FPGA Implementation of 32-bit RISC-V Processor with Web-Based Assembler-Disassembler

Etki Gür, Zekiye Eda Sataner, Yusuf H. Durkaya, Salih Bayar
Dept. of Electrical and Electronics Engineering
Faculty of Engineering, Marmara University
Göztepe, 34722 Istanbul, Turkey

{etkigur, zekiyeeda.sataner, yusufdurkaya}@marun.edu.tr
salih.bayar@marmara.edu.tr

***Abstract*—In this study, a pure structural implementation based, 32-bit open source RISC-V processor is presented. The proposed processor is designed using Verilog and it is implemented on Cyclone IV 4CE115 FPGA device available on Altera DE2-115 Board. Additionally, web-based assembler and disassembler tools are developed and published as a part of this project. Before using the target RISC-V processor, the user can generate machine code using the web-based assembler tool. Then, the generated machine code can be downloaded onto the RISC-V processor using UART. The web-based assembler and disassembler tools are developed with technologies such as HTML5, CSS and JavaScript. The proposed processor is a fully functional processor that uses RV32I base integer instructional set with 37 instructions. The amount of hardware resources used by the whole processor circuit is about 43.7% of the Cyclone IV 4CE115 FPGA device and the maximum frequency achieved for the processor is 150MHz without using any timing constraint.**

I. INTRODUCTION

The integrated circuit technology has developed monumentally in the last few decades. Thanks to these developments, there has been drastic changes in chip technology and implementations in a single chip are widespread now. With these, the significance of Instruction Set Architectures (ISAs) has increased rapidly. Reduced Instruction Set Computer RISC based ARM architectures already dominated the mobile systems like Android and Apple and MIPS based architectures used by most gaming consoles RISC-V is a new open ISA based on RISC and it is developed to support research and education [1]. Unlike other ISAs, RISC-V is an open source instruction set architecture first developed by University of California, Berkeley and everyone is allowed to design and manufacture. RISC-V is designed for modern devices like mobile phones, small embedded systems and cloud computers [2]. It is also designed to support 32-bit, 64-bit and 128-bit addresses. Compared to ARM microprocessors, RISC-V is faster and has small footprint size.With all these features, RISC-V can replace ARM microprocessors [3].

In this work, we represent a hardware design architecture for RV32I base integer system for 32-bit addresses only. RV32I is a form of RISC-V that was designed to be adequate for generating compiler targets and supporting modern system environments [4]. In addition, a minimal hardware required was taken into consideration while designing the ISA. The implementation is single core, in-order, non-bus based, single cycle architecture that supports RV32I base integer instruction set fully functionally. One of our goals was to decrease the power needed and lower the cost. It is also targeted to create a non-complex algorithm. The scope of application differs from embedded systems to signal processing, sensor technology and various other areas. After we noticed the lack of an instruction test system for RISC-V, we have decided to develop a web-based assembler and disassembler. To the best of our knowledge, we are the first, who develop a web-based assembler and disassembler for RISC-V architecture. For the connection between web-based assembler-disassembler and the Field Programmable Gate Array (FPGA), the Universal Asynchronous Receiver-Transmitter (UART) protocol was used.

This paper is presented as follows: Section II introduces RISC-V(RV32I) instruction set and various basic instructions. Section III explains the architecture and processor design. Section IV gives insight on simulations and FPGA design. Last but not least, section V explains the conclusion and future work.

II. RV32I BASE INTEGER INSTRUCTION SET

A. Characteristics

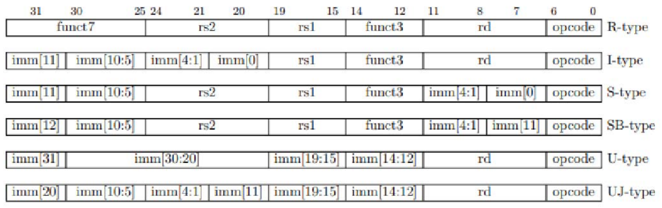
RV32I has four core instruction formats (R/I/S/U). All are a fixed 32 bits in length and must be aligned on a four-byte boundary in memory. Immediate instruction type is split into five different immediate types (I/S/B/U/J). Sign extension uses the left-most bit in instruction. There are a further two variants of the instruction formats (SB/UJ) based on the handling of immediate values [5].

It contains 47 different and unique instructions. RV32I contains 47 unique instructions, though an implementation might cover the eight SCALL/SBREAK/RD [1] instructions with a single SYSTEM hardware instruction that always traps, reducing hardware instruction count to 40 total.

