International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 3, Issue 8, August 2014

An Advanced Universal Asynchronous Receiver Transmitter (UART) Design & Implementation By Using VERILOG

G. Bhanu Priya[1], B. Lakshman Murthy[2] , P. Pragathi[3]
1- PG student, QIS Institute of Technology, Ongole, India.
2- Assistant Professor, QIS Institute of Technology, Ongole, India.
3- Assistant Professor, QIS Institute of Technology, Ongole, India.

Abstract—
The proposed paper illustrate the advanced technique for implementation of UART using FPGA with the help of Verilog description language. UART is a serial communication protocol which allows the full duplex communication in serial link, it is an essential to computers and allows them to communicate with low speed peripheral devices, such as the keyboard, the mouse, modems etc. Thus, the UART or Universal Asynchronous Receiver/ Transmitter is the most important component required in serial communication. In This paper the UART main components namely transmitter, receiver and baud rate generator which is nothing but the frequency divider for fast transferring the data are synthesized on FPGA kits with Spartan3E.

1. Introduction

The UART consists of three main components namely transmitter, receiver and baud rate generator which is nothing but the frequency divider**.** The universal asynchronous receiver/transmitter is shortened as UART. The word "asynchronous" indicates that UART recover character timing information from the data stream, using designated "start" and "stop" bits to indicate the framing of each character. The history of the first UART -like devices was rotating mechanical commutates, these sent 5-bit codes for mechanical teletypewriters, their after replaced more code 8 bit,10 bit FIFO etc. The UART is used in between the slow and the fast peripheral devices for example: computer and motor drives or peripherals or in between the controller and LCD projectors, due to this reason, UART is used mostly for the short distance, low speed and is of low cost.

This paper uses the Verilog description language to adopt the core functions of UART and assimilate them into a FPGA chip. It has three main components i.e. transmitter, receiver and the BRG (baud rate generator). Hear we are using the state machines for transmitter and receiver due to this our proposed UART becomes more stable, reliable, compact and less complex for serial data communication. Due to which, the consumption of LUTs, slice flip flops or in short the area consumption of the chip becomes less. We have also tested our design for the errors which arises during transmission of data to analyze that our output of the receiver is free from the errors or not. Thus we are tested it for parity and CRC errors.

2. Differences of VHDL and Verilog

VHDL means very high speed Integrated circuit hardware description language and Verilog - means verify the logic. Verilog is an alternative language to VHDL for specifying RTL for logic synthesis VHDL similar to Ada programming language in Syntax Verilog similar to C/Pascal programming language. VHDL is a strongly and richly typed language. Derived from the Ada programming language, its language requirements make it more verbose than Verilog. The additional verbosity is intended to make designs self-documenting. Also, the strong typing requires additional coding to explicitly convert from one data type to another. VHDL does not define any simulation control or monitoring capabilities within the language. These capabilities are tool dependent. Due to this lack of language-defined simulation control commands and also because of VHDL's user defined type capabilities, the VHDL community usually relies on interactive GUI environments for debugging design problems.

Verilog is a weakly and limited typed language. Its heritage can be traced to the C programming language and an older HDL called Hilo. All data types in Verilog are predefined in the language. Verilog recognizes that all data types have a bit-level representation. The supported data representations (excluding strings) can be mixed freely in Verilog. Simulation semantics in Verilog are more ambiguous than in VHDL. This ambiguity gives designers more flexibility in applying optimizations, but it can also (and often does) result in race conditions if careful coding guidelines are not followed. It is possible to have a design that generates different results on different vendors' tools or even on different releases of the same vendor's tool. Verilog defines a set of basic simulation control capabilities (system tasks) within the language.

ISSN: 2278 – 909X

3. EXPERIMENTAL DETAILS

The three main components of the UART such as transmitter, receiver and the baud rate generator are described below:

3.1. UART TRANSMITTER MODULE:

The function of the transmitter module is to convert the 8 bit data into the single bit data. In this module, when our load signal is high the data_in is stored into the holding register. The holding register is shifted to the intermediate register with the start bit of zero and this this intermediate register is have 9 bits. Once the shift signal is high the least significant bit of the intermediate register. After Tx_STA signal is high the START bit comes at the output of the transmitter and served as the input to the receiver. When entire data has been sent the transmitter provides a parity bit which is served as input to the receiver. The transmitter generates the remainder which is given as the input to the receiver and receiver provides us the output.

