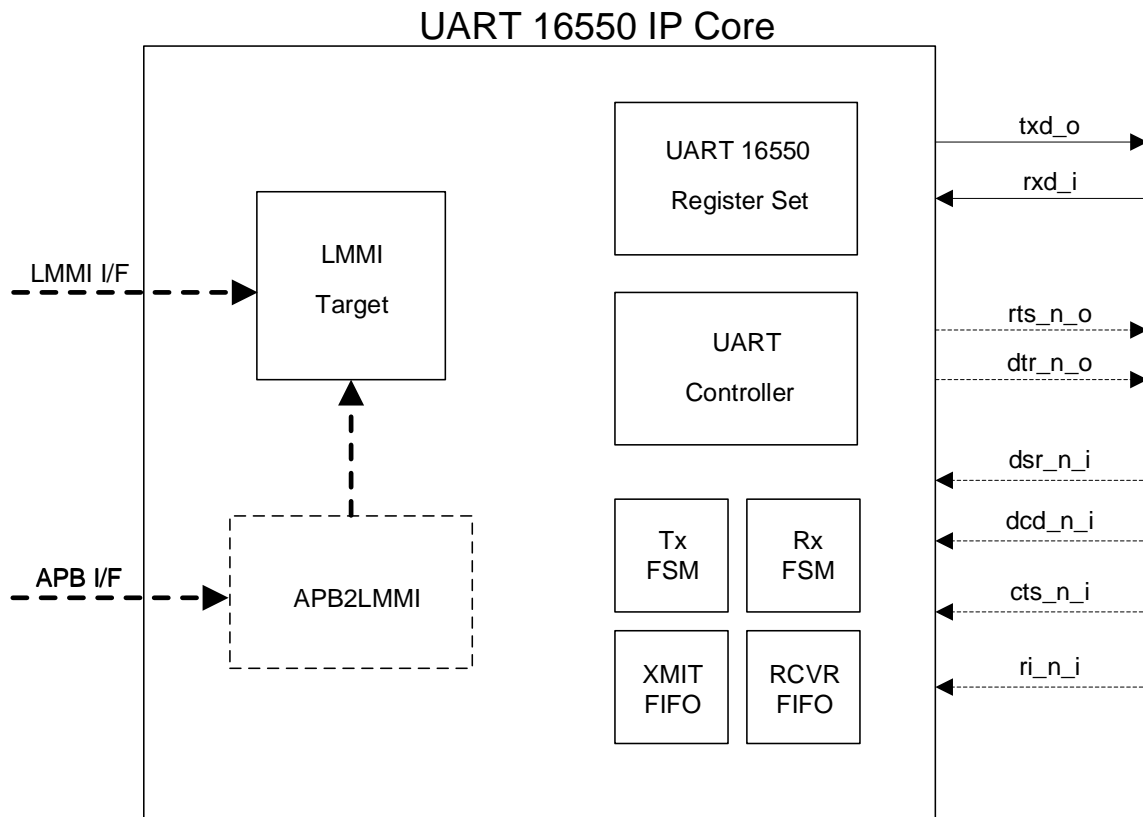


Figure 2.1. Functional Block Diagram

2.2. Clocking and Reset

The UART 16550 IP requires a system clock: `clk_i`. You need to provide this clock via an external source and route it to an appropriate pin on the FPGA. The system clock (externally sourced) provides a timing reference and is used to operate the IP's internal logic. Take note, the externally sourced clock frequency needs to match the system clock frequency set in the IP configuration GUI.

The UART 16550 IP contains an asynchronous active low reset: `rst_n_i`. When asserted, the output ports and registers are forced to their default values. The reset assertion can be asynchronous but reset negation should be synchronous. The IP contains internal logic to synchronously de-assert the internal reset once `rst_n_i` is de-asserted, so you do not need to worry about implementing your own de-assertion logic.

2.3. User Interfaces

2.3.1. Selectable Memory-Mapped Interface

The memory-mapped interface of UART 16550 IP is selected by the *Enable APB* attribute; it can be APB or LMMI. Register access are done via the selected memory-mapped interface. These interfaces are not described in this user guide, references to their respective specifications are provided below.

For LMMI interface, refer to the [Lattice Memory Mapped Interface and Lattice Interrupt Interface \(FPGA-UG-02039\)](#) document for information and timing diagram of the LMMI. Take note of the following information when checking LMMI timing diagram in the said document:

- `lmmi_ready_o` is always asserted; thus, write and read transactions have no wait state.
- Read latency is one clock cycle.

For APB interface, refer to the [AMBA 3 APB Protocol v1.0 Specification](#) for information and timing diagram of the APB interface. Take note of the following information when checking APB timing diagram in the said document:

- Write transaction has one wait state.
- Read transaction has one wait state.

2.4. Programming Flow

2.4.1. Initialization

The following UART register fields should be set properly before performing UART transaction:

- Line Control Register: `even_parity_sel`, `parity_en`, `stop_bit_ctrl`, `char_len_sel`
- Divisor Latch Registers: `divisor_msb`, `divisor_lsb`

These should match the corresponding setting in the communicating UART/MODEM for the serial transaction to be successful. Note that the reset values of these Register fields are configurable during IP generation. Thus, in some applications, initialization step is not necessary.

2.4.2. FIFO Interrupt Mode Operation

When Received Data Available Interrupt is enabled (`IER.rda_int_en=1`), Received Data Available and Character Timeout interrupts occur as follows:

1. The Receiver Data Available trigger is issued to the Host when the FIFO has reached its programmed trigger level, and the interrupt is cleared as soon as the FIFO falls below the programmed trigger level.
2. The IIR Receiver Data Available (`int_prio=2'b10`) indication also occurs when the FIFO trigger level is reached, and like the interrupt, it is cleared when the FIFO drops below the trigger level.
3. The Data Ready Bit (`LSR.data_rdy`) is set as soon as a character is transferred from the shift register to the RCVR FIFO. It is reset (logic low) when the FIFO is empty.

4. A FIFO character timeout interrupt occurs if all the following conditions exist:
 - There is at least one character in the FIFO.
 - The most recent serial character received was longer than four continuous character times ago (one character is eight data bits, one start bit, one parity bit and two stop bits).
 - The most recent Host read of the FIFO was longer than four continuous character times ago.
 - The timer resets when the Host reads one character from the RCVR FIFO.
5. With IER.thre_int_en set to 1'b1, the Transmitter Holding Register interrupt occurs when the XMIT FIFO is empty; it is cleared as soon as the Transmitter Holding Register is written to.

2.4.3. Transmit Operation

The steps for the transmitting character data through the UART 16550 IP Core is shown below. This step assumes that the IP core is not performing transmit operation or at least the XMIT FIFO is empty.

1. If MODEM control signals are used, set MCR.rts_ctrl=1'b1. This asserts rts_n_o signal, telling the peripheral device that the UART would like to transmit data. In response to this, peripheral device asserts cts_n_i if it is ready to receive data.
2. Write data to THR. You can write up to 16-character data.
3. Set IER.thre_int_en=1'b1 to enable Transmit Holding Register Empty interrupt
4. Wait for Transmit Holding Register Empty interrupt to assert.
 - Using interrupt: Wait for interrupt assertion and check that IIR[3:0]=4'b0010.
 - Polling: Read LSR until the thr_empty bit asserts. Note that Step 3 can be skipped for polling mode.
5. If you need to send more characters, write data to THR. You can write up to 16-character data.
6. Repeat Steps 4-5 until all characters are sent.
7. Optional: If MODEM control signals are used, poll LSR until xmitr_empty bit is high. This ensure that the UART transmission of last character is completed. Set MCR.rts_ctrl=1'b0 to negate the rts_n_o signal.

2.4.4. Receive Operation

The steps for the receiving character data through the UART 16550 IP Core is shown below. This step assumes that the IP core is not performing receive operation.

1. Optional: If MODEM control signals are used, set IER.ms_int_en=1'b1 to enable MODEM Status Interrupt. When interrupt occurs due to change in dsr_n_i signal (MSR.dds_r=1'b1) and MSR.cds_r=1'b1, this means the peripheral device is requesting to send data to UART. Set MCR.dtr_ctrl=1 to indicate that UART is ready to receive data.
2. Enable the following interrupts:
 - Received Data Available Interrupt (IER.rda_int_en=1'b1) – to notify the host that received data in FIFO reaches trigger level setting (FCR.trig_lvl) or when character timeout occurs.
 - Receiver Line Status interrupt (IER.rls_int_en=1'b1) – to notify the host of receive status such as error and break condition.
3. Wait for Receive Data Available Interrupt or Character Timeout Interrupt. Wait for interrupt assertion and check that IIR[3:0]=4'b0100 (Receive Data Available) or 2'b1100 (Character Timeout). If Receiver Line Status Interrupt asserts (IIR[3:0]=4'b0110), read the LSR to determine the cause.
4. If Receiver Line Status Interrupt does not occur, read the character data from RBR:
 - If Receive Data Available Interrupt occurs, read trigger level amount of data from RBR.
 - If Character Timeout Interrupt occurs, read LSR. If LSR.data_rdy=1'b1, read RBR. Continue reading RBR until LSR.data_rdy becomes 1'b0.
5. Repeat Steps 3-4 until all expected data are received.
6. Optional: When interrupt occurs due to change in dsr_n_i signal (MSR.dds_r=1'b1) and MSR.cds_r=1'b0, this means the peripheral device is done sending data to UART. Set MCR.dtr_ctrl=0 to indicate that UART is no longer available to receive data.

