ECE 485 Project #2

Jay Mundrawala

November 23, 2009

Abstract

This project describes the design and implementation of a multicycle MIPS datapath. Along with it are testbenches that issue the appropriate signals for a few signals.

1 Introduction

The goal of this project was to implement a MIPS datapath capable of executing a small set of instructions. For this project, we were to implement the following instructions:

- LW
- SW
- ADD
- SUB
- AND
- OR
- SLT
- BEQ
- J
- BNE
- ADDI
- SLL
- LUI
- JAL

The source code for this datapath is available in the appendix and in a git repository on github. This is available at `http://github.com/whois/ECE485_P2`.

2 Design

The design for a multicycle MIPS datapath is quite simple. It is shown in Figure 1. The data path has the following signals coming from the control unit, which describe how the datapath should operate: PCWrite, IorD, MemRead, MemWrite, IRWrite, RegDst, MemToReg, RegWrite, ALUSrcA, ALUSrcB, ALUOp, and PCSource. Tables 1 through 6 enumerate the possible values for these signals and Listings 19 contains their definitions.

Signal name	Effect when deasserted	Effect when asserted
PCWrite	None	PC is written
PCWriteCondEq	None	PC is written of zero is asserted
PCWriteCondNEq	None	PC is written of zero is deasserted
IorD	Instruction is Fetched	Data is fetched
MemRead	None	Memory at given address is read
MemWrite	None	Data is written to given address
IRWrite	None	Instruction register is written
RegWrite	None	Data is written to register
ALUSrcA	Port A of ALU gets PC	Port A of ALU gets register A's output

Table 1: Actions of 1-bit control signals

ALUSrcB	
Signal value	Effect
ASB_REGB	Port B of ALU gets register B's output
ASB_FOUR	Port B of ALU gets 4
ASB_SEXT	Port B of ALU gets instruction[15-0] sign extended to 32 bits
ASB_SEXTS	Same as previous except value is shifted left by 2

Table 2: Actions of ALUSrcB signal

ALUOp	
Signal value	Effect
AOP_AND	ALU will 'and' the two operands
AOP_OR	ALU will 'or' the two operands
AOP_ADD	ALU will add the two operands
AOP_SUB	ALU will subtract A - B
AOP_SLT	ALU will output '1' if $A < B$, '0' otherwise
AOP_NOR	ALU will NOR the two operands
AOP_SLL	ALU will shift B left by instruction[10-6]

Table 3: Actions of ALUOp signal

With these signals defined, we can begin by defining each functional block in Figure 1. First, we have the PC. This is the program counter and stores the address of the next instruction to be executed. The memory block is our RAM. For this project, the test benches only output one value when the memory is read, and that is the instruction we are testing. This allows us to test the datapath without creating a memory module. Nothing is ever written to the memory; when testing LW we analyze the value on the

PCSource	
Signal value	Effect
PS_PCINC	Value to PC is $PC + 4$
PS_ALUOUT	Value to PC is that of the register ALU OUT
PS_JMP	Value to PC is $PC[31 - 28] + Instr[25 - 0] << 2$

Table 4: Actions of PCSource signal

RegDst	
Signal value	Effect
RD_RT	Write register is set to instr[25-21]
RD_RD	Write register is set to instr[20-16]
RD_RA	Write register is set to $R31$

Table 5: Actions of RegDst signal

write data line when it is time to write to the memory. The instruction register holds the values of the instruction being executed. The memory data register(MDR), stores the value read from memory. This will be written every clock cycle since the value only need be stored for one clock cycle. Thus, it does not need its own control signal and its enable can be attached to the clock. The register file, denoted by Registers in Figure 1, stores the 32 registers. Register A and B store the output of the register file for one clock cycle. The ALU does all the arithmetic computations. Its functionality is fully described by Table 3. The ALU OUT register holds the output of the ALU for one clock cycle. PCWrite Generate generates the enable signal to write to the PC. In the case of this design, it is described by the following equation: $G = PCWrite + PCWriteCondEq * zero + PCWriteCondNEq * \overline{zero}$.

2.1 Instruction Fetch, Instruction Decode, and Branch Target Computation

There are two steps that each instruction execution has in common. The first is the IF stage. Each instruction needs to be fetched from memory. To do this, we need to read the instruction from memory and store it into the instruction register. This is also the step where we must update the PC to $PC + 4$. So, with that in mind, we can define the signals needed to complete this step. First, IorD needs to be deasserted. This will select the PC as the address for memory. MemRead needs to be set, so that the instruction is fetched from memory. IRWrite needs to be asserted so that the instruction is stored after it is read. This is enough to actually fetch the instruction, but the PC still needs to be updated. Thus, we require ALUSrcA to be deasserted to select PC for port A of the ALU, and ALUSrcB to be set to ASB_FOUR to select 4 for the second port($PC + 4$). PCSource needs to be set to PS_PCINC inidcating that $PC + 4$ should be written to the PC. Finally, PCWrite needs to be asserted so that the value is written to the PC. One thing that should be noted is that all writes happen only when the clock is high, except for the IRWrite. This is a bug, however it does not affect the system in any way.

The next step is the instruction decode stage. In this stage, we do two things. First, we read the registers from the register file. This is done automatically by the instruction register. Since the ALU is not being used in this stage, its a good time to calculate the

MemToReg	
Signal value	Effect
MTR_ALUOUT	Register write data gets the output of the ALU OUT register
MTR_MDR	Register write data gets the output of the MDR register
MTR_PC	Register write data gets the PC

