

Introduction

The Lattice WISHBONE UART provides an interface between the WISHBONE UART system bus and an RS232 serial communication channel.

Figure 1 shows the major blocks implemented in the UART in non-FIFO mode. This UART reference design contains a receiver and a transmitter. The receiver performs serial-to-parallel conversion on the asynchronous data frame received from the serial data input SIN. The transmitter performs parallel-to-serial conversion on the 8-bit data received from the CPU. In order to synchronize the asynchronous serial data and to insure the data integrity, Start, Parity and Stop bits are added to the serial data. An example of the UART frame format is shown in Figure 2. In FIFO mode, the RBR (Receiver Buffer Register) in the RXCVER block and the THR (Transmit Hold Register) in the TXMTT block become 16-word-deep FIFOs. In non-FIFO mode, these are simple registers.

Both Verilog and VHDL versions of the reference design are available. Lattice design tools are used for synthesis, place and route and simulation. The design can be targeted to multiple Lattice device families.

Figure 1. UART Block Diagram

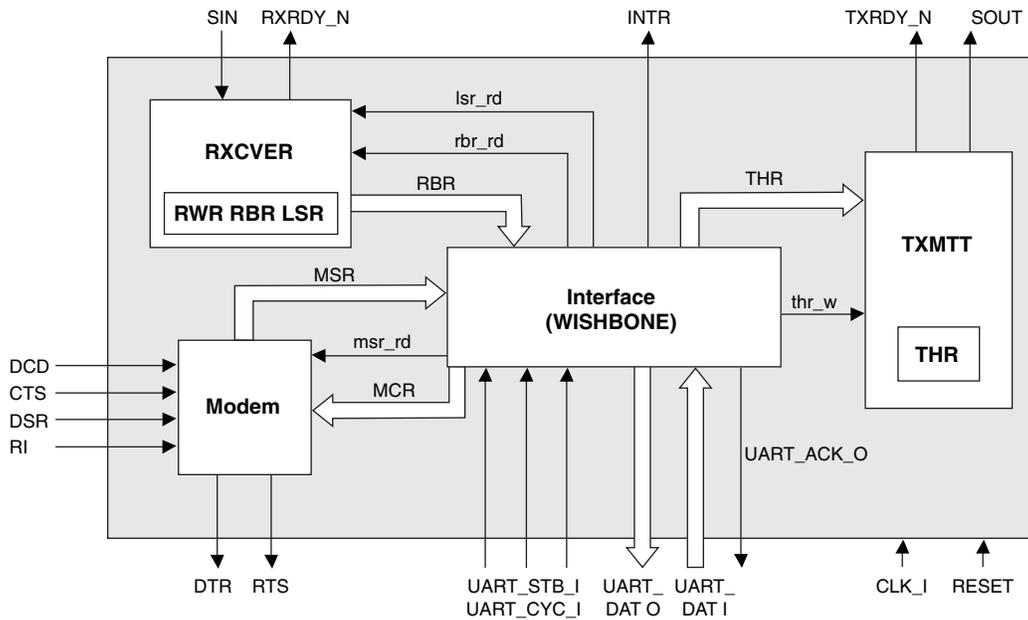
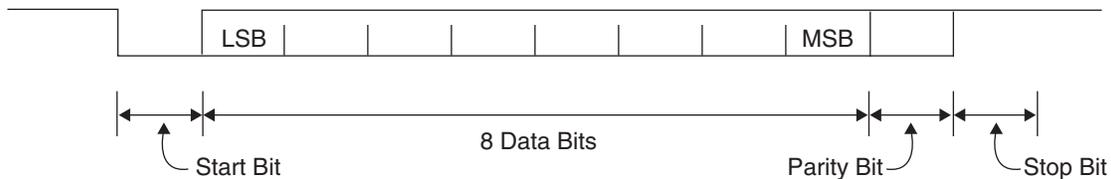


Figure 2. UART Frame Format



© 2010 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Features

The UART includes the following features:

- WISHBONE B.3 interface
- Similar to the NS16450 UART. The optional FIFO mode is implemented in the UART.
- Insertion or extraction of standard asynchronous communication bits (start, stop, and parity) to or from the serial data
- Holding and shifting registers which eliminate the need for precise synchronization between the CPU (WISHBONE interface) and serial data
- A common interrupt line for all internal UART data and error events. Interrupt conditions include receiver line errors, receiver buffer available, transmit buffer empty, and detection of status flag change.
- Fully prioritized interrupt system control
- Optional modem function (not presently supported)
- Support for instantiating the UART multiple times
- Fully programmable serial interface characteristics
- 5-, 6-, 7-, or 8-bit characters
- Even-, odd-, or no-parity bit generation and detection
- 1-, 1.5-, or 2-stop bit generation and detection
- False-start bit detection
- Line-break generation and detection
- Interactive control signaling and status reporting capabilities

Compatibility

This UART implementation supports a higher operating speed and lower access time compared to the NS16450 function. It is functionally equivalent to the NS16450 with the following variations:

1. The Programmable Clock and Baud Rate Generator is not implemented. The UART internal clock Clk16X runs at 16x of the receiver/transmitter baud clock. Users can create this clock from the on-board system clock (Check the Device) or via a separate Divide-by-N clock divisor when implementing in other Lattice CPLD devices to obtain the desired baud clock frequency with the PLD. The associated control registers (Divisor Latch) are also not implemented.
2. The External Crystal Oscillator connections (XTAL1 and XTAL2) are not supported. Alternatively, an external user-specified CLK clock drives the transmitter and receiver clock input either directly or through the PLL.
3. The auxiliary user-defined output pins are not supported, because the user can modify the reference design, and output any internal signals in the Lattice device.
4. The user-accessible Scratchpad register is not supported because all Lattice devices have the built-in User Electronic Signature (UES) register feature.
5. The bidirectional bus D[15:0] is divided into an input bus DIN[15:0] and another output bus DOUT[15:0]. In case the design is used in a larger CPLD design as an embedded module, the DOUT [15:0] should directly drive from the output data multiplexer and not be tri-stated.
6. The internal loop-back capability for on-board diagnostics is not supported because all Lattice devices have ISP™ capability, and can perform in-system test with various patterns.

Theory of Operation

This UART reference design contains a receiver and a transmitter. The receiver performs serial-to-parallel conversion on the asynchronous data frame received from the serial data input SIN. The transmitter performs parallel-to-serial conversion on the 8-bit data received from the CPU.

The design consists of the following modules:

- Uart_core.v/Uart_core.vhd Top level
- Interface.v/Interface.vhd WISHBONE to UART connect
- Rxcver.v/Rxcver.vhd UART receiver
- Transmit.v/Transmit.vhd UART transmitter

Signal Descriptions

Table 1. Signal Descriptions

Signal	Type	Description
Global Reset and Clock Interface		
RESET	Input	Reset signal, active high.
CLK	Input	Clock signal 16 times the receiving/transmitting baud rate clock frequency.
WISHBONE Interface		
UART_ADR_I	Input	Address from WISHBONE interface.
UART_DAT_I	Input	Data input from WISHBONE interface.
UART_STB_I	Input	Strobe input indicating that the slave is selected.
UART_CYC_I	Input	Cycle signal indicating that a valid bus cycle is in progress.
UART_WE_I	Input	Write enable input, 0 = read, 1 = write.
UART_BTE_I	Input	Burst-type extension.
UART_SEL_I	Input	Select input array, which indicates where the valid data is expected on a data bus.
UART_DAT_O	Output	Data output to WISHBONE interface.
UART_ACK_O	Output	Acknowledge output, indicating the termination of a normal bus cycle.
INTR	Output	Interrupt request, active high, used to request service from the CPU whenever one of the following conditions occurs: - Receiver line errors - Receiver buffer available - Transmit buffer empty - Modem status flag change detected
UART Receiver Interface		
SIN	Input	RS232 serial data input
RXRDY_N	Output	Receiver Ready – When there is data in the Receiver Buffer Register, this signal will be low active. Once activated, it will become inactive when there is no character in the Receiver Buffer Register.
UART Transmitter Interface		
TXRDY_N	Output	Transmitter Ready – When there is no data in Transmitter Holding Register, this signal will be low active. Once activated, it will become inactive after the character is loaded into the Transmitter Holding Register.
SOUT	Output	RS232 serial data output.