2.5. Operations Details

2.5.1. Data Format

The character data is written to THR and read from RBR in little endian format as shown in [Figure 2.2](#).

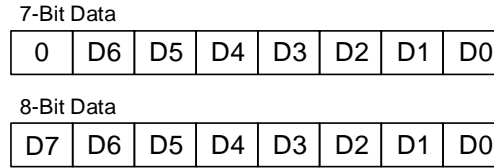


Figure 2.2. Tx/Rx Data Format

2.5.2. Timing Diagrams

During transmission, data is shifted out by the transmitter logic in LSB first format as shown in [Figure 2.3](#). This timing diagram shows the transmission of 1 character with flow control. UART IP Core asserts `rts_n_o`, MODEM `cts_n_i` signal as a response. The UART 16550 IP Core then sends the Start Bit, character data and Stop Bit on the `txd_o` line.

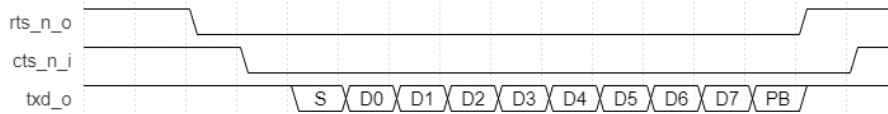


Figure 2.3. Transmit Operation Timing Diagram with Flow Control Signals (1 byte)

When transmitting multiple data bytes, `rts_o` and `cts_i` are held low after the stop bit for the first byte to immediately start the next byte transfer. This is shown in [Figure 2.4](#). In this diagram, one stop bit is generated.

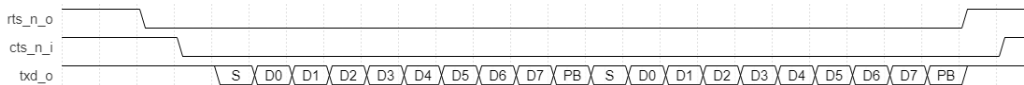


Figure 2.4. Transmit Operation Timing Diagram with Flow Control Signals (2 bytes)

[Figure 2.5](#) shows the transmit operation from the MODEM side. The MODEM asserts (logic 0) the `dsr_n_i` line signaling to the UART IP that it has data to send. The UART IP then asserts (logic 0) the `dtr_n_o` line when it is ready to receive data. The MODEM then initiates the transaction on the `rxd_i`.

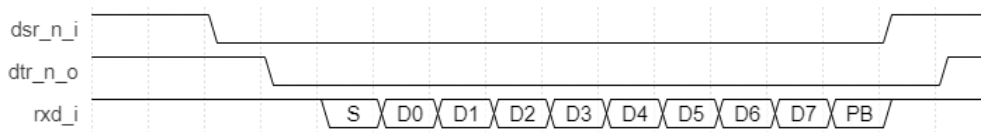


Figure 2.5. Receive Operation Timing Diagram with Flow Control Signals (1 byte)

When receiving multiple data bytes, `dsr_n_i` and `dtr_n_o` are held low after the stop bit for the first byte so that the external MODEM device can immediately start the next byte transfer. This is shown in [Figure 2.6](#).

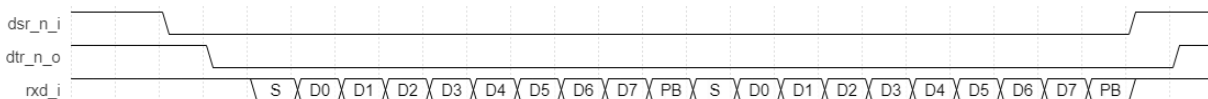


Figure 2.6. Transmit Operation Timing Diagram with Flow Control Signals (2 bytes)

3. IP Parameter Description

The configurable attributes of the UART 16550 IP are shown in [Table 3.1](#) and are described in [Table 3.2](#). You can configure the IP by setting the attributes accordingly in the IP Catalog’s Module/IP wizard of the Lattice Radiant software.

Table 3.1. Attributes List

Attribute	Selectable Values	Default	Dependency on Other Attributes
General			
Enable APB	Checked, Unchecked	Unchecked	—
System Clock Frequency (MHz) [2 - 150]	2–150	50	—
Serial Data Width	5, 6, 7, 8	8	—
Stop Bits	1, 2	1	—
Parity			
Parity Type	No Parity, Even, Odd	No Parity	—
Enable Stick Parity	Checked, Unchecked	Unchecked	Active if <i>Parity Type</i> != No Parity
Baud Rate			
Baud Rate Type	Standard, Custom	Standard	—
UART Standard Baud Rate	2400, 4800, 9600, 14400, 19200, 28800, 38400, 56000, 57600, 115200	115200	Active if <i>Baud Rate Type</i> == Standard
UART Custom Baud Rate [2400 - 1000000]	2400–1000000	115200	Active if <i>Baud Rate Type</i> == Custom
UART Actual Baud Rate	—	Calculated	Calculated based on the formula: <i>Actual Baud Rate</i> = (<i>System Clock Frequency</i>) / (16 × <i>Divisor</i>) Where: <i>Divisor</i> = (<i>System Clock Frequency</i>) / (16 × (<i>Standard or Custom Baud Rate</i>))
MODEM Signal Enables			
Enable Request to Send	Checked, Unchecked	Unchecked	—
Enable Data Terminal Ready	Checked, Unchecked	Unchecked	—
Enable Data Set Ready	Checked, Unchecked	Unchecked	—
Enable Data Carrier Detect	Checked, Unchecked	Unchecked	—
Enable Clear to Send	Checked, Unchecked	Unchecked	—
Enable Ring Indicator	Checked, Unchecked	Unchecked	—
Interrupt			
Received Data Available Interrupt	Checked, Unchecked	Unchecked	—
Transmitter Holding Register Empty Interrupt	Checked, Unchecked	Unchecked	—
Receiver Line Status Interrupt	Checked, Unchecked	Unchecked	—
MODEM Status Interrupt	Checked, Unchecked	Unchecked	—
Receiver FIFO Interrupt Trigger Level (bytes)	1, 4, 8, 14	4	Active if <i>Received Data Available Interrupt</i> is checked

Table 3.2. Attributes Descriptions

Attribute	Dependency on Other Attributes
General	
Enable APB	Selects memory-mapped interface for register access by the host. Checked: AMBA APB Unchecked: LMMI
System Clock Frequency (MHz)	Specifies the target frequency of system clock. This is used for baud rate calculation.
Serial Data Width	Specifies the default data bit width of UART transactions by setting the reset value of char_len_sel field of Line Control Register. 7: Reset value is 2'b10. 8: Reset value is 2'b11.
Stop Bits	Specifies the default number of stop bits to be transmitted and received by specifying the reset value of stop_bit_ctrl bit of Line Control Register. 1: Reset value is 1'b0. 2: Reset value is 1'b1.
Parity	
Parity Type	Specifies the absence/presence of parity as well as the default type (odd/even) of parity by setting the reset values of parity_en and even_parity_sel bits of Line Control Register. No Parity: reset values are parity_en=1'b0 and even_parity_sel=1'b0 Even: reset values are parity_en=1'b1 and even_parity_sel=1'b1 Odd: reset values are parity_en=1'b1 and even_parity_sel=1'b0
Enable Stick Parity	Enables the generation and checking of Stick parity by specifying the reset value of stick_parity_en bit of Line Control Register. Checked: Reset value is 1'b1. Unchecked: Reset value is 1'b0.
Baud Rate	
Baud Rate Type	Selects between Standard Baud Rate and Custom Baud Rate for the reset value of Divisor Latch Register.
Standard Baud Rate	Specifies the baud rate of UART transactions from the standard baud rate options.
Custom Baud Rate	Specifies the baud rate of UART transactions based on user input.
Actual Baud Rate	Calculates the actual baud rate that can be achieved based on Custom Baud Rate input. This also specifies the reset values of Divisor Latch Register as follows: {divisor_msb,divisor_lsb} = 'System Clock Frequency' / (16 × (Standard or Custom Baud Rate))
MODEM Signal Enables	
Enable Request to Send	Enables the presence of rts_n_o signal on the generated IP. Checked: Signal is available. Unchecked: Signal is removed.
Enable Data Terminal Ready	Enables the presence of dtr_n_o signal on the generated IP. Checked: Signal is available. Unchecked: Signal is removed.
Enable Data Set Ready	Enables the presence of dsr_n_i signal on the generated IP. Checked: Signal is available. Unchecked: Signal is removed and internally set to 1'b0 (asserted).
Enable Data Carrier Detect	Enables the presence of dcd_n_i signal on the generated IP. Checked: Signal is available. Unchecked: Signal is removed and internally set to 1'b1 (negated).
Enable Clear to Send	Enables the presence of cts_n_i signal on the generated IP. Checked: Signal is available. Unchecked: Signal is removed and internally set to 1'b1 (asserted). This should be unchecked only if the peripheral device is can always accommodate data transmission.

