ISSN: 2321-7782 (Online)

Volume 2, Issue 9, September 2014

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Customized Processor Design and its Run Time Configuration

Priyanka Balu Bhor[1]
Department of E&TC Engineering
DYPCOE, Akurdi
Pune – India

Prof. R. Arokia Priya[2]
Department of E&TC Engineering
DYPCOE, Akurdi
Pune – India

Dr. P. Malathi[3]
Department of E&TC Engineering
DYPCOE, Akurdi
Pune – India

Abstract: Soft-core's are nowadays becoming a vital part of an ASIC designer. These soft-cores when combined with FPGA's can prove to be very advantageous. FPGA's can be customized according to need of the designer and a lot of resource can be saved for other use on the FPGA. Hence a technique is built wherein the FPGA can be loaded at run time. The system basically has three vital components. One is the soft-core UART, the other is the soft-core Processor, and the third is the software which does the user FPGA interaction. Software converts the assembly language to machine code and sends it to the FPGA.FPGA has a soft core written for UART and receives the machine code which is then processed by the soft core CPU implemented on the FPGA. The UART soft-core and the processor soft-core are designed in verilog language using the Xilinx ISE design suite 14.1. The software tool which is an assembly code editor and compiler is designed using Visual Basic 6.0. The simulation results show how the design works as expected. Also the result is displayed on the seven segment display of xc3s200a FPGA board for hardware testing purpose. The design implemented works at a frequency of 89.5MHz.

Keywords: Programmable gate array, Soft-core, Universal Asynchronous Receiver Transmitter Design.

I. INTRODUCTION

A microprocessor which is defined in HDL and can be synthesized in programmable hardware like FPGA is called soft core processor. Soft-core processors provide a lot of advantages over hardcore processors. Soft-core processors are platform independent, immune to obsolescence, flexible, and are less costly. Universal asynchronous receiver transmitter is widely used in many applications where data has to be communicated between the PC and peripherals.

There exist 2 types of processors - (1) Complex instruction set computers (2) Reduced instruction set computers (RISC). One example of RISC processor is MIPS (Microprocessor without interlocked pipeline stages), which is a 32 bit processor designed with single cycle implementation. Three types of instructions in MIPS are: (1) immediate (I-type) (2) Jump (J-type) (3) Register (R-type). Every instruction type has a different encoding.

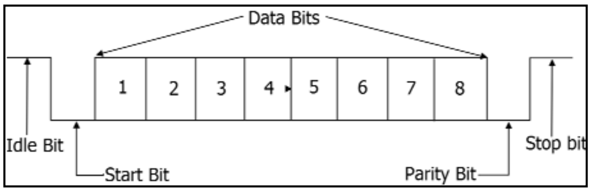


Figure 1-UART frame format

Because memory was expensive in old days, designers enhanced the complication of instructions to reduce program length. This brought up a new design style called the CISC (computer instruction set computer). But long run time cost and low universal property resulted in great disparity of instructions.

A. Register type instruction

(31 - 26)	(25-21)	(20-16)	(15 - 11)	(10 - 6)	(5 - 0)
OPCODE	RS	RT	RD	SHIFT	FUNCTION

B. Immediate type instruction

(31 - 26)	(25-21)	(20-16)	(15 - 0)
OPCODE	RS	RT	ADDRESS /IMMEDIATE VALUE

C. Jump type instruction

(31 - 26)	(25 - 0)
OPCODE	BRANCH TARGET ADDRESS

RISC CPU has more advantages over CISC. It is faster, simple, easier to implement. It is extensively used in embedded systems. MIPS stand for microcomputer without interlocked pipeline stages. MIPS instruction format -3 different instruction formats used for instruction are Register format (R-type), immediate format (I-type), Branch type format (J-type). Data flow - R-format data path, R-I format data path, Load word data path, and Memory word data path. Pipeline design- Pipeline decomposition is as follows (a) IF instruction fetch (b) instruction decodes (ID) (c) Execution (EXE) (d) Memory input/output (MEM) (e) Write back (WB).

II. Related Work

In the past there have been many implementations of MIPS soft-core processors and UART soft-cores. Some of those implementations are summarized here.

Mounika et al. [3] proposes a work in which implementation of MIPS processor is done. In this work the minimum period of design is 40ns which implies the operating frequency of design to be 25MHZ. This work implements MIPS processor in VHDL language.

