Implementation and Customization of UART in Xilinx FPGA

T.D. Manoj Kumar Reddy, Lecturer, JNTUA-CEA, Ananthapuramu, Andhra Pradesh,India-515002.

Abstract—**Standard UART logics whether they are in micro-controller or in FPGA will contain all the functions related to the UART. This implementations are complex and may not be require in a specific applications. The full-fledged implementation will consume more micro cells and puts burden on the remaining logics. The present project "Implementation and customization of UART in Xilinx FPGA" will explain how minimization can be done in UART design so that the required functionalities can be fulfilled with considerable reduction in the FPGA utilization. The proposed project is one of the example.**

Keywords— **FPGA,UART, NVRAM,VHDL**

I. Introduction

A UART (Universal Asynchronous Receiver Transmitter) is a device that allows reception and transmission of data in serial and asynchronous fashion. It is a serial communication protocol that sends parallel data to serial line. It typically used with RS-232 standard. UART allows communication between computer and several kinds of devices such as printer, scanner etc interconnected via RS-232 cable. The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. 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.. The UART module provides asynchronous serial communication with external devices such as modems and other computers. The UART can be used to control the process of breaking parallel data from the PC down into serial data that can be transmitted and vice versa for receiving data. Each UART contains a shift register which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is much more cost effective than parallel transmission through multiple wires.

The UART [1] consists of one receiver module and one transmitter module. Those two modules have separate inputs and outputs for most of their control lines, the lines that are shared by both modules are the bi-directional data bus, master clock. Since the data stream has no clock, data recovery depends on the transmitting device and the receiving device operating at close to the same bit rate. The UART receiver is responsible for the synchronization of the serial data stream and the recovery of data characters. Data transmission can be made in serial way that contains 11 blocks. One start bit, eight data bits, one parity and one stop bit. . The start bit signals the receiver that a new character is coming. The next five to eight bits, depending on the code set employed, represent the character. First least significant bit(LSB) is transmitted. Following the data bits may be a parity bit. The parity is set to be '0' or '1' depending on the number of '1's transmitted. When even parity is used the number should be even. When odd parity is used the number should be odd. If chosen parity is not represented in this block transmission error should take place. The next one or two bits are always in the mark (logic high, i.e., '1') condition and called the stop bit(s). they signal the receiver that the character is completed. The transmission and reception lines should hold '1' under idle condition.

The UART[8] allows the devices to communicate without the need to be synchronized. The UART system can tolerate a moderate amount of system noise without losing any information.. UART in FPGA[2] will have single ended transmit and receive signals can be converted in to differential signals using high speed differential line drivers. The voltages of FPGA and Line Drivers are different., so level converter circuit need to be present in the board. In this customized UART when ever particular character is received on the receive signal ,particular task has to be performed and the data need to be transmitted. The digital design [9] of the customized UART is implemented in FPGA using VHDL[10].The address, data and control signals for NVRAM are generated with in the FPGA[2] and stored in to the NVRAM[3].Power to the FPGA is carried out by using DC-DC converter[6] and voltage regulator.RS-422 communication is done by RS-422 transceiver [4].10MHz reference clock is used for the design of customization of UART. Reset signal is generated externally by using reset IC[5] and is used for the design of Customized UART. For this application of the UART one start bit,8 data bits, no parity and followed by one stop bit.

II. Block Diagram

A. UART block diagram

The block diagram of the customized UART[1] is as shown in the fig 1.The UART consisting of baud rate generator transmitter and receiver with control logic. The logic can be implemented by using VHDL language and programmed in FPGA. It is interfaced with NVRAM for the storage of data and RS-422 transceiver is used for the communication with the external world.

