PIPELINED PROCESSOR

Group 1

Content

Introduction

Central processing unit is one of the most important creations in the human’s history. It carries out the instructions and the data from memory. It is the key element of computers. In the previous project, we have completed a single cycle processor, however, the efficiency is a big problem. In this experiment, we design a pipelined processor and achieve it using the VHDL.

Group member

You Li

Shichao Xu

Shuqi Zhang

Contributions

You: and basic element implementation as well as stall implementation

Shichao: forwarding implementation and pipelined data-path implementation

Shuqi: debugging all the codes and implementing the testing files.

In addition, this report is written by all the group members.

Design overview

Each instruction procedure is divided in to five stages and run in pipelined style, which is shown in the following picture.

IF	ID	EXE	MEM	WB				
	IF	ID	EXE	MEM	WB			
		IF	ID	EXE	MEM	WB		
			IF	ID	EXE	MEM	WB	
				IF	ID	EXE	MEM	WB

We followed the design of the pipelined processor in the lecture slide 12, and using the same basic components in the single-cycle processor.

We divide our processor into two rails: data rail and control rail. Each rail has its own stage modules and pipeline registers. The same stage of both rails are connected by wires, so that they can communicate.

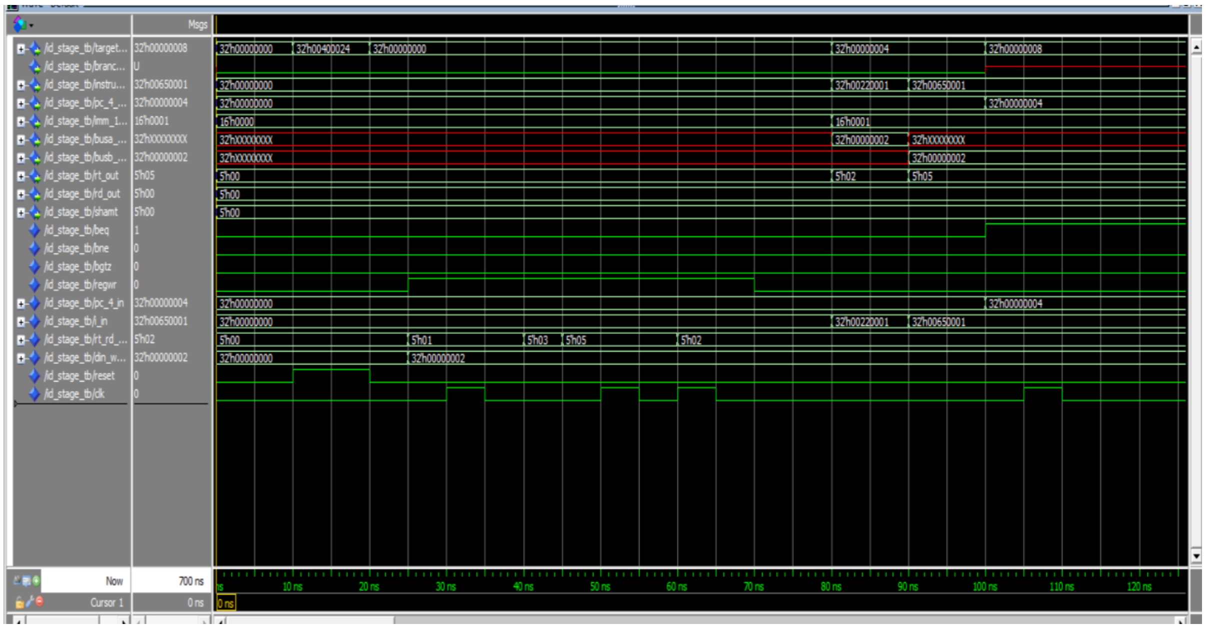
IF stage

The ID stage consists of the Instruction Memory, an adder which adds a constant 4, and a MUX select from a feedback address or the next instruction. The feedback address may come from ID stage by adding the immediate number in the case of forwarding, or come from the EX/MEM registers in regular cases. The result address will read corresponding instruction from the memory.