Table 6: Actions of MemToReg signal

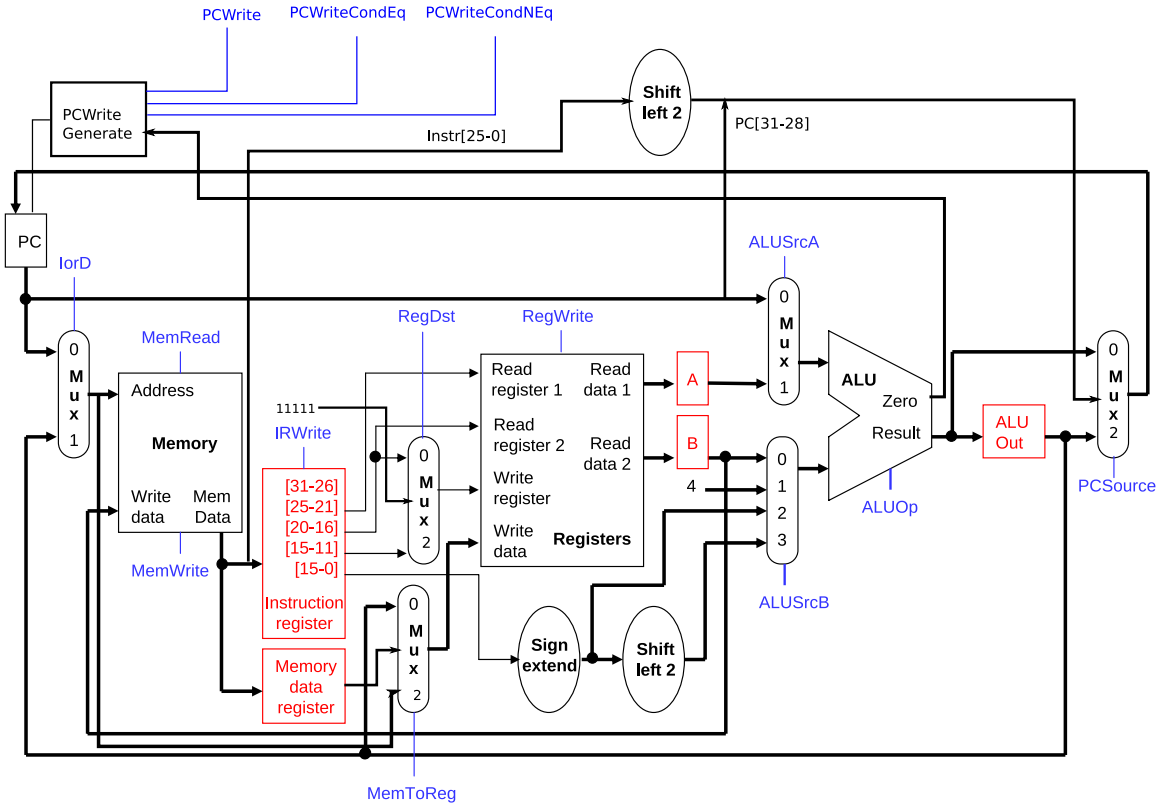

Figure 1: MIPS multicycle datapath

branch target address, regardless of if the instruction is a branch or not. This way, the target is ready incase it is a branch instruction. To do this, ALUSrcA is deasserted, indicating port A of the ALU gets the value of the PC. ALUSrcB is set to indicate the sign extended and shifted value of instruction[15-0]. These two will be added and stored in ALU OUT.

Figure 2 shows these two steps. One thing to note is that the first clock cycle is just a reset; it assigns values to all the signals so they are known. Thus, for this figure, and all future figures, the cycle begins in the second clock cycle. The important thing to note in this figure is that the PC is being assigned $PC + 4$ after executing the IF stage and IR has been assigned an instruction in that stage. In the next stage, register A and register B are assigned values that were read from the register file. The final clock cycle is bogus as the state machine was not updated.

2.2 LW/SW Instructions

LW and SW are the only two instructions allowed to access memory. They both have one step in common, and that is the memory address computation stage. In this step,

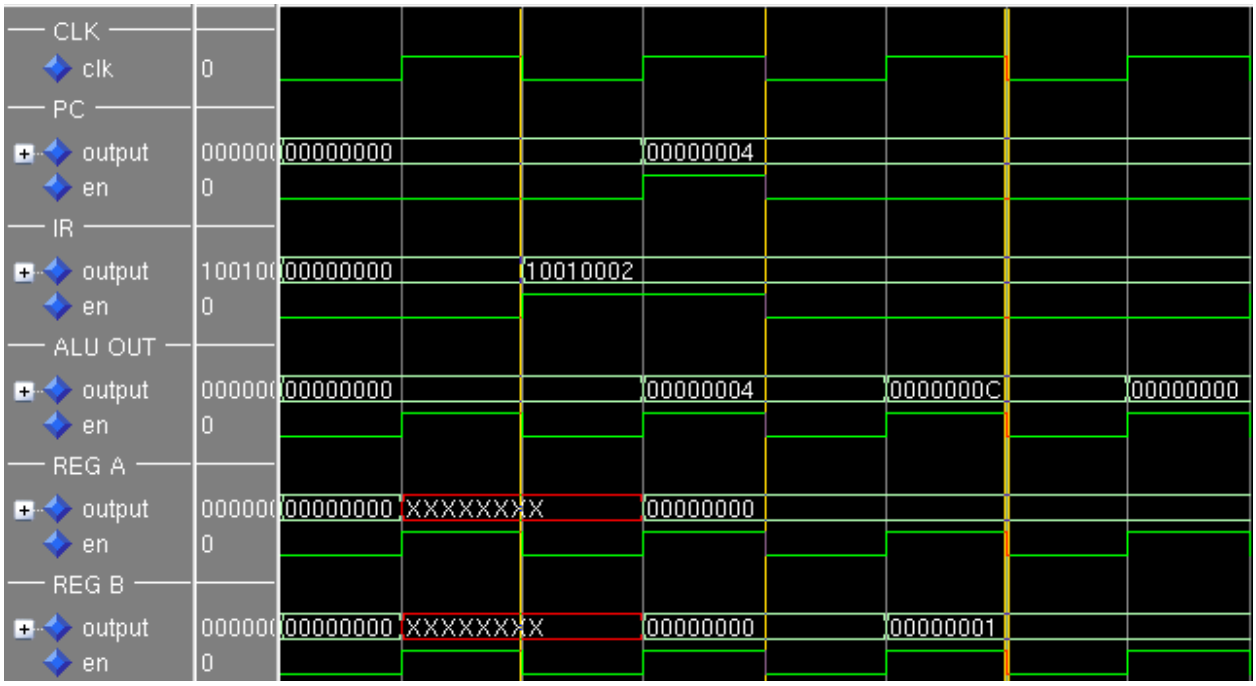

Figure 2: IF and ID stages. Both the first and last clock cycle have no meaning. The middle two are the one of interest.

ALUSrcA will be set to use register A. ALUSrcB will be set to use the bottom 16 bits of the instruction sign extended to 32 bits. These will be added by the ALU and used for the memory access.

Next is the memory access. The previously computed address is used here. Thus, in both the cases of LW and SW IorD is set to use the ALU OUT register. For LW, MemRead is asserted, and for SW, MemWrite is asserted.

At this point, SW is complete, but LW still has to write back. Thus, RegWrite is asserted, MemToReg is set to use the value stored in MDR, and RegDst is set to use RT.

Figure 3 shows the operation of LW. The first clock cycle is a reset. The second shows the IF stage. The third shows the ID stage. The fourth is where the memory address computation is occuring. Here, $0x00FF$ is being added to $R0$ as specified by the instruction. The ALU outputs FF as expected. Next, is the memory access... nothing happens here since the value of memory is kept constant for simplicity. Next, in the final clock cycle, the write back occurs, and $R1$ gets the value of the memory.