Register Descriptions

Table 2. Register Map

Register Name	Offset	31-16	15-8	7	6	5	4	3	2	1	0
RBR (Receive buffer Register)/ THR (Transmit Holding Register)	0x8	—	—	D7	D6	D5	D4	D3	D2	D1	D0
IER (Interrupt Enable Register)	0x1			0	0	0	0	MSI	RLSI	THRI	RBRI
IIR (Interrupt Identification Register)	0x2			0	0	0	0	0	ID1	ID0	STAT
LCR (Line Control Register)	0x3			0	SB	SP	EPS	PEN	STB	WLS1	WLS0
LSR (Line Status Register)	0x5			0	TEMT	THRE	BI	FE	PE	OE	DR
DIV (Baud Rate Divisor Register)	0x7			Divisor bits[15:0].							

Table 3. Interrupt Enable Register (IER, Addr =001)

7	6	5	4	3	2	1	0
0	0	0	0	MSI	RLSI	THRI	RBRI

RBRI: Receiver Buffer Register Interrupt (1 = Enable, 0 = Disable)

THRI: Transmitter Hold Register Interrupt (1 = Enable, 0 = Disable)

RLSI: Receiver Line Status Interrupt (1 = Enable, 0 = Disable)

MSI: MODEM Status Interrupt (1 = Enable, 0 = Disable)

Table 4. Interrupt Identification Register (IIR, Addr=010)

7	6	5	4	3	2	1	0
0	0	0	0	0	ID1	ID0	STAT

Table 5. Four Prioritized Interrupt Levels and Sources

Level	ISR Bit [3:0]	Source of Interrupt	Interrupt Reset Control
None	Xxx1	None	None
Highest	0110	LSR error Flags (OE/PE/FE/BI)	Reading LSR
Second	0010	LSR Receiver data ready flag (DR)	Reading RBR
Third	0010	LSR flag THR empty (THRE)	Reading IIR or Writing THR

The Line Control Register (LCR) is used to specify the asynchronous data communication format.

Table 6. Line Control Register

7	6	5	4	3	2	1	0
0	SB	SP	EPS	PEN	STB	WLS1	WLS0

WLS1:0 - Word Length Select (Bits 1-0):

- 00 - 5 bit
- 01 - 6 bit
- 10 - 7 bit
- 11 - 8 bit

STB - Stop Bit Length (Bit 2):

- 0 - 1 stop
- 1 -
 - WLS1:0 = 00: 1.5 stop
 - WLS1:0 = 01: 2 stop
 - WLS1:0 = 10: 2 stop
 - WLS1:0 = 11: 2 stop

Parity enable:

- 0 - Parity bit disable
- 1 - Parity bit enable

Stick parity:

- 0 - Stick parity disable
- 1 -
 - When PEN, EPS, or SP is set (that is, 1), parity is sent or checked for 0.
 - When PEN or SP is set, EPS is clear, parity is sent or checked for 1.

Tx break:

- 0 - Disable break assertion
- 1 - Assert break. SOUT is driven low active (break character) as long as this bit is 1. It has no effect on transmitter logic.

Table 7. Line Status Register

7	6	5	4	3	2	1	0
0	TEMP	THRE	BI	FE	PE	OE	DR

DR: Receiver Data Ready

DR indicates status of RBR. It will be set to logic 1 when RBR data is valid and will be reset to logic 0 when RBR is empty. When line errors (OE/PE/FE/BI) happen, DR will also be set to logic 1 and RBR will be updated to reflect the Data bits portion of the frame. Pin RxRDYn is actually a complement of this bit.

OE: Overrun Error

This bit will be set when the next character is transferred into RBR before the previous RBR data is read by the CPU. Even though DR will still be 1 when OE is set to logic 1, the previous frame data stored in RBR which is not read by the CPU is trashed and can't be recovered.

PE: Parity Error

This bit will be set to logic 1 only when the Parity is enabled and the Parity bit is not at the logic state it should be. For Even Parity, the Parity bit should be 1 if an odd number of 1s in the Data bits is received; otherwise, the Parity bit should be 0. For Odd Parity, the Parity bit should be 1 if an even number of 1s in the Data bits is received; otherwise, the Parity bit should be 0. For Stick Parity '1', the Parity bit should be 1. For Stick Parity '0', the Parity bit should be 0.

