IJITKM *Volume 6 • Number 2 • July-December 2013 pp.-165-170*

FPGA-based implementation of UART

Kamal Kumar Sharma[1] Parul Sharma[2]
[1] Professor; [2]Assistant Professor
Dept. of Electronics and Comm Engineering, E-max School of Engineering and Applied Research, Ambala
kamalsharma111@gmail.com,erparul_sharma@yahoo.com

Abstract
This paper focuses on the hardware implementation of a high throughput Universal Asynchronous Receiver & Transmitter (UART) using FPGA. The UART described in this paper consist of the transmitter, the receiver and the baud rate generator. This has been implemented using Verilog Hardware Description Language and simulated using ModelSim SE 6.0d. The Verilog description is synthesized on the Field Programmable Gate Array Devices (FPGA) such as Virtex4 and Sparten3and a comparative study is done between the different characteristics. The maximum frequency of operation in case of Virte4 is 289.151and in 155.473MHz in case of Sparten3.But the total power consumption in case of Virtex4 is 268mw and 93mw.The main focus of this paper is to design a synthesizable UART and to study its characteristics.
Keywords: UART, FPGA *&* Verilog

1. Introduction

A Universal Asynchronous Receiver and Transmitter (UART) is an indispensible component used for communication with serial input and serial output devices. Serial communication [1], [2], [3] is an essential to computers and allows them to communicate with the low speed peripherals devices such as keyboard, mouse, modems etc. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. Figure 2 illustrates a basic UART data packet. A data packet is generally consist of 1 start bit, which is always a logic 0, followed by a programmable number of data bits (typically between 5 to 8), and a programmable number of stop bits (1,1/2 or 2) [2]. The stop bit always remains at logic 1.

Fig 2: Basic data format of UART: 1 star bit, 8 data bit & 1 stop bit

Thus, a standard UART can transmit 10 bits of data byte. In the UART, the two systems transmitter and the receiver do not share a clock signal & they contain separate local clock [2]. s. Since common clock is not shared, a known data transfer rate (baud rate) must be agreed upon before the transmission of data bit. In all cases the transmitting and receiving. The transmitter shifts out the data starting with the LSB first. Once the required baud rate will be established (prior to initial Communication), both the transmitter and the receiver's internal clock is set to the same frequency (though not the same phase). The receiver "synchronizes" its internal clock to that of the transmitter's at the beginning of every data packet received. This allows the receiver to sample the data bit at the bit-cell centre [1]. So, it is necessary to study how it transmit and receive the data. In this paper UART (Universal Asynchronous Receiver Transmitter) is built using Verilog & design has been synthesized on Xilinx ISE 10.1 tool, and simulated using ModelSim SE 6.0d, at the end the design has been targeted on Xilinx Virtex4 & Spartan 3 FPGA. The UART consists of following five modules.

- u_xmit1.v
- u_rec1.v
- baud1.v
- inc1.h
- uart1.v

This paper is arranged as follows:
After giving the brief introduction in section 1. Verilog Designing of UART has been discussed in section 2. The section 3 gives the results & discussion of simulation and synthesis. & the final section draws the conclusion & Future work.

2. Verilog Implementation of UART Modules

2.1Transmitter Module

The transmitter of -UART is composed of bit cell counter, transmitted bit counter, a serializer and a state machine. Like the receiver counter-part, the design is minimalist and contains no error detecting logic. The Xmitted bit counter is used to keep track of the number of data bits cumulated so far. When this count will become the pre-set limit (i.e.8), then the state-machine will stops accepting more data bits. This counter has 2 control inputs: enabitcountH and rstbitCountH. When the former is active high, the counter is advanced by 1, when the latter is active high the counter is cleared to 0.
The width of this counter is of 4-bits by default. The main function of bit cell counter counter is to generate a delay in units of uartclk (Baud rate Period/16). This is an up counter & the signal countEnableH will control it. When countEnableH becomes active high the counter is in reset state. When this signal is active low, the counter will count up by 1. Figure 5 illustrates the functional block diagram.