Attribute	Dependency on Other Attributes
Enable Ring Indicator	Enables the presence of ri_n_i signal on the generated IP. Checked: Signal is available. Unchecked: Signal is removed and internally set to 1'b1 (negated).
Interrupts	
Received Data Available Interrupt	Specifies the reset value of rda_int_en bit of Interrupt Enable Register. Checked: Reset value is 1'b1. Unchecked: Reset value is 1'b0.
Transmitter Holding Register Empty Interrupt	Specifies the reset value of thre_int_en bit of Interrupt Enable Register thre_int_en. Checked: Reset value is 1'b1. Unchecked: Reset value is 1'b0. It is recommended to uncheck this setting to avoid asserting interrupt upon power-up.
Receiver Line Status Interrupt	Specifies the reset value of rls_int_en bit of Interrupt Enable Register. Checked: Reset value is 1'b1. Unchecked: Reset value is 1'b0.
MODEM Status Interrupt	Specifies the reset value of ms_int_en bit of Interrupt Enable Register. Checked: Reset value is 1'b1. Unchecked: Reset value is 1'b0.
Receiver FIFO Interrupt Trigger Level (bytes)	Specifies the reset value of trig_lvl field of FIFO Control Register. 1: Reset value is 2'b00. 4: Reset value is 2'b01. 8: Reset value is 2'b10. 14: Reset value is 2'b11.

4. Signal Description

This section describes the UART 16550 IP ports.

Table 4.1. UART 16550 IP Core Signal Description

Port Name	I/O	Width	Description
Clock and Reset			
clk_i	In	1	System clock.
rst_n_i	In	1	Asynchronous active low reset. The reset assertion can be asynchronous but reset negation should be synchronous. When asserted, output ports and registers are forced to their reset values.
Interrupt Port			
int_o	Out	1	Interrupt signal.
Serial/MODEM Interface			
txd_o	Out	1	Serial Output. Serial data output to the communication link. Reset value: 1'b1
rx_d_i	In	1	Serial Input. Serial data input from the communication link.
rts_n_o	Out	1	Request to Send. When asserted, this informs the peripheral device (MODEM) that the UART is ready to exchange data. This signal is controlled by writing to the rts_ctrl bit of MODEM Control Register. Reset value: 1'b1
dtr_n_o	Out	1	Data Terminal Ready. When low, this informs the MODEM that the UART is ready to establish a communication link This signal is controlled by writing to the dtr_ctrl bit of MODEM Control Register. Reset value: 1'b1
dsr_n_i	In	1	Data Set Ready. When low, this indicates that the peripheral device (MODEM) is ready to establish a communication link with the UART.
dcd_n_i	In	1	Data Carrier Detect. When low, this indicates that the peripheral device (MODEM) has detected the data carrier.
cts_n_i	In	1	Clear to Send. When low, this indicates that the peripheral device (MODEM) is ready to exchange data. When this signal transitions to high logic in the middle of character transmission, the current character is completed, and the next character is pended until this signal becomes low again.
ri_n_i	In	1	Ring Indicator. When low, indicates that the peripheral device (MODEM) has received a telephone-ringing signal.
LMMI Interface*			
lmmi_request_i	In	1	Start transaction.
lmmi_wr_rdn_i	In	1	Write = 1'b1, Read = 1'b0
lmmi_offset_i	In	4	Register offset, starting at offset 0
lmmi_wdata_i	In	8	Output data bus
lmmi_rdata_o	Out	8	Input data bus
lmmi_rdata_valid_o	Out	1	Read transaction is complete and lmmi_rdata_o contains valid data.
lmmi_ready_o	Out	1	IP is ready to receive a new transaction. This is always asserted (tied to 1'b1).
APB Interface*			
apb_psel_i	In	1	Select signal. Indicates that the completer device is selected, and a data transfer is required.
apb_paddr_i	In	6	Address signal.
apb_pwdata_i	In	32	Write data signal. Bits [31:8] are not used.
apb_pwrite_i	In	1	Direction signal. Write = 1, Read = 0
apb_penable_i	In	1	Enable signal. Indicates the second and subsequent cycles of an APB transfer.
apb_pready_o	Out	1	Ready signal. Indicates transfer completion. The completer uses this signal to extend an APB transfer.

Port Name	I/O	Width	Description
apb_pslverr_o	Out	1	Error signal. Indicates a transfer failure. This signal is tied to 1'b0.
apb_prdata_o	Out	32	Read data signal. Bits [31:8] are not used.

***Note:** If *Enable APB* is Checked, APB Interface generated. Otherwise, LMMI Interface is generated.

5. Register Description

The register address map, shown in [Table 5.1](#), specifies the available IP Core registers. The offset of each register is dependent on the *Enable APB* attribute setting as follows:

- LMMI (Unchecked): the offset increments by 1
- APB (Checked): the offset increments by four to allow easy interfacing with the Processor and System Buses. In this case, each register is 32-bit wide wherein the lower 8 bits are used, and the upper 24 bits are unused. The unused bits are treated as reserved – write access is ignored and read access returns 0.

Table 5.1. Registers Address Map

Offset LMMI	Offset APB	Register Name	Access Type	Description
0x00	0x00	RBR	RO	Receive Buffer Register. Active when LCR.dlab=0.
0x00	0x00	THR	WO	Transmitter Holding Register. Active when LCR.dlab=0.
0x01	0x04	IER	RW	Interrupt Enable Register. Active when LCR.dlab=0.
0x02	0x08	IIR	RO	Interrupt Identification Register.
0x02	0x08	FCR	WO	FIFO Control Register.
0x03	0x0C	LCR	RW	Line Control Register.
0x04	0x10	MCR	RW	MODEM Control Register.
0x05	0x14	LSR	RO	Line Status Register.
0x06	0x18	MSR	RO	MODEM Status Register.
0x00	0x00	DLR_LSB	RW	Divisor Latch Register LSB. Active when LCR.dlab=1.
0x01	0x04	DLR_MSB	RW	Divisor Latch Register MSB. Active when LCR.dlab=1.
0x07 – 0x0F	0x1C – 0x3C	Reserved	RSVD	Reserved. Write access is ignored and 0 is returned on read access.

The behavior of registers to write and read access is defined by its access type, which is defined in [Table 5.2](#).

Table 5.2. Access Type Definition

Access Type	Behavior on Read Access	Behavior on Write Access
RO	Returns register value	Ignores write access
WO	Returns 0	Updates register value
RW	Returns register value	Updates register value
RSVD	Returns 0	Ignores write access

5.1. Receive Buffer Register (RBR)

The Receive Buffer Register is the interface to the Receiver Buffer/FIFO (RCVR FIFO). Reading from this register pops and returns the output data in the RCVR FIFO. If read is performed during RCVR FIFO empty, the last data in the FIFO is returned.

This register can be accessed when LCR.dlab=0.

Table 5.3. Receive Buffer Register

Field	Name	Access	Width	Reset
[7:0]	rx_data	RO	8	0x00

- rx_data
Receive Data. Contains the received data from UART.

5.2. Transmitter Holding Register (THR)

The Transmitter Holding Register is the interface to the Transmitter Buffer/FIFO (XMIT FIFO). Writing to this register pushes the write data to the XMIT FIFO. If write is performed during XMIT FIFO full, data is lost. The host can only know if the XMIT FIFO is empty or not (via LSR.thr_empty), it cannot know how many characters are currently in the XMIT FIFO. Thus, it is recommended to initiate the write sequence to this register only when XMIT FIFO is empty wherein the host can write up to 16-character data.

This register can be accessed when LCR.dlab=0.

Table 5.4. Transmitter Holding Register

Field	Name	Access	Width	Reset
[7:0]	tx_data	WO	8	0x00

- tx_data
Transmit Data. Contains the transmit data to UART.

5.3. Interrupt Enable Register (IER)

This register enables the four types of UART interrupts. When enabled, each interrupt can individually activate the interrupt (int_o) output signal. Each interrupt can individually be enabled/disabled by writing to the corresponding bit in the IER. Disabling an interrupt prevents it from being indicated as active in the IIR and from activating the int_o output signal. On the other hand, the UART status including Line Status and MODEM Status Registers are not affected by IER setting.

This register can be accessed when LCR.dlab=0.

Table 5.5. Interrupt Enable Register

Field	Name	Access	Width	Reset
[7:4]	reserved	RSVD	3	–
[3]	ms_int_en	RW	1	Refer to attribute: <i>MODEM Status Interrupt</i>
[2]	rls_int_en	RW	1	Refer to attribute: <i>Receiver Line Status Interrupt</i>
[1]	thre_int_en	RW	1	Refer to attribute: <i>Transmitter Holding Register Empty Interrupt</i>
[0]	rda_int_en	RW	1	Refer to attribute: <i>Received Data Available Interrupt</i>

- ms_int_en
MODEM Status Interrupt Enable. This bit controls the enabling of MODEM Status Interrupt.
 - 1'b0 – Interrupt is disabled
 - 1'b1 – Interrupt is enabled
- rls_int_en
Receiver Line Status Interrupt Enable. This bit controls the enabling of Receiver Line Status Interrupt.
 - 1'b0 – Interrupt is disabled
 - 1'b1 – Interrupt is enabled
- thre_int_en
Transmitter Holding Register Empty Interrupt Enable. This bit controls the enabling of Transmitter Holding Register Empty Interrupt.
 - 1'b0 – Interrupt is disabled
 - 1'b1 – Interrupt is enabled
- rda_int_en
Received Data Available Interrupt Enable. This bit controls the enabling of Received Data Available Interrupt and Timeout Interrupt (timeout_int).
 - 1'b0 – Interrupt is disabled
 - 1'b1 – Interrupt is enabled

5.4. Interrupt Identification Register (IIR)

To minimize software overhead during data character transfers, the UART 16550 IP Core prioritizes interrupts into four levels and records these in the Interrupt Identification Register. When the host reads the IIR, the IP Core only indicates the highest priority pending interrupt. After clearing the source of the highest priority pending interrupt, IIR updates to the next priority pending interrupt. The summary of controlling (asserting and clearing) the interrupt is described in [Table 5.7](#).