FE: Framing Error

FE will be reset to logic 0 whenever SIN is sampled high at the center of the first Stop bit, regardless of how many Stop bits the UART is configured to.

BI: Break Interrupt

BI will be set to logic 1 whenever SIN is low for longer than the whole frame (the time of Start bit + Data bits + Parity bit + Stop bits), not at the SIN rising edge where Break is negated. If SIN is still low after BI is reset to logic 0 by reading LSR, BI will not be set to logic 1 again. Since Break is also a Framing error, FE will also be set to 1 when BI is set.

THRE: THR Empty

THRE will be set to logic 1 whenever THR is empty which indicates that the transmitter is ready to accept new data to transmit. Pin TxRDYn is actually a complement of this bit.

TEMT: Both THR and TSR are Empty

This bit will be set to logic 1 when THRE is set to 1 and the last Data bit in the TSR is shifted out through SOUT.

The four error flags (OE, PE, FE and BI) of LSR will be reset to logic 0 after a LSR read.

Design Module Description

UART Clock Frequencies

The UART has a single clock input. The UART uses the WISHBONE CLK_I input frequency to transfer data between the LatticeMico8™ microprocessor and the UART control and status registers. CLK_I is also used to transfer and receive data on the UART's SOUT and SIN pins.

Receiver

The receiver (RXCVER) section contains an 8-bit receiver buffer register (RBR) and receiver shift register (RSR). In FIFO mode, the RBR is a 16-word-deep FIFO.

Since the serial frame is asynchronous to the receiving clock, a high-to-low transition of the SIN pin will be treated as the start bit of a frame. However, in order to avoid receiving incorrect data due to SIN signal noise, the False Start Bit Detection feature is implemented in the design. This requires the start bit to be low at least 50% of the receiving baud rate clock cycle. Since the internal clock Clk16X is 16 times the receiving/transmitting baud rate clock frequency, the start bit must be low at least 8 Clk16X clocks to be considered as a valid start bit.

Once a valid 8 Clk16X clocks start bit is received, the data bits and parity bit will be sampled every 16 Clk16X clocks (the receiving baud rate). If the start bit is exactly 16 Clk16X clocks long, each of the following bits will be sampled at the center of the bit itself. LSR will be updated to show the received frame status when any of the line errors (overrun error, parity error, framing error, break) is detected.

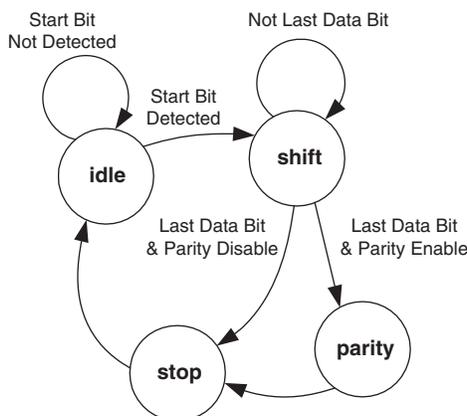
Whenever the Framing error is detected, the UART assumes that the error was due to the start bit of the following frame and tries to resynchronize it. To do this, it samples the start bit twice based on the Clk16X clock. If both sam-

ples of the SIN are low, the UART will take in the following frame's data bits after the 8 Clk16X clock's start bit is sampled. The resynchronization will not occur for the framing error caused by the break.

When the data is available in RBR, the RxRDYn will be low active, and the Receiver Data available flag (DR) in LSR will be set to logic 1 to inform the CPU that the data is ready to be read.

The behavior of the receiver is controlled by the FSM shown in Figure 3.

Figure 3. Receiver State Machine



The Receiver State Machine includes the following states:

- **Idle:** When reset or after the stop state, the receiver FSM is reset to this state. When in this state, it waits for SIN to be changed from high to low and stay low for half a bit duration to be considered a valid start bit. Once a valid start bit is detected, the FSM switches to the “shift” state.
- **Shift:** When the FSM is in this state, it waits one bit duration for each data bit to shift into the RSR. After the last data bit is shifted in, the FSM switches to the “parity” state if parity is enabled. Otherwise, it switches to the “stop” state.
- **Parity:** When the FSM is in this state, it waits for one bit duration and then samples the parity bit. Once the parity bit is sampled, the FSM switches to the “stop” state.
- **Stop:** Whether the stop-bit length is configured to be 1, 1.5, or 2 bits long, the FSM always waits for one bit duration and then samples the stop bit. As long as a logic 1 is sampled at the stop bit, the framing error flag (FE) in LSR is not set. The receiver does not check whether the stop bit is in the right length as configured. The FSM switches back to the “idle” state after sampling the stop bit.