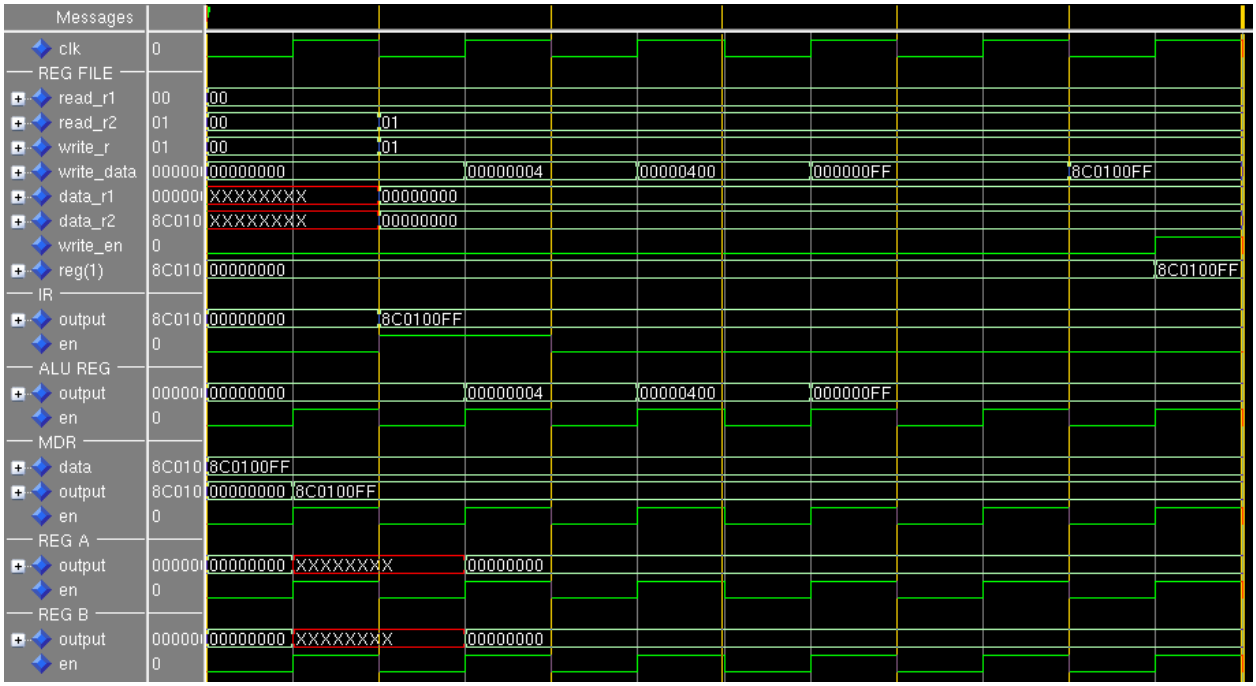

Figure 3: Shows the execution of a LW instruction

Similarly, Figure 4 shows the operation of SW. The first 4 cycles behave the same way. The final cycle is different in that it asserts a MemWrite signal and writes the data to memory.

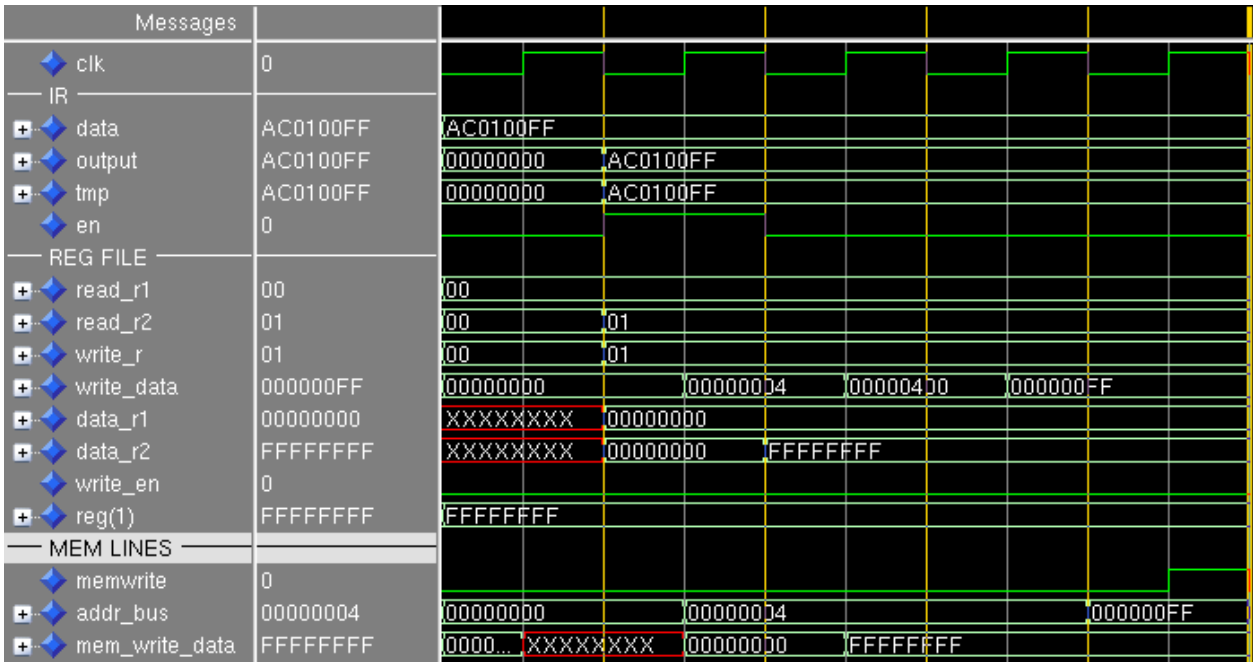

Figure 4: Shows the execution of a SW instruction

2.3 R Type Instructions

This section shows the correct functionality of the R Type instructions. Only add and sll are described in detail. The waveforms for the remaining are at the end of the section.

No further description is needed as only changing the ALU operation for the R Type instructions is done.

In general, R Type instructions consist of first performing the ALU operation specified, and writing that value back to the registers. For example, Figure 5 shows the waveform for an add instruction. First there is the reset cycle, then IF, and ID. Next is the execution stage. Here, the ALU takes the values read from the register file and performs and add on them. And the end of this cycle, write_data has the value of $R0 + R1$. In the next cycle, the write back cycle, $R2 <= R0 + R1$ as expected.