B. Register organization

There are 31 general-purpose registers x1–x31, which hold integer values. Register x0 is hardwired to the constant 0. Standard software calling convention uses register x1 to hold the return address on a call.

TABLE I
INSTRUCTIONS FORMAT


31 30 25 24 21 20 19 15 14 12 11 8 7 6 0
funct7 | rs2 | rs1 | funct3 | rd | opcode | R-type
imm[11] | imm[10:5] | imm[4:1] | imm[0] | rs1 | funct3 | rd | opcode | I-type
imm[11] | imm[10:5] | rs2 | rs1 | funct3 | imm[4:1] | imm[0] | opcode | S-type
imm[12] | imm[10:5] | rs2 | rs1 | funct3 | imm[4:1] | imm[11] | opcode | SB-type
imm[31] | imm[30:20] | imm[19:15] | imm[14:12] | rd | opcode | U-type
imm[20] | imm[10:5] | imm[4:1] | imm[11] | imm[19:15] | imm[14:12] | rd | opcode | UJ-type

It is also possible to encode an ISA with 16 registers but it lowers the efficiency and increases the instruction count. To avoid intermediate instruction sizes RV32I encoded with 32 registers.

C. Instructions

RV32I instructions are classified based on the following instruction bits; instruction[30], function3[14:12], Opcode[6:0]. It has 7 different instruction category and every category is explained in Table I.

TABLE II
INSTRUCTION TYPE, OPCODE AND DEFINITION

CATEGORY	OPCODE	INSTRUCTION	DEFINITION
Integer Register-Immediate Instructions (I-Type)	0010011	addi, slti, sltiu, ori, xori, andi, slli, srli, srai	addi rd, rs1, imm Adds rs1 and imm and then it writes the answer to the rd
Integer Register-Register Operations (R-Type)	0110011	add, sub, sll, slt, sltu, xor, srl, sra, or, and	add rd , rs1, rs2 Adds rs1 and rs2 and then it writes the answer to the rd
Integer Computational Instructions (U-Type)	0110111 0010111	lui auipc	lui rd,imm Load immediate into upper bits of word
Unconditional Jumps (UJ, I-Type)	1101111	jal, jalr	jal rd,imm Jumps to adress and place return adress to the rd
Conditional Branches (SB-Type)	1100011	beq, bne, blt, bltu, bge, bgeu	beq rs1,rs2,imm If rs1 and rs2 is equal, branchs into the immediate value adress
Load Instruction (I-Type)	0000011	lw, lh, lb, lhu, lbu	lw rd,rs1,imm Adds rs1 and imm and then writes the corresponding value from memory to rd
Store Instruction (S-Type)	0100011	sw, sh, sb	sw rs1,rs2,imm Adds rs1 and imm and then writes the value of rd to the corresponding adress in memory

III. DEVELOPMENT AND IMPLEMENTATION OF RISC-V ON FPGA

32-bit designs are sufficient for embedded applications that do not require complex processing. A single cycle design was deemed suitable for this study. In the following sections, the design will be explained in detail.

A. Overview

There are various modules that make a single cycle datapath. All modules run at the same time in a single cycle, the control signals determine the progression that will produce the required output. After the instruction is fetched from the instruction memory, the process starts with decode and control stage [6].

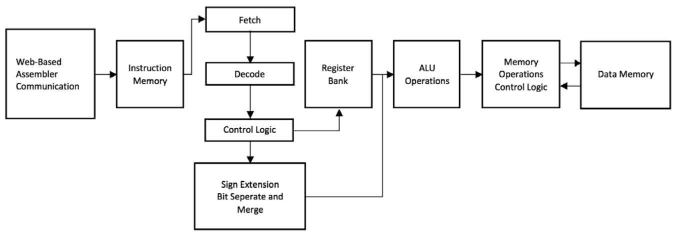

Web-Based Assembler Communication
Instruction Memory
Fetch
Decode
Control Logic
Sign Extension Bit Seperate and Merge
Register Bank
ALU Operations
Memory Operations Control Logic
Data Memory

Fig. 1. Processor Design

Register file contains 32 registers from 0 to 31. The zero register (0th register) has a constant value of 0. Also, this module comprises the register addressing and read and write logic. ALU deals with arithmetic register-register and register-immediate operations for this module. ALU does not calculate the address for jump and branch operations. For branch instructions, ALU only calculates the condition. Sign-Extension, Bit separate and Merge unit provides properly reordered and sign extended immediate operands for ALU. One of the operands comes from this unit whilst the other one comes from Register Bank. The value is selected in this module according to the appropriate instruction. The memory operation control logic module has been added to generate various store/load instructions. Thanks to this module memory reading and recording is performed.