Xizhi Li et al. [4] proposes an architecture wherein both the processor and application specific integrated circuit coexist on the same FPGA chip. The resource utilization has been optimized to a great extent. Sequencing logic is stored in single block RAM of FPGA, reducing resource utilization. The maximum operating clock rate is 64MHz.

Rupali. S. Balpande et al. [5] analyzes MIPS theory based on RISC CPU instruction set. The work presents the instruction set. MIPS instruction set is analyzed which includes analysis of instruction format including Instruction type, register type and jump type instructions. Also different data path models including R format data path, RI format data path, load word data path and memory word data path. Simulation results for instruction fetch and instruction decode modules are presented, which are simulated in quartus II 4.3 using VHDL language.

Tirath Ramdas et al. [6] present a design of MIPS soft-core processor on RC200 development board from Celoxica. The design is developed in Handel-C language which is a high level HDL language that targets FPGA's. The integer MIPS has the conventional set of instructions and also has additional instruction which is application specific. The instruction is branch if pixel not included (bpni). This instruction improves noise reduction program of system. The MIPS software is basically designed to improve the face detection system. The system performance implies that the number of LUT's used by the system is 5454, and the clock rate is 45 MHz.

Ali El Kateeb et al. [7] presents a design of soft processing core which is used for sensor network application. The design is implemented using Xilinx 8.2i design suite and is tested using Xilinx xc3s700a chip. The work describes motivation for using a soft-core processor. Processor architecture instruction set and interrupts have been analyzed. The processor basically is used for image processing application. The processor architecture is designed to perform only the basic logical and arithmetic operations. The architecture works at a frequency of 50 MHz.

III. SYSTEM IMPLEMENTATION

As it has been mentioned in the past literature there are different implementations of both MIPS processor and UART soft core. In this proposed architecture a technique is proposed where both the soft core UART and soft core MIPS processor can be used together to build a system.

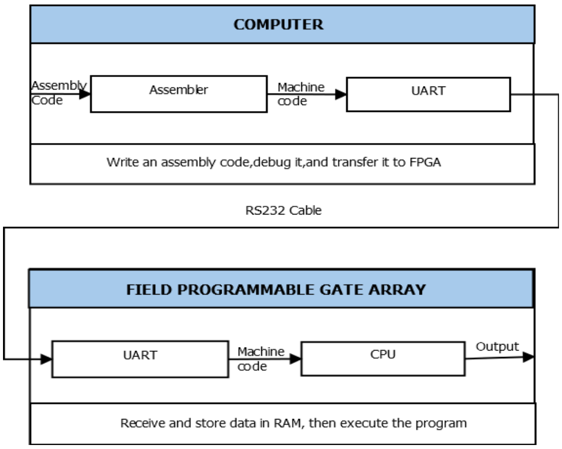

COMPUTER
Assembly Code
Assembler
Machine code
UART
Write an assembly code,debug it,and transfer it to FPGA
RS232 Cable
FIELD PROGRAMMABLE GATE ARRAY
UART
Machine code
CPU
Output
Receive and store data in RAM, then execute the program

Figure 2 - Basic Block diagram

This system will be able to make changes to the MIPS soft core processor at run time using the UART soft core. Basically the idea is to convert the ROM to RAM. And hence run time loading will be possible. The system will be using UART soft core to send the code at run time to the FPGA. Hence the overhead of re-implementing and reloading the whole code in the FPGA is reduced.

The literature mentioned in [3]-[7] implementations does not support run time loading of code. Also the UART that will be implemented here will have different properties than the ones presented in the literature [8]-[9]. The UART implementation will be both customizable and reliable.

A. The UART implementation

Assembly code after being converted to machine code will have to be sent to the instruction memory in the microprocessor in FPGA for execution. Transfer of this data is done by using serial communication where the data or the machine code would be sent one bit at a time over a communication channel between the FPGA and PC.FPGA does not support a UART soft core hence it can be implemented with customizable baud rates.

B. Software tool

The software tool can be designed using a software language to facilitate the communication between microprocessor and the user through the UART. This tool can be used as

1. Assembly editor
2. For checking the assembly code
3. Baud rate selection
4. Data bits list
5. Connect serial port
6. Send data

C. Microprocessor (Instruction Flow)

The processor is 32 bits. That means a single instruction is 32 bit wide. The data path is 8 bit wide. So in order to construct a single instruction 4 words are to be concatenated.