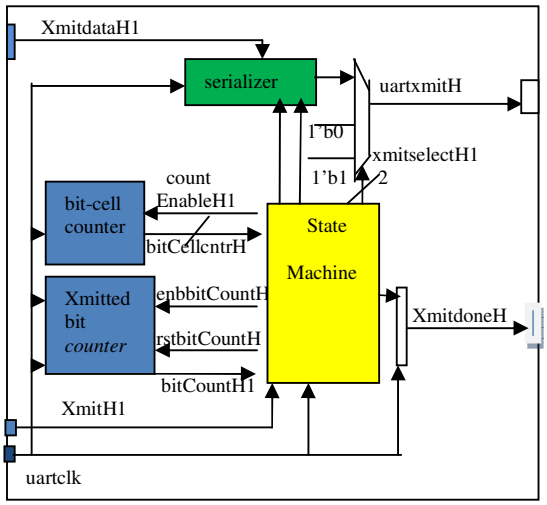


Fig 5: Transmitter Block diagram

The serializer in transmitter is a 8 bit parallel-in-serial-out shift register. It is controlled by 2 control inputs: loadshiftRegH and shiftEnaH1. When the first signal will be active high it will loads the parallel data into the shift register. And an active high on the latter signal will shifts the loaded data out by 1 bit. A mux is present on the uartxmitH1signal. This functionality of mux is to select the start-bit (logic 0), user data (from the shift register) and the stop-bit (logic 1). The state machine is a simple 5 state Mealy type. Figure 6 illustrates the state flow.

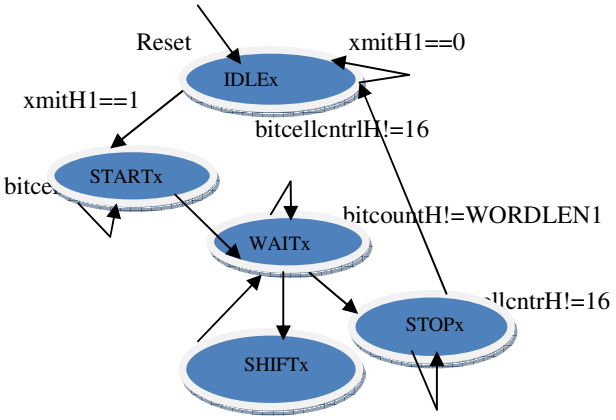


Fig 6: State-Machine of transmitter

When the system reset, the state machine defaults to IDLEx state. In this state, the state machine idles for as long as no transmit command is given [1]. When xmitH1 becomes active high (for 1 uart pulse), then the serializer will be loaded and it will transitions to STARTx state. In STARTx state, the uart_xmitH mux will be set to 1'b0 (start bit), and it will be waited for 1 baud tick (16 uart pulse) before transitioning to WAITx state. In WAITx state, the uartxmitH1 mux will be set to point to the shift register, and 1 baud tick will be waited. When the wait is complete and all bits (WORDLEN1) have been transmitted then the state machine will transitions to STOPx state, otherwise it will go back the SHIFTx state. In the SHIFTx state, the shift-register will be shifted by 1 bit and transitions to WAITx state. In STOPx state, the uartxmitH1 mux set to 1'b1 (stop bit), 1 baud tick will be waited and then transitions to IDLEx state[1].

2.2Receiver Module

The receiving part of UART consists of a Machine which controls the different states of the receiver, a de-serialise, and a support logic. The main functionality of the receiver is to detect the start-bit, then to convert the following 8 bit serial data into parallel(de-serialize), then to detect the stop-bit, and make the data will be available to the host, in this the parity bit is not taken. Figure 3 explains the functionality of the receiver. There no error checking logic is taken by default. The u_baud1.v will generate the signal uartclk which is 16*Baud-Rate. The clocks present within the receiver module will be driven by this clock.