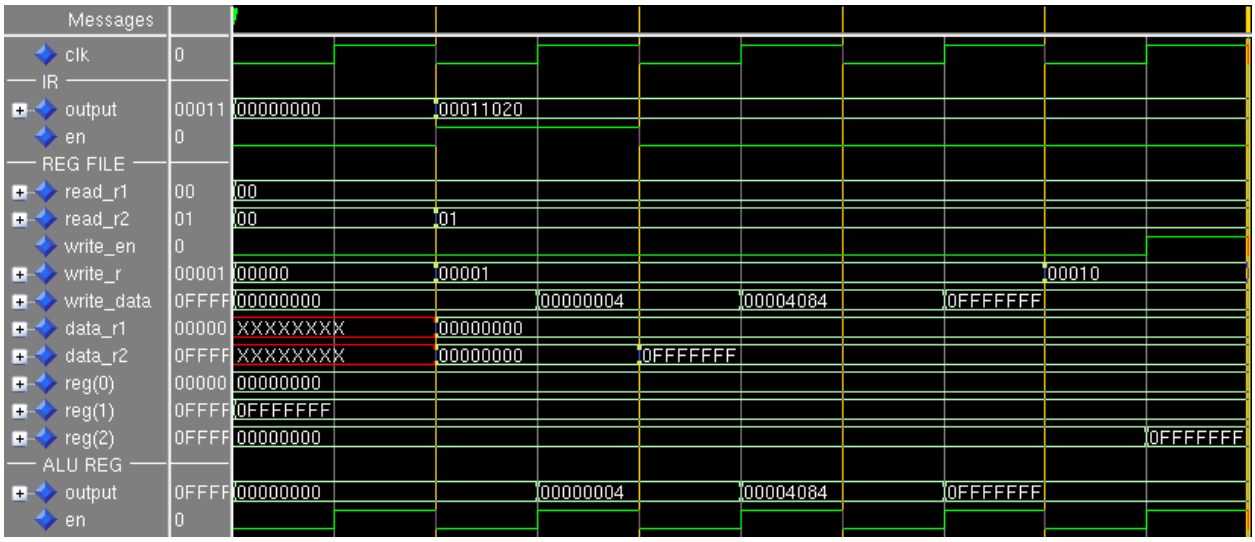

Figure 5: Shows the execution of a add instruction

The SLL is similar, except instead of adding 2 operands, it takes the B operand to the ALU and shifts it by an amount specified in the instruction. This is shown in Figure **??**. The first clock cycle is the reset, followed by the IF, followed by ID, followed by the execution. In the execution, ALUSrcA is irrelevant. Operand B(R1) is '1', which will be left shifted by the amount of 2. Thus, the expected value is 4, and that is the value that is written back in the final clock cycle.

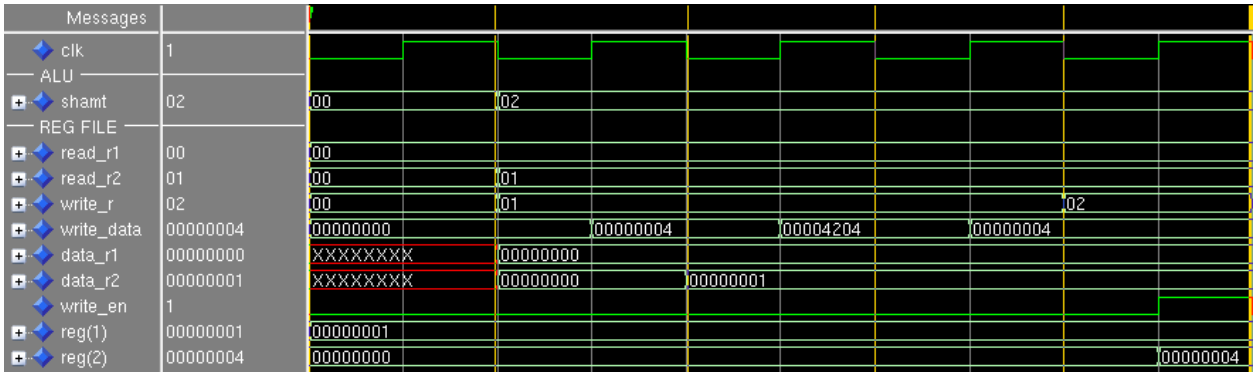

Figure 6: Shows the execution of a SLL instruction

The following are the figures for SUB, AND, OR, and SLT. Whats important to look at in these instructions are the operands and the final write back values. With that, it can be seen that each functions correctly.