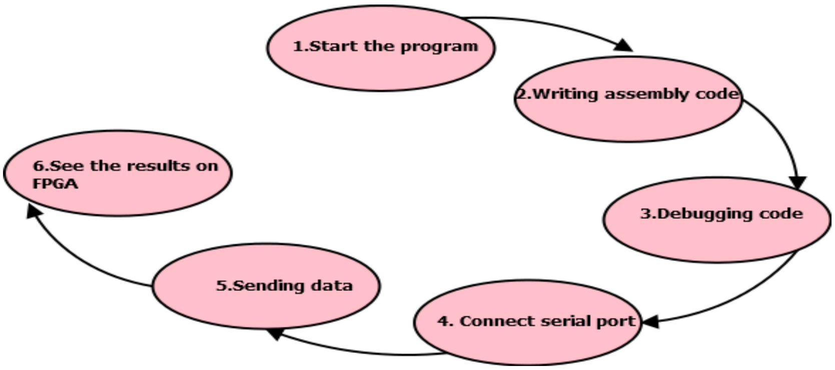


Figure 3-The steps of operation for the whole system

The machine code converted by software tool is sent to the FPGA 8 bits at a time. Now, in order to construct a 32 bit instruction four bytes are concatenated to form one instruction. This memory configuration and the flow for this procedure are presented below in figure 4.

The incoming bytes are stored in memory location named “ram”. The end of reception of the code is understood by last four null bytes. They indicate the code end. And now the loading step is stopped and the execution of these stored instructions is started.