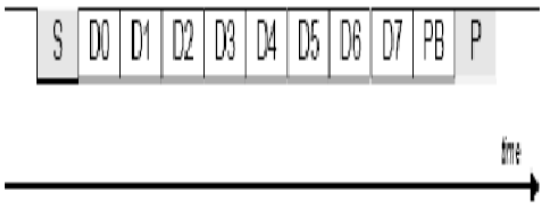


Figure 1 Basic UART Transmission Flow

When a word is given to the UART for Asynchronous transmission, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. After the Start Bit, the individual bits of the word of data are sent, with the Least\ Significant Bit (LSB) being sent first. Each bit in the transmission is transmitted for exactly the same amount of time as all of the other bits, and the receiver "looks" at the wire at approximately halfway through the period assigned to each bit to determine if the bit is a 1 or a 0 as shown on Figure 1. For example, if it ta

kes two seconds to send each bit, the receiver will examine the signal to determine if it is a 1 or a 0 after one second has passed, then it will wait two seconds and then examine the value of the next bit, and so on. Then at least one Stop Bit is sent by the transmitter. Because asynchronous data is "self-synchronous", if there is no data to transmit, the transmission line can be idle.

3.2. BAUD RATE GENERATOR of UART:

Baud is a measurement of transmission speed in asynchronous communication. Because of advances in modem communication technology, this term is frequently misused when describing the data rates in newer devices. Traditionally, a Baud Rate represents the number of bits that are actually being sent over the media, not the amount of data that is actually moved from one UART device to the other. The Baud count includes the overhead bits Start, Stop and Parity that are generated by the sending UART and removed by the receiving UART. This means that seven-bit words of data actually take 10 bits to be completely transmitted.

The baud rate of a data communications system is the number of symbols per second transferred. A symbol may have more than two states, so it may represent more than one binary bit (a binary bit always represents exactly two states). Therefore the baud rate may not equal the bit rate, especially in the case of recent modems, which can have (for example) up to nine bits per symbol. Usually the baud rate of a modem will not be equal to the bit rate and is of no interest to the end user--only the data rate, (in bits per second) is.

Baud Rate =Clock Frequency/ (Sampling Rate) * (Divisor)

Baud rate refers to the number of signal or symbol changes that occur per second. A symbol is one of several voltage, frequency, or phase changes.

What baud really refers to is modulation rate or the number of times per second that a line changes state, this is not always the same as bits per second (BPS). If two serial devices are connected together using direct cables then baud and BPS are in fact the same. Thus, if you are running at 19200 BPS, then the line is also changing states 19200 times per second. But when considering modems, this isn't the case. As modems transfer signals over a telephone line, the baud rate is actually limited to a maximum of 2400 baud. This is a physical restriction of the lines provided by the phone company. The increased data throughput achieved with 9600 or higher baud modems is accomplished by using sophisticated phase modulation, and data compression techniques.

Once the start bit has been sent, the transmitter sends the actual data bits. There may either be 5, 6, 7, or 8 data bits, depending on the number you have selected. Both receiver and the transmitter must agree on the number of data bits, as well as the baud rate. Almost all devices transmit data using either 7 or 8 data bits.

ISSN: 2278 – 909X

When notice that when only 7 data bits are employed, you cannot send ASCII values greater than 127. Likewise, using 5 bits limits the highest possible value to 31. After the data has been transmitted, a stop bit is sent. A stop bit has a value of 1 - or a mark state - and it can be detected correctly even if the previous data bit also had a value of 1. This is accomplished by the stop bit's duration. Stop bits can be 1, 1.5, or 2 bit periods in length.

3.3. UART RECEIVER MODULE :

The receiver of micro-UART is composed of a control state-machine, de-serializer, and support logic. The main goal of the receiver is to detect the start-bit, then de-serialize the following bit stream, detect the stop-bit, and make the data available to the host. Figure 3 illustrates the functional block diagram of the receiver. The design is minimalist, and no error checking logic is present by default. All of these features are to become user enhancements.