B. Micro-Architecture

In general, the system consists of fetch, decode and control stage. Fetch stage calculates the current program counter and predicts the next PC. It keeps the PC value in the 32-bit register and calculates the next PC value as PC + 4. However, when this process is performed, what is the instruction is examined, and if the jump and branch processes are performed, the PC value is adjusted accordingly. After the instruction is decoded, the control stage determines the progress.

During the decode stage, the instructions taken from the instruction memory are resolved. This analysis will determine the signals to go to the control process. They are the key determinants of progress in a single cycle. The control signals opcode(0 to 6 bits), funct3(12 to 14 bits) and the 30th bit of the instruction are processed in the control stage. 12 signals

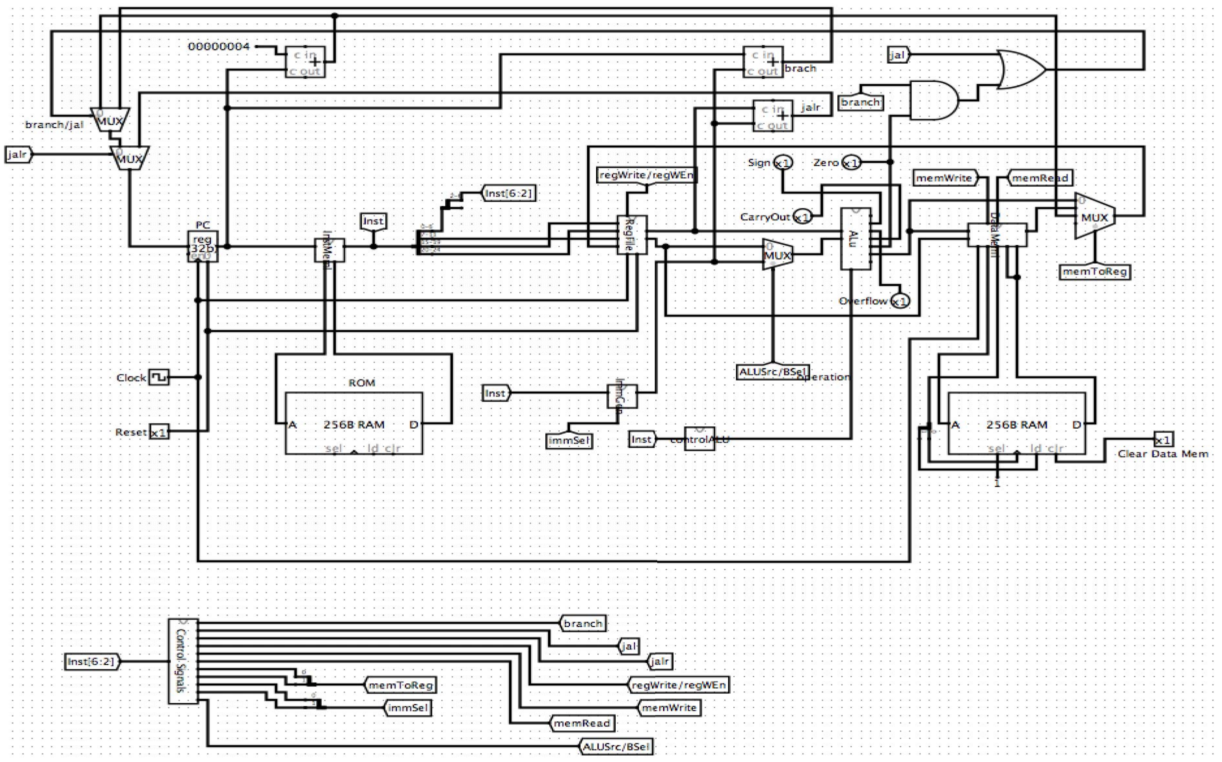


Fig. 2. Micro architecture of RV32I ISA based RISC-V

are output from the control stage depending on the design. For example, the 4-bit X signal for ALU control specifies the operations to be performed in the ALU connected to the instruction. The MemRead signal also specifies whether to

read from memory. It controls the modules by working in a similar way on the remaining signals.

Depending on the instructions, the changed instantaneous values are converted to 32-bit so that they can be processed by the Sign-Extension module. It is transmitted to the relevant module. The selected operation is performed by two 32 bit inputs to the ALU module by ALU control. The output is either sent to the memory operations, or sent to the Register File for writing to the register depending on the control signals. Memory operations load and store instructions transfer a value between the registers and memory. LW, LH, LB, LBU, LHU instructions load 8-bit, 16-bit and 32-bit value from memory into rd. 8-bit and 16-bit values load from memory, then sign-extends or zero extends to 32-bits before storing in rd. SW, SH, SB instructions store 32-bit, 16-bit, and 8-bit values from the low bits of register rs2 to memory [7].