Transmitter

The serial transmitter (TXMITT) section consists of an 8-bit transmitter hold register (THR) and transmitter shift register (TSR) when the UART is in non- FIFO mode. When the UART is in FIFO mode, THR is a 16-word-deep FIFO.

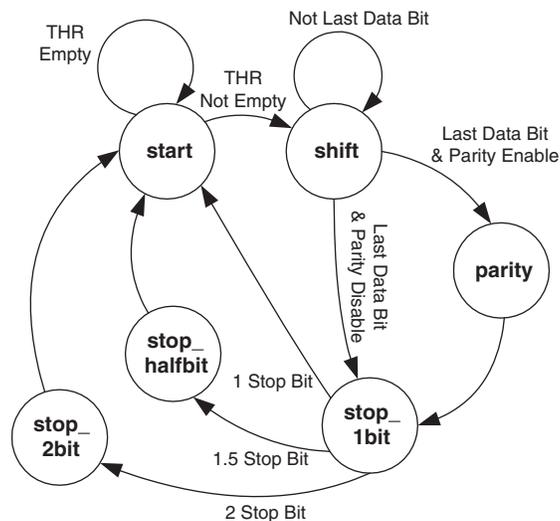
The UART provides two methods of indicating the status of THR: a txrdy_n output signal or a Transmit Holding Register empty (THRE) flag in the line status register (LSR). When THR is empty, the txrdy_n pin becomes low active, and the THR empty flag in LSR is set to a logic 1. A write to the THR interferes with a transmission in progress if THRE is active or trdy_n is de-asserted. After the data is loaded in THR, the THR empty flag in LSR is reset to logic 0, and the txrdy_n pin goes inactive high.

The serial data transmission is automatically enabled after the data is loaded into THR. First, a start bit (logic 0) is transmitted and the data in THR is automatically parallel-loaded to TSR. The data bits are shifted out of TSR, followed by the parity bit, if parity is enabled. Finally, the stop bit (logic 1) is generated to indicate the end of the frame. After a frame is fully transmitted, another frame is transmitted immediately if THR is not empty. This automatic

sequencing causes the frames to be transmitted back to back, which increases the transmission bandwidth. The SOUT pin is held high when no transmission is in progress.

The behavior of the transmitter is controlled by the finite state machine (FSM) as shown in Figure 4.

Figure 4. Transmit State Machine



The transmitter state machine includes the following states:

- **Start:** When the UART is reset, the FSM transmitter is reset to this state. When in this state, the transmitter waits to assert the start bit. A start bit is asserted as soon as THR is not empty. Once a low SOUT (start bit) is asserted, the FSM switches to the shift state.
- **Shift:** When the FSM is in this state, it waits for the last (most significant) data bit to be shifted out. After the last data bit is shifted out, the FSM switches to the “parity” state if parity is enabled. Otherwise, it switches to the stop_1bit state.
- **Parity:** When the FSM is in this state, the last data bit is still in transmission. When the transmission is complete, the FSM asserts the parity bit. Once the parity bit is asserted, the FSM switches to the stop_1bit state.
- **Stop_1bit:** Whether the stop bit is configured to be 1, 1.5, or 2 bits long, the FSM always switches to this state, waits for a baud clock cycle, and then asserts the stop bits. For one stop bit, the FSM switches back to the start state and waits to assert the start bit of another frame. For 1.5 stop bits, it switches to the “stop_halfbit” state and stays there for just half a baud clock cycle before switching to the start state. For two stop bits, it switches to the stop_2bit state and then switches back to the “start” state. The stop bit is asserted at the time that the FSM leaves the stop_1bit state.
- **Stop_halfbit:** This state is for 5-bit data bits with a 1.5 stop bit. The FSM stays in this state for only half a baud clock cycle and then switches to the start state.
- **Stop_2bit:** When the FSM is in this state, the first stop bit is in transmission. It waits for a baud clock cycle and then asserts the second stop bit and switches to the start state.