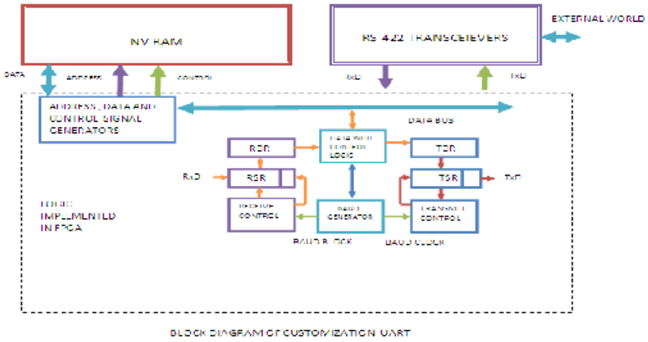


Fig1:Block diagram of customized UART

1) Transmitter

Transmission operation is simpler and can be done using shift registers. As soon as when the data is deposited in the shift register , the UART hardware generates a start bit, shifts the required number of data bits out to the line, generates and appends the parity bit (if used), and appends the stop bits. Full-duplex operation requires characters to be sent and received at the same time. In this application 16 bit counter is used. with this 16 bit counter, 8 MSB bits are taken as one byte of data and next 8 bit LSB bits are taken as one byte of data. One start bit, 8 data bits and one stop bits are concadded and shifted out by using transmission shift register. Transmitter shift register takes data from the data bus. When the character is received by the receiver the control is enabled and the transmission can happen. In this customization of UART, the desired received character on the Rxd will enable the transmission shift register and transmission can takes place These bytes can be stored in NVRAM for every 5ms. By using generated internal signals the data will be shifted out at desired time.10 MHz is used as reference clock for this application

2) Receiver

The UART is controlled by the clock signal which runs at a multiple (say, 16) of the data rate. Each data bit is a length of 16 clock pulses. The received data will be concadded with the receiving shift register. The receiver looks for the start bit and test the recognizes the start bit. If the apparent start bit checks for at least one-half of the bit time, if it is valid and signals the start of the new character. If not, the spurious pulse is ignored. After detecting the start bit, the receiver checks for the data bits and stop bit. If the particular character received on the receiver from the input terminal matches the desired character then it will enable the transmission process.

3) Baud rate Generator

Baud rate generator generated different types of baud rates with respect to reference clock. By using the selection the desire baud rate can be selected. This process can be done by dividing the reference clock by 16 and intern divided by particular value which matches the desired baud rate. The generated baud rates can be multiplexed and the desired baud rate can be selected. Transmitter and receiver uses the same baud rate for the communication. The common baud rates are 4800,9600,19200,38400 etc.,

B. Power supply scheme block diagram

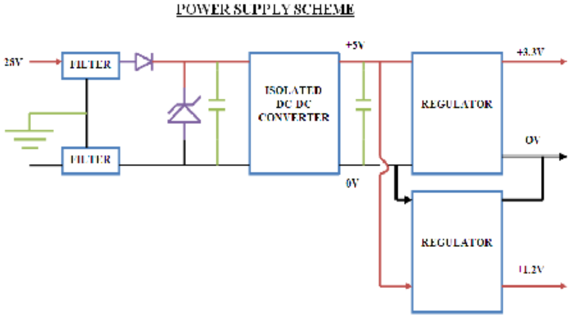


Fig2:Block diagram of the power supply scheme

The block diagram of the power supply scheme is as shown in the fig 2.The variable DC supply 28V is used for generation of 5V ,3.3Vand 1.2V dc supply. The 28V supply is applied to the isolated DC-DC converter through filter and passive components. The diode and zenar diode is used for protection of the circuit in case of reverse connection. The isolated DC-DC converter converts 28V into 5V.Intern this 5V is converted in to 3.3V and 1.2V by using different voltage regulators. . The generated 5V is used to power the RS-422 converter. The generated 3.3V and 1.2V are used to power the FPGA.

C. Interfacing block diagram of UART

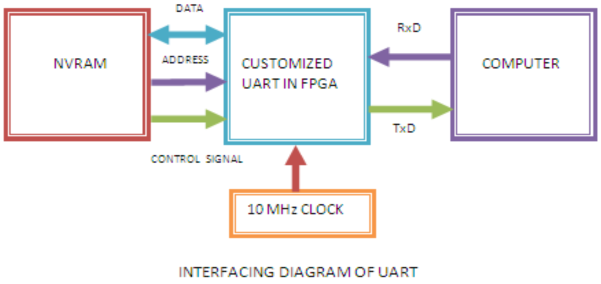


Fig.3: Interfacing diagram of UART

The Interfacing block diagram of UART is as shown in the fig 3.it mainly consisting of FPGA and Level converter and RS-422 transceiver, , NVRAM and the computer, . The design of the UART is implanted in Xilinx SPAETAN-3 FPGA.VHDL is used for the design in Xilinx tool and hardware programming can be done by using JTAG Programmer. Reference 10 MHz clock is used for the UART design. By using selection lines baud rate of UART is selected. TxD and RxD are the outputs of the transmitter and receiver sections of the UART. These single-ended signals are can be converted into differential by using MAX1490 RS-422 transceiver. The single ended signals from FPGA are of 3.3V can be converted into 5V by using level converter . FPGA and NVRAM operates at 3.3V and RS-422 transceiver operates at 5V. Apart from UART design some internal logic can be implemented to generated the signals for writing data into the NVRAM for every 5 ms. Some of the signals such as chip Enable(CE),Write Enable(WE), Output Enable(OE), Address, Data can be generated using some internal logic implemented in FPGA. These signals are used for writing the NVRAM.