Table 5.6. Interrupt Identification Register

Field	Name	Access	Width	Reset
[7:6]	fifos_en	RO	2	2'b11
[5:4]	reserved	RSVD	2	—
[3]	timeout_int	RO	1	1'b0
[2:1]	int_prio	RO	2	2'b00*
[0]	int_pending	RO	1	1'b1*

*Note: The reset value of IER.thre_int_en is set by the *Transmitter Holding Register Empty Interrupt* attribute. Thus, depending on the value of this attribute, the value of int_prio and int_pending may immediately change after reset negation. Below are the observable reset values of {int_prio, int_pending} based on the *Transmitter Holding Register Empty Interrupt* attribute:

- When Unchecked, {int_prio, int_pending} = 3'b001 (No interrupt pending).
 - When Checked, {int_prio, int_pending} = 3'b010 (*Transmitter Holding Register Empty Interrupt* pending).
- **fifos_en**
FIFOs Enabled. Fixed to logic 2'b11 to indicate that both write and read FIFOs are enabled.
 - **timeout_int**
Timeout Interrupt. This interrupt indicates that there is at least one character in RCVR FIFO and FIFO is not updated for a duration of four-character UART transaction. See the [Programming Flow](#) section for more information.
 - 1'b0 – Interrupt is negated
 - 1'b1 – Interrupt is asserted
 - **int_prio**
Interrupt Priority. This bit indicates the highest priority pending interrupt. For example, if all interrupts are asserted, int_prio returns 2'b11 when read. After Receiver Line Status Interrupt is cleared, int_prio becomes 2'b10 to indicate the next high priority pending interrupt.
 - 2'b11 – Receiver Line Status Interrupt
 - 2'b10 – Received Data Available Interrupt
 - 2'b01 – Transmitter Holding Register Empty Interrupt
 - 2'b00 – MODEM Status Interrupt
 - **int_pending**
Interrupt Pending. This bit indicates that an interrupt is pending.
 - 1'b0 – Interrupt is pending according to int_prio and timeout_int.
 - 1'b1 – No pending interrupt

Table 5.7. Interrupt Control Function

IIR[3:0]	Priority	Interrupt Type*	Interrupt Assertion Cause	Interrupt Clearing Control
4'b0001	—	None	None	—
4'b0110	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
4'b0100	Second	Received Data Available	RCVR FIFO reached the trigger level (FCR.trig_lvl)	Reading the Receiver Buffer Register until RCVR FIFO drops below trigger level
4'b1100	Second	Character Timeout	RCVR FIFO has at least 1 character and has not been updated for a duration of 4-character UART transaction	Reading the Receiver Buffer Register
4'b0010	Third	Transmitter Holding Register Empty	XMIT FIFO is empty	Writing to XMIT FIFO.

IIR[3:0]	Priority	Interrupt Type*	Interrupt Assertion Cause	Interrupt Clearing Control
4'b0000	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Registers

*Note: If the interrupt type is disabled in IER, the interrupt control function treats the interrupt as not asserted even if the source of interrupt is asserted. Hence, the IIR value is affected by the IER.

5.5. FIFO Control Register (FCR)

This is a write only register at the same offset as the IIR, which is a read only register. This register is used to control the XMIT FIFO and XCVR FIFO.

Table 5.8. FIFO Control Register

Field	Name	Access	Width	Reset
[7:6]	trig_lvl	WO	2	Refer to attribute: <i>Receiver FIFO Interrupt Trigger Level</i>
[5:4]	reserved	RSVD	2	–
[3]	reserved ¹	RSVD	1	–
[2]	xmit_fifo_rst	WO	1	1'b0
[1]	rcvr_fifo_rst	WO	1	1'b0
[0]	reserved ²	RSVD	1	–

Notes:

- DMA signaling of NS-PC16550D is not implemented.
- FIFO enable control is not implemented because FIFO is always enabled.

- trig_lvl
Sets trigger level of Received Data Available interrupt. See the [Programming Flow](#) section for more information.
 - 2'b00 – Trigger level 1 byte
 - 2'b01 – Trigger level 4 bytes
 - 2'b10 – Trigger level 8 bytes
 - 2'b11 – Trigger level 14 bytes
- xmit_fifo_rst
Resets the XMIT FIFO. This bit is auto clear, after writing 1'b1 to this bit it automatically clears to 1'b0.
 - 1'b0 – Writing 0 is ignored
 - 1'b1 – Resets XMIT FIFO for 1 clock cycle. FIFO content is cleared.
- rcvr_fifo_rst
Resets the RCVR FIFO. This bit is auto clear, after writing 1'b1 to this bit it automatically clears to 1'b0.
 - 1'b0 – Writing 0 is ignored
 - 1'b1 – Resets RCVR FIFO for 1 clock cycle. FIFO content is cleared.

5.6. Line Control Register (LCR)

This configures character length, number stop bits, and parity bits. The most significant bit of the line control register defines the state of the DLAB (Divisor Latch Access Bit) and affects the selection of certain UART Transceiver registers.

Table 5.9. Line Control Register

Field	Name	Access	Width	Reset
[7]	dlab	RW	1	1'b0
[6]	break_ctrl_en	RW	1	1'b0
[5]	stick_parity_en	RW	1	Refer to attribute: <i>Enable Stick Parity</i>
[4]	even_parity_sel	RW	1	Refer to attribute: <i>Parity Type</i>
[3]	parity_en	RW	1	Refer to attribute: <i>Parity Type</i>
[2]	stop_bit_ctrl	RW	1	Refer to attribute: <i>Stop Bits</i>

Field	Name	Access	Width	Reset
[1:0]	char_len_sel	RW	2	Refer to attribute: <i>Serial Data Width</i>

- **dlab**
Divisor Latch Access Bit. The RBR/THR and IER shares offset address with DLR_LSB and DLR_MSB respectively. This bit selects which registers are enabled for access from the host.
 - 1'b0 – Enables access to RBR/THR and IER
 - 1'b1 – Enables access to DLR_LST and DLR_LSB
- **break_ctrl_en**
Break Control Enable. When break control enabled, it causes serial output (txd_o) to go and stay at logic 0, which is interpreted as long stream of 0 bits by the receiving UART - the *break condition*. The Break Control Bit acts only on txd_o and does not control the transmitter logic.
 - 1'b0 – Disables break control
 - 1'b1 – Enables break control
- **stick_parity_en**
Stick Parity Bit. When Bits 3, 4 and 5 of this register are logic 1, the parity bit is transmitted and checked as a logic 0. When Bits 3 and 5 are logic 1 and Bit 4 is logic 0 then the parity bit is transmitted and checked as a logic 1. Stick parity is usually used for checking the parity check behavior of the external UART/MODEM. The parity bit will “stick” if there is an error in the data bits, regardless of the next data packet correctness, until it is cleared.
 - 1'b0 – Disables stick parity
 - 1'b1 – Enables stick parity
- **even_parity_sel**
Even Parity Select bit. This bit is enabled when parity_en is 1'b1.
 - 1'b0 – Odd parity. An odd number of logic 1s are transmitted and checked in to data character bits.
 - 1'b1 – Even parity. An even number of logic 1s are transmitted and checked in to data character bits.
- **parity_en**
Parity Enable bit. Enables transmission and checking of parity bit.
 - 1'b0 – Disables parity
 - 1'b1 – Enables parity
- **stop_bit_ctrl**
Stop Bit Control. Specifies number of stop bits. The receiver only checks the first stop bit regardless of the setting.
 - 1'b0 – One stop bit is generated in the transmitted data.
 - 1'b1 – If char_len_sel==2'b00, one and a half stop bits are generated. Otherwise, two stop bits are generated.
- **char_len_sel**
Character length select. Selects a character length.
 - 2'b00 – 5 bits
 - 2'b01 – 6 bits
 - 2'b10 – 7 bits
 - 2'b11 – 8 bits

5.7. MODEM Control Register (MCR)

This controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM).

Table 5.10. MODEM Control Register

Field	Name	Access	Width	Reset
[7:5]	reserved	RSVD	3	–
[4]	loopback_en	RW	1	1'b0
[3]	out2_ctrl*	RW	1	1'b0
[2]	out1_ctrl*	RW	1	1'b0

Field	Name	Access	Width	Reset
[1]	rts_ctrl	RW	1	1'b0
[0]	dtr_ctrl	RW	1	1'b0

***Note:** Auxiliary user output signals of NS-PC16550D are not implemented. These bits are only used for diagnostic testing of UART (loopback_en=1'b1).

- loopback_en
Provides a local loopback feature for diagnostic testing of the UART. The following occur when loopback mode is enabled:
 - The transmitter Serial Output (txd_o) is set to 1'b1 (marking state)
 - The receiver Serial Input (rxd_i) is disconnected.
 - The output of the Transmitter Shift Register is *looped back* into the Receiver Shift Register input.
 - The four MODEM Control inputs (dsr_n_i, cts_n_i, ri_n_i, and dcd_n_i) are disconnected. These are internally connected to MCR bits as follows:
 - Internal dsr_n_i – dtr_ctrl
 - Internal cts_n_i – rts_ctrl
 - Internal ri_n_i – out1_ctrl
 - Internal dcd_n_i – out2_ctrl
 - The MODEM Control outputs (dtr_n_o, rts_n_o) are forced to their inactive state (high).