The signal uart_clk is 16xBaud-Rate generated by the baud rate generator u_baud.v This clock is used to drive all of the clock within the receiver module. The incoming data uart_dataH is fed to the dual-rank synchronizer before feeding to de-serializer. Note that this synchronizer is absolutely essential since the data present on uart_dataH is synchronous to the transmitter's clock, and not on the receiver's clock.

The de-serializer is a simple serial-to-parallel shift register. It has 1 control input shiftH from the state machine. When this signal is active high, the de-serializer shifts the data over by 1 bit. The default shift register width is 8 bits. Note that the LSB is shifted infirst.

The received bit counter is used to keep track of the number of data bits cumulated so far. when this count becomes the pre-set limit (word_len parameter from inc.h), then the state-machine stops accepting more data bits. This counter has 2 control inputs: countH and rst CountH. When the former is active high, the counter is advanced by 1, when the latter is active high; the counter is cleared to 0. Note that this is a synchronous counter.

4. ANALISATION OF RESULTS

Transmitter:

The state machine is a simple 5 state Mealy type. Figure 2 illustrates the state flow. Just as in the case of the receiver, 1 baud tick is equal to 7 uart_clk ticks. Upon system reset, the state machine defaults to x_IDLE state. In this state, the state machine idles for as long as no transmit command is given. But when Tx_STS become active high (for 1 uart_tick), then the serializer is loaded and the state machine transitions to x_START state.

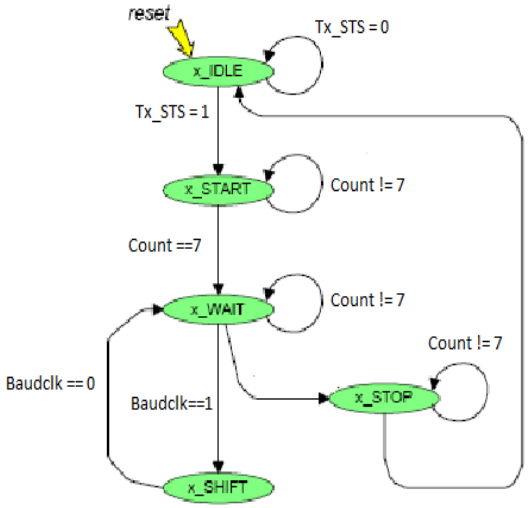


Figure 2 State Machine of UART Transmitter

In x_START state, the uart_xmitH mux is set to 1'b0 (start bit), and 1 baud tick is waited (16 uart_ticks) before transitioning to x_WAIT state. In x_WAIT state, the uart_xmitH mux is set to point to the shift register, and 1 baud tick is waited. After the wait is complete, if all bits (WORD_LEN) have been transmitted then the state machine transitions to x_STOP state, otherwise it goes to the x_SHIFT state. In the x_SHIFT state, the shift-register is shifted by 1 bit, and transitions to x_WAIT state.
In x_STOP state, the uart_xmitH mux is set to 1'b1 (stop bit), 1 baud tick is waited and then transitions to x_IDLE state.

Receiver:

The state-machine is a simple 5 state, Mealy type (output is function of present state and input). Figure 3 illustrates the state flow. The state-machine ties all of the functional units previously described. Upon system reset, the state machine defaults to r_START state. In this state, the state-machine looks for the start-bit. This condition is detected by the transition of the incoming data (which at idle is logic 1) to a logic 0. Once the start-bit is detected, it transitions to r_CENTER state.