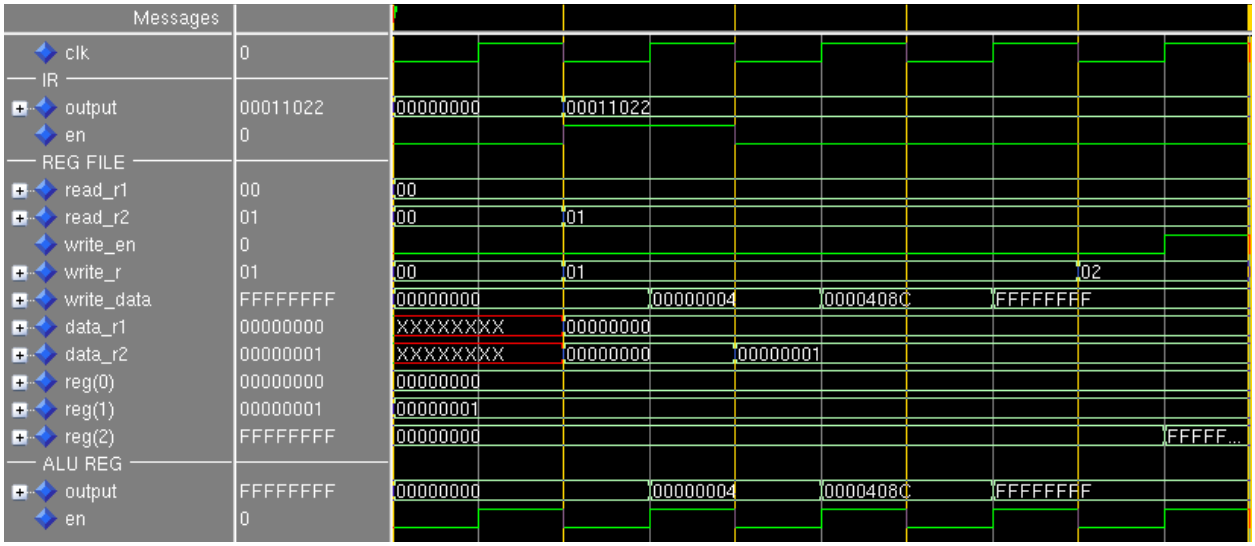

Figure 7: Shows the execution of a SUB instruction

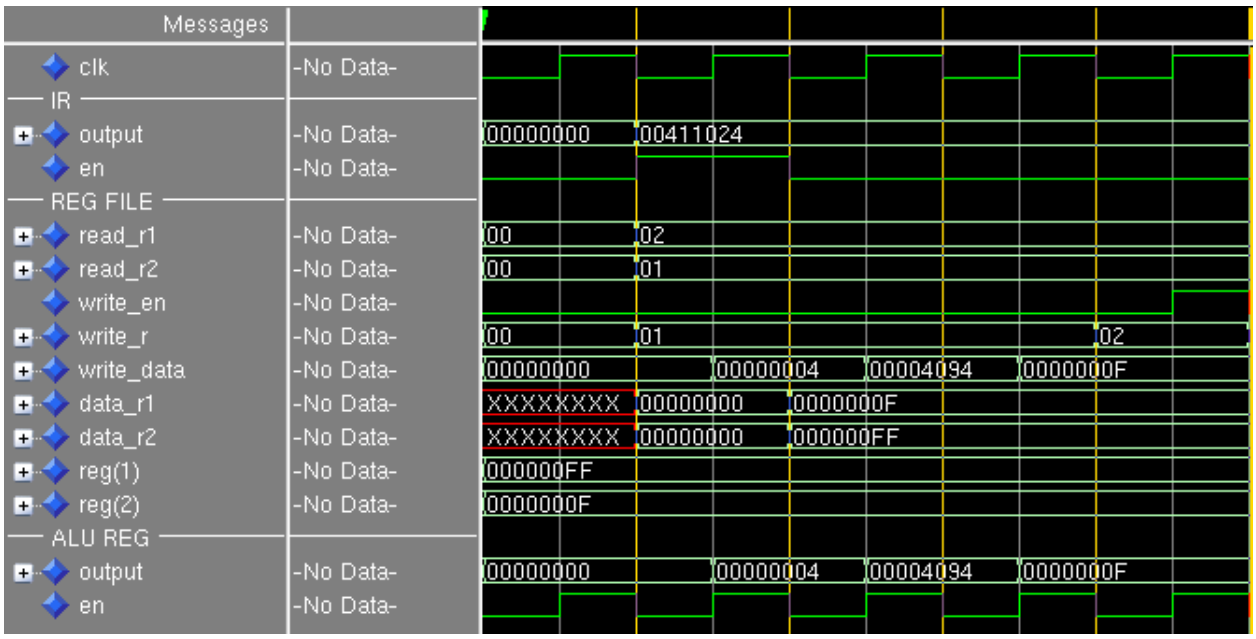

Figure 8: Shows the execution of a AND instruction

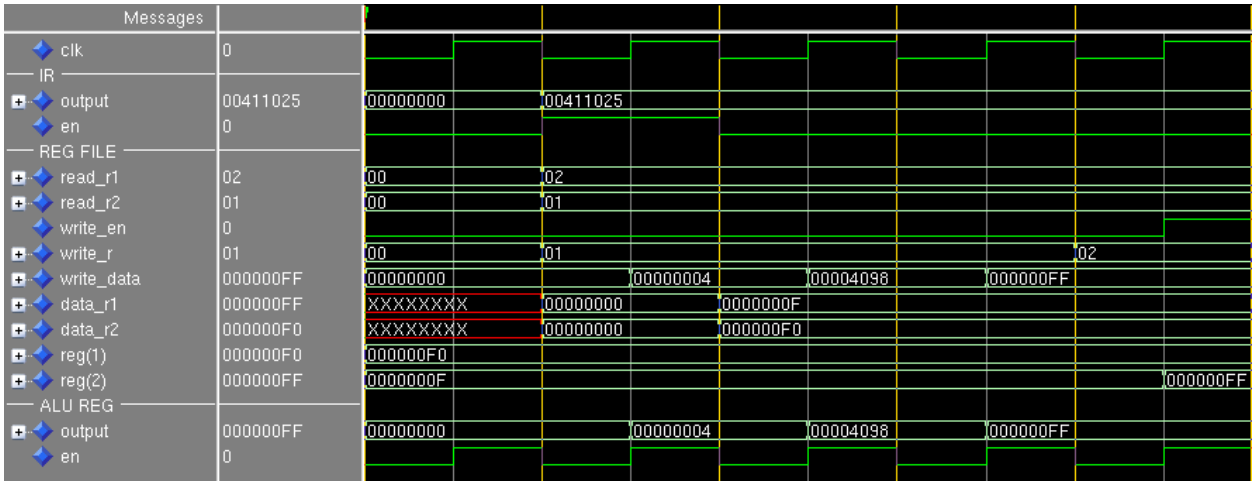

Figure 9: Shows the execution of a OR instruction

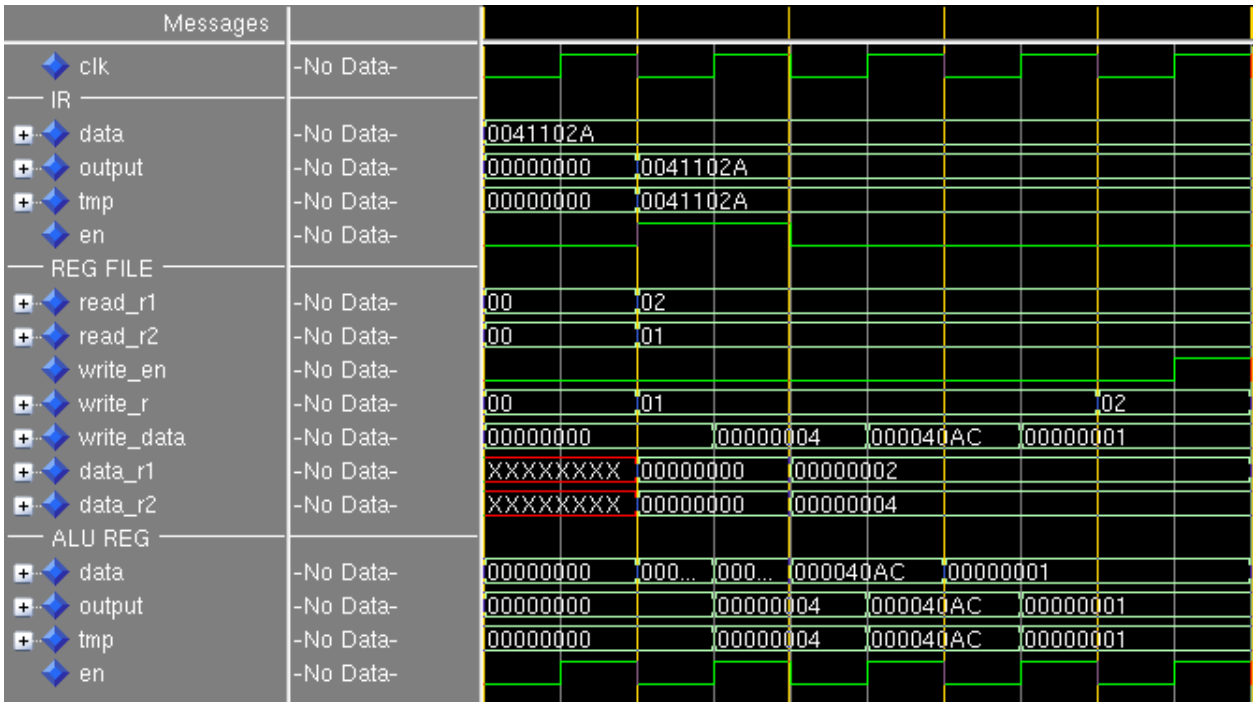

Figure 10: Shows the execution of a SLT instruction

2.4 Immediate Instruction

Only one immediate instruction required testing, and the was ADDI. Its very similar to ADD, except that ALUSrcB is set to use the sign extended lower 16 bits of the instruction. Its waveform is shown in Figure 11. The first cycle is reset, the second is instruction fetch, third is instruction decode, fourth is execution. The lower 16 bits of this instruction are 8, which is being added to $R0$. The final clock cycle shows that 8 is written back as expected.