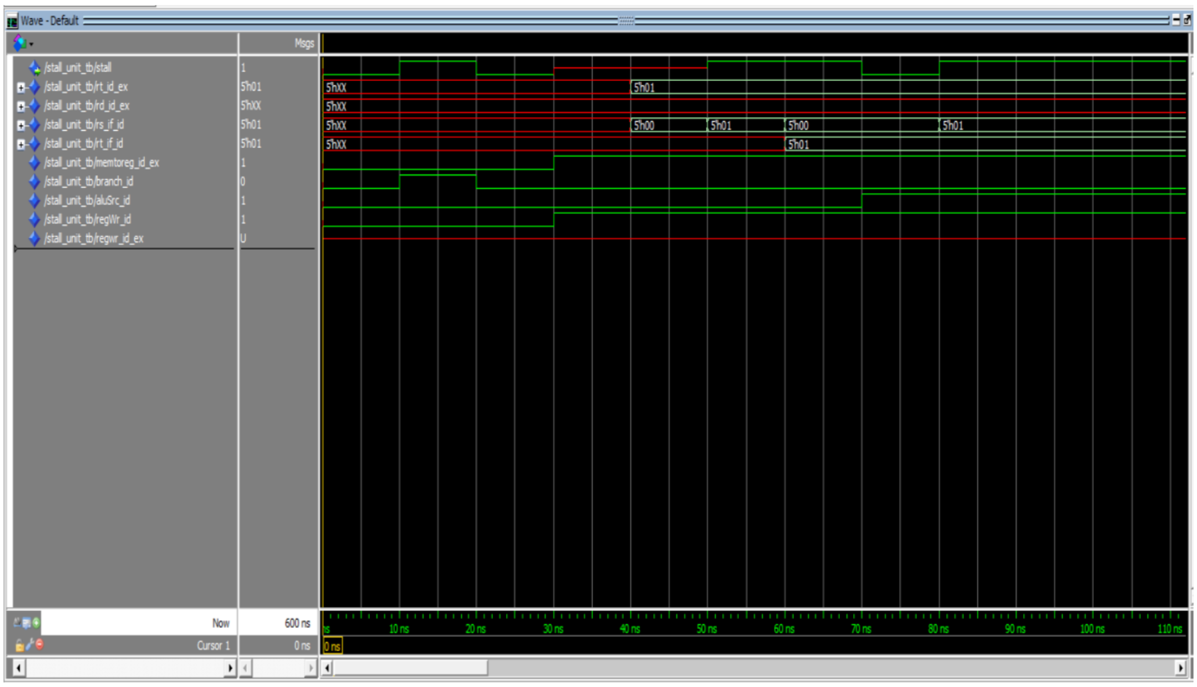
If the stall input is enabled, PC address will keep unchanged for the current clock cycle.

ID stage

We put the control signal module in the control rail. An instruction decoder will decode the instruction provided by IF stage, and send it to the control signal module.


Msgs
/id_stage_tb/target... 32'h00000008
/id_stage_tb/branc... U
/id_stage_tb/instru... 32'h00650001
/id_stage_tb/pc_4_... 32'h00000004
/id_stage_tb/imm_1... 16'h0001
/id_stage_tb/busa_... 32'hXXXXXXXX
/id_stage_tb/busb_... 32'h00000002
/id_stage_tb/rt_out 5'h05
/id_stage_tb/rd_out 5'h00
/id_stage_tb/shamt 5'h00
/id_stage_tb/beq 1
/id_stage_tb/bne 0
/id_stage_tb/bgtz 0
/id_stage_tb/regwr 0
/id_stage_tb/pc_4_in 32'h00000004
/id_stage_tb/i_in 32'h00650001
/id_stage_tb/rt_rd_... 5'h02
/id_stage_tb/din_w... 32'h00000002
/id_stage_tb/reset 0
/id_stage_tb/clk 0
Now 700 ns
Cursor 1 0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns
110 ns
120 ns

To support forwarding and stall, we add additional components in the data rail. Specifically, we add an Adder to compute the next instruction, thus the processor does not need to wait until the EX/MEM registers to settle down. We add a branch control module, and directly analyze beq, bne, bgtz control signals using a full ALU. The result of next instruction address will be sent back to IF stage.