In the loopback mode, data that is transmitted is immediately received. This feature allows the host to verify the transmit and received data paths of the UART. In this mode, the receiver and transmitter interrupts are operational. The MODEM Control Interrupts are also operational, but the interrupt sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control input signals. The interrupts are still controlled by the Interrupt Enable Register.

 - 1'b0 – Loopback mode is disabled
 - 1'b1 – Loopback mode is enabled
- out2_ctrl
Auxiliary output 2 control. This bit is only used in loopback mode because auxiliary output 2 signal is not implemented.
 - 1'b0 – Sets internal dcd_n_i to logic 1 (inactive level)
 - 1'b1 – Sets internal dcd_n_i to logic 0 (active level)
- out1_ctrl
Auxiliary output 2 control. This bit is only used in loopback mode because auxiliary output 2 signal is not implemented.
 - 1'b0 – Sets internal ri_n_i to logic 1 (inactive level)
 - 1'b1 – Sets internal ri_n_i to logic 0 (active level)
- rts_ctrl
Request to Send Output Control
 - 1'b0 – Sets rts_n_o to logic 1
 - 1'b1 – Sets rts_n_o to logic 0
- dtr_ctrl
Data Terminal Ready Output Control
 - 1'b0 – Sets dtr_n_o to logic 1
 - 1'b1 – Sets dtr_n_o to logic 0

5.8. Line Status Register (LSR)

This provides status information to the Host concerning data transfer. This register is not affected by Interrupt Enable Register. For example, if *Receiver Line Status Interrupt* is disabled, interrupt is not generated but the host can still read the actual status of the transfer from this register.

Table 5.11. Line Status Control Register

Field	Name	Access	Width	Reset
[7]	rcvr_fifo_err	RO	1	1'b0
[6]	xmitr_empty	RO	1	1'b1
[5]	thr_empty	RO	1	1'b1
[4]	break_cond	RO	1	1'b0
[3]	framing_err	RO	1	1'b0
[2]	parity_err	RO	1	1'b0
[1]	overrun_err	RO	1	1'b0
[0]	data_rdy	RO	1	1'b0

- **rcvr_fifo_err**
Receiver FIFO error status. The rcvr_fifo_err asserts when there is at least one parity error, framing error or break indication in the FIFO. This bit is cleared when the host reads the LSR and there are no subsequent errors in the Receiver FIFO.
 - 1'b0 – Indicates RCVR FIFO has no data that generates error status or break condition
 - 1'b1 – Indicates RCVR FIFO contains at least 1 data that generates error status or break condition
- **xmitr_empty**
Transmitter Empty indicator.
 - 1'b0 – Indicates XMIT FIFO or Transmitter Shift Register has data
 - 1'b1 – Indicates XMIT FIFO and Transmitter Shift Register are both empty
- **thr_empty**
Transmitter Holding Register empty indicator. The thr_empty asserts after the last data in FIFO is sent to the Transmitter Shift Register. This bit negates when at least one byte is written to the XMIT FIFO. The thr_empty bit generates interrupt when IER.thr_int_en is set to logic 1.
 - 1'b0 – Indicates XMIT FIFO contains at least one data
 - 1'b1 – Indicates XMIT FIFO is empty
- **break_cond**
Break condition indicator. Break condition occurs when the received data input is held in the Spacing state (logic 0) for longer than a full word transmission time (that is, the total time of Start bit, Data bits, Parity bit and Stop bits). The break_cond bit is reset whenever the Host reads the contents of the Line Status Register. This error is associated with the specific character in the FIFO to which it applies. This error is asserted when its associated character is at the output end of the FIFO. When break occurs, only one zero character is loaded into the FIFO. The next character transfer is enabled after rxd_i goes to the marking state and receives the next valid start bit. The break_cond generates interrupt when asserted if Receiver Line Status Interrupt is enabled (IER.rls_int_en == 1).
 - 1'b0 – No break condition
 - 1'b1 – Break condition occurs
- **framing_err**
Framing Error Indicator. Framing Error occurs when a received character did not have a valid stop bit. This bit asserts whenever the Stop bit following the last data bit or parity bit is detected as a logic 0 bit (Spacing level). The framing_err bit is reset whenever the Host reads the contents of the Line Status Register. This error is associated with the specific character in the FIFO it applies to. This error is asserted when its associated character is at the output end of the FIFO. The framing_err generates interrupt when asserted if Receiver Line Status Interrupt is enabled (IER.rls_int_en == 1).
 - 1'b0 – No framing error
 - 1'b1 – Framing error occurs
- **parity_err**
Parity Error Indicator. Parity error occurs when a received character does not have a correct even or odd parity as selected by LCR.even_parity_sel. The parity_err bit is reset whenever the Host reads the contents of the Line Status Register. This error is associated with the specific character in the FIFO to which it applies. This error is asserted when

its associated character is at the output end of the FIFO. The `parity_err` generates interrupt when asserted if Receiver Line Status Interrupt is enabled (`IER.rls_int_en == 1`).

- 1'b0 – No parity error
- 1'b1 – Parity error occurs
- `overrun_err`
Overrun Error Indicator. Overrun error occurs when RCVR FIFO is full, and the next character has been fully received. In this case, the new character is lost. The `overrun_err` generates interrupt when asserted if Receiver Line Status Interrupt is enabled (`IER.rls_int_en == 1`).
 - 1'b0 – No overrun error
 - 1'b1 – Overrun error occurs
- `data_rdy`
Data Ready Indicator. This bit asserts when a complete incoming character is received and transferred into the Receiver Buffer Register or the FIFO. This bit is reset to a logic 0 by reading all the data in the Receiver Buffer Register or the FIFO.
 - 1'b0 – Receive Buffer Register has no data
 - 1'b1 – Receive Buffer Register has data

5.9. MODEM Status Register (MSR)

This provides the current state of the control lines from the MODEM (or peripheral device) to the Host via the following fields: `cdcdi`, `cri`, `cdsr`, `ccts`. In addition to this, the other four bits (`ddcdi`, `teri`, `ddsr`, `dcts`) of the MODEM Status Register provide change information. These bits are set to a logic 1 when the corresponding control input from the MODEM changes state. They are reset to logic 0 when the Host reads the MODEM Status Register.

Table 5.12. Line Control Register

Field	Name	Access	Width	Reset
[7]	<code>cdcdi</code>	RO	1	1'b0. Refer to signal description listed below.
[6]	<code>cri</code>	RO	1	1'b0. Refer to signal description listed below.
[5]	<code>cdsr</code>	RO	1	1'b0. Refer to signal description listed below.
[4]	<code>ccts</code>	RO	1	1'b0. Refer to signal description listed below.
[3]	<code>ddcdi</code>	RO	1	1'b0. Refer to signal description listed below.
[2]	<code>teri</code>	RO	1	1'b0. Refer to signal description listed below.
[1]	<code>ddsr</code>	RO	1	1'b0. Refer to signal description listed below.
[0]	<code>dcts</code>	RO	1	1'b0. Refer to signal description listed below.

- `cdcdi`
Complement of Data Carrier Detect input. If the attribute *Enable Data Carrier Detect* is Checked, the reset value is not observable because this bit is updated in the next cycle.
 - 1'b0 – `dcd_n_i` is negated (logic 1)
 - 1'b1 – `dcd_n_i` is asserted (logic 0)
- `cri`
Complement of Ring Indicator input. If the attribute *Enable Ring Indicator* is Checked, the reset value is not observable because this bit is updated in the next cycle.
 - 1'b0 – `ri_n_i` is negated (logic 1)
 - 1'b1 – `ri_n_i` is asserted (logic 0)
- `cdsr`
Complement of Data Set Ready input. If the attribute *Enable Data Set Ready* is Checked, the reset value is not observable because this bit is updated in the next cycle.
 - 1'b0 – `dsr_n_i` is negated (logic 1)
 - 1'b1 – `dsr_n_i` is asserted (logic 0)

- ccts
Complement of Clear to Send input. If the attribute *Enable Clear to Send* is Checked, the reset value is not observable because this bit is updated in the next cycle.
 - 1'b0 – cts_n_i is negated (logic 1)
 - 1'b1 – cts_n_i is asserted (logic 0)
- ddcdi
Delta Data Carrier Detect Indicator. Indicates the dcd_n_i input has changed state. The ddcdi generates interrupt when asserted if MODEM Status Interrupt is enabled (IER.ms_int_en == 1).
 - 1'b0 – No change in dcd_n_i input from the last read of Modem Status Register
 - 1'b1 – The dcd_n_i input toggles
- teri
Trailing Edge of Ring indicator detector. Indicates ri_n_i input has changed state from a low to a high state. The teri generates interrupt when asserted if MODEM Status Interrupt is enabled (IER.ms_int_en == 1).
 - 1'b0 – No low to high transition in ri_n_i input from the last read of Modem Status Register
 - 1'b1 – The ri_n_i input toggles from low to high state
- ddsr
Delta Data Set Ready indicator. Indicates dsr_n_i input has changed state since last time it is read by the Host. The ddsr generates interrupt when asserted if MODEM Status Interrupt is enabled (IER.ms_int_en == 1).
 - 1'b0 – No change in dsr_n_i input from the last read of Modem Status Register
 - 1'b1 – The dsr_n_i input toggles
- dcts
Delta Clear to Send indicator. Indicates cts_n_i input has changed state since last time it is read by the Host. The dcts generates interrupt when asserted if MODEM Status Interrupt is enabled (IER.ms_int_en == 1).
 - 1'b0 – No change in cts_n_i input from the last read of Modem Status Register
 - 1'b1 – The cts_n_i input toggles

5.10. Divisor Latch Register (DLR_MSB, DLR_LSB)

The UART 16550 IP Core contains a programmable baud generator that can divide the reference clock input (clk_i) by divisors of 1 to (2¹⁶-1) and producing a 16 times clock for driving the internal transmitter and receiver logic. Two 8-bit registers (DLR_MSB, DLR_LSB) store the divisor in a 16-bit binary format. These Divisor Latch Registers must be loaded during initialization to ensure proper operation of the baud generator. Upon loading either of the Divisor Latch Registers, a 16-bit Baud Counter is immediately loaded.