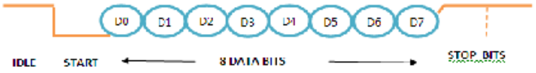


Fig4: UART data out format

UART data out format is as shown in the fig 4. The sequence is Idle condition , one start bit ,eight data bits(LSB first) followed by one or two stop bits. Considering Fig 2, the UART is developed in the FPGA, the single ended signals Tx and Rx are converted in to 5v and fed to RS-422 transceiver and the outputs are differential signals and communicated to the external world.RS-422 to USB is used for communicating with the computer. With the use of Hyper terminal the communication can be monitored in the computer. By selecting parameters like Baud rate, data bits, parity, number of stop bits, flow control communication can be established. The Received data can be captured in a file and can be viewed by using HEX pert software tool.

III DESIGN OF CUSTOMIZED UART IN FPGA

D. Software Design flow:-

The digital logic of this project is explained with reference to the flow chart shown in the Fig 5.With respect to reference clock LATCH signal is generated for every 5 ms. By using this LATCH signal, one more signal STABLE LATCH will be generated which will be divided by 2 of the LATCH signal. MSB bits of the 16 bit counter given to the counter 16MSB byte and LSB bits of the 16 bit counter given to the counter16LSB byte. The data will be stored in NVRAM for every 5 ms. In addition to this the control signals will be generated from the FPGA. Active low WRITE ENABLE pulse(WE) of 400 ns will be generated when the LATCH signal occurs. This signal will becomes low when count of the counter becomes 50003 to 50006. Complement of WRITE ENABLE can be used for the OUTPUT ENABLE(OE). The signal will becomes high when count of the counter becomes 50003 to 50006. A low signal is generated for CHIP SELECT(CS). The counter will be reset when the count becomes 50009.When control signal R/WSEL signal is low write operation will be performed, when signal is high read operation will be performed, default condition is write mode. Address will be incremented with respect to LATCH SIGNAL. The 19 bit address will be generated in the FPGA. When STABLE LATCH becomes high counter16MSB byte will be transferred and stored in NVRAM and STABLE LATCH becomes low counter16LSBwill be transferred and stored in NVRAM. This process repeated for every 5 ms. When R/WSEL signal is high read operation will be performed, Address will be incremented with respect to LATCH. 19 bit read address will be generated by FPGA and used for reading the NVRAM. CHIP SELECT (CS) will be low, WRITE ENABLE will be high and OUTPUT ENABLE (OE) will be low during the read operation. A 10 MHZ reference clock has been using for the design.19,200 baud rate will be generated with the 10 MHZ, the clock is first divide by 260 and then divided by 2.This can be used for the transmit as well as receive in the UART .

In the UART receive mode the 10 MHZ clock is first divide by 26 and then divided by 2.This test clock is used for the detection of the start bit. Counterz is used for the counting of the test clocks when RxD start bit becomes low. When the RxD becomes high counter will be reset. When RxD is low and counterz becomes 8 the start bit is zero generate the signals start and Dataen and the counterz will be reset. When start becomes high, counterb is used to count 80 test clock pulses and Dataen signal becomes low. When count becomes 90 and the RxD becomes high counterb counter will be reset. When counterb becomes 80 the Stop signal becomes high and counterb becomes 90, Stop signal becomes low. When dataen is high, falling edge of baud clock,8 bit SISO shift register 1 to 7 bits are concadded with RxD and assigned to same serial in serial out shift register 0 to 7 bits. When Stop becomes high the shift register output assigned to other shift register. when $ symbol is given on RxD, R/WSEL becomes high, read process will occur. when # comes on RxD, R/WSEL becomes low it will be in writing mode R/W SEL becomes high UART transmission occurs ,start bit(zero) concadded with 8 bit data concadded with one stop bit(one) and assigned to transmit data shift register. When stable latch is high as well as low, LSB bit of transmit data shift register is assigned to TxD. This process carried out with respect to the baud clock. Baud rate generator using reference clock. Different baud rates can be generated by using selection lines.