Interrupt

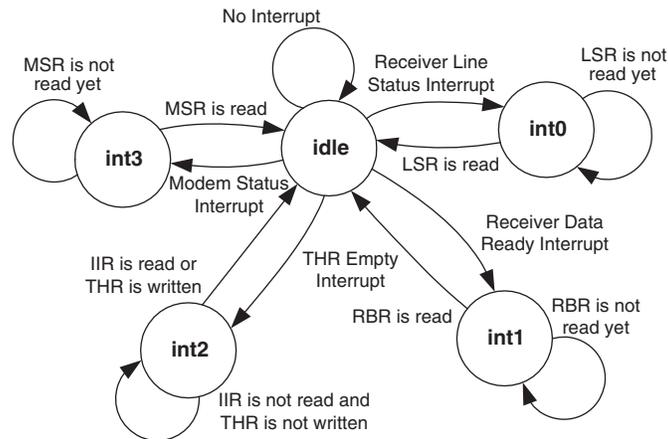
The common interrupt request pin (INTR) goes to high active when any interrupt conditions are matched and enabled by the interrupt enable register (IER).

The UART prioritizes interrupts into four levels to minimize external software interaction, and it records these in the interrupt identification register (IIR). The four levels of interrupt conditions in order of priority are receiver line status, received data ready, transmitter holding register empty, and modem status_register. Performing a read cycle on IIR

freezes all interrupts and indicates the highest priority pending interrupt to the CPU. No other interrupts are acknowledged until the pending interrupt is serviced. Whenever the IIR is read, the current pending interrupt is cleared. Any pending lower-priority interrupt becomes visible in the IIR after the previous IIR read.

The behavior of the interrupt is controlled by the FSM, as shown in Figure 5.

Figure 5. Interrupt State Machine



The interrupt state machine has the following states.

- **Idle:** When the UART is reset, the interrupt FSM is reset to this state. When in this state, it waits for the enabled interrupt conditions to be true, and then it switches to the interrupt state with the highest priority.
- **int0:** The FSM switches to this state when the highest-priority level interrupt occurs. It stays at this state until the LSR is read.
- **int1:** The FSM switches to this state when the second-priority level interrupt occurs. It stays at this state until the RBR is read.
- **int2:** The FSM switches to this state when the third-priority level interrupt occurs. It stays at this state until the IIR is read or after THR is written.
- **int3:** The FSM switches to this state when the fourth-priority level interrupt occurs. It stays at this state until the MSR is read.

The interrupt continues to be generated as long as the corresponding enable bit in IER is set and the corresponding interrupt condition is matched.

WISHBONE

The LatticeMico8 microprocessor interfaces to the UART control and status registers by using the WISHBONE bus. The registers are accessed using classic mode data cycles. Each read and write is performed as a single cycle transfer (that is, not in burst mode).

All of the control registers are 32-bit aligned, and the SEL_I inputs are ignored. When the control registers are read or written to, only the least significant eight bits are valid, with the exception of the baud-rate divisor register, which is 16 bits.

When the LatticeMico8 microprocessor initiates a bus cycle, it starts by asserting STB_I and CYC_I. The UART responds to the cycle with UART_ACK_O at the first rising edge following the STB_I and CYC_I assertion. The UART_ACK_O stays active for a single CLK_I cycle, terminating the transfer and accepting or returning data.

Testbench Description

The Verilog files needed for simulation include:

- `uart_tb_transmit.v/uart_tb_receive.v` – Testbench for transmit and receive
- `transmit_test.v` – Test case for transmitter test
- `receive_test.v` – Test case for receiver test
- `Eval_params.v` – Parameter file for the simulation
- `Transmit_test` – This UART transmitter test writes a known pattern to the Transmit Hold Registers (THR) and verifies that the SOUT line transmits this pattern correctly.

There are a total of 20 tests with different combinations of data bits, parity and stop bits. Each test writes four different data patterns.