ISSN: 2278 – 909X

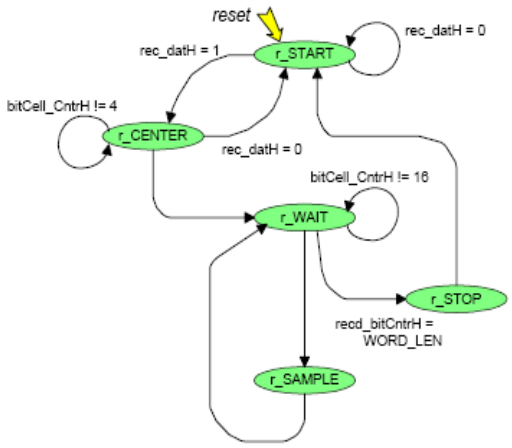


Figure 3 State Machine of UART Receiver

In r_CENTER state, the state-machine wait for ½ bitcell in order to find the bit cell center. A bit-cell is 1 baud "tick" and correspond to16 uart_clk ticks. So ½ bit-cell corresponds to 8 uart_ticks. The bit cell counter is used to generate this delay. The reason for waiting for 4 uart_ticks (not 8), is that the synchronizer uncertainty adds upto 2 uart_ticks. Additionally, counter overhead also adds upto 2 ticks. Although it is theoretically optimum to sample the incoming data at it cell center, the micro-UART design allows some margin of error (due to the fact that the data is oversampled at x16). Once the bit-cell center is found (after having waited 4 uart_ticks), if the state of the rec_dataH (synchronized incoming data) is still low, then the state machine transitions to r_WAIT state. If rec_dataH is high, then this is not a valid start bit, so the state machine transitions back to r_START state.

A spurious noise in the UART data line can produce this kind of effect. The r_WAIT state simply waits for 1 baud tick (16 uart_ticks). Note that the previous state, r_CENTER, aligned the incoming data to the center of the start-bit bit-cell. Once 1 baud tick is waited, the incoming data can be sampled into the de-serializer. If all WORD_LEN bits have been sampled, then the state machine transitions to r_STOP state, otherwise, it transitions to r_SAMPLE state.

In r_SAMPLE state, the state of rec_dataH is sampled into the de-serializer. In r_STOP state, the state of rec_dataH is "sensed" to check for logic 1. This bit is not sampled into the de-serializer. Note that no error condition is generated should the expected stop bit is not active high. Before transitioning to r_START state, a status signal is generated, rec_readyH, to indicate that a valid data is available for reading. Notice that rec_readyH is flopped, or piped, before being made available. This practice reduces critical path timing constraints.

5. Simulation Results:

Transmitter:

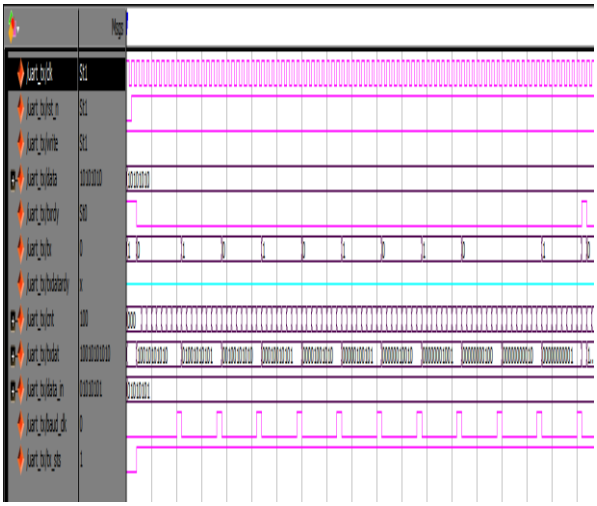


Figure 4 Simulation Result for Transmitter

The description of the above Figure is simulation result of UART- Transmitter using Verilog. Here given 8bit date 10101010 and transmitted by controllers like Clock (clk), Reset(rst) and write pins. We can able to see output result at Tx pin.