E. Hardware design:-

Hardware design is done by interfacing of power supply scheme and interfacing diagram. In power supply scheme the 28V DC is converted in to 5V by DC-DC converter and this 5V intern converted in to required 3.3V and 1.2V.This is connected to FPGA, NVRAM and transceivers. The required logic of UART and required logic for NVRAM is designed by using VHDL. The code is complied ,synthesized and simulated by using XILINX- ISE tool and the programming file(bit file) generated and the SPARTAN-3AN FPGA is programmed by using ISE- iMPACT tool using JTAG programmer. The FPGA is interfaced with the MAX-1490B RS-422 transceiver. The output of the transceiver connected to the external world. In the case of connection with the computer it will interfaced with RS-422 to USB.

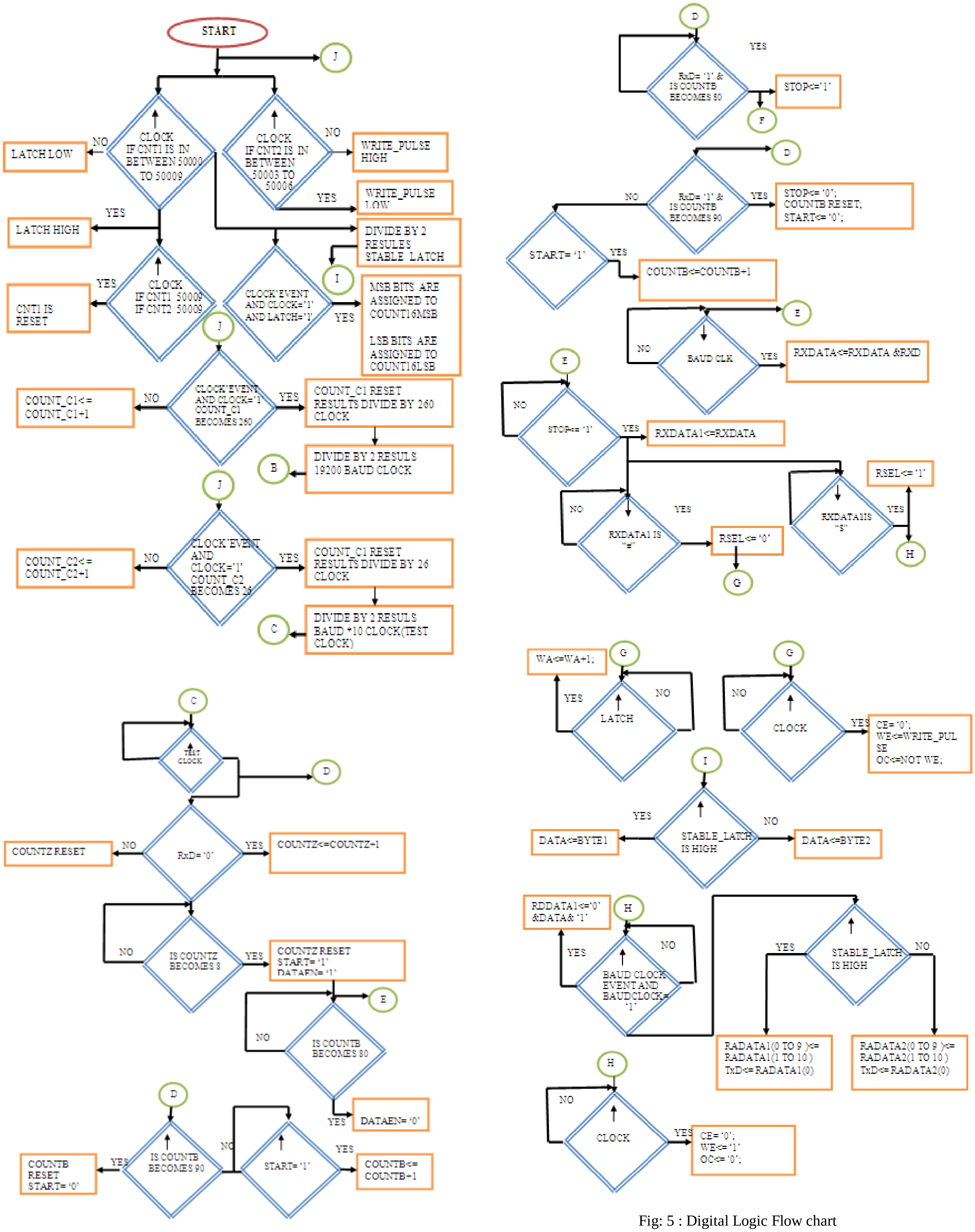

START
J
LATCH LOW
NO
CLOCK IF CNT1 IS IN BETWEEN 50000 TO 50009
CLOCK IF CNT2 IS IN BETWEEN 50003 TO 50006
NO
WRITE_PULSE HIGH
YES
WRITE_PULSE LOW
YES
LATCH HIGH
DIVIDE BY 2 RESULES STABLE LATCH
I
YES
CNT1 IS RESET
CLOCK IF CNT1 50009 IF CNT2 50009
CLOCK EVENT AND CLOCK='1' AND LATCH='1'
YES
MSB BITS ARE ASSIGNED TO COUNT16MSB
LSB BITS ARE ASSIGNED TO COUNT16LSB
J
COUNT_C1<= COUNT_C1+1
NO
CLOCK EVENT AND CLOCK='1' COUNT_C1 BECOMES 260
YES
COUNT_C1 RESET RESULTS DIVIDE BY 260 CLOCK
DIVIDE BY 2 RESULS 19200 BAUD CLOCK
B
J
COUNT_C2<= COUNT_C2+1
NO
CLOCK EVENT AND CLOCK='1' COUNT_C2 BECOMES 26
YES
COUNT_C1 RESET RESULTS DIVIDE BY 26 CLOCK
DIVIDE BY 2 RESULS BAUD *10 CLOCK(TEST CLOCK)
C
C
TEST CLOCK
D
COUNTZ RESET
NO
RxD='0'
YES
COUNTZ<=COUNTZ+1
NO
IS COUNTZ BECOMES 8
YES
COUNTZ RESET START='1' DATAEN='1'
E
NO
IS COUNTB BECOMES 80
YES
DATAEN='0'
D
COUNTB RESET START='0'
YES
IS COUNTB BECOMES 90
NO
START='1'
YES
COUNTB<= COUNTB+1
D
YES
RxD='1' & IS COUNTB BECOMES 80
STOP<='1'
F
D
NO
RxD='1' & IS COUNTB BECOMES 90
YES
STOP<='0'; COUNTB RESET; START<='0';
START='1'
YES
COUNTB<=COUNTB+1
E
NO
BAUD CLK
YES
RXDATA<=RXDATA &RXD
E
NO
STOP<='1'
YES
RXDATA1<=RXDATA
RSEL<='1'
NO
YES
RXDATA1 IS "#"
RSEL<='0'
G
RXDATA1IS "$"
YES
H
WA<=WA+1;
G
YES
LATCH
NO
G
NO
CLOCK
YES
CE='0'; WE<=WRITE_PULSE OC<=NOT WE;