If the operation is to be summarized, the PC calculates the address of the instruction to be executed. The inputs required for the operation of the modules are found by the control unit. The values from the register file are reached to get the register values to be used for instruction (rs1, rs2, rd). The inputs required for ALU operations are selected from the multiplexers. The input selection is again determined by the inputs provided by the control unit. According to control signals ALU output is written to the register file or used to determine the address in the data memory or to determine the address in the PC. The PC specifies the address of the next instruction.

IV. SIMULATION AND FPGA IMPLEMENTATION

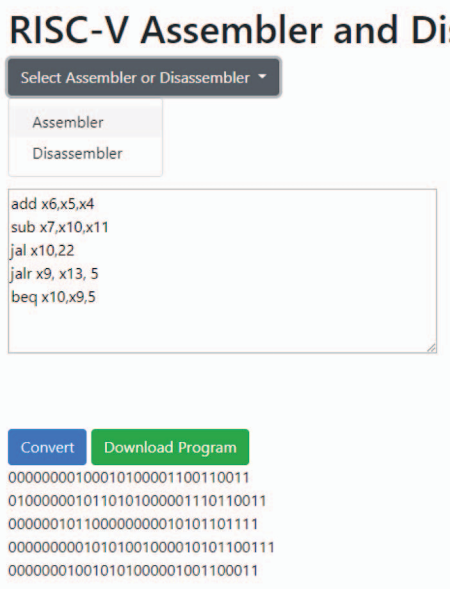
Using the web-based assembler and disassembler that we have developed, we are giving inputs to the program. Communication between this system and FPGA provided by the UART. The given input goes to FPGA and simulation works based on the input. The proposed processor is designed using Verilog and it is implemented on Cyclone IV 4CE115 FPGA device available on Altera DE2-115 Board.

A. *Web-Based Assembler-Disassembler*

Due to the lack of a web-based RISC-V Assembler and Disassembler, people spend unnecessary time translating the assembly code to machine code. The application that takes the RISC-V Assembly Code as an input from a ’text area’ in a Web-site then converts it to the Machine Code as an output was developed.This application sends the generated machine code into the FPGA via USB using UART-serial communication in local network. A website that completes bidirectional translation between machine code and assembly language was published.

B. *Simulation*

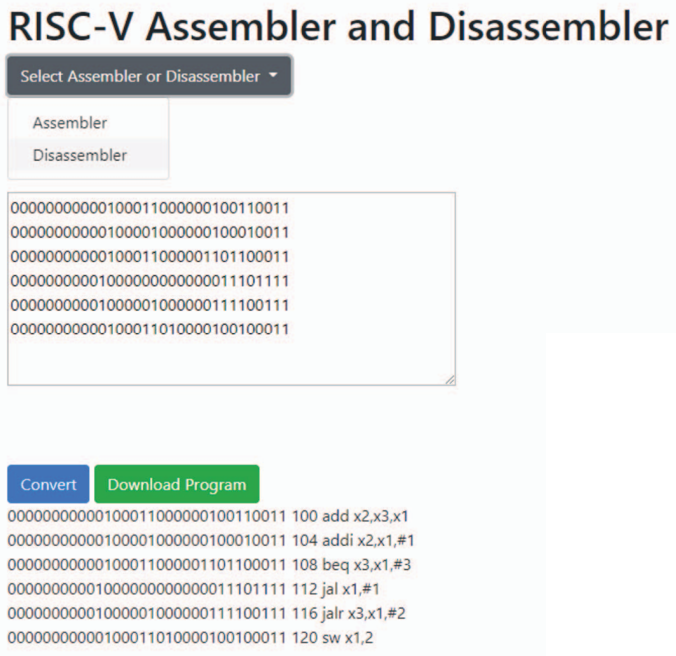
All the simulation procedures have been done in Logisim. Logisim is an educational tool for designing and simulating


RISC-V Assembler and Disassembler
Select Assembler or Disassembler
Assembler
Disassembler
Convert
Download Program

```
add x6,x5,x4
sub x7,x10,x11
jal x10,22
jalr x9, x13, 5
beq x10,x9,5
```

```
00000000100010100001100110011
01000000101101010000001110110011
00000010110000000000010101101111
00000000010101001000010101100111
0000000100101010000001001100011
```