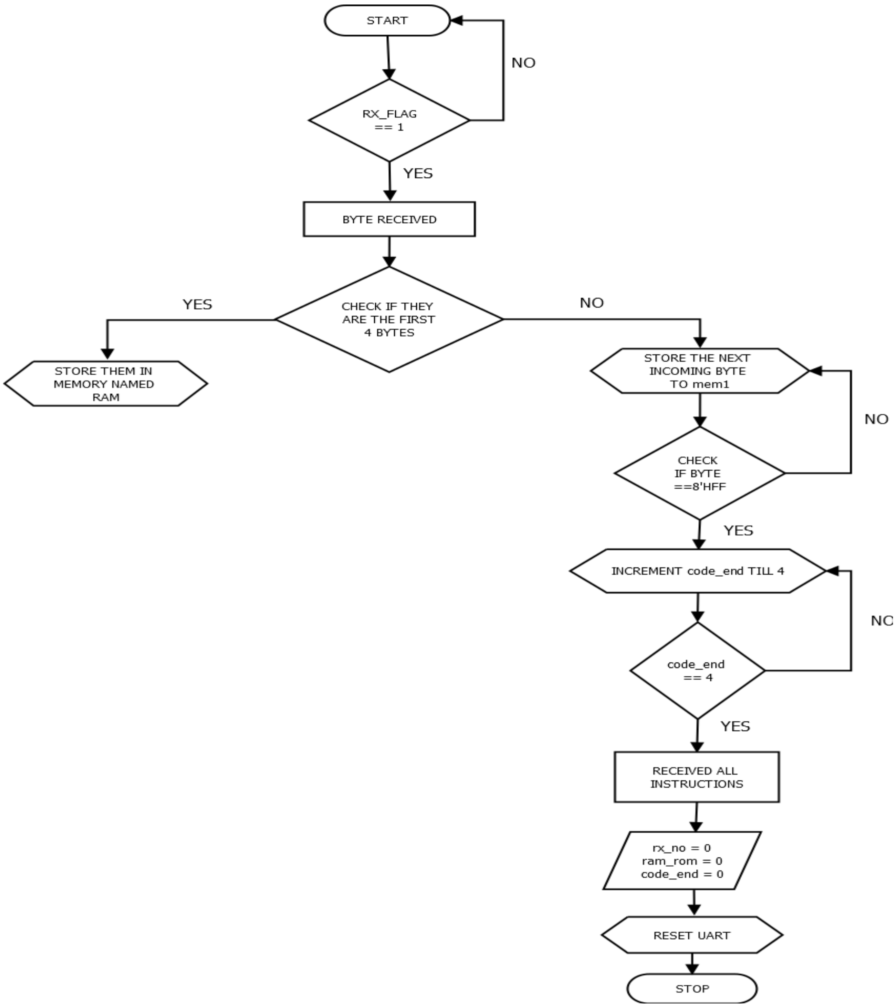

START
NO
RX_FLAG == 1
YES
BYTE RECEIVED
YES
CHECK IF THEY ARE THE FIRST 4 BYTES
NO
STORE THEM IN MEMORY NAMED RAM
STORE THE NEXT INCOMING BYTE TO mem1
NO
CHECK IF BYTE ==8'HFF
YES
INCREMENT code_end TILL 4
NO
code_end == 4
YES
RECEIVED ALL INSTRUCTIONS
rx_no = 0
ram_rom = 0
code_end = 0
RESET UART
STOP

Figure 4-Instruction flow

IV. Results

A) RTL schematics of Modules implementing the overall design.

The overall design is implemented in Xilinx ISE design tool 14.1. The RTL schematics for all modules which together implement the design are presented in figures 5, 6, 7 and 8. These modules are MIPS top module, ALU module, Program Counter and Hex to ASCII module for seven segment display.

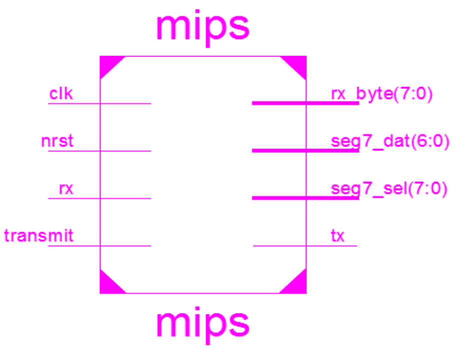


Figure-5 RTL Schematic of MIPS top module

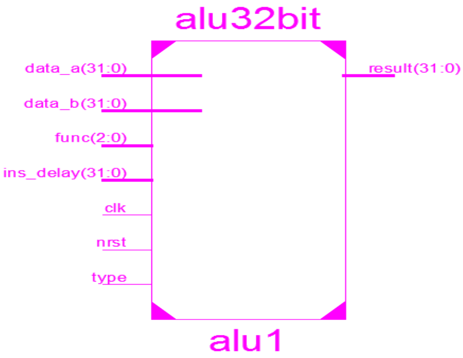


Figure-6 RTL Schematic of ALU module

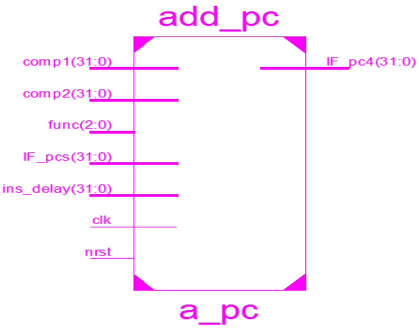


Figure-7 RTL Schematic of Program Counter

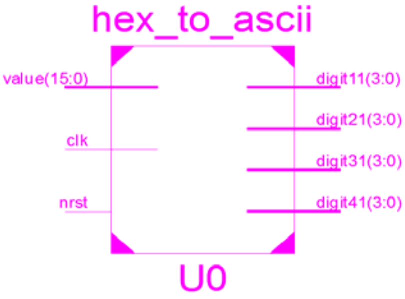
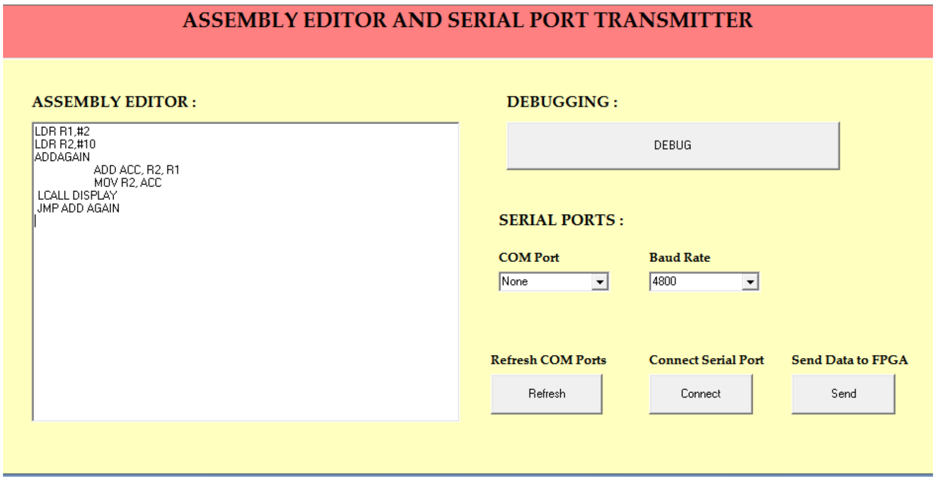