I
YES
DATA<=BYTE1
STABLE_LATCH IS HIGH
NO
DATA<=BYTE2
RDDATA1<='0' &DATA&'1'
H
YES
NO
BAUD CLOCK EVENT AND BAUDCLOCK='1'
YES
STABLE_LATCH IS HIGH
NO
RADATA1(0 TO 9)<= RADATA1(1 TO 10) TxD<=RADATA1(0)
RADATA2(0 TO 9)<= RADATA2(1 TO 10) TxD<=RADATA2(0)
H
NO
CLOCK
YES
CE='0'; WE<='1' OC<='0';

Fig: 5 : Digital Logic Flow chart

Sl.no	Selection input	Input clock frequency	Baud rate
1	000	10MHz	38400
2	001	10MHz	19200
3	010	10MHz	9600
4	011	10MHz	4800
5	100	10MHz	2400
6	101	10MHz	1200
7	110	10MHz	600
8	111	10MHz	300

Table 1:Baud rate generator selection

In UART design different types of baud rates are generated. By using the selection inputs baud rate can be selected. This customized UART can be used for different types of baud rates by using the selection as shown in the Table 1.

IV. CRITICALITY OF THE PROJECT

In the PCB of customized UART, the component density is very high, testing is very critical. The testing of internal logic of FPGA is carried out by configuring some of the pins as test points and routing the required monitoring signal to that test point since monitoring on the high density FPGA is not possible.

The sequence of events has to be followed as per the flowchart. To achieve this different latch signals were implemented in different stages. These latch signals are used to transfer the bytes at fixed interval of time and generating the signals for storage. The chain processing has to be monitor after completing total activity. For this a non-volatile static RAM is provided which is loaded with the processed data at the fixed interval of time. The counters MSB byte which is coded in fixed format and the counters LSB byte which is also coded in a fixed format and written in to the static RAM continuously.

As and when, a request come to the FPGA on the UART, what ever data will be available in the NVRAM to be transmitted to the computer through RS-422 port .The UART logic was fully implemented and tested with in the FPGA. This method saved an external UART and interconnection logics and the space .The customized UART is designed and implemented in FPGA as per the requirement by the user.