Receiver:

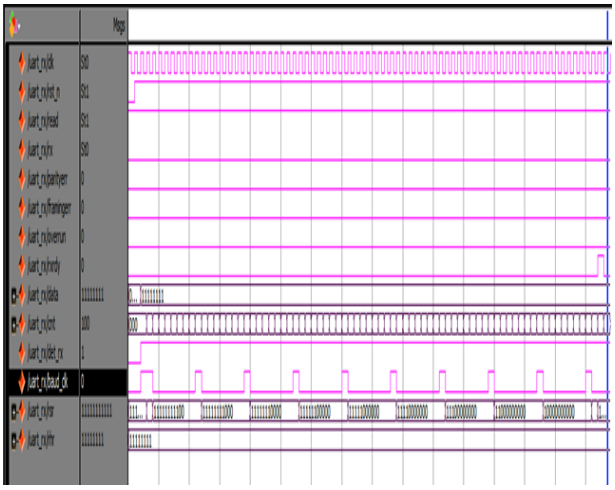


Figure 5 Simulation Result for UART Receiver

The description of the above Figure is simulation result of UART- Transmitter using Verilog. Here receiving 8bit date 10101010 by controllers like Clock (clk), Reset(rst), Read and Rx pins. We can able to see output result at Rx_data pin.

ISSN: 2278 – 909X

Top Level UART:

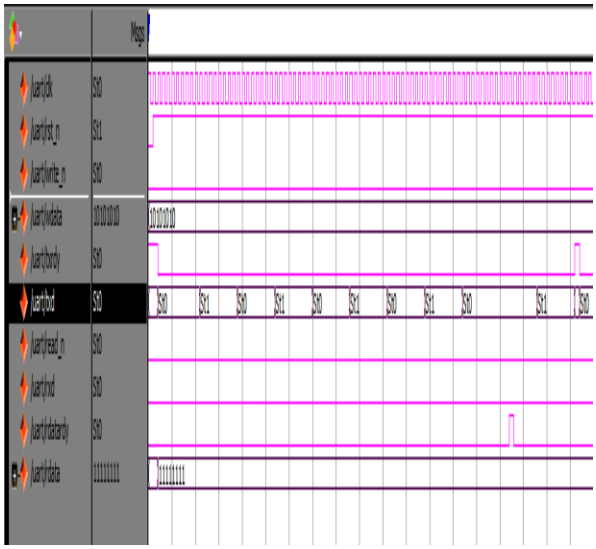
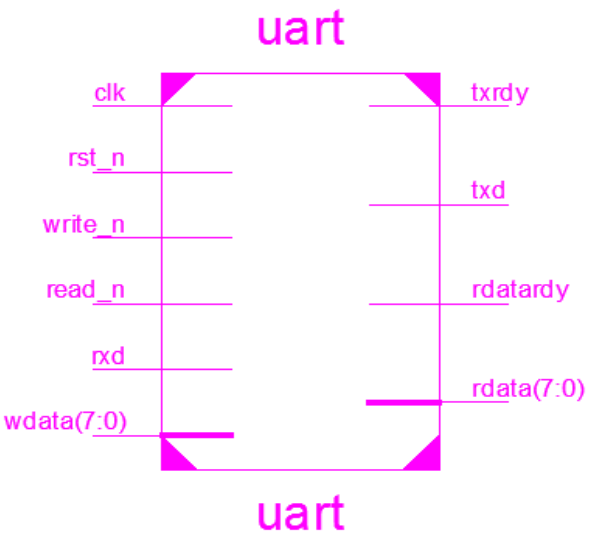


Figure 6 Simulation Result for UART

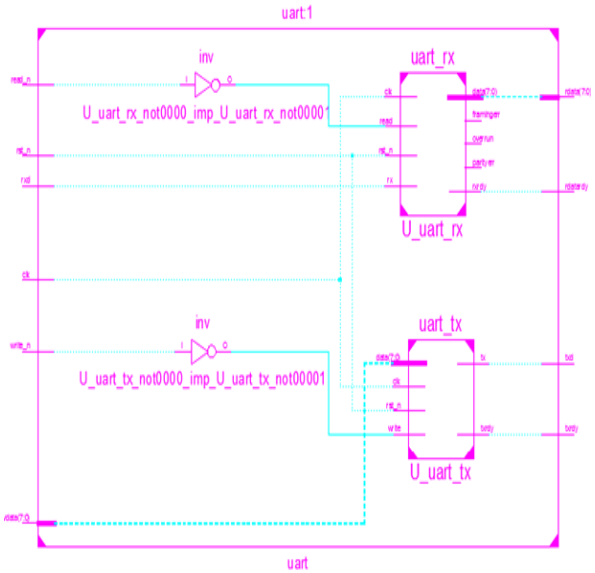
The description of the above Figure is simulation result of UART using Verilog. Here transmitted 8bit date 10101010 by controllers like Clock (clk), Reset (rst), Write pins and Its Received by Rx_data by the read pin. We can able to see output result at Rx_data pin. When the data is ready to transmit Tx_data is high. Total 8bit data is received by receiver Rx_data will be High.