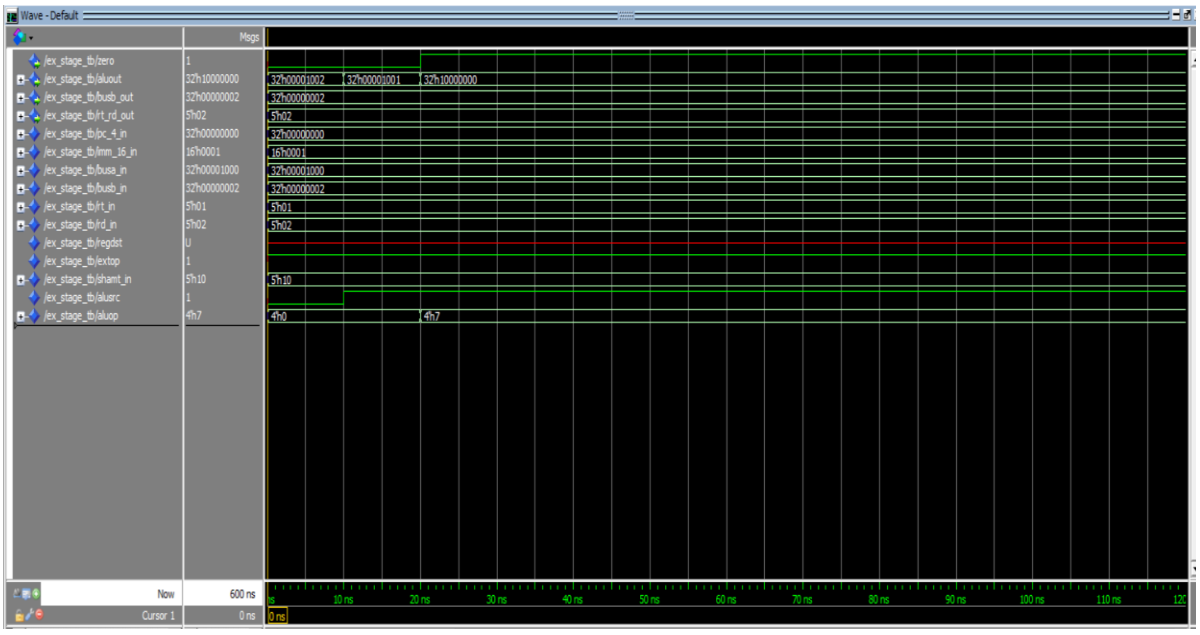


```
Wave - Default
                                    Msgs
/stall_unit_tb/stall                1
/stall_unit_tb/rt_id_ex             5'h01
/stall_unit_tb/rd_id_ex             5'hXX
/stall_unit_tb/rs_if_id             5'h01
/stall_unit_tb/rt_if_id             5'h01
/stall_unit_tb/memtoreg_id_ex       1
/stall_unit_tb/branch_id            0
/stall_unit_tb/aluSrc_id            1
/stall_unit_tb/regWr_id             1
/stall_unit_tb/regwr_id_ex          U

Now                                 600 ns
Cursor 1                            0 ns
```

EX stage

For this stage, we have an extender controlled by ExtOp, a shifter and Adder to compute next instruction address, a MUX selects from immediate number and the second operand by AluSrc, a MUX controlled by RegDst for the write back data, and definitely the ALU and corresponding ALU control translator.



Mem stage

Basically only the data memory is in this stage.

WB stage

A MUX will select from data memory output and ALU output by MemtoReg, and send back to register file.

Control Unit

The instructions and the most of the corresponding control signals are shown in the following graph.

The main problem we faced is that the difference between the interface of each part of the basic elements. So we designed special control unit of each components to solve the interface difference. Just like in the ALU component, it use four bits operation code, which is three bits in the ALU control. So we add a special control unit as the decoder to fit the difference.

instr	aluop	regDst	aluSrc	memtoReg	regWr	memRd	memwr	extop	bne
Add	0000	1	0	0	1	0	0	0	0
Addi	0000	0	1	0	1	0	0	0	0
Addu	0000	1	0	0	1	0	0	0	0
Sub	0001	1	0	0	1	0	0	0	0
Subu	0001	1	0	0	1	0	0	0	0
And	0100	1	0	0	1	0	0	0	0
Or	0110	1	0	0	1	0	0	0	0
Lw	0000	0	1	1	1	1	0	1	0
Sw	0000	0	1	0	0	0	1	1	0
Sll	0111	1	0	0	1	0	0	0	0
Slt	0010	0	0	0	0	0	0	0	0
Sltu	0011	0	0	0	0	0	0	0	0
bne	0001	0	0	0	0	0	0	0	1
bgtz	0001	0	0	0	0	0	0	0	1
beq	0001	0	0	0	0	0	0	0	0