These registers can be accessed when LCR.dlab=1.

Table 5.13. Line Control Register

DLR_MSB Field	Name	Access	Width	Reset
[7:0]	divisor_msb	RW	8	Refer to attribute: <i>Baud Rate</i>

Table 5.14. Line Control Register

DLR_MSB Field	Name	Access	Width	Reset
[7:0]	divisor_lsb	RW	8	Refer to attribute: <i>Baud Rate</i>

- divisor_msb
The upper byte of the Divisor Latch Register.
- divisor_lsb
The lower byte of the Divisor Latch Register.

The divisor ({divisor_msb, divisor_lsb}) is calculated as follows: ({divisor_msb, divisor_lsb}) = 'System Clock Frequency' / (16 × (Standard or Custom Baud Rate)).

For the example, [Table 5.15](#) shows the values of the Divisor Latch Register for the UART IP supported standard baud rates grid for a system clock of 55.296 MHz.

Table 5.15. Standard Baud Rates Grid with DLR Values for 55.296 MHz System Clock

Supported Baud Rate Grid	Divisor Latch (divisor_msb, divisor_lsb)
2400	1440
4800	720
9600	360
14400	240
19200	180
28800	120
38400	90
56000	62
57600	60
115200	30

6. Designing with the IP

This section provides information on how to generate the UART 16550 IP using the Lattice Radiant software, and how to run simulation and synthesis. For more details on the Lattice Radiant Software, refer to the [Lattice Radiant Software 2023.1 User Guide](#).

6.1. Generating and Instantiating the IP

You can use the Lattice Radiant software to generate IP modules and integrate them into the device's architecture. The steps below describe how to generate the UART 16550 IP in the Lattice Radiant software.

To generate the UART 16550 IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **UART 16550** under **IP, Processors, Controllers, and Peripherals** category. The **Module/IP Block Wizard** opens as shown in [Figure 6.1](#). Enter values in the **Instance name** and the **Create in** fields and click **Next**.

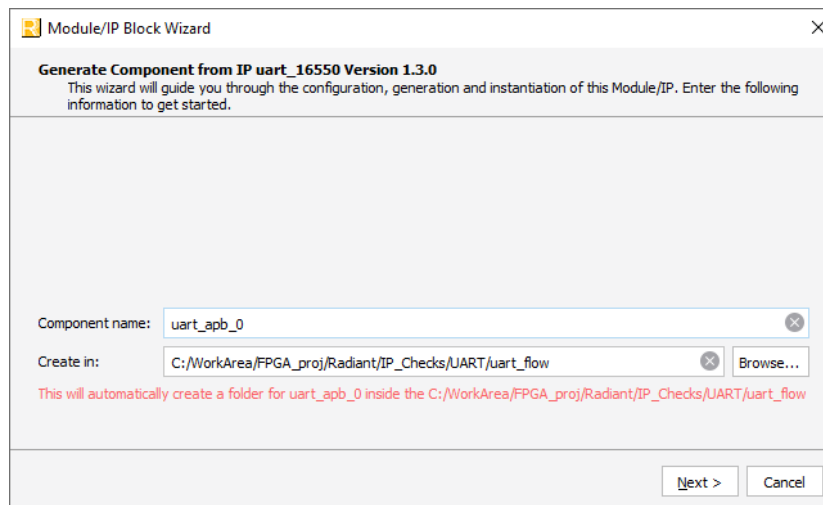


Figure 6.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected UART 16550 IP using drop-down lists and check boxes. [Figure 6.2](#) shows an example configuration of the UART 16550 IP. For details on the configuration options, refer to the [IP Parameter Description](#) section.

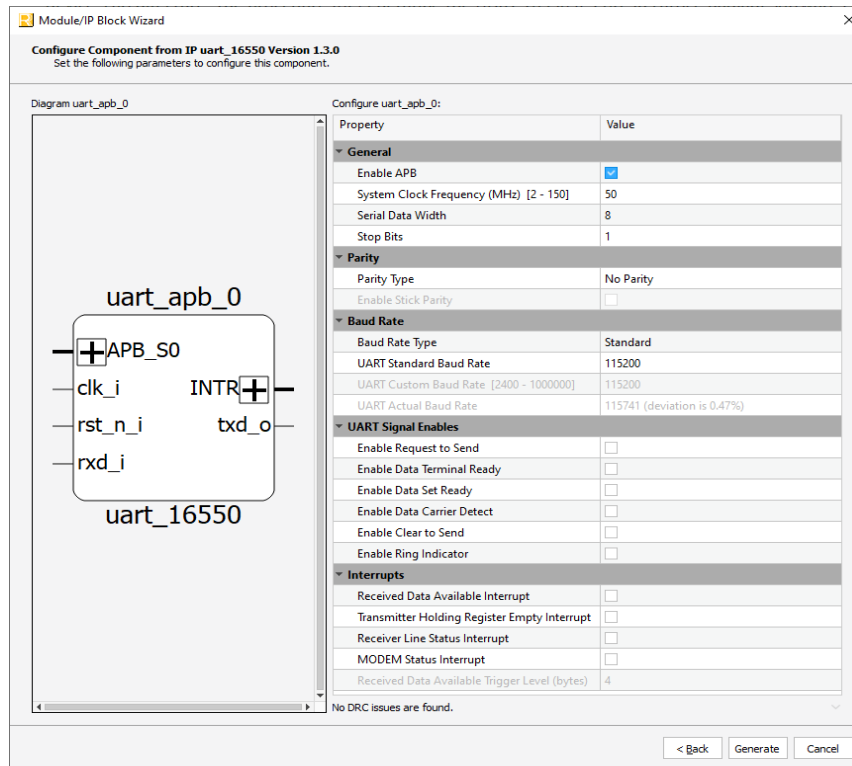


Figure 6.2. IP Configuration

- Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results as shown in Figure 6.3.

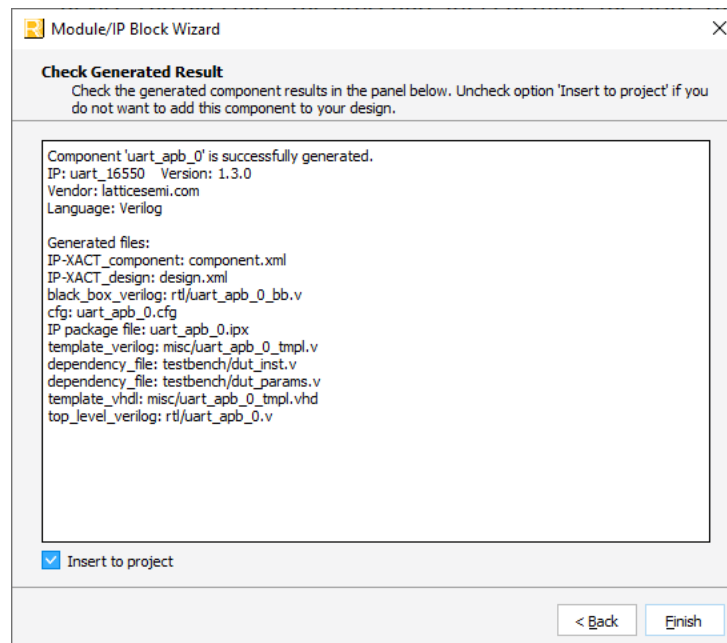


Figure 6.3. Check Generated Result

- Click **Finish**. All the generated files are placed under the directory paths in the **Create in** and the **Instance name** fields shown in Figure 6.1.

6.1.1. Generated Files and File Structure

The generated UART 16550 IP Core package includes the closed-box (<Instance Name>_bb.v) and instance templates (<Instance Name>_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (<Instance Name>.v) that can be used as an instantiation template for the IP core is also provided. You may also use this top-level reference as the starting template for the top-level for their complete design. The generated files are listed in [Table 6.1](#).


Table 6.1. Generated File List

Attribute	Description
<Instance Name>.ipx	This file contains the information on the files associated to the generated IP.
<Instance Name>.cfg	This file contains the parameter values used in IP configuration.
component.xml	Contains the ipxact:component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Instance Name>.v	This file provides an example RTL top file that instantiates the IP core.
rtl/<Instance Name>_bb.v	This file provides the synthesis closed-box.
misc/<Instance Name>_tmpl.v misc /<Instance Name>_tmpl.vhd	These files provide instance templates for the IP core.

6.2. Running Functional Simulation

You can run functional simulation after the IP is generated.

To run functional simulation:

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** shown in [Figure 6.4](#).