- `Receive_test` – Similar to transmitter test, the receiver test sends a known pattern that is captured at the receiver side SI N line. Frame and parity error generation is tested by inserting known error conditions into the testbench.

The VHDL simulation files include:

- `uart_tb_transmit.vhd` – This UART transmitter test writes a known pattern to the Transmit Hold Registers (THR) and verifies that the SOUT line transmits this pattern correctly.
- `uart_tb_receive.vhd` – Similar to the transmitter test, the receiver test sends a known pattern that is captured at the receiver-side SI N line. Frame and parity error generation is tested by inserting known error conditions into the testbench.

Design Flow

Lattice design tools are used for synthesis, place and route and simulation. In addition to the place and route/fitter engine, the Lattice ispLEVER® design tool includes Synplify®/SynplifyPro® from Synplicity® and Active-HDL® from Aldec®. The details of the design flow can be found in the README.txt file that comes with the reference design.

Timing Specifications

The timing diagrams below show an 8-bit pattern “55” written to the Transmit Hold Register. This in turn initiates a UART transmit on the SOUT line after reading TEMT and once the transmit empty flag is set in the Line Status Register.

Figure 6. UART Transmit Timing Diagram

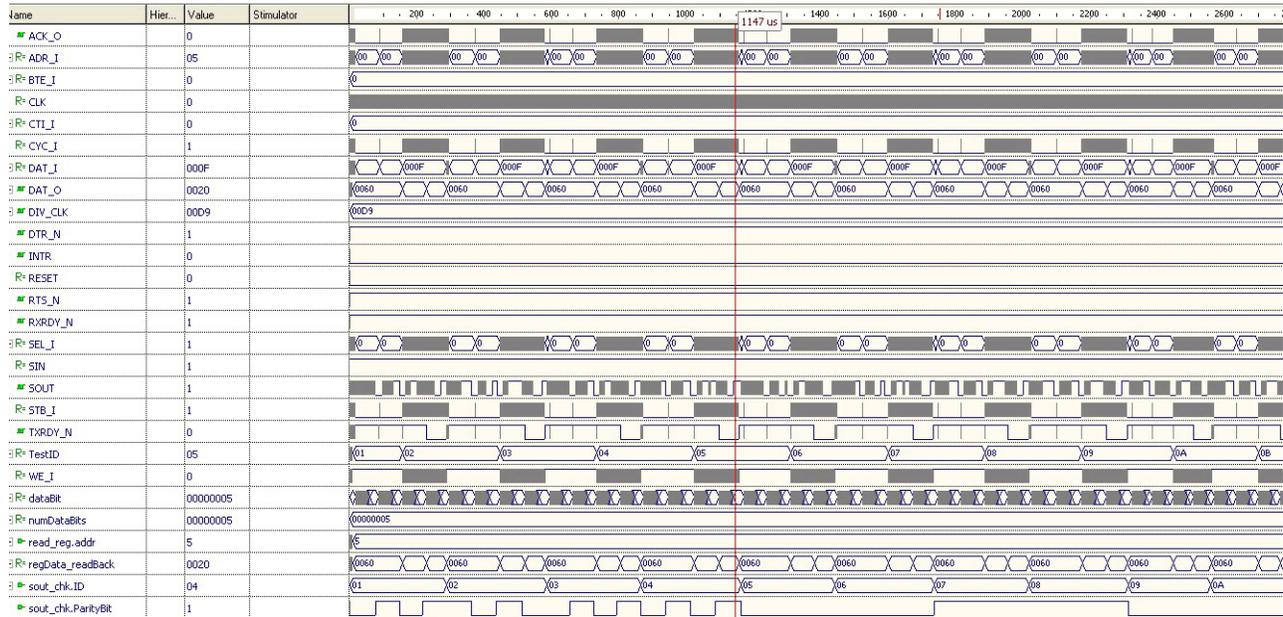
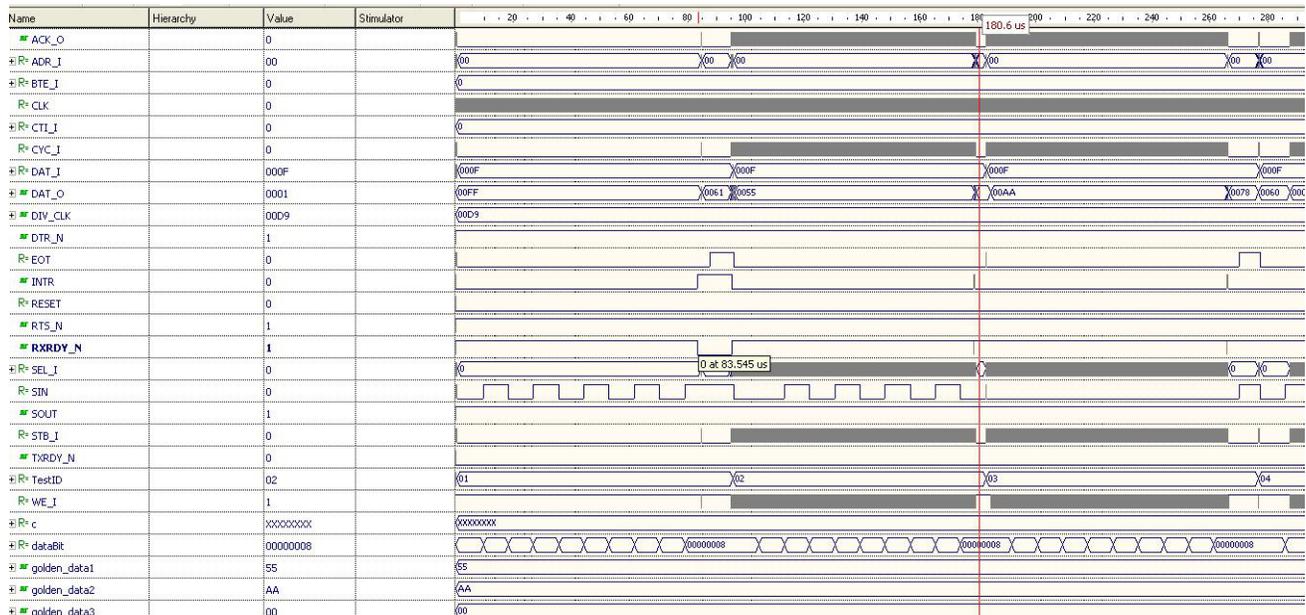