V. SIMULATION AND HARDWARE REULTS

The digital logic for customized UART is implemented by using VHDL language in XILINX-ISE tool. The VHDL code is complied and synthesized. The simulation was done by ISE-SIMULATOR. The simulation results are as shown in the Fig 6,7 and 8. The signals like CLOCK, ADDRESS, DATA, LATCH, CE, OE, WE, LATCH, STABLE_LATCH, Tx, Rx, EN ETC., are observed in the simulator as well as in the hardware. The UCF file was created for Translate, mapping and place and root and bit file has been generated. The generated bit file was programmed in FPGA by using XILINX-ISE JTAG Programmer.

The 28V DC power supply is used for the hardware. The hard ware test has been carried out by giving inputs and monitored the outputs at corresponding points by using oscilloscope and digital multimeter. The data can be logged in the NVRAM and when it is in read operation, the results of the data was monitored at the HyperTerminal by using RS-422 to USB converter. The results can be captured in text file and corresponding binary values of the ASCII code can be verified by using converter tool. At different time intervals different data will be generated and at one particular value the data will be reset and repeats again. The captured results of the data are as shown in the fig.6 ,7 and 8.

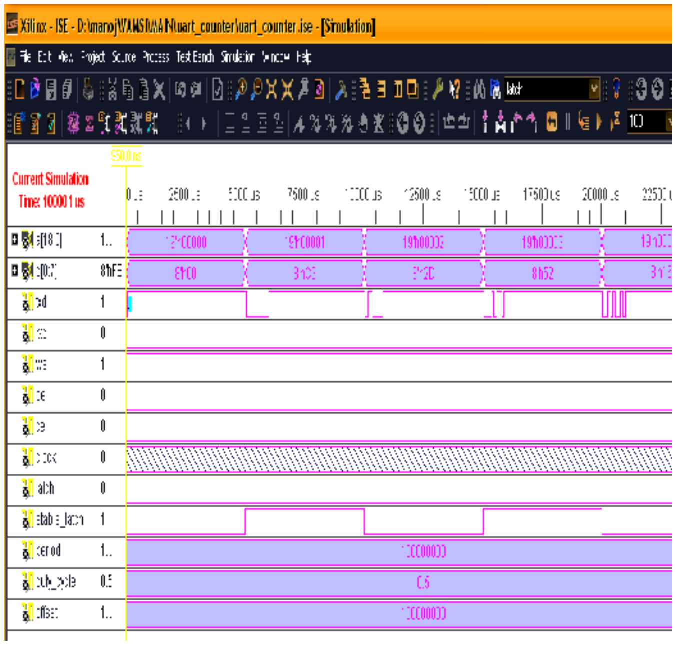

Xilinx - ISE - D:\manoj\WAMSIMAIN\uart_counter\uart_counter.ise - [Simulation]
Current Simulation Time: 1000001 us

Fig 6 :Simulation results for stable latch, Data, Address and Tx

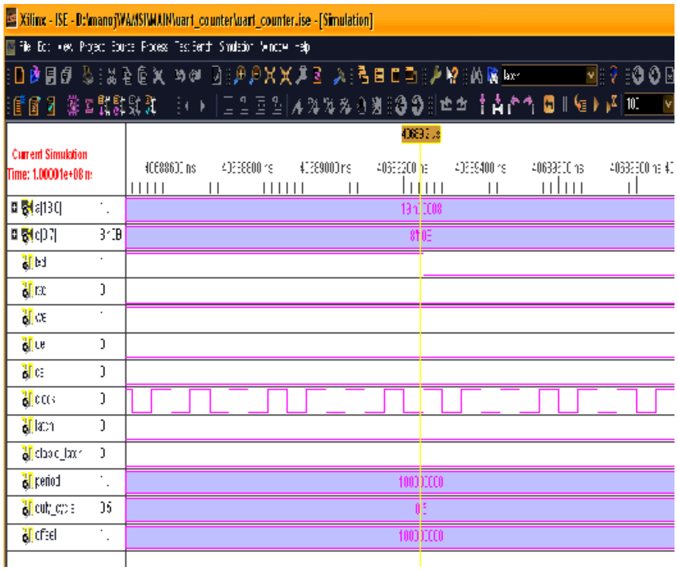

Xilinx - ISE - D:\manoj\WAMSIMAIN\uart_counter\uart_counter.ise - [Simulation]
Current Simulation Time: 1.00001e+08 ns

