RISC V Integer Processor Implementation

Design Specifications:

The processor has been designed based on the RISC V-32I instruction set architecture and is pipelined with five stages. It supports three basic categories of instructions:

1. Computational Instructions (Typically referred to as R-type or I-type instructions), that directly involve computations (necessarily integer-based in this case) performed in the Algorithmic and Logical Unit.
2. Memory Instructions (Instructions that help the processor communicate with the memory).
3. Control Transfer Instructions (Instructions that involve transfer of control, typically useful for execution of loops and functions).

Under the above specified categories, 12 instructions were selected from the RISC V instructions list for the design. The instructions along with their formats are specified below (more details can be found on the website):

1. add rd, rs1, rs2

funct7	rs2	rs1	funct3	rd	opcode

2. sub rd, rs1, rs2

funct7	rs2	rs1	funct3	rd	opcode

3. addi rd, rs1, imm

imm[11:0]	rs1	funct3	rd	opcode

4. slli rd, rs1, shamt

funct7	shamt	rs1	funct3	rd	opcode

5. srli rd, rs1, shamt

funct7	shamt	rs1	funct3	rd	opcode

6. srai rd, rs1, shamt

funct7	shamt	rs1	funct3	rd	opcode

7. lw rd, imm (rs1)

imm[11:0]	rs1	funct3	rd	opcode

8. sw rs2, imm(rs1)

imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode

9. beq rs1, rs2, label

imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode

10. bge rs1, rs2, label

imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode

11. jal ra, label

imm[20|10:1|11|19:12]	ra	opcode

12. jalr ra, rs1, imm

imm[11:0]	rs1	funct3	rd	opcode

Memory Details:

- Both instruction and data memory are assumed to be byte-addressable.
- The address size for both memories is 32 bits.
- Word size is 32 bits with misaligned addressing allowed in the current design.
- Although the current design contains instructions that interact in terms of only words, it can be scaled to include instructions that deal with bytes, half-words and double-words.

Register File Specifications:

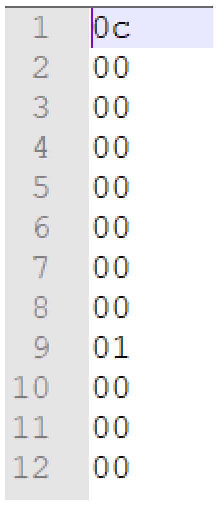
- The register file consists of 32 registers with each register being 32 bits in size.
- The 0th register is a special register, called the zero register that always contains 0x00000000 and can't be written into by any instruction.
- The 1st register is reserved for storing return address of jump statements.
- The general purpose registers are as per the RISC-V ISA.

Other specifications:

- The entire design has been coded in Verilog HDL on Xilinx ISE platform and simulated on Xilinx ISim.
- Since it is a pipelined implementation, there is a possibility of data or control hazards.
- To minimize the impact of or to resolve data hazards, measures like forwarding and stalling have been implemented.
- To reduce delays due to the evaluation of conditions for conditional control transfer instructions, a 'predict not taken' branch prediction scheme is being currently used with flushing enabled in case of a hazard, however, some other scheme like a branch prediction buffer with a saturation counter can be accommodated into the design in future.

Simulation Details:

- The processor was tested with a program to store 12 consecutive numbers of the Fibonacci series into the memory with the address of first two numbers being already available.
- The data memory before running the program is shown below:



```
0c
00
00
00
00
00
00
00
01
00
00
00
```

(The first location contains 0000000CH, which is the number of Fibonacci series values to be found out, second and third locations respectively contain 0 and 1 which are the first two numbers in the Fibonacci series).