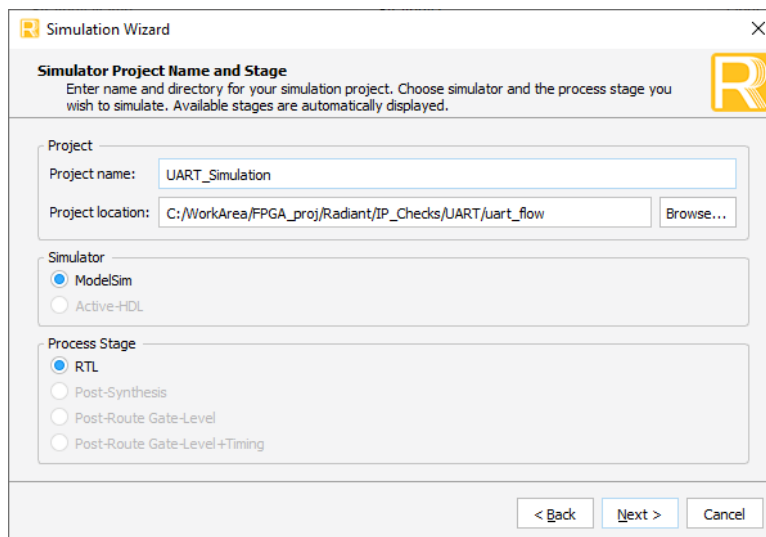


Figure 6.4. Simulation Wizard

- Click **Next** to open the **Add and Reorder Source** window as shown in [Figure 6.5](#).

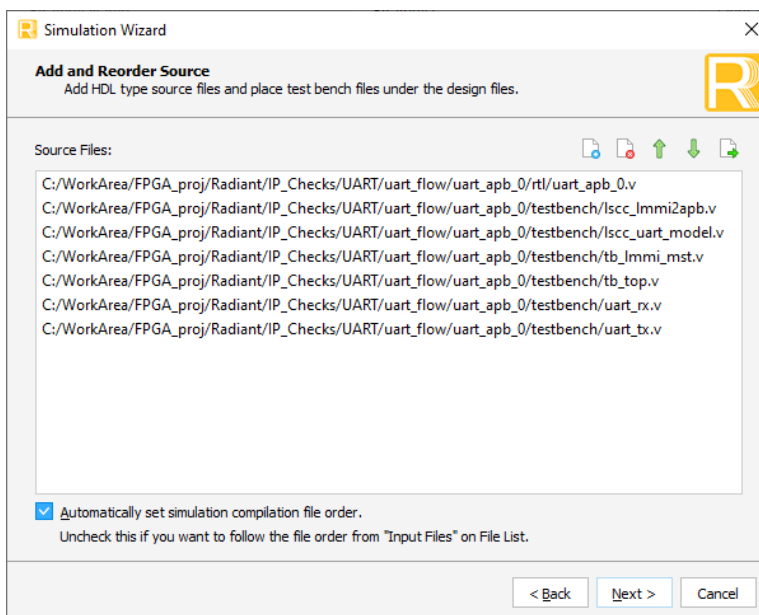


Figure 6.5. Add and Reorder Source

- Click **Next** to open the **Parse HDL files for simulation** window as shown in [Figure 6.6](#).

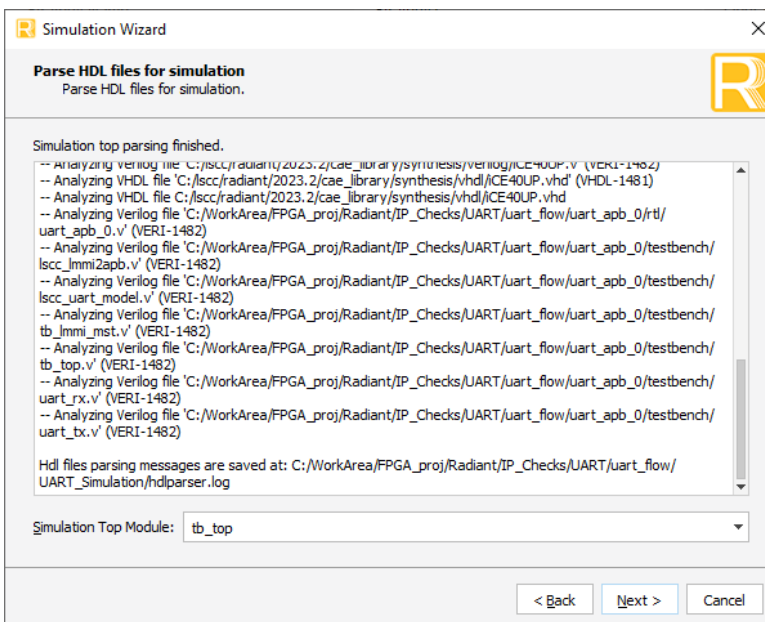


Figure 6.6. Parse HDL Files for Simulation

4. Click **Next**. The **Summary** window is opened as shown in [Figure 6.7](#).

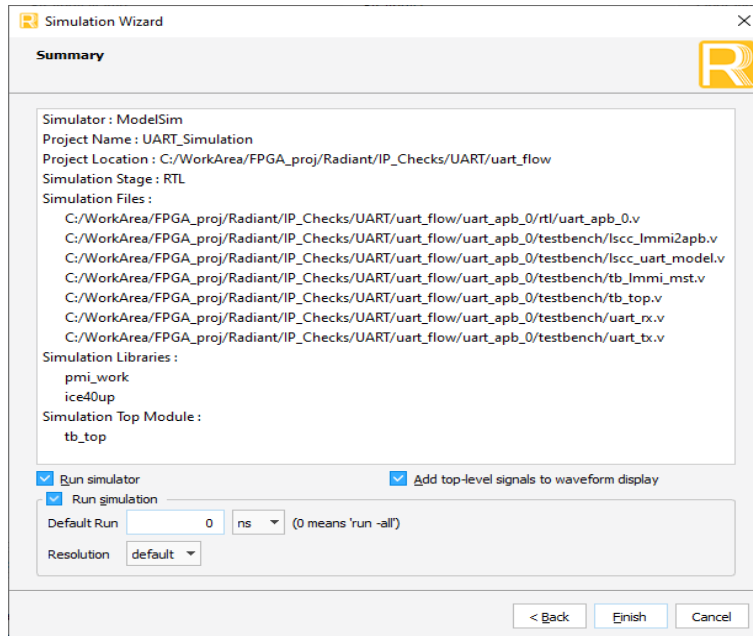


Figure 6.7. Summary Window

5. Click **Finish** to run the simulation.

Note: It is necessary to follow the procedure above until it is fully automated in the Lattice Radiant software.

The waveform in [Figure 6.8](#) shows an example simulation result.

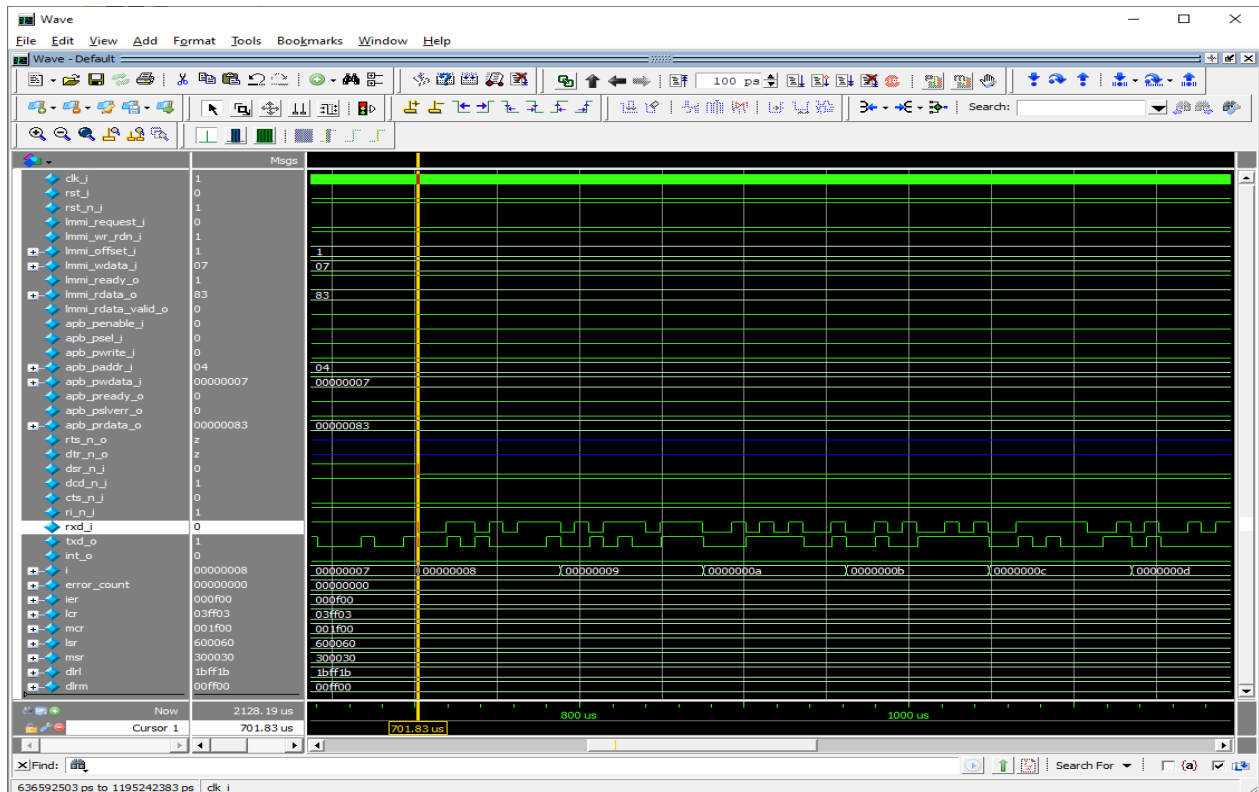


Figure 6.8. Simulation Waveform

7. Debugging

This section lists possible issues and suggested troubleshooting steps that you can follow.