Figure 8-RTL Schematic of Hex to Ascii module.

B) Software tool: Assembly Editor and Serial Port Transmitter

The very basic requirement of the project that is the UART receiver has been implemented and the simulation results are found to be satisfactory. The design of the GUI (Graphical user interface) is shown in figure 9.



ASSEMBLY EDITOR AND SERIAL PORT TRANSMITTER

ASSEMBLY EDITOR :

```
LDR R1,#2
LDR R2,#10
ADDAGAIN
                ADD ACC, R2, R1
                MOV R2, ACC
 LCALL DISPLAY
 JMP ADD AGAIN
```

DEBUGGING :

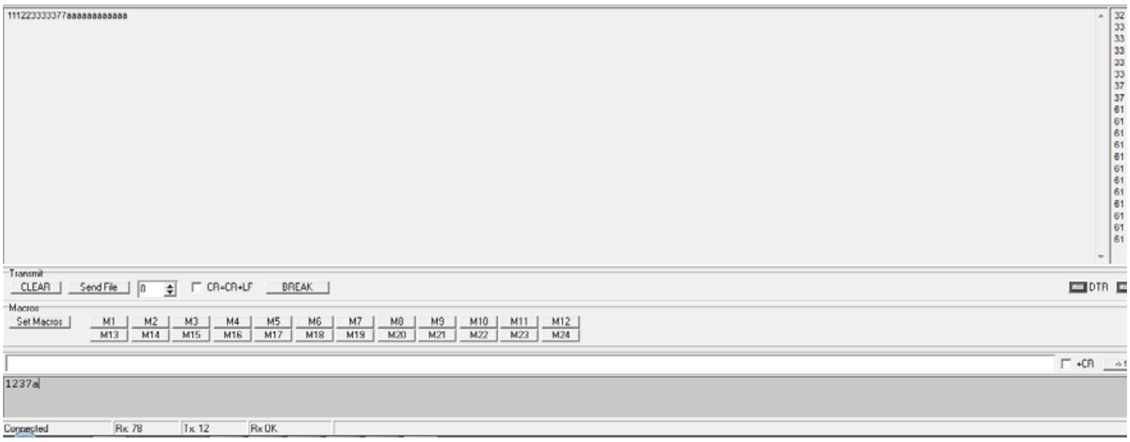
SERIAL PORTS :

COM Port: None

Baud Rate: 4800

Figure 9-Editing text in software tool

C) UART Transmitter Receiver Design on FPGA.



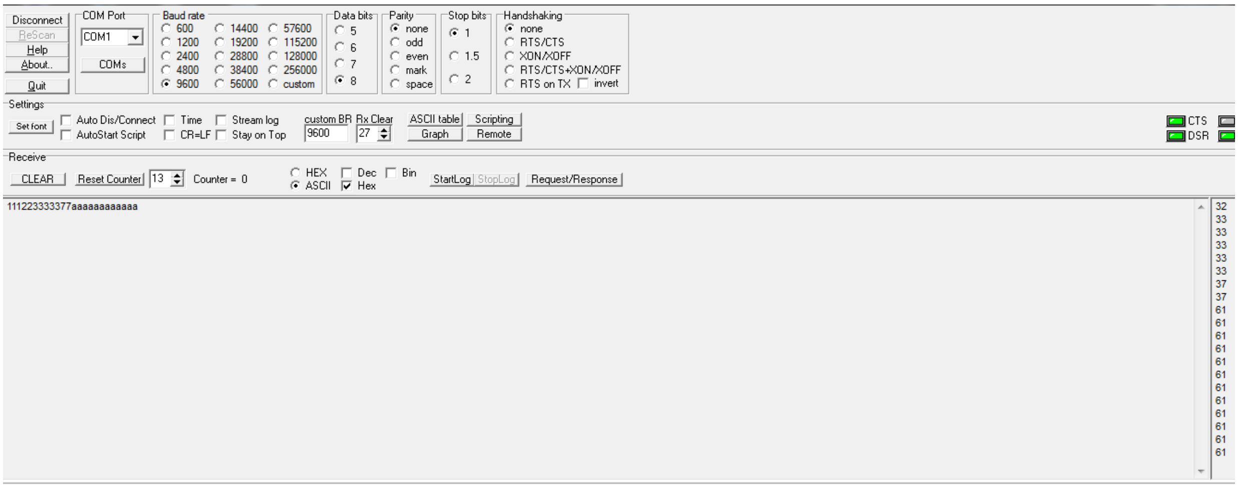
```
111223333377aaaaaaaaaaaa
```

```
32
33
33
33
33
33
37
37
61
61
61
61
61
61
61
61
61
61
61
61
```