6. Synthesis Result:

Top Level UART:



Synthesis Result for Transmitter and Receiver:



The synthesis result contains a table 1 which shows the comparison between the old research and proposed UART. While doing comparison we will find that our research shows great significance as our all the parameters are consuming less area and consumes less power to operate.

Table 1

Serial No.	Comparison between old research and proposed UART		
	Parameters	Virtex4v	Spartan-3E
1	Number of slices	63	32
2	LUTs	102	73
3	GCLKs	2	1
4	Slice Flip-Flop	46	24
5	Maximum Freq	284.075 MHz	107.082MHz

CONCLUSION

In this we use Verilog description language for obtaining the modules of Universal asynchronous receiver and transmitter. After studying the comparative analysis between Verilog and VHDL we conclude that there is a difference in between the number of slices, LUTs, GCLKs and the maximum frequency. The results are reliable and quiet stable and has high flexibility with high integration. We can see in table no 1.

ISSN: 2278 – 909X

Reference

[1] Himanshu Patel, Sanjay Trivedi, R.Neelkanthan, V. R. Gujraty " a robust uart architecture based on recursive running sum filter for better noise Performance".

[2] Intel® MCS-51 microcontroller Family User's manual

[3] Synopsys DesignWare® DW8051 Macro Cell Databook

[4]M.Delvai,U.Eisenmann,W. Elmenreich,"Intelligent UARTModule forReal-Time Applications"First Workshop on Intelligent Solutions in Embedded Systems(WISES),pages177-185, Vienna,Austria, June 2002

[5] Xilinx® Virtex FPGA datasheets

[6] Spartan-3 FPGA Family:Complete Data Sheet

[7] Wilfried Elenmenriech, Marti Delvai."Time triggered communication with UARTs." 4th IEEE international workshop on factory communicationsystem, Sweden, August 2002.

[8] Raffaele Gallo, Martin Delvai,Wilfried Elmenreich, Andreas Steininger."Revision and Verification of an Enhanced UART", Austria, 2004.

[9]M.Delvai,U.Eisenmann,W.Elmenreich,"Intelligent UART Module for Real-Time Applications." First Workshop on Intelligent Solutions in E mbedded Systems (WISES), pages 177-185,Vienna, Austria, June 2002.

[10] Mahat N.F, "Design of a 9-bit UART module based on Verilog HDL", in the proceedings of 10th IEEE International Conference on Semiconductor Electronics (ICSE), 19-21st Sept. 2012, DOI: 10.1109/SMElec.2012.6417210,pp.570-573

Mr. LAKSHMAN MURTHY is currently working as an assistant professor in ECE Department, QIS Institute of technology, Ongole, A.P, India. He received his B.Tech degree in the department of Electronics and Communication Engineering, from Narasaraopeta Engineering College, Narasaraopeta (Affliated to JNTU Kakinada). He received his M.Tech from S R M University, Chennai. His research interest in the area of VLSI.

Ms.P.PRAGATHI is currently working as an assistant professor in ECE Department, QIS Institute of technology, Ongole, A.P, India. She received her B.Tech degree in the department of Electronics and Communication Engineering, from QIS Engineering College,Ongole (Affliated to JNTU Kakinada). She received her M.Tech from Vignan University Vadlamudi, Guntur. Her research interest in the area of VLSI.

Ms.G.Bhanu Priya received her B.Tech degree from the department of Electrical and Electronics Engineering from NBKR Institute Of Science and Technology (Affiliated to SVU, Tirupati). She is pursuing M.Tech in QIS Institute of Technology (Affiliated to JNTU Kakinada), Ongole, AP, India. Her current research interest in the area of VLSI.

ISSN: 2278 – 909X