Design and Implementation of RISC-V Subsystems using Verilog HDL

Shruthi[1], Dr. Jamuna S[2]

[1]PG Student (M.Tech in VLSI Design and Embedded Systems), Department. of ECE, DSCE Bangalore, Karnataka
[2]Professor, Department. of ECE, DSCE, Bangalore, Karnataka

---***---

Abstract - *RISC-V is a modern Instruction Set Architecture (ISA) with standard open architecture which is designed for different applications. RISC-V processor is suitable for applications in embedded processors, IoT applications, machine learning, and military applications. To design high-performance systems, the speed of operations called throughput and the number of calculations per unit time become essential. We often go for a technique called pipelining. Pipelining is a standard feature used in RISC-V processors. Pipelining involves executing multiple instructions per cycle. The five-stage pipeline (fetch, decode, execute, memory, write back) processor is implemented. This work includes the design of functional blocks of 32-bit RISC-V processor like Branch Prediction Unit (BPU), Forwarding unit, Floating point unit, Floating point register. This pipelining feature may lead to different types of hazards like structural hazard, data hazard and control hazard in the processor design. BPU and Forwarding unit are used to handle control and data hazard respectively. Floating point register unit is designed to improve the speed of the processor. These subsystems are used in different stages of pipelining and are implemented using Verilog HDL.*

Keywords – RISC-V ISA (Instruction Set Architecture), Pipelining, Branch Prediction Unit (BPU), forwarding unit, Floating point register, Floating point unit, Verilog HDL.

1. INTRODUCTION

RISC-V is one of the standard free and open architecture used in different domains like academics and industrial applications. Nowadays, the processors are evaluated by the performance, speed of operation, number of calculations per unit time and Instruction set architecture (ISA).

Pipelining is a unique feature supported in RISC-V ISA. It can improve the performance of the processor by increasing the throughput. However, it also introduced hazards to the processor. The hazard can be categorizes into three different types: structural hazard, data hazard and control hazard. These hazards can be handled with the implementation of functional blocks like Branch Prediction unit, forwarding unit which handles control hazard data hazard respectively and there by increases the throughput of the processor. Due to its open and free ISA this can also be used in military and defense applications.

This paper includes the design and implementation of sub-blocks of 32-bit, 5-stage superscalar pipelined architecture. The sub-blocks include Floating point unit, Branch prediction unit, forwarding unit, operand logic and floating point register file. These blocks are implemented in Xilinx ISE using Verilog HDL.

Section 2 briefs on the related work. Section 3 introduces an overview of the processor architecture. Section 4 describes implementation of sub blocks. Section 5 discusses the simulation results. Conclusion is provided in section 6.

2. RELATED WORK

[2] Briefs on different cases for choosing the RISC-V open ISA. The different cases include (a).Case for a free open ISA, (b).Case for RISC as free, open ISA style, (c).Case for using an existing RISC free, open ISA and (d).Case for RISC-V as the RISC-free open ISA.

[3] Exposes a 32-bit processor which includes 5 stages of pipeline. It supports different set of instructions like integer, atomic and multiply/divide instructions. It has memory subsystems and memory error control. The design is implemented on Virtex-board and is able attain 100MHz peak clock frequency.

[4] Articulates the processor micro architecture design and the possible consequences of instruction set architecture on micro architecture design. The simulation of this micro architect is carried out using Blue-spec Systemverilog and is synthesized on FPGA. Comparison of the synthesis result is done with respect to same efforts on RISC-V based processor.

The work presented in [5] is centered on open source RISC-V ISA. The processor is designed for aiming low cost embedded devices. The processor is executed in verilog hardware description language and is then prototyped FPGA board. It was able to achieve maximum frequency of 32MHz and the power is evaluated using Xilinx power analyzer and is found to be 7.93mW.

In [6] single precision arithmetic operations along with the simple ALU (Arithmetic and Logic unit), general purpose registers, program and data memories and complete instruction set for a pipelined 32-bit RISC processor is implemented in Altera DE2 FPGA.