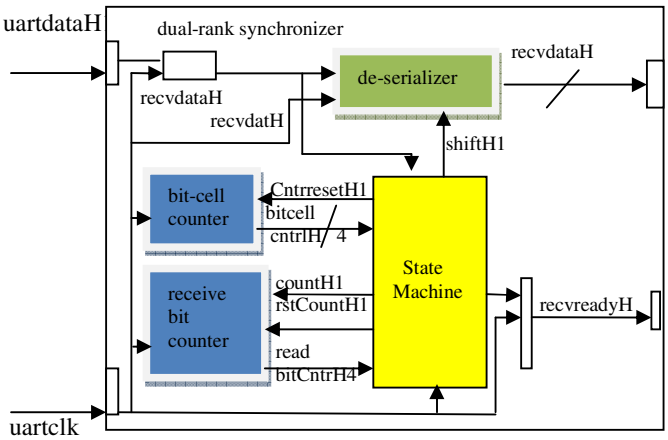


Fig 3: Receiver Block Diagram

Before giving the incoming data i.e. uartdataH1 to de – serializer it is fed to the dual-rank synchronizer. This dual-rank synchronizer is an essential part of receiver because the data present on uartdataH1is synchronous to the transmitter's clock, and not on the receiver's clock. The de-serializer functionality is same as that of serial-to-parallel shift register. It has 1 control input shiftH1 from the state machine [1]. When this signal is going to be active high, the deserializer will shifts the data over by 1 bit. By default the width of shift register is 8 bits. The LSB will be shifted in first. The receive bit counter keeps the track of the number of data bits received so far. When this count is equal to the pre-defined limit (i.e.8), then the state-machine will stops receiving more data bits.

This counter has 2 control inputs: countH1 and rstCountH1 [1]. When the former is active high, the counter is advanced by 1, when the latter is active high, the counter is cleared to 0[1]. Note that this is a synchronous counter [1]. This width of

this counter is of 4-bits by default. The main function of bit cell counter is to generate a delay in units of uartclk (Baud rate Period/16). This is an upcounter & the signal cntrresetH1 will control it . When cntrresetH1 becomes active high the counter is in reset state. When this signal is active low, the counter will count up by 1. The state-machine is a simple 5 state, Mealy type (output is function of present state and input). Figure 4 explains the state diagram of state machine.

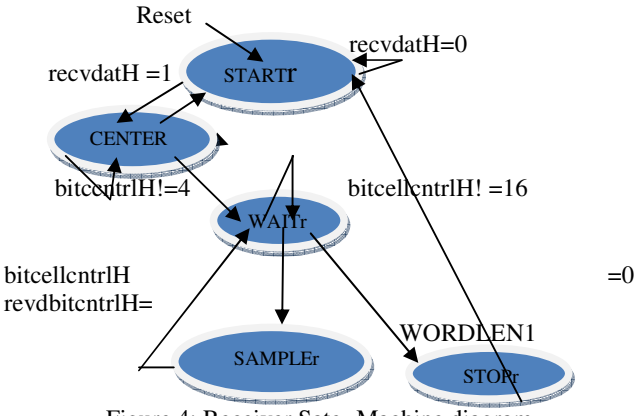


Figure 4: Receiver Sate -Machine diagram

The state-machine ties all of the functional units previously described. When the system reset, the state machine by defaults will be in STARTr state. In this state, the state-machine looks for the start-bit [1]. This condition is going to be detected by the
transition of the incoming data (which is idle at logic 1) to a logic 0. Once the start-bit will be detected, it transitions to CENTERr state. In CENTERr state, the state-machine waits for ½ bit cell in order to find the bit-cell centre [1]. A bit-cell is 1 baud "tick" and corresponds to16 uartclk ticks. So ½ bit-cell corresponds to 8 uart pulses. . Once the bit-cell centre is found (after having waited 4 uart pulses), if the state of the recvdataH (synchronized incoming data) is still low, then the state machine transitions to WAITr state. If recdataH1 is high, then this is not a valid start bit, so the state machine transitions back to STARTr state. This type of effect can be produced by noise signal in the UART data line. The WAITr state simply waits for 1 baud tick (16 uart pulses) [1]. Note that the previous state, CENTERr, aligned the incoming data to the center of the start bit-cell. Once 1 baud tick is waited, the incoming data can be sampled into the de-serializer. If all WORDLEN1 (8 by default) bits have been sampled, then the state machine transitions to STOPr state, otherwise, it transitions to SAMPLEr state.

2.3Functionality of Baud Rate Generator

The functionality of baud rate generator is very simple. It will generate the uartclk from the external clock (sysclk) of the system. The uartclk is equal to 16 times the baud rate. The baud rate and the system clock rate are same as specified in the inc1.h file. These parameters are specified during the synthesis process [1]. In this design the system clock is 25.175MHz and the baud rate is 9600kbps.The required maximum frequency of uart_clk (16*baud rate) is 9600*16=153600.