Figure 1 Control Signals

The test wave is shown below:

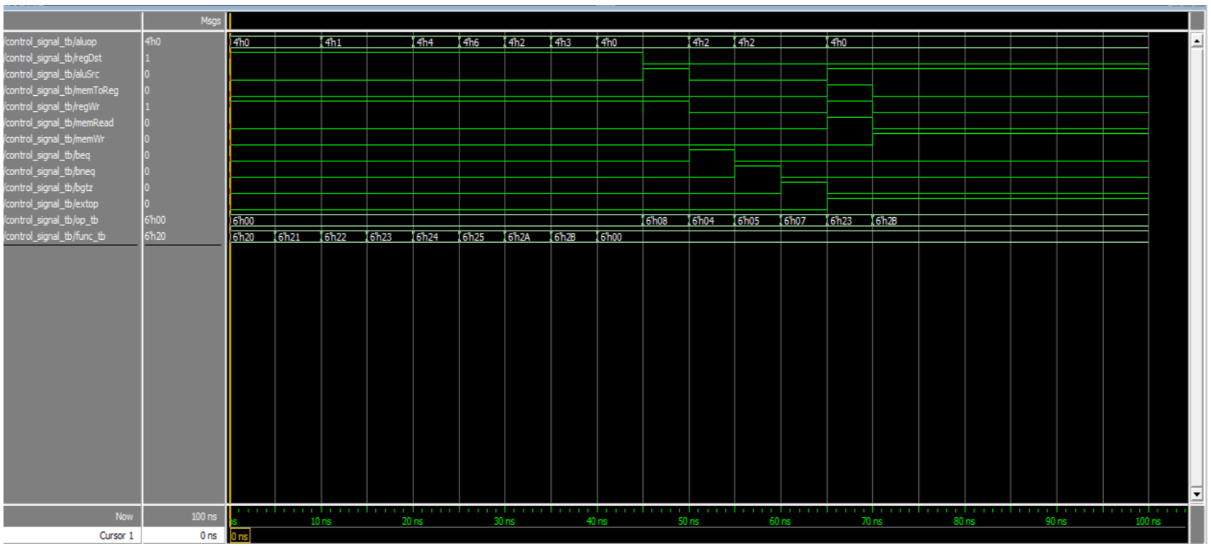
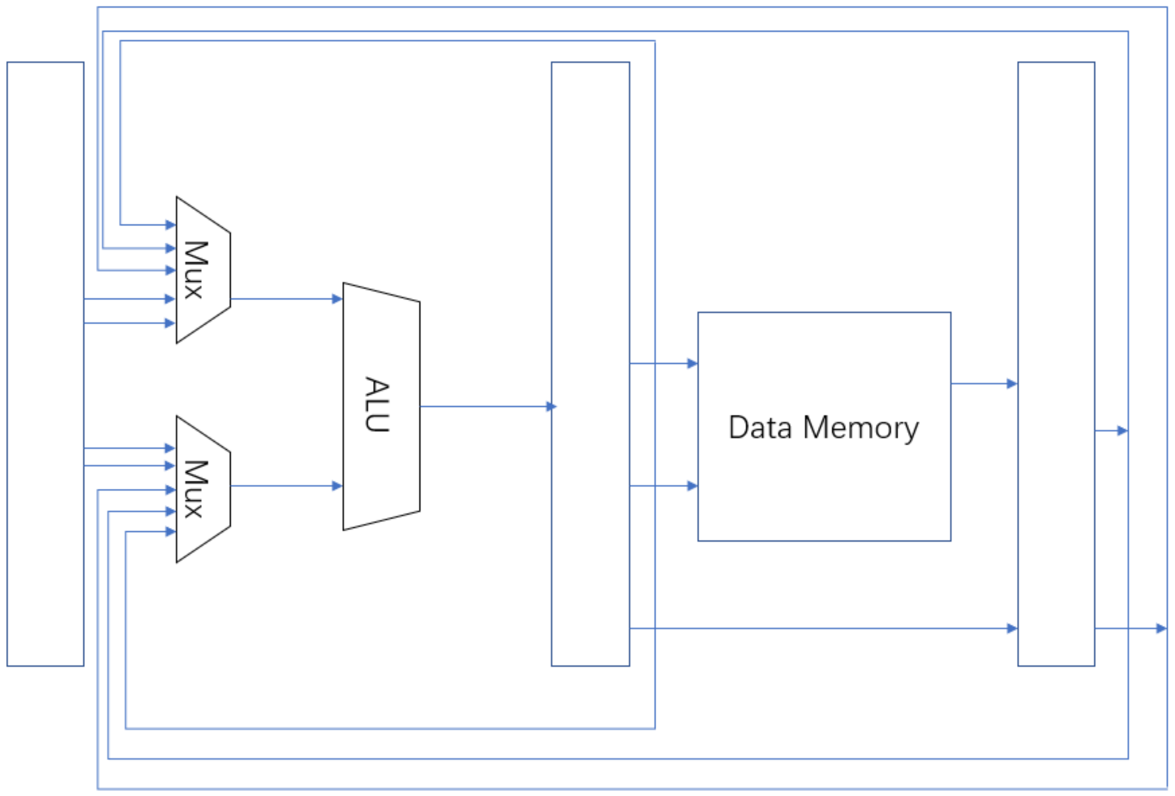