Fig 7 :Simulation results for clock,Address,Data and Tx

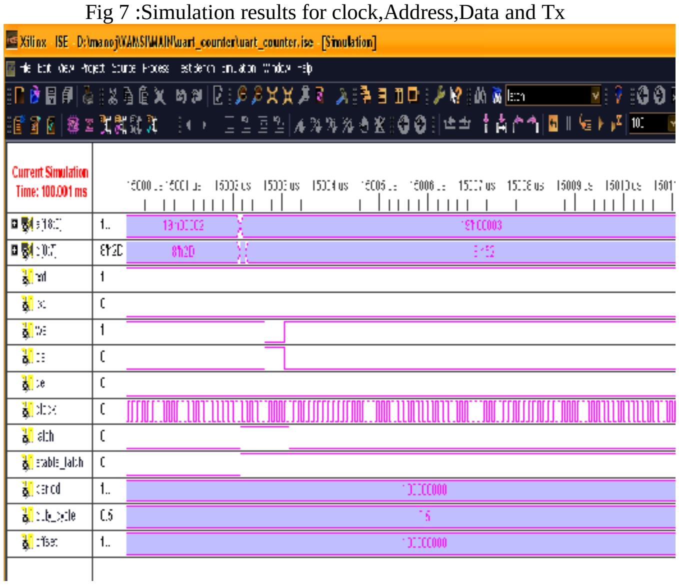
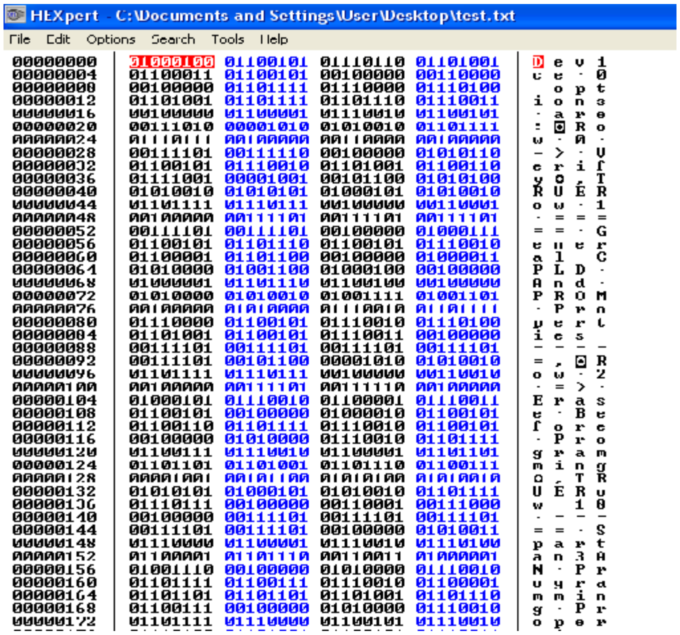

Xilinx ISE D:\manoj\VANSIMAIN\uart_counter\uart_counter.ise [Simulation]
Current Simulation Time: 100.001 ms
8'h2D
enable_latch

Fig 8 :Simulation results for latch ce, oe and we

The hardware results of 16 bit counter is as shown in fig 9. the last column in fig 9 corresponds to ASCII character. The format of 16 bit counter MSB bits and 16 bit counter MSB bits is as shown in fig.10.