3. PROPOSED WORK

Fig-1 shows 32-bit, 5-stage superscalar pipelined architecture [1]. Instruction Fetch (IF), Instruction decode (ID), Execute (EX), Memory(MEM) and write-back (WB) are the different stages of pipelining. In Instruction fetch stage, program counter (PC) is used to generate program memory. The program counter value is then utilized to fetch instructions from IF stage.

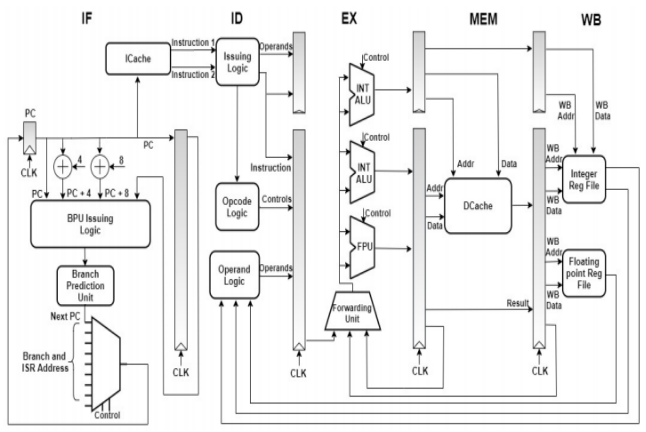

Fig-1:5-stage superscalar pipeline architecture [1]

Two instructions are from I-Cache are fetched in issuing logic and sent to decode stage. Instruction unit issues the instructions to pipeline, depending on data dependencies between different instructions. The data that is present in data memory is read by decode stage and it decodes the instructions and hence thereby gives select signals for the multiplexers so that it can feed the data to forwarding unit of execution stage. Operand and opcode logic are obtained in the instruction decode stage and are fed to the execution stage for further operations. In Execution stage of Fig-1, there are three ALUs out of which two are integer ALUs and one floating point Unit with the forwarding unit. The ALUs are responsible to perform arithmetic and logic operations. Forwarding unit here toggles the multiplexer so that the data from different stages are used in EX stage. The results from execution stage are transferred to memory stage when instructions like load and store are used and for the instructions other than load or store the results are forwarded to write back stage. Different register files for integer and floating point instructions are used in WB stage.

4. DESIGN IMPLEMENTATION OF SUBBLOCKS

This section of paper includes design implementation of different sub-blocks of processor like floating point unit, forwarding unit, branch prediction unit, operand logic and floating point register file.

1). Floating Point Unit (FPU)

This unit is designed to compute different arithmetic operations on floating point numbers. The floating point unit that is represented in this work includes three arithmetic operations; they are addition, subtraction and multiplication.

IEEE 754 standard provides a method for computation with floating-point numbers [7].

Single precision floating point format occupies 32 bits in computer memory. The 32 bit format of floating point number is represented in Fig.2 includes sign bit of 1-bit length, 8-bit exponent and mantissa is of 24 bits length among which 23 bits are stored and 1 bit is implicit 1.

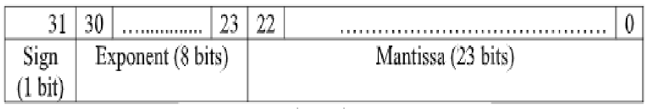

Fig-2: IEEE 754 standard format

31st bit is sign bit used to determine the sign of a number. If sign bit is 1 it denotes negative number and if it is 0 it represents positive number. Exponent is 8 bit length and is a signed integer from -128 to 127. Mantissa or significant an implicit leading bit with value 1 and it has 23 bits to the right side of binary point.