1237a

Connected Rx 78 Tx 12 Rx OK

Figure 10-Hyperterminal showing transmission of data at the bottom of diagram, the characters "1237a" are sent to the FPGA and this data is received by FPGA, and again transmitted back to the HyperTerminal.



COM Port: COM1

Baud rate: ○ 600 ○ 1200 ○ 2400 ○ 4800 ● 9600 ○ 14400 ○ 19200 ○ 28800 ○ 38400 ○ 56000 ○ 57600 ○ 115200 ○ 128000 ○ 256000 ○ custom

Data bits: ○ 5 ○ 6 ○ 7 ● 8

Parity: ● none ○ odd ○ even ○ mark ○ space

Stop bits: ● 1 ○ 1.5 ○ 2

Handshaking: ● none ○ RTS/CTS ○ XON/XOFF ○ RTS/CTS+XON/XOFF ○ RTS on TX □ invert

Settings: □ Auto Dis/Connect □ AutoStart Script □ Time □ CR=LF □ Stream log □ Stay on Top custom BR 9600 Rx Clear 27

Receive: 13 Counter = 0 ○ HEX ● ASCII □ Dec ☑ Hex □ Bin

```
111223333377aaaaaaaaaaaa
```

```
32
33
33
33
33
33
37
37
61
61
61
61
61
61
61
61
61
61
61
61
```

Figure 11-The characters received by FPGA is transmitted back to the HyperTerminal.

The UART receiver transmitter is designed in Xilinx ISE design suite 14.1. The functional test of UART is done by dumping the code to Spartan 3 xc3s200a FPGA kit. The characters sent through HyperTerminal are received and stored in

FPGA by the receiver module implemented and sent back to HyperTerminal by the transmitter module designed at the FPGA end.

D) Simulation results showing expected results for MIPS modules and the run time configuration of memory.

The simulation result for design is presented in figure 12. The waveforms show the program counter which increments by 4 each time because each instruction is a 32 bit value divided into 4 bytes each. The instruction is fetched, decoded and executed to perform the required operations. The module presented here is the counter which increments by user defined value of 5.

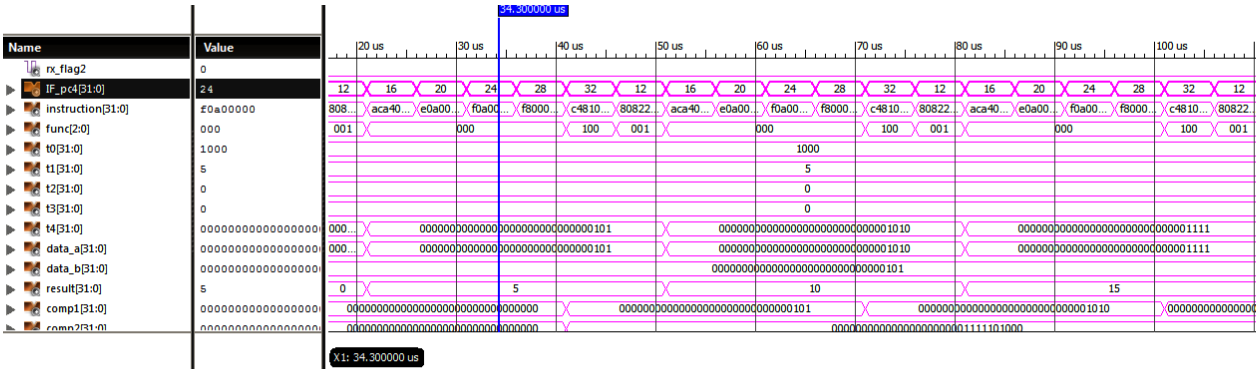


Figure 12-Simulation results showing result of Counter which increments by user defined value '5'.