HEXpert - C:\Documents and Settings\User\Desktop\test.txt
File Edit Options Search Tools Help

```
00000000  01000100 01100101 01110110 01101001  D e v i
00000004  01100011 01100101 00100000 00110000  c e · 0
00000008  00100000 01101111 01110000 01110100    o p t
00000012  01101001 01101111 01101110 01110011  i o n s
00000016  00100000 01100001 01110010 01100101  · a r e
00000020  00111010 00001010 01010010 01101111  : ◙ R o
00000024  01110111 00100000 00110000 00100000  w · 0 ·
00000028  00111101 00111110 00100000 01010110  – > · V
00000032  01100101 01110010 01101001 01100110  e r i f
00000036  01111001 00001001 00101100 01010100  y ○ , T
00000040  01010010 01010101 01000101 01010010  R U É R
00000044  01101111 01110111 00100000 00110001  o w · 1
00000048  00100000 00111101 00111101 00111101  · = = =
00000052  00111101 00111101 00100000 01000111  = = · G
00000056  01100101 01101110 01100101 01110010  e n e r
00000060  01100001 01101100 00100000 01000011  a l   C
00000064  01010000 01001100 01000100 00100000  P L D ·
00000068  01000001 01101110 01100100 00100000  A n d ·
00000072  01010000 01010010 01001111 01001101  P R O M
00000076  00100000 01010000 01110010 01101111  · P r o
00000080  01110000 01100101 01110010 01110100  p e r t
00000084  01101001 01100101 01110011 00100000  i e s
00000088  00111101 00111101 00111101 00111101  – – – –
00000092  00111101 00101100 00001010 01010010  = , ◙ R
00000096  01101111 01110111 00100000 00110010  o w · 2
00000100  00100000 00111101 00111110 00100000  · = > ·
00000104  01000101 01110010 01100001 01110011  E r a s
00000108  01100101 00100000 01000010 01100101  e · B e
00000112  01100110 01101111 01110010 01100101  f o r e
00000116  00100000 01010000 01110010 01101111  · P r o
00000120  01100111 01110010 01100001 01101101  g r a m
00000124  01101101 01101001 01101110 01100111  m i n g
00000128  00001001 00101100 01010100 01010010  ○ , T R
00000132  01010101 01000101 01010010 01101111  U É R o
00000136  01110111 00100000 00110001 00111000  w   1 8
00000140  00100000 00111101 00111101 00111101  · – – –
00000144  00111101 00111101 00100000 01010011  = = · S
00000148  01110000 01100001 01110010 01110100  p a r t
00000152  01100001 01101110 00110011 01000001  a n 3 A
00000156  01001110 00100000 01010000 01110010  N · P r
00000160  01101111 01100111 01110010 01100001  o g r a
00000164  01101101 01101101 01101001 01101110  m m i n
00000168  01100111 00100000 01010000 01110010  g · P r
00000172  01101111 01110000 01100101 01110010  o p e r
```

Fig 9: NVRAM data at HyperTerminal when impact signal is given

Counter 16bit MSB(15)	Counter 16bit MSB(14)	Counter 16bit MSB(13)	Counter 16bit MSB(12)	Counter 16bit MSB(11)	Counter 16bit MSB(10)	Counter 16bit MSB(9)	Counter 16bit MSB(8)

BYTE MSB16 FORMAT

Counter 16bit LSB(7)	Counter 16bit LSB(6)	Counter 16bit LSB(5)	Counter 16bit LSB(4)	Counter 16bit LSB(3)	Counter 16bit LSB(2)	Counter 16bit LSB(1)	Counter 16bit LSB(0)

BYTE_LSB16 FORMAT

Fig 10: 16 bit counter byte format

VI. CONCLUSIONS

The required flowchart as per the logic of bits of 16 bit counter is implemented in the FPGA by using VHDL and stored in the NVRAM for every 5 ms. The FPGA is operated at 10Mhz basic clock and the required signals are verified with appropriate timings. These results are transmitted to the computer system on a RS-422 port using an UART developed with the FPGA as per the requirement. The outputs on the computer are verified for different baud rates. The FPGA will have enough free macro cells available and can be reconfigured for any new application or additions for future design.

ACKNOWLEDGMENT

I sincerely thank Mr.K.Prabhakar, Director (Tech.), ANALOGIC CONTROLES INDIA LIMITED for giving me an opportunity and supporting me in carrying out this research work. I also thank my HOD of ECE Dept. and other faculty who help to complete my research work.

REFERENCES

[1] “Irda and UART Design” –Dec-200- http://www.xilinx.com
[2] “Spatran-3 AN FPGA Family Data sheet” Dec-2007- http://www.xilinx.com
[3 “bq4015ly NON VOLATILE SRAM”-may1999-Revised-january2010 -http://www.ti.com
[4] “MAXIM-RS-485,RS-422 TRANSCIVERS” –Revised-Dec-1994- http://www.datasheetcatalog.com
[5] “Low power microprocessor supervisory circuits-SP705”-2009 copyright 2000-http://www.datasheetcatalog.com
[6] “Gaia converter HI-REL DC-DC CONVERTER MGDM-04:4W POWER”-2006 Revised Aug-2006--http://www.gaia-converter.com
[7] “National Semiconductor –low power quad operational amplifiers”- August-2000. - http://www.national.com
[8] “Micro-processors and interfacing”-Revised-2006-Douglas v. Hall
[9] “Digital Design”-Fourth edition -2008-John- F warlkerly
[10] “A VHDL PRIMER”-Third edition-2010-J.Bhasker.

Author Institutional affiliations and Addresses

T.D. Manoj Kumar Reddy, *M.Tech –(DECS)*

Lecturer, ECE Department, Jawaharlal Nehru Technological University, Anantapur, Andhra Pradesh, India.
manoj.deshai@gmail.com