3. RESULTS & DISCUSSION:

The hardware of the system design mainly consists of the UART (Universal Asynchronous Receiver Transmitter) part that will be connected to any serial communication device. The main goal of this work is to design a UART by using HDL for serial data transmission. The generated simulated result of UART shows the serial transmission of 8 bit data at the baud rate of 9600 kbps by using the crystal frequency of 25.175MHz is shown in Fig.7 .The transmitter converts the 8 bit parallel data into serial and take 8*16 clk pulses for transmitting the 8 bit data transmitter .The signal xmitdone is active high when the 8 bit data has been successfully transmitted. When it completes the transmission of first 8 eight bit data then it start transmitting remaining data. Similarly the receiver collect the 8 bit serial data and convert it into parallel data. When receiver receives the 8 bit data then signal recvreadyH becomes high. RTL schematics of baud rate generator, transmitter and receiver is shown in Figure.8, Fig.9 and Fig.10 respectively .The synthesis flow for the UART has been targeted to two flexible high performance FPGA Architectures available from Xilinx called Virtex4FX and the Sparten3 families . The synthesis has been done when the optimization goal of the designing is area. The table.1 shows the comparison between the synthesized results between the two FPGA families.

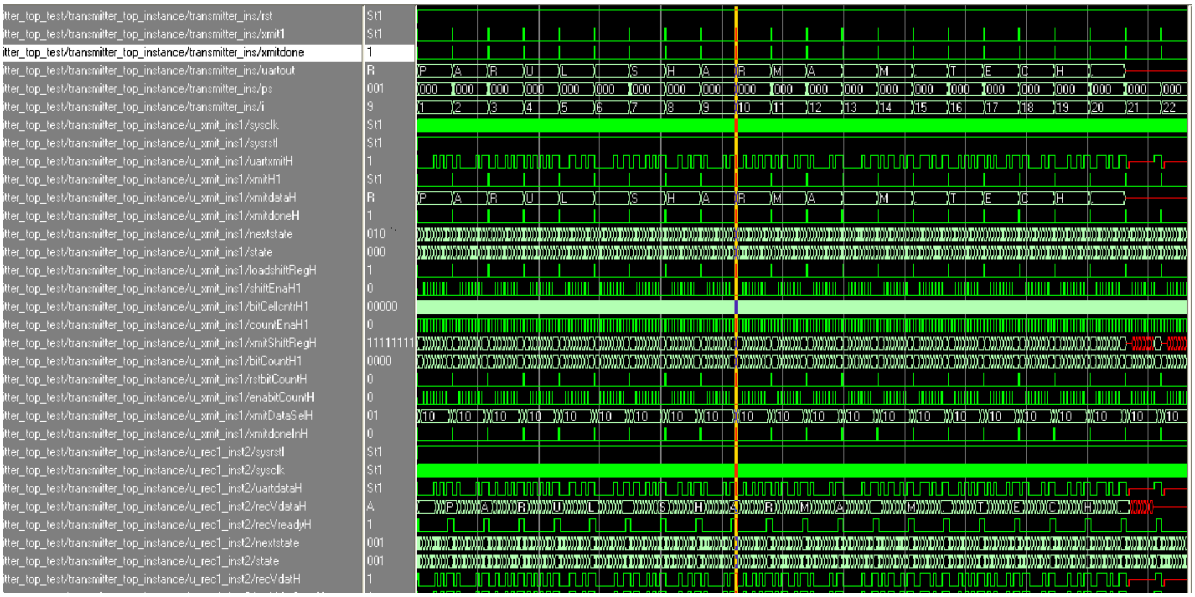


Fig 7: Simulated Result of UART

Fig 8:RTL Schematic of Baud Rate Generator

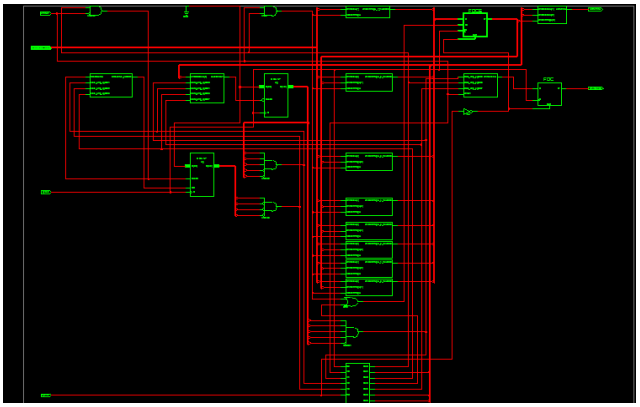
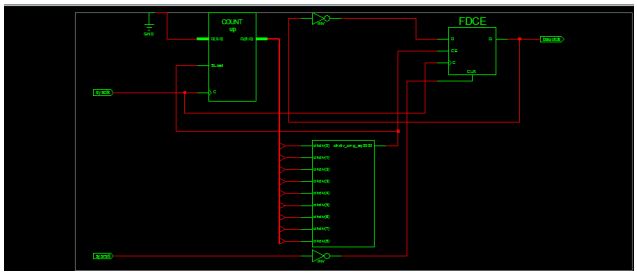


Fig 9: RTL Schematic of Transmitter



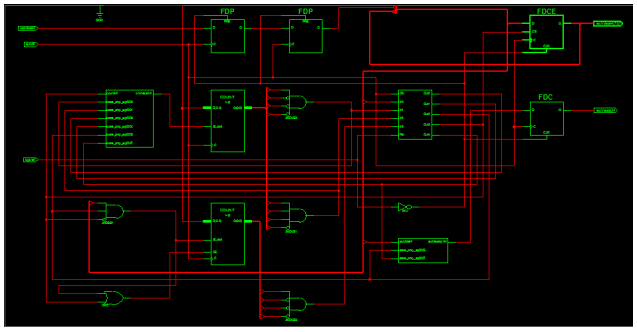


Fig 10: RTL Schematic of Receiver

Synthesis Results

Parameters	Virtex-4vfx12-sf363-10	Sparten-3s1000fg320-4
Number of Slices	63	59
Number of Slice flip-flops	67	69
Number of LUT'S	109	112
Number of IOB'S	12	12
Number of GCLKS	2	25
Minimum input arrival time	3.204ns	4.431ns