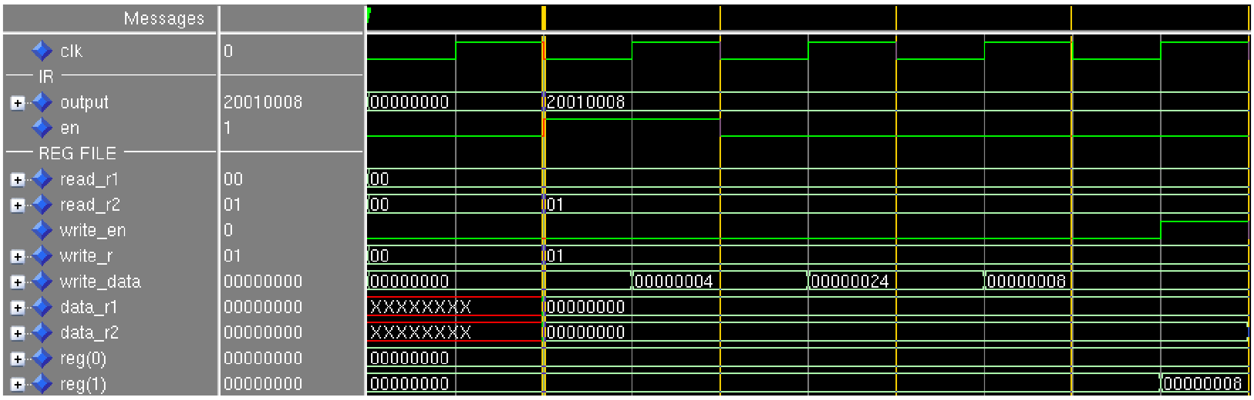

Figure 11: Shows the execution of a ADDI instruction

2.5 Branch and Jump

This section shows the correct functionality of BNE and JAL. BEQ and JMP are shown at the end of this section for completeness, however they are not annotated as the instructions are very similar to BNE and JAL.

The correct functionality for BNE is shown in Figure 12. For branch instructions, the target address is available after the ID stage. Thus, as soon as the appropriate signals have progagated from the ALU, we can branch. For BNE, the control unit will set ALUSrcA to register A. ALUSrcB will be set to allow register B. These two will be compared in the ALU during the execution stage. Along with those signals, BNE needs to assert the PCWriteCondNEq signal to indicate a branch when not equal. The branch occurs in the execution stage, as that is when PC is written with its new value. In Figure 12, the lower 16 bits are 2. Sign extended and shifted left by two makes 8. Thus, $PC + 4 + 8 = C$, as the original PC started off at 0. This is the value that is written to the PC as can be seen.

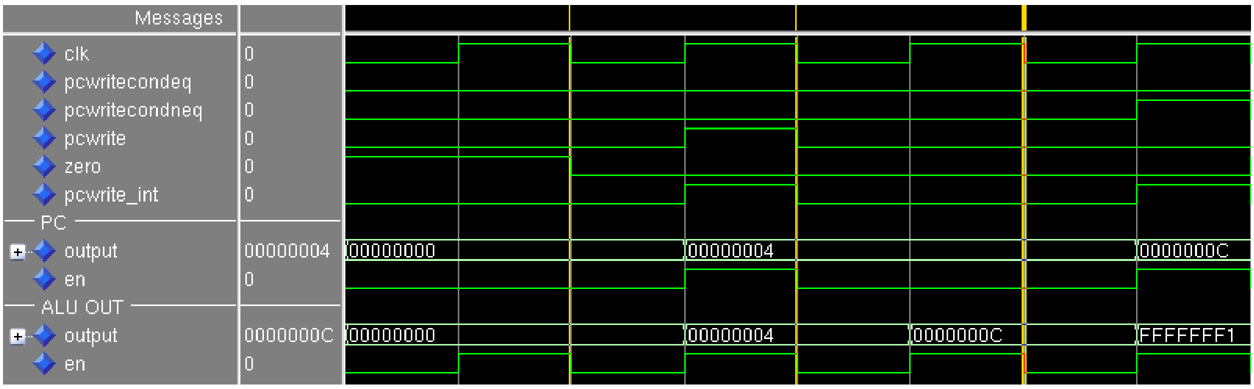

Figure 12: Shows the execution of a BNE instruction

Jumping is similar to a branch, however it uses instruction[26-0] shifted left by 2 for the lower 28 bits. The higher bits are determined by PC[31-28]. JAL, shown in Figure 13, does more than just jump. It also writes the value of the PC to $R31$. The instruction if Figure 13 stores the value of PC as expected into $R31$ in the final clock cycle. The lower 26 bits of the instruction are 8, thus $0x20$ is the expected jump value, the same value that PC gets in the final clock cycle.

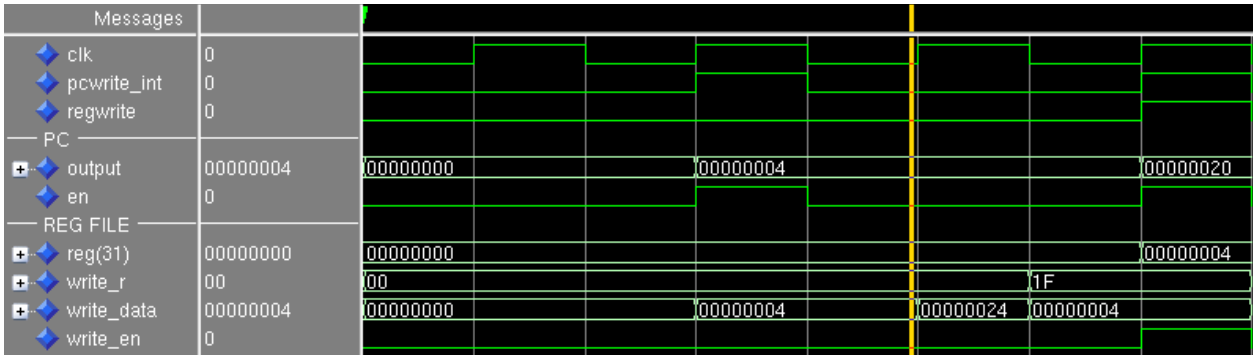

Figure 13: Shows the execution of a JAL instruction

The following figures(13 and 15 show similar instructions J and BEQ.