Floating point multiplication:

$X3=X1*X2= (-1)^{s1} (M1 \times 2^{E1}) * (-1)^{s2} (M2 \times 2^{E2})$.....(1)

Equation (1) shows the notation of multiplication. S1 and S2 represent the sign bits , E1 and E2 represent exponent bits and M1 and M2 are Mantissa bits of multiplier and multiplicand X1 and X2 respectively.

The result of multiplication i.e. X3 is obtained by the following steps.

1. If any one of the operand that is X1 or X2 is 0 or if both the operands are 0, then the result of multiplication that is X3 is set to zero.

2. XOR S1 and S2 which are signed bits of the multiplicand and the multiplier respectively, to obtain the resultant sign bit.
3. The resultant mantissa (M3) is obtained by multiplying M1 and M2 (M1*M2), the resultant M3 is then truncated or rounded off for 24 bits.
4. The exponent of the result (E3) is obtained adding E1 and E2 and subtracting this sum by the bias value. E3=E1+E2-bias.
5. Normalization of the result obtained in above step is done by incrementing or decrementing the exponent.

Floating Point addition and subtraction:

$$X3 = X1 + X2 = (M1 \times 2^{E1}) +/- (M2 \times 2^{E2}) \ldots\ldots\ (2)$$

Equation-(2) shows the notation of addition/subtraction. The result of addition/subtraction i.e. X3 is obtained by the following steps.

1. If the exponents E1 and E2 are equal, then add the X1 and X2.
2. Magnitude of X1 should be higher than magnitude of X2, else interchange both the operands such that magnitude of X1 is greater than magnitude of X2.
3. Since the magnitude of X1 is greater than X2, the resultant exponent value is evaluated as E3=E1.
4. Exponents difference i.e. Exponent difference = (E1-E2), is calculated.
5. To make X1 and X2 equal, left shift the decimal point of M2 by the difference obtained in previous step.
6. Depending on sign bits S1 and S2, perform addition or subtraction.
7. If S1 and S2 are equal then perform addition else perform subtraction of mantissas.
8. Normalize the resultant mantissa (M3) if needed. Initial exponent result E3=E1 and the format 1.m3 must be adapted as concerned to the normalization of mantissa.

2). Branch Prediction Unit (BPU)

This unit is present in fetch stage. It decides the branch instruction; the branch prediction is carried out in the execution stage of the pipelining architecture. When the prediction is predicted wrongly then the PC value will be updated. To assign the PC value we have next PC logic in Fetch stage, as shown in Fig-3.

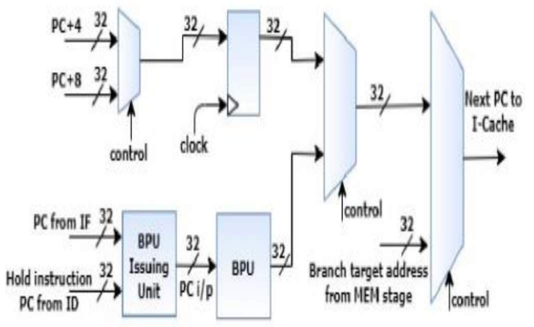

Fig-3: Next-PC Logic [1]

Whenever the PC value returns to the same address or whenever rollback is declared in decode stage, then PC+4 that is the next program counter value is assigned as the address of next instruction else it assigns the next PC value that is PC+8. In the instruction fetch stage, if the prediction is a branch instruction, then the program counter value of that particular instruction is loaded from the BPU. And if the prediction by BPU is incorrect, then the accurate value of program counter from the memory stage will be loaded [1].

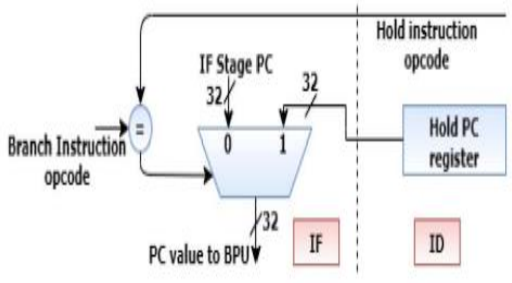

Fig-4: Branch Prediction Issuing Unit [1]

Figure 4 represents the branch prediction issuing unit, which basically assigns the PC value to branch prediction unit. This issuing unit for branch prediction is implemented in fetch stage. If the instruction that is held in decode stage is a branch instruction, then the program counter value from hold register is forwarded to branch prediction unit and if the instruction that is held in decode stage is not a branch instruction, then the program counter value from instruction fetch is forwarded to branch prediction unit [1].