Figure 2 Waveform for Control Unit

Forwarding

The forwarding technique is shown in the following picture.

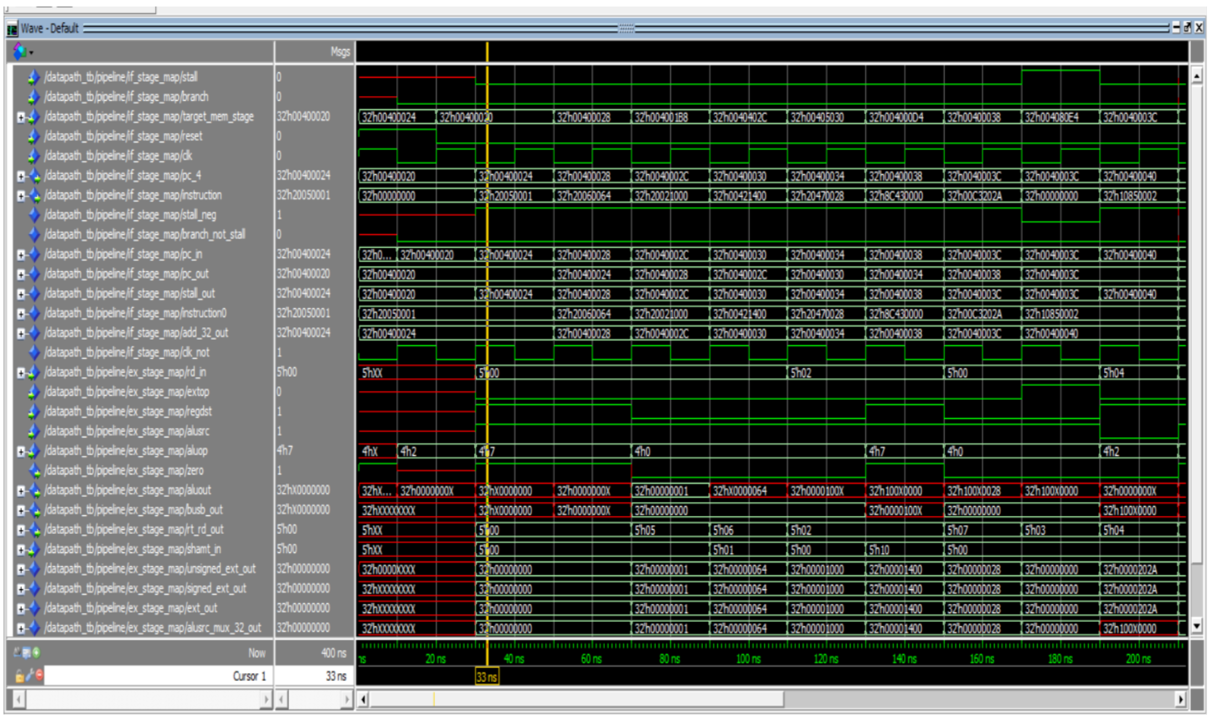


In the EXE stage, the ALU will also receive the data from EXE/MEM, MEM/WB register, which depends on the instruction dependency.