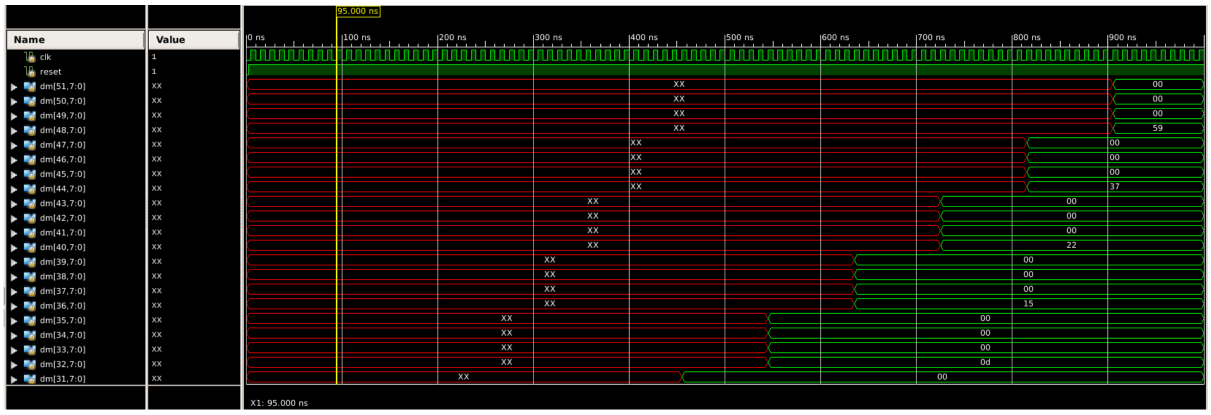
- It is assumed that the register t0 contains address of the starting location of Fibonacci series, i.e., t0 contains 00000004H. Also, the register t2 contains address of the location where required number of Fibonacci series values is stored, i.e., t2 contains 00000000H.
- All the instructions in the program are listed below in order:

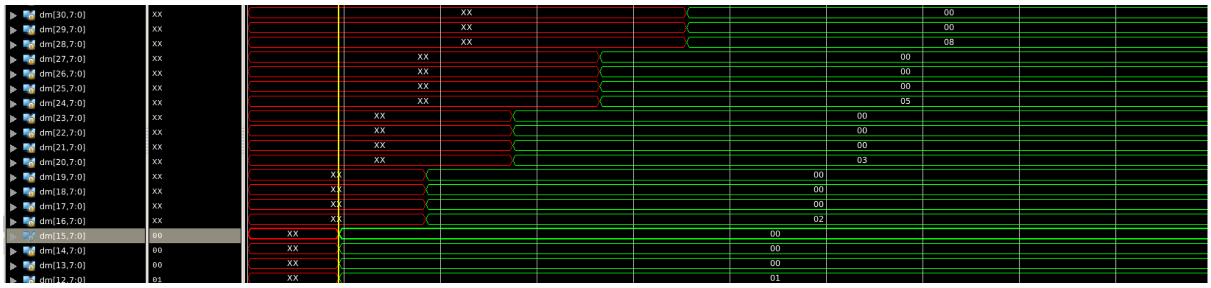
1.	lw t1, 0 (t2)	0003A303H
2.	addi t1, t1, -2	FFE30313H
3.	L1: lw t3, 0 (t0)	0002AE03H
4.	lw t4, 0 (t0)	0042AE83H
5.	add t4, t3, t4	01DE0EB3H
6.	sw t4, 8, t0	01D2A423H
7.	addi t0, t0, 4	00428293H
8.	addi t1, t1, -1	FFF30313H
9.	beq t1, zero, Exit	00030463H

10. jal ra, L1 FE5FF0EFH

11. Exit: some instruction related to another program.

- The following diagram shows the simulation window, where it can be seen that at the end of the program, the data memory is filled with 12 Fibonacci series numbers in contiguous locations from 1st to 48th location in little-endian format:


95.000 ns
Name
Value
clk
reset
dm[51,7:0]
dm[50,7:0]
dm[49,7:0]
dm[48,7:0]
dm[47,7:0]
dm[46,7:0]
dm[45,7:0]
dm[44,7:0]
dm[43,7:0]
dm[42,7:0]
dm[41,7:0]
dm[40,7:0]
dm[39,7:0]
dm[38,7:0]
dm[37,7:0]
dm[36,7:0]
dm[35,7:0]
dm[34,7:0]
dm[33,7:0]
dm[32,7:0]
dm[31,7:0]
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
59
37
22
15
0d
X1: 95.000 ns


dm[30,7:0]
dm[29,7:0]
dm[28,7:0]
dm[27,7:0]
dm[26,7:0]
dm[25,7:0]
dm[24,7:0]
dm[23,7:0]
dm[22,7:0]
dm[21,7:0]
dm[20,7:0]
dm[19,7:0]
dm[18,7:0]
dm[17,7:0]
dm[16,7:0]
dm[15,7:0]
dm[14,7:0]
dm[13,7:0]
dm[12,7:0]
08
05
03
02
01