3). Forwarding Unit

Forwarding unit is used to control the data hazards that appear from data dependences from different stages of pipeline. This unit is implemented in execution stage.

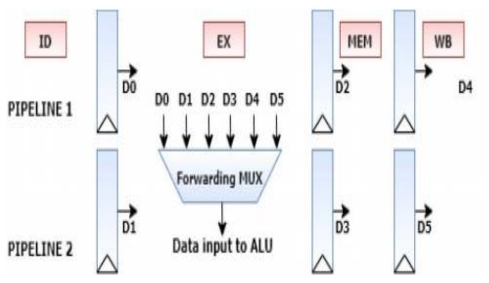

Fig-5: Forwarding Unit [1]

The processor shown in Fig-1 has 2- integer ALUs and 1-Floating Point Unit and hence it requires three forwarding unit, one of which is shown in Fig-5. When the instruction is fetched from MEM and WB at the same time or when the data hazard occurs between instructions in MEM and WB stage; the data that is present in MEM stage is fed as output to ALU as it is having the latest instruction [1].

4). Operand Logic

Operand and opcode logic is implemented in Instruction Decode stage. This is accountable for accessing data that is present in data memory. Based on the instruction that it accessed from the memory operands and opcode are generated, which are then fed to the next stage that is execution stage in which the operations on operands are performed based on opcode logic.

5). Floating Point register file

This is present in the write back stage. It includes the data and address from the memory stage. Data present in floating point register file is used by floating point unit of execution stage. For high performance applications floating point computational logic has been an essential component [1].

5. RESULTS

The subsystems that are discussed in section III are designed Verilog HDL using Xilinx ISE and ModelSim Altera-6.4a. The simulation results for the same are shown below.

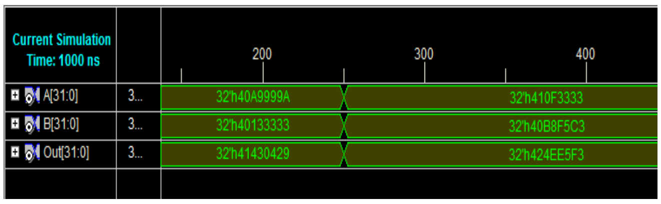
Current Simulation Time: 1000 ns
200
300
400
A[31:0] 3... 32'h40A9999A 32'h410F3333
B[31:0] 3... 32'h40133333 32'h40B8F5C3
Out[31:0] 3... 32'h41430429 32'h424EE5F3

Fig-6: Simulation result of multiplication of two floating point values

Fig-6 shows the simulation waveform of floating point multiplication for different values of multiplier and multiplicand. The multiplication result for inputs a=5.3, b =2.3 and a=8.9 b=5.7 respectively with a delay of 100ns are shown.

Fig-7 represents the simulated waveform for addition of floating point number with different values of addends.

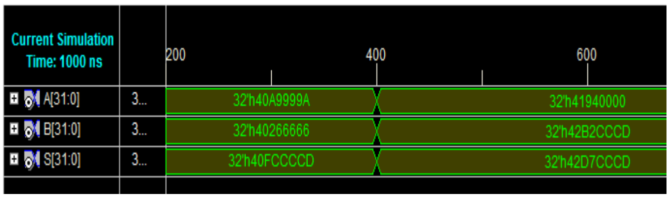
Current Simulation Time: 1000 ns
200
400
600
A[31:0] 3... 32'h40A9999A 32'h41940000
B[31:0] 3... 32'h40266666 32'h42B2CCCD
S[31:0] 3... 32'h40FCCCCD 32'h42D7CCCD

Fig-7: Simulated waveform of floating point addition

It represents addition result for inputs a=5.3, b =2.6, a=18.5, b=89.4 and a=21.6, b=10.2 respectively with a delay of 100ns.