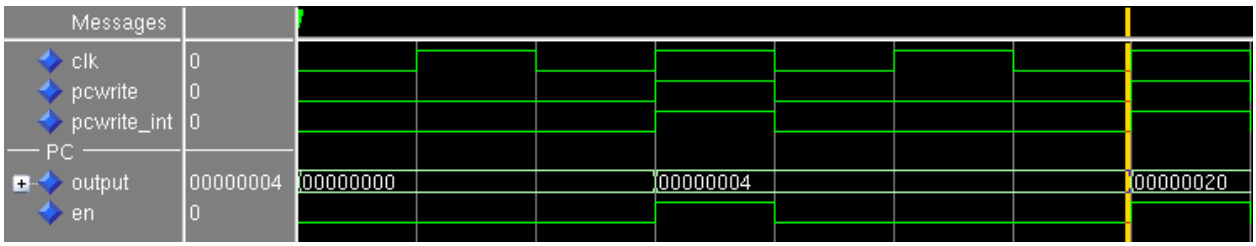

Figure 14: Shows the execution of a J instruction

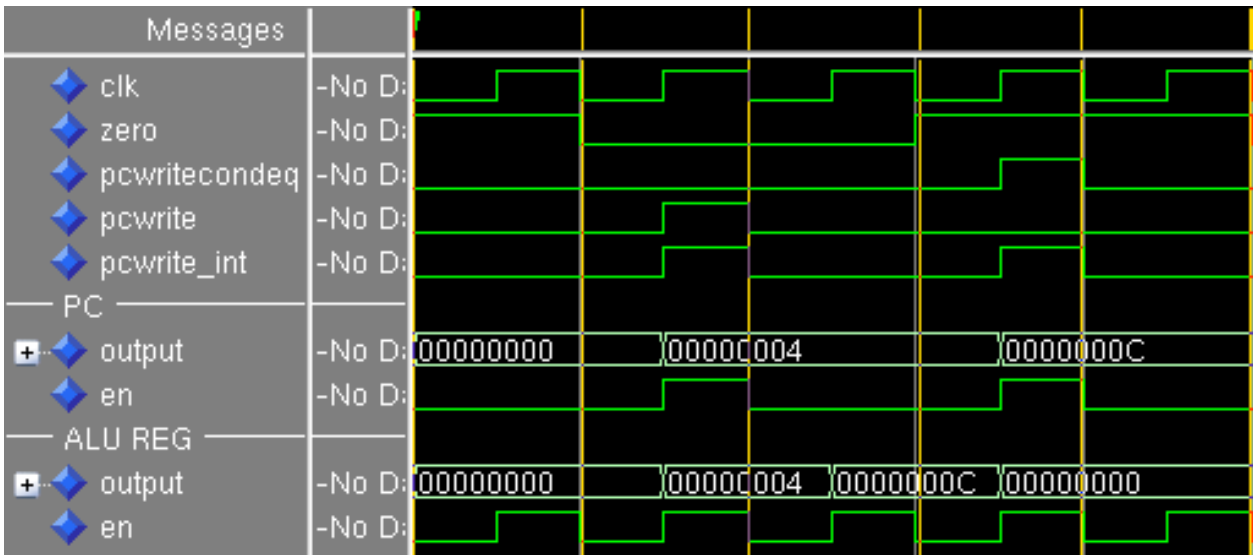

Figure 15: Shows the execution of a BEQ instruction

2.6 LUI

Since the ALU already has the capability to do shift operations on ALU port B operands, nothing needs to be added to the datapath. This instruction can be executed by 3 instructions already described. The instructions needed are SLL, which is done by 16 bits. Next, we can use an AND the lower upper 16 bits of the register we want to LUI into with 1's. This clears out the top bits. This leaves one final OR with the result of SLL and the register we want to LUI into.

2.7 Exception Handling

Exception handling requires two additional registers and addition control signals. The registers needed are the cause register and the EPC. This datapath must allow for two different types of exceptions. First, the arithmetic overflow exception. This can only occur during the execution stage, and may not be caused by the control unit. The second is undefined instruction. Here, there are two possibilities. First, there can be an unkwown opcode. Second, there can be an unknown function for an R-Type. The design here is to let the ALU assert arithmetic overflow exceptions, and the control unit will assert the unknown operation exception. When these are asserted, there will be a jump to some hard-wired memory address, in this case $0x4$ for simplicity.

3 Unfinished

1. ALU does not generate overflow
2. There is no testbench for exception handling

Other than that, and a bug with the IR that does not produce any bad results, everything works.

A Code

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : mips_lib.vhdl
5  -- Creation Date : 06/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13
14 package mips_lib is
15     constant DATA_WIDTH : integer              := 32;
16     constant ADDR_WIDTH : integer              := 32;
17
18     constant rOpcode    : bit_vector(5 downto 0) :="000000";
19     constant jOpcode    : bit_vector(5 downto 0) :="000010";
20     constant jalOpcode  : bit_vector(5 downto 0) :="000011";
21     constant addiOpcode : bit_vector(5 downto 0) :="001000";
22     constant andiOpcode : bit_vector(5 downto 0) :="001100";
23     constant beqOpcode  : bit_vector(5 downto 0) :="000100";
24     constant bneOpcode  : bit_vector(5 downto 0) :="000101";
25     constant lwOpcode   : bit_vector(5 downto 0) :="100011";
26     constant swOpcode   : bit_vector(5 downto 0) :="101011";
27
28     constant addFunc    : bit_vector(5 downto 0) :="100000";
29     constant subFunc    : bit_vector(5 downto 0) :="100010";
30     constant andFunc    : bit_vector(5 downto 0) :="100100";
```

```
31      constant orFunc      : bit_vector(5 downto 0) :="100101";
32      constant sltFunc     : bit_vector(5 downto 0) :="101010";
33      constant sllFunc     : bit_vector(5 downto 0) :="000000";
34
35      constant UDEXP       : std_logic_vector(31 downto 0) := x"00000003";
36      constant OVFEXP      : std_logic_vector(31 downto 0) := x"00000001";
37
38
39
40      type t_aluSrcA     is (ASA_PC, ASA_REG_A);
41      type t_aluSrcB     is (ASB_REGB, ASB_FOUR, ASB_SEXT, ASB_SEXTS);
42      type t_aluOp       is (AOP_AND, AOP_OR, AOP_ADD, AOP_SUB, AOP_SLT,
            AOP_NOR, AOP_SLL);
43      type t_pcSrc       is (PS_PCINC, PS_ALUOUT, PS_JMP, PS_FOUR);
44      type t_regDst      is (RD_RT, RD_RD, RD_RA);
45      type t_iord        is (IOD_PC, IOD_ALUOUT);
46      type t_memToReg    is (MTR_ALUOUT, MTR_MDR, MTR_PC);
47      type t_comp        is (eq, ne, gt, lt, lte, gte);
48      type t_microinstr is (
49           s_if,     --Instruction Fetch
50           s_id      --Instruction Decode
51      );
52
53
54  end package;
```

Listing 1: mips_lib.vhdl

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File           : reg.vhdl
5  -- Creation Date : 06/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13
14 ----------------------------------------------------------------------------
15 Entity reg is
16 ----------------------------------------------------------------------------
17 generic
18 (
19     SIZE  : natural := 32;
20     DELAY : time := 0 ns
21 );
22
23
24 port
25 (
26     en     : in std_logic;
27     data   : in std_logic_vector((SIZE-1) downto 0);
28     output : out std_logic_vector((SIZE-1) downto 0)
29 );
30 end entity;
31
```

```
32
33 ----------------------------------------------------------------------------
34 Architecture reg_1 of reg is
35 ----------------------------------------------------------------------------
36     signal tmp : std_logic_vector((SIZE-1) downto 0) := (others => '0');
37 begin
38     process(en)
39     begin
40         if(en='1') then
41             tmp <= data;
42         end if;
43     end process;
44     output <= tmp;
45 end architecture reg_1;
```