Maximum output required time	4.677ns	7.241ns
Maximum frquency	289.151MHz	155.473MHz
Total Power Consumption	268mw	93mw

Table 1: Comparison between Synthesized results of two FPGA families

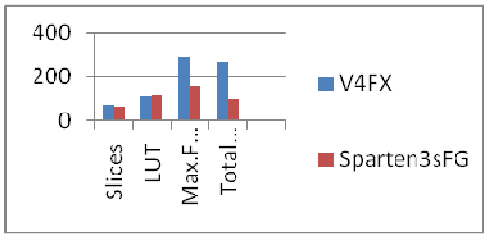


Fig 11: Comparison chart of parameters for Virtex4fx & Sparten3sfg FPGA devices

4. Conclusion

A UART is an device which is used for serial data transmission in communications systems. UARTs have enhanced features that can increase data throughput while preventing data loss and data errors. The UART system was designed using Verilog in a high level design method. All modules of the design have been simulated using ModelSim SE 6.0d and implemented using Xilinx ISE 10.1 tool & Virtex4 with 363 & Sparten3 FPGA with 320 input/output pins is used as a target device. From the results we conclude that there is small difference between no. of slices, LUTs, GCLKs but here is a large difference in total power consumption between the two FPGA devices. The power consumed Sparten3FG is less than V4FX. So, Sparten3FG is a better choice for the implementation of this UART because it consumes less power.

5. Future Work

This work can also be modified in order to determine any error occur in the data bit during the transmission by using the parity–check codes. The synthesized can also be compared by considering the optimization goal as area. This UART can also be designed for higher baud rates according to the applications in order to increase the speed of transmission.

6. References

[1] Micro-UART available at www.cmosexod.com

[2] Chig-Chang Wong & Yu-Han Lin "A reusable UART IP Design and its Application in Mobile Robots", Dec 2006.

[3] Tomasi, Wayne, Advanced electronic communication systems, Third Edition, Prentice-Hall, United States of America, 1994

[4] Norhuzaimin j. Maimun, H.H. "The design of high speed UART" Asia-Pacific Conference on Applied Electromagnetic (APACE 2005). Dec.2005

[5] Digital UART Design in HDL available from
www.QuicKLogic .com.