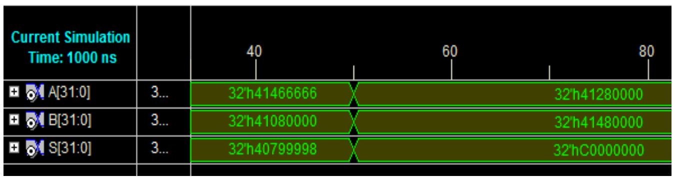
Current Simulation Time: 1000 ns
40
60
80
A[31:0] 3... 32'h41466666 32'h41280000
B[31:0] 3... 32'h41080000 32'h41480000
S[31:0] 3... 32'h40799998 32'hC0000000

Fig-8: Simulated waveform of floating point subtraction

Fig-8 shows the simulation result of floating point subtraction for different values of minuend and subtrahend. It represents the subtraction result for inputs a=12.4, b =8.5, a=10.5, b=12.5 and a=5.2, b=1.1 respectively with a delay of 100ns.

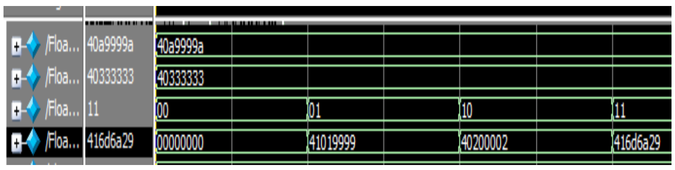
/Floa... 40a9999a 40a9999a
/Floa... 40333333 40333333
/Floa... 11 00 01 10 11
/Floa... 416d6a29 00000000 41019999 40200002 416d6a29

Fig-9: Simulation result of FPU

The simulation results of floating point unit (FPU) are represented in Fig-9; it shows different operations performed on operands based on the select lines. As the designed floating point unit include only three operations like addition subtraction and multiplication the result for select line value “00” is set to zero. If the select lines are 01, 10 and 11, it performs addition subtraction and multiplication respectively.

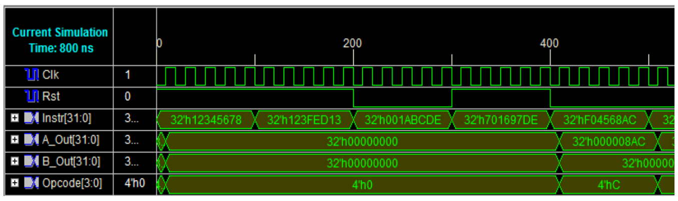

Fig-10: Simulation result of operand logic

Fig-10 shows the simulation result of and operand and opcode logic.

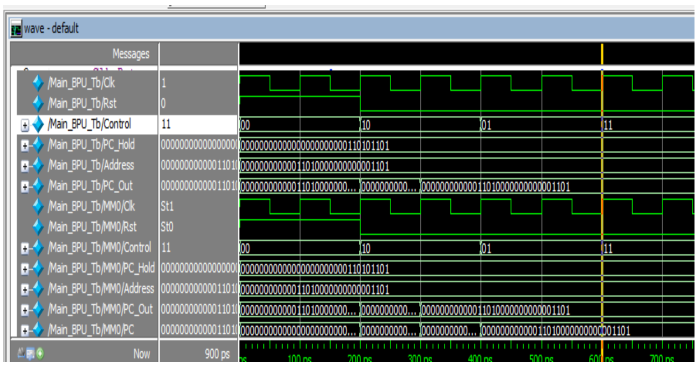

Fig-11: Simulation result of branch prediction unit

Fig-11 shows the simulation result of a branch prediction unit. It will update the program counter value.