Figure 7 is the timing simulation diagram showing the Receiver frame.

Figure 7. UART Receive Timing Diagram



Implementation

Table 8. Performance and Resource Utilization¹

Device Family	Language	Speed Grade	Utilization (LUTs)	f _{MAX} (MHz)	I/Os	Architecture Resources
MachXO™ 1	Verilog	-5	253	>60	52	N/A
	VHDL	-5	256	>60	52	N/A
MachXO2™ 2	Verilog	-4	274	>60	52	N/A
	VHDL	-4	267	>60	52	N/A
LatticeXP2™ 3	Verilog	-5	323	>60	52	N/A
	VHDL	-5	312	>60	52	N/A

1. Performance and utilization characteristics are generated using LCMXO2280C-5T144C, with Lattice ispLEVER® 8.1 SP1 and Lattice Diamond™ 1.1 software. When using this design in a different device, density, speed, or grade, performance and utilization may vary.
2. Performance and utilization characteristics are generated using LCMXO2-1200HC-4TG144CES, with Lattice ispLEVER 8.1 SP1 and Lattice Diamond 1.1 software. When using this design in a different device, density, speed, or grade, performance and utilization may vary.
3. Performance and utilization characteristics are generated using LFXP2-5E-5TN144C, with Lattice ispLEVER 8.1 SP1 and Lattice Diamond 1.1 software. When using this design in a different device, density, speed, or grade, performance and utilization may vary.

References

- [LatticeMico32 UART](#)
- [MachXO Family Data Sheet](#)
- [LatticeXP2 Family Data Sheet](#)
- [LatticeMico32 Processor Reference Manual](#)

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
 +1-503-268-8001 (Outside North America)
 e-mail: techsupport@latticesemi.com
 Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
August 2009	01.1	Added VHDL source file and testbench.
December 2009	01.2	Added two VHDL modules to complete the VHDL design.
		Added support for LatticeXP2 device family.
November 2010	01.3	Added support for MachXO2 device family.
		Updated for Lattice Diamond software support.
		Updated the VHDL test bench.