Fig. 3. RV32I Web-Based Assembler


RISC-V Assembler and Disassembler
Select Assembler or Disassembler
Assembler
Disassembler
Convert
Download Program

```
00000000000100011000000100110011
00000000000100001000000100010011
00000000000100011000001101100011
00000000001000000000000011101111
00000000001000001000000111100111
00000000000100011010000100100011
```

```
00000000000100011000000100110011 100 add x2,x3,x1
00000000000100001000000100010011 104 addi x2,x1,#1
00000000000100011000001101100011 108 beq x3,x1,#3
00000000001000000000000011101111 112 jal x1,#1
00000000001000001000000111100111 116 jalr x3,x1,#2
00000000000100011010000100100011 120 sw x1,2
```

Fig. 4. RV32I Web-Based Disassembler

logic circuits. 32-bit RISC-V processor was implemented in a structural base and fully supports RV32I base integer instruction set. 256KB RAMs was used for both instruction memory and data memory. Some set of test cases were developed with 38 different instructions and tested on Logisim Simulation. A schematic of Logisim circuit can be seen in Figure 2.

C. Prototype

We have developed an experimental FPGA prototype for this work. The proposed processor is designed using Verilog and it is implemented on Cyclone IV 4CE115 FPGA device available on Altera DE2-115 Board. Both the instruction memory and data memory is designed with 256KB RAMs. To display the output, the LCD on FPGA was used. The amount of hardware resources used by the whole processor circuit is about 43.7% of the Cyclone IV4CE115 FPGA device and the maximum frequency achieved for the processor is 150MHz without using any timing constraint.

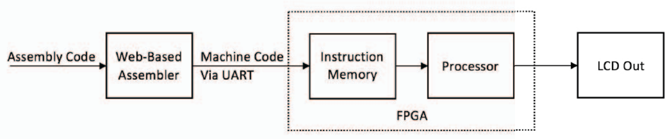


Fig. 5. Work Flow Diagram

V. CONCLUSION

In this paper, we have presented an 32-bit open source RISC-V processor and our aim was to develop a simple, single-cycle embedded system to overcome the complexities in usage of modern embedded system designs. 256 KB RAMs are used for both instruction memory and data memory. Maximum frequency achieved for the processor is 150MHz without using any time constraint and 43.7% amount of hardware resources used by the processor circuit. As a future work, we are intended to build a faster processor version for specific and general applications. With the fast growing community of RISC-V, it will be a core topic in various areas like Internet of Things [8] or Real-Life Embedded Systems.

REFERENCES

[1] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanović. The risc-v instruction set manual, volume i: User-level isa, version 2.0. Technical Report UCB/EECS-2014-54, EECS Department, University of California, Berkeley, May 2014.

[2] Wikipedia. RISC-V — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=RISC-Voldid=843228532, 2018. [Online; accessed 04-June-2018].

[3] C. Duran, D. L. Rueda, G. Castillo, A. Agudelo, C. Rojas, L. Chaparro, H. Hurtado, J. Romero, W. Ramirez, H. Gomez, J. Ardila, L. Rueda, H. Hernandez, J. Amaya, and E. Roa. A 32-bit risc-v axi4-lite bus-based microcontroller with 10-bit sar adc. In 2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS), pages 315–318, Feb 2016.

[4] D. K. Dennis, A. Priyam, S. S. Virk, S. Agrawal, T. Sharma, A. Mondal, and K. C. Ray. Single cycle risc-v micro architecture processor and its fpga prototype. In 2017 7th International Symposium on Embedded Computing and System Design (ISED), pages 1–5, Dec 2017.

[5] D. Patterson. 50 years of computer architecture: From the mainframe cpu to the domain-specific tpu and the open risc-v instruction set. In 2018 IEEE International Solid - State Circuits Conference - (ISSCC), pages 27–31, Feb 2018.

[6] A. Raveendran, V. B. Patil, D. Selvakumar, and V. Desalphine. A risc-v instruction set processor-micro-architecture design and analysis. In 2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA), pages 1–7, Jan 2016.

[7] A. Waterman, Y. Lee, R. Avizienis, H. Cook, D. Patterson, and K. Asanovic. The risc-v instruction set. In 2013 IEEE Hot Chips 25 Symposium (HCS), pages 1–1, Aug 2013.

[8] C. Duran, L. E. Rueda G., A. Amaya, R. Torres, J. Ardila, L. Rueda D., G. Castillo, A. Agudelo, C. Rojas, L. Chaparro, H. Hurtado, J. Romero, W. Ramirez, H. Gomez, H. Hernandez, and E. Roa. A system-on-chip platform for the internet of things featuring a 32-bit risc-v based microcontroller. In 2017 IEEE 8th Latin American Symposium on Circuits Systems (LASCAS), pages 1–4, Feb 2017.