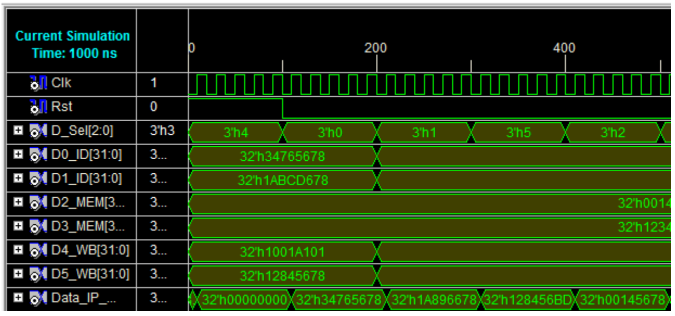

Fig-12: Simulation result of forwarding unit

The above Fig-12 shows the simulation results of forwarding unit, which forwards the data from different stages like decode, memory and write-back stage to ALU in execution stage.

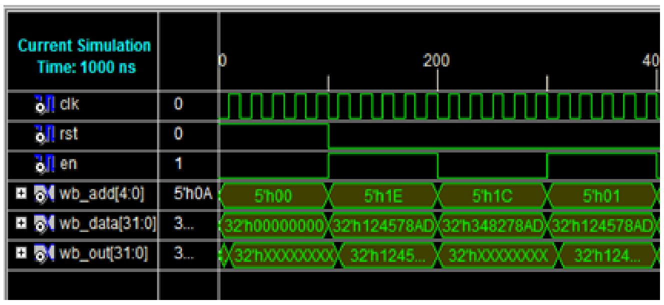

Fig-13: Simulation result of floating point register file

Fig-13 shows the simulation results of floating point register file; it outputs the data based on WB address and WB data.

6. CONCLUSION

This paper briefs on the 32-bit, 5-stage superscalar architecture. The processor supports the pipelining feature which increases the performance of the architecture. 5-stages of pipeline are IF, ID, EX, MEM and WB stage. This work particularly includes the implementation on sub-blocks of the processor like Floating point unit, Branch prediction unit, forwarding unit, operand logic and floating point register file presented in different stages of the pipelined processor. Implementation of these blocks is done using Verilog HDL and is simulated in Xilinx ISE.

REFERENCES

[1] Gokulan T, Akshay Muraleedharan, Kuruvilla Varghese, “Design of a 32-bit dual pipeline superscalar RISC-V processor on FPGA” 23rd Euromicro Conference on Digital System Design(DSD) 2020

[2] K. Asanovi and D. A. Patterson, “Instruction sets should be free: The case for risc-v,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146, Aug 2014

[3] Suseela Budi, Pradeep Gupta, Kuruvilla Varghese, Amrutur Bharadwaj, “A RISC-V ISA compatible processor IP for SoC”, International Symposium on Devices, Circuits and Systems, March 2018

[4] S. Budi, P. Gupta, K. Varghese and A. Bharadwaj, "A RISC-V ISA compatible processor IP for SoC," 2018 International Symposium on Devices, Circuits and Systems (ISDCS), 2018.

[5] D. K. Dennis et al., "Single cycle RISC-V micro architecture processor and its FPGA prototype," 2017 7th International Symposium on Embedded Computing and System Design (ISED), 2017.

[6] Jinde Vijay Kumar, Chintakunta Swapna, Boya Nagaraju, Thogata Ramanjappa, "FPGA based implementation of pipelined 32-bit RISC processor with Floating point unit", Journal of Engineering Research and Applications. ISSN : 2248-9622, Vol. 4, Issue 4(Version 5), April 2014, pp.01-07

[7] "IEEE Standard for Floating-Point Arithmetic," in IEEE Std 754-2019 (Revision of IEEE 754-2008), vol., no., pp.1-84, 22 July 2019