7.1. Debug Methods

7.1.1. Invalid Data

When the UART shows invalid data, the information received might appear garbled, incomplete, missing entirely, contain unexpected characters, or fail verification checks. In some cases, communication might even timeout or generate specific error messages. Here are some guidelines on how to resolve this issue:

1. Verify Baud Rate Settings
 - Confirm the configured baud rate in both transmitting and receiving devices. Ensure they are identical. UART 16550 IP Core baud rate settings can be configured in the IP configuration GUI as shown in [Figure 6.2](#).
 - Double-check any configuration tools or communication software used for baud rate settings. Typos or inconsistencies can cause issues.
2. Inspect Supplied Clock
 - Ensure the external clock source is properly connected and functioning. Check for loose connections or damaged components.
 - Validate the frequency of the supplied clock matches the expected value. It should match the System Clock Frequency value configured in UART 16550 IP Core configuration GUI as shown in [Figure 6.2](#).
3. Utilize Loopback Test
 - Connect the transmit and receive pins of the UART using a loopback connector. This allows testing the functionality without relying on external communication.
 - Transmit a known test pattern and verify if the received data matches at the configured baud rate. This helps to isolate the issues to the UART itself or the communication pathway.

7.2. Debug Tools

You can use the tools described in the subsections to debug UART 16550 IP design issues.

7.2.1. Reveal Analyzer

Reveal continuously monitors signals within the FPGA for specific conditions, which can range from simple to quite complex. When the trigger condition occurs, Reveal can save signal values preceding, during, and following the event for analysis, including a waveform presentation. The data can be saved to a value change dump file (.vcd), which can be used with tools such as Siemens EDA ModelSim™, or to an ASCII tabular format that can be used with tools such as Excel.

Before running Reveal Analyzer, use Reveal Inserter to add Reveal modules to your design. In these modules, specify the signals to monitor, define the trigger conditions, and other options. Reveal supports multiple logic analyzer cores using hard/soft JTAG interface. You can have up to 15 modules, typically one for each clock region of interest. When the modules are set up, regenerate the bitstream data file to program the FPGA.

The main purpose is narrowing down to problem areas during debug cycles using a divide and conquer method into many small functional blocks to control and monitor the status of each block.

Refer to [Reveal User Guide for Radiant Software](#) for details on how to use Reveal Analyzer.

7.2.2. Siemens EDA ModelSim Lattice FPGA Edition

The Siemens EDA ModelSim tool is an OEM simulation tool that is closely linked to the Radiant software environment. It is not run as part of the Process implementation flow. ModelSim adds support for Linux-based FPGA development and improves overall simulation performance for faster design verification.

Refer to the [Lattice Radiant Software User Guide](#) for more information on Siemens EDA ModelSim Lattice FPGA Edition.

8. Design Considerations

8.1. Compatibility

- Standards
Ensure your application adheres to the supported communication standards like RS-232, RS-422, RS-485, and EIA.
- National Semiconductor PC16550D
Verify compatibility with the register set, data transfer protocol, and interrupt generation of this specific UART variant.

8.2. Configuration

- Serial Data Width
Choose the appropriate Serial Data Width (5, 6, 7, or 8 bits) for your communication needs.
- Stop Bits
Configure the number of stop bits (1 or 2) for the transmit operation.
- Parity
Select the desired parity mode (even, odd, or stick) for both transmit and receive operations.
- Baud Rate
Utilize the programmable divisor latch to set the custom baud rate for your application. Refer to the [Register Description](#) section for more information.

8.3. Interrupts

- Interrupt Identification
Utilize the provided Interrupt Identification Register to effectively handle different interrupt sources. Refer to the [Register Description](#) section for more information.

Appendix A. Resource Utilization

Table A.1 shows a sample resource utilization of the UART_16550 default configuration on the iCE40UP5K, LIFCL-40-9BG400I, LFCPNX-100-9LFG672I, LAV-AT-E70-3LFG1156I, and LAV-AT-G70-3LFG1156I devices using Lattice Synthesis Engine of Lattice Radiant Software Version 2023.2.

Table A.1. Resource Utilization

Device	PFU Registers	LUT4s	EBRs
iCE40UP5K	622	1275	0
LIFCL-40-9BG400I	620	912	0
LFCPNX-100-9LFG672I	620	912	0
LAV-AT-E70-3LFG1156I	619	895	0
LAV-AT-G70-3LFG1156I	619	895	0

References

- [Lattice Memory Mapped Interface and Lattice Interrupt Interface \(FPGA-UG-02039\)](#)
- [AMBA 3 APB Protocol v1.0 Specification](#)
- [Lattice Radiant Software 2023.1 User Guide](#)
- [Reveal User Guide for Radiant Software](#)
- [Lattice Radiant Software web page](#)
- [Lattice Insights web page](#) for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.4, April 2024

Section	Change Summary
All	<ul style="list-style-type: none"> Minor adjustments to ensure that the document is consistent with Lattice Semiconductor's inclusive language policy. Made editorial fixes. Renamed the document from <i>UART 16550 IP Core - Lattice Radiant Software</i> to <i>UART 16550 IP</i>.
Disclaimers	Updated boilerplate.
Inclusive Language	Added boilerplate.
Abbreviations in This Document	<ul style="list-style-type: none"> Replaced the word <i>acronyms</i> in this section to <i>abbreviations</i>. Added FPGA, GUI and MODEM to the list of abbreviations.
Introduction	<ul style="list-style-type: none"> Moved the paragraph of the 1. Introduction section to the 1.1. Overview of the IP section. Moved the previous 3.1. Licensing the IP and 3.4. Hardware Evaluation sections to the 1.4. Licensing Information section and updated the content. Added the 1.5. IP Validation Summary and 1.6. Minimum Device Requirements sections and updated the heading numbers of the remaining sections accordingly. Added <i>Naming</i> to the title of the 1.7. Naming Conventions section.
Functional Description	<ul style="list-style-type: none"> Added an introductory paragraph to this section. Added <i>IP Architecture</i> to the title of the 2.1. IP Architecture Overview section. Updated Figure 2.1. Functional Block Diagram. Added the 2.3. User Interfaces section and updated the heading numbers of the remaining sections accordingly. Moved the previous 2.5. Programming Flow section to the 2.4. Programming Flow section, updated the content and heading numbers accordingly. Moved the previous 2.6. Data Format, and 2.7. Timing Diagrams sections into the 2.5. Operations Details section and updated the section heading numbers accordingly.
IP Parameter Description	<ul style="list-style-type: none"> Moved the previous 2.3. Attribute Summary section to this section, renamed the section to 3. IP Parameter Description, and updated the section heading number and table numbers accordingly.
Signal Description	Moved the previous 2.2. Signal Description section to this section, and updated the section heading number and table number accordingly.
Register Description	Moved the previous 2.4. Register Description section to this section, and updated the section heading numbers and table numbers accordingly.
Designing with the IP	<ul style="list-style-type: none"> Moved the previous 3.2. Generation and Synthesis section to the 6.1. Generating and Instantiating the IP section, and updated the content, heading numbers, figure numbers and table number accordingly. Updated Figure 6.1. Module/IP Block Wizard – Figure 6.3. Check Generated Result. Moved the previous 3.3. Running Functional Simulation section to the 6.2. Running Functional Simulation section. Replaced previous step 3 with new step 3 – step 5 in the 6.2. Running Functional Simulation section. Updated Figure 6.5. Add and Reorder Source and Figure 6.8. Simulation Waveform.
Debugging	Added this section.
Design Considerations	Added this section.
Resource Utilization	<ul style="list-style-type: none"> Added the phrase <i>Lattice Radiant Software Version 2023.2</i> to the paragraph of this section. Added the following devices to the paragraph of this section and to Table A.1. Resource Utilization: <ul style="list-style-type: none"> LFCPNX-100-9LFG672I LAV-AT-E70-3LFG1156I LAV-AT-G70-3LFG1156I

Section	Change Summary
References	<p>Updated this section and added the following references:</p> <ul style="list-style-type: none"> • Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039) • AMBA 3 APB Protocol v1.0 Specification • Lattice Radiant Software 2023.1 User Guide • Reveal User Guide for Radiant Software • Lattice Radiant Software web page • Lattice Insights web page for Lattice Semiconductor training courses and learning plans
Technical Support Assistance	Added the <i>Lattice Answer Database</i> information.

Revision 1.3, June 2021

Section	Change Summary
Introduction	<ul style="list-style-type: none"> • Removed last paragraph. • Updated Table 1.1. UART 16550 IP Quick Facts. <ul style="list-style-type: none"> • Revised Supported FPGA Families • Revised Targeted Devices • Revised Lattice Implementation • Revised reference to the Lattice Radiant Software User Guide and removed link.
References	<ul style="list-style-type: none"> • Removed reference to iCE40 UltraPlus web page. • Revised reference to the Lattice Radiant Software User Guide and removed link.

Revision 1.2, June 2020

Section	Change Summary
Introduction	Updated Table 1.1 to add Certus-NX as supported device.
Functional Description	Updated Table 2.2.

Revision 1.1, February 2020

Section	Change Summary
Introduction	Updated Table 1.1. to add LIFCL-17 as targeted device.

Revision 1.0, December 2019

Section	Change Summary
All	Initial release.



www.latticesemi.com