Testbench

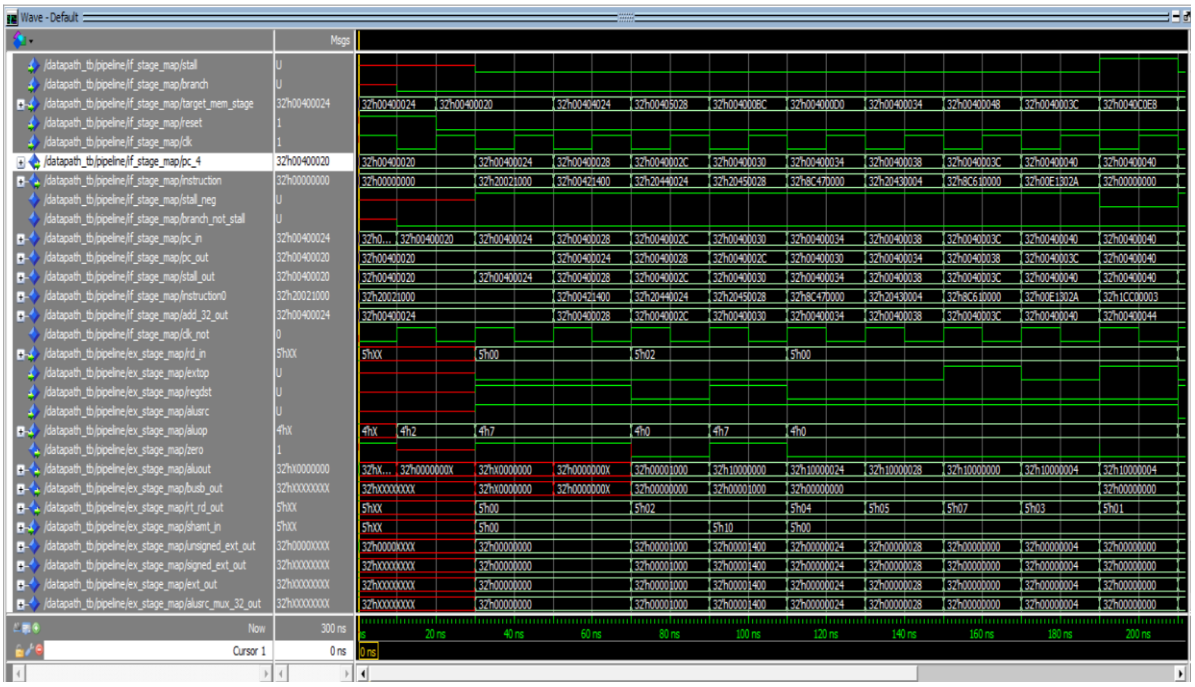
In addition to building testbenches to correct each functional components independently, we focus on observing the synthesized function of our Pipeline Processor. Clock signal and reset signal are generated in cpu_testbench, performing as inputs to Datapath. Clock Cycle Time is 20 ns. Reset signal is set to '1' for the first cycle to initiate the program counter. Below are our wave diagrams:

a. bills_branch.dat



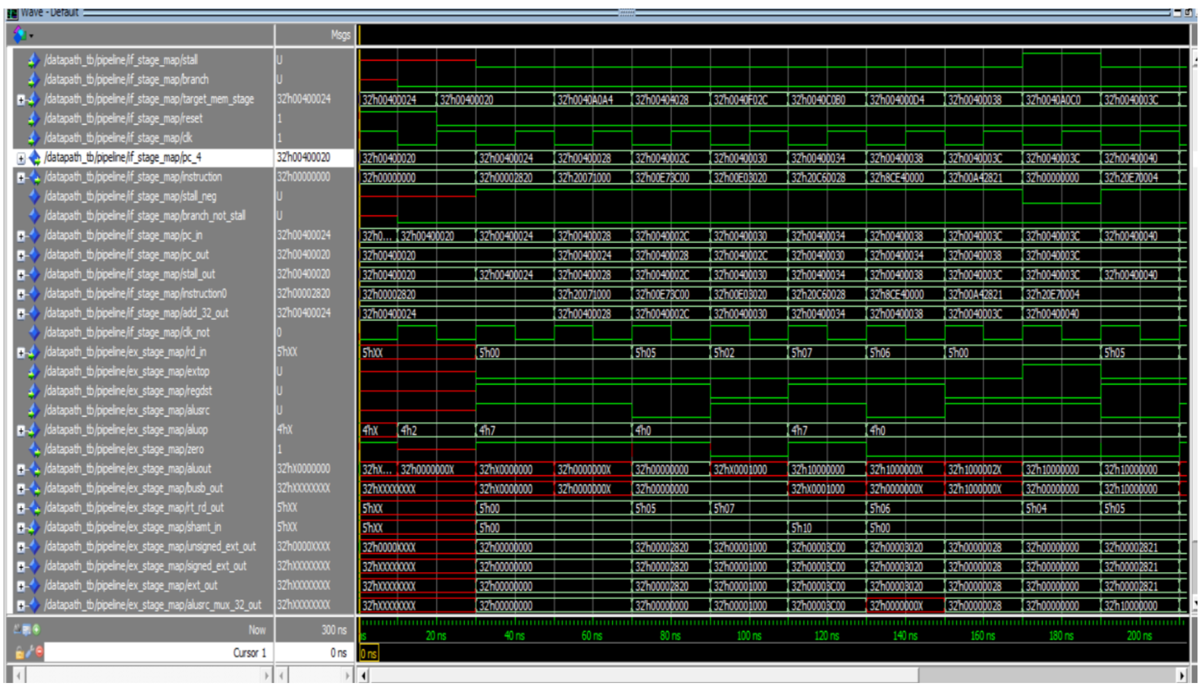
```
Wave - Default
                                                          Msgs
/datapath_tb/pipeline/if_stage_map/stall                  0
/datapath_tb/pipeline/if_stage_map/branch                 0
/datapath_tb/pipeline/if_stage_map/target_mem_stage       32'h00400020
/datapath_tb/pipeline/if_stage_map/reset                  0
/datapath_tb/pipeline/if_stage_map/clk                    0
/datapath_tb/pipeline/if_stage_map/pc_4                   32'h00400024
/datapath_tb/pipeline/if_stage_map/instruction            32'h20050001
/datapath_tb/pipeline/if_stage_map/stall_neg              1
/datapath_tb/pipeline/if_stage_map/branch_not_stall       0
/datapath_tb/pipeline/if_stage_map/pc_in                  32'h00400024
/datapath_tb/pipeline/if_stage_map/pc_out                 32'h00400020
/datapath_tb/pipeline/if_stage_map/stall_out              32'h00400024
/datapath_tb/pipeline/if_stage_map/instruction0           32'h20050001
/datapath_tb/pipeline/if_stage_map/add_32_out             32'h00400024
/datapath_tb/pipeline/if_stage_map/clk_not                1
/datapath_tb/pipeline/ex_stage_map/rd_in                  5'h00
/datapath_tb/pipeline/ex_stage_map/extop                  0
/datapath_tb/pipeline/ex_stage_map/regdst                 1
/datapath_tb/pipeline/ex_stage_map/alusrc                 1
/datapath_tb/pipeline/ex_stage_map/aluop                  4'h7
/datapath_tb/pipeline/ex_stage_map/zero                   1
/datapath_tb/pipeline/ex_stage_map/aluout                 32'hX0000000
/datapath_tb/pipeline/ex_stage_map/busb_out               32'hX0000000
/datapath_tb/pipeline/ex_stage_map/rt_rd_out              5'h00
/datapath_tb/pipeline/ex_stage_map/shamt_in               5'h00
/datapath_tb/pipeline/ex_stage_map/unsigned_ext_out       32'h00000000
/datapath_tb/pipeline/ex_stage_map/signed_ext_out         32'h00000000
/datapath_tb/pipeline/ex_stage_map/ext_out                32'h00000000
/datapath_tb/pipeline/ex_stage_map/alusrc_mux_32_out      32'h00000000
                                                   Now    400 ns
                                              Cursor 1    33 ns
```