V. CONCLUSION

The systems that are mentioned in the literature above [3]-[7] do not have run time loading. The proposed system is able to load the assembly code at run time through UART and the software. In all a UART module has been successfully designed in generic form which is operating fine with no error. UART and Processor has been implemented using Xilinx ISE 14.1 simulator. The software tool is designed in visual studio 6.0 version. For a higher end FPGA the resource utilization of the designed system is very low leaving plenty space for implementing other modules in the FPGA. The assembly code that is sent to FPGA is that of a counter. The counter is incremented or decremented by user defined value. The simulation results show how the design works as expected. Also the result is displayed on the seven segment display of xc3s200a FPGA board for hardware testing purpose. The design implemented works at a frequency of 89.85MHz. The design uses a small part of the FPGA resources. The number of LUT's used by the design is 2382 and those used by other implementation [6] are 5454. Hence the FPGA area used by the design is 2.25 times less than that of other implementation [6]. Hence if any modification in the CPU design is required at run time, it is possible in this architecture. In this way the overall system is made very flexible and reliable.

References

1. D. A. Patterson and J. L. Hennessy, Computer Organization And Design. third ed.
2. Wikipedia, "Website-en.wikipedia.org."
3. M. Mounika and A. Shankar, "Design and implementation of 32-bit risc (mips) processor," vol. 4, October 2013.
4. Xizhi Li and Tiecai Li, "ECOMIPS: an economic MIPS CPU design on FPGA," in Proc. of the 4th IEEE International Workshop on System-on- Chip for Real-Time Applications, pp. 291-294, 2004
5. Balpande, R.S. and Keote, R.S., "Design of a system based on FPGA ,Instruction Fetch & instruction decode Decode Module of 32-bit RISC (MIPS) Processor" in Proc. of International Conference on Communication Systems, pp. 409-413, 2011.
6. Ramdas, T. Li-Minn Ang and Egan, G., "FPGA implementation of an integer MIPS processor in Handel-C and its application to human face detection," in Proc. of IEEE Region 10 Conference, Vol. 1, pp. 36-39, 2004.
7. A. R. Ali El Kateeb and L.Azzawi, "Wireless sensor nodes processor architecture and design,"
8. Yongcheng Wang; Kefei Song; , "A new approach to realize UART,"Electronic and Mechanical Engineering and Information Technology(EMEIT), 2011 International Conference on , vol.5, no., pp.2749-2752, 12-14 Aug. 2011.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.
9. Chun-zhi, He; Yin-shui, Xia; Lun-yao, Wang; , "A universal asynchronous receiver transmitter design," Electronics, Communications and Control (ICECC), 2011 International Conference on , vol., no., pp.691-694, 9-11 Sept. 2011
10. B. Roy, "Platform-independent customizable uart soft-core," in Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on, pp. 692{694, 2012.

11. N. Patel, V. Patel, and V. Patel, “Vhdl implementation of uart with status register," in Communication Systems and Network Technologies (CSNT), 2012 International Conference on, pp. 750-754, 2012.
12. F. Yi-yuan and C. Xue-jun, “Design and simulation of uart serial communication module based on vhdl," in Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on, pp. 1-4, 2011.
13. B.Jeevan and M.Neeraja, “Design and simulation of uart protocol based on verilog," Sept 2012.
14. Xilinx, Virtex-6 Family Overview.
15. Altera, Cyclone IV FPGA Device Family Overview.
16. Xilinx, “www.xilinx.com."
17. Harris, “Digital design and computer architecture, elsevier,"

AUTHOR(S) PROFILE

Priyanka Balu Bhor, is pursuing her Master in Engineering degree in VLSI and Embedded System from D. Y. Patil College of Engineering, Pune, affiliated to University of Pune. She also holds Bachelor of Engineering in Electronics and Telecommunication from Indira College of Engineering and Management, Pune.