b. sort_corrected_branch.dat



```
Wave - Default
                                                        Msgs
/datapath_tb/pipeline/if_stage_map/stall                U
/datapath_tb/pipeline/if_stage_map/branch               U
/datapath_tb/pipeline/if_stage_map/target_mem_stage     32'h00400024
/datapath_tb/pipeline/if_stage_map/reset                1
/datapath_tb/pipeline/if_stage_map/clk                  1
/datapath_tb/pipeline/if_stage_map/pc_4                 32'h00400020
/datapath_tb/pipeline/if_stage_map/instruction          32'h00000000
/datapath_tb/pipeline/if_stage_map/stall_neg            U
/datapath_tb/pipeline/if_stage_map/branch_not_stall     U
/datapath_tb/pipeline/if_stage_map/pc_in                32'h00400024
/datapath_tb/pipeline/if_stage_map/pc_out               32'h00400020
/datapath_tb/pipeline/if_stage_map/stall_out            32'h00400020
/datapath_tb/pipeline/if_stage_map/instruction0         32'h20021000
/datapath_tb/pipeline/if_stage_map/add_32_out           32'h00400024
/datapath_tb/pipeline/if_stage_map/clk_not              0
/datapath_tb/pipeline/ex_stage_map/rd_in                5'hXX
/datapath_tb/pipeline/ex_stage_map/extop                U
/datapath_tb/pipeline/ex_stage_map/regdst               U
/datapath_tb/pipeline/ex_stage_map/alusrc               U
/datapath_tb/pipeline/ex_stage_map/aluop                4'hX
/datapath_tb/pipeline/ex_stage_map/zero                 1
/datapath_tb/pipeline/ex_stage_map/aluout               32'hX0000000
/datapath_tb/pipeline/ex_stage_map/busb_out             32'hXXXXXXXX
/datapath_tb/pipeline/ex_stage_map/rt_rd_out            5'hXX
/datapath_tb/pipeline/ex_stage_map/shamt_in             5'hXX
/datapath_tb/pipeline/ex_stage_map/unsigned_ext_out     32'h0000XXXX
/datapath_tb/pipeline/ex_stage_map/signed_ext_out       32'hXXXXXXXX
/datapath_tb/pipeline/ex_stage_map/ext_out              32'hXXXXXXXX
/datapath_tb/pipeline/ex_stage_map/alusrc_mux_32_out    32'hXXXXXXXX
Now     300 ns
Cursor 1     0 ns
```

c. unsigned_sum.dat



```
Wave - Default
                                                        Msgs
/datapath_tb/pipeline/if_stage_map/stall                U
/datapath_tb/pipeline/if_stage_map/branch               U
/datapath_tb/pipeline/if_stage_map/target_mem_stage     32'h00400024
/datapath_tb/pipeline/if_stage_map/reset                1
/datapath_tb/pipeline/if_stage_map/clk                  1
/datapath_tb/pipeline/if_stage_map/pc_4                 32'h00400020
/datapath_tb/pipeline/if_stage_map/instruction          32'h00000000
/datapath_tb/pipeline/if_stage_map/stall_neg            U
/datapath_tb/pipeline/if_stage_map/branch_not_stall     U
/datapath_tb/pipeline/if_stage_map/pc_in                32'h00400024
/datapath_tb/pipeline/if_stage_map/pc_out               32'h00400020
/datapath_tb/pipeline/if_stage_map/stall_out            32'h00400020
/datapath_tb/pipeline/if_stage_map/instruction0         32'h00002820
/datapath_tb/pipeline/if_stage_map/add_32_out           32'h00400024
/datapath_tb/pipeline/if_stage_map/clk_not              0
/datapath_tb/pipeline/ex_stage_map/rd_in                5'hXX
/datapath_tb/pipeline/ex_stage_map/extop                U
/datapath_tb/pipeline/ex_stage_map/regdst               U
/datapath_tb/pipeline/ex_stage_map/alusrc               U
/datapath_tb/pipeline/ex_stage_map/aluop                4'hX
/datapath_tb/pipeline/ex_stage_map/zero                 1
/datapath_tb/pipeline/ex_stage_map/aluout               32'hX0000000
/datapath_tb/pipeline/ex_stage_map/busb_out             32'hXXXXXXXX
/datapath_tb/pipeline/ex_stage_map/rt_rd_out            5'hXX
/datapath_tb/pipeline/ex_stage_map/shamt_in             5'hXX
/datapath_tb/pipeline/ex_stage_map/unsigned_ext_out     32'h0000XXXX
/datapath_tb/pipeline/ex_stage_map/signed_ext_out       32'hXXXXXXXX
/datapath_tb/pipeline/ex_stage_map/ext_out              32'hXXXXXXXX
/datapath_tb/pipeline/ex_stage_map/alusrc_mux_32_out    32'hXXXXXXXX
Now     300 ns
Cursor 1     0 ns
```

Acknowledgement

Thanks for the help of Professor Memik’s slides! Thanks for TA’s instructions on ModelSim!