Listing 2: Register

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : reg_file.vhdl
5  -- Creation Date : 06/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use ieee.std_logic_arith.all;
14 use ieee.std_logic_unsigned.all;
15 ----------------------------------------------------------------------------
16 Entity reg_file is
17 ----------------------------------------------------------------------------
18 generic
19 (
20     SIZE : natural :=32;
21     DELAY : time := 0 ns
22 );
23 port
24 (
25     clk        : in std_logic;
26     write_en   : in std_logic;
27     read_r1    : in std_logic_vector(4 downto 0);
28     read_r2    : in std_logic_vector(4 downto 0);
29     write_r    : in std_logic_vector(4 downto 0);
30     write_data : in std_logic_vector((SIZE-1) downto 0);
31     data_r1    : out std_logic_vector((SIZE-1) downto 0);
32     data_r2    : out std_logic_vector((SIZE-1) downto 0)
33 );
34 end entity;
35
36
37 ----------------------------------------------------------------------------
38 Architecture reg_file_1 of reg_file is
39 ----------------------------------------------------------------------------
40     type t_reg is array (0 to 31) of std_logic_vector(31 downto 0);
41     signal reg : t_reg := ((others => (others=>'0')));
42
```

```
43 begin
44     process(clk)
45     begin
46         if(clk'event and clk='0') then
47             data_r1 <= reg(CONV_INTEGER(read_r1)) after DELAY/2;
48             data_r2 <= reg(CONV_INTEGER(read_r2)) after DELAY/2;
49         elsif(clk'event and clk='1') then
50             if(write_en ='1') then
51                 reg(CONV_INTEGER(write_r)) <= write_data after DELAY;
52             end if;
53         end if;
54         reg(0) <= (others => '0');
55     end process;
56 end architecture reg_file_1;
```

Listing 3: Register File

```
1  ----------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : alu.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------
9
10 library IEEE;
11 use work.mips_lib.all;
12 use IEEE.STD_LOGIC_1164.ALL;
13 use IEEE.numeric_std.all;
14 ----------------------------------------------------------------------
15 Entity alu is
16 ----------------------------------------------------------------------
17 port
18 (
19     op_A      : in std_logic_vector(31 downto 0);
20     op_B      : in std_logic_vector(31 downto 0);
21     shamt     : in std_logic_vector(4 downto 0);
22     alu_ctrl  : in t_aluOp;
23     f         : out std_logic_vector(31 downto 0);
24     zero      : out std_logic;
25     overflow  : out std_logic
26 );
27 end entity;
28
29
30
31 ----------------------------------------------------------------------
32 Architecture alu_1 of alu is
33 ----------------------------------------------------------------------
34 -- from http://www.csee.umbc.edu/~squire/download/bshift.vhdl
35   function to_integer(sig : std_logic_vector) return integer is
36     variable num : integer := 0;  -- descending sig as integer
37   begin
38     for i in sig'range loop
39       if sig(i)='1' then
40         num := num*2+1;
41       else
42         num := num*2;
```

```
43        end if;
44      end loop;  -- i
45      return num;
46    end function to_integer;
47      CONSTANT DELAY : time := 0 ns;
48      signal value    : std_logic_vector(31 downto 0);
49  begin
50      process(alu_ctrl, op_A, op_B)
51      begin
52          case alu_ctrl is
53              when AOP_AND =>
54                  value <= op_A and op_B after DELAY;
55              when AOP_OR =>
56                  value <= op_A or op_B after DELAY;
57              when AOP_ADD =>
58                  value <= std_logic_vector(signed(op_A) + signed(op_B))
                        after DELAY;
59              when AOP_SUB =>
60                  value <= std_logic_vector(signed(op_A) - signed(op_B))
                        after DELAY;
61              when AOP_SLT =>
62                  if(signed(op_A) < signed(op_B)) then
63                      value <= (others => '0');
64                      value(0) <=  '1';
65                  else
66                      value <= (others => '0');
67                  end if;
68              when AOP_NOR =>
69                  value <= NOT (op_A or op_B) after DELAY;
70              when AOP_SLL =>
71                  value <= to_stdlogicvector(to_bitvector(op_B) sll
                        to_integer(shamt));
72              when others =>
73                  value <= (others => '1');
74          end case;
75        end process;
76        f <= value;
77        zero <= '1' when value = x"00000000" else
78                '0';
79  end architecture alu_1;
```

Listing 4: ALU

```
1  ----------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File           : datapath.vhdl
5  -- Creation Date : 06/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------
16 Entity datapath is
```

```
17 ------------------------------------------------------------------------------
18     port(
19             clk             : in std_logic;
20     -- Control Unit
21             PCWriteCondEq : in std_logic;
22             PCWriteCondNEq : in std_logic;
23             PCWrite         : in std_logic;
24             IorD            : in t_iord;
25             MemRead         : in std_logic;
26             MemWrite        : in std_logic;
27             MemToReg        : in t_memToReg;
28             IRWrite         : in std_logic;
29             RegWrite        : in std_logic;
30             RegDst          : in t_regDst;
31             ALUSrcA         : in t_aluSrcA;
32             ALUSrcB         : in t_aluSrcB;
33             PCSource        : in t_pcSrc;
34             ALUOp           : in t_aluOp;
35             UndefInstrEx    : in std_logic;
36             OverflowEx      : out std_logic;
37             Exception       :in std_logic;
38
39     --Memory
40             mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
41             mem_read        : out std_logic;
42             addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
43             mem_write_data  : out std_logic_vector((DATA_WIDTH-1) downto 0)
44         );
45
46 end entity;
47
48
49
50 ------------------------------------------------------------------------------
51 Architecture datapath_1 of datapath is
52 ------------------------------------------------------------------------------
53     component reg
54         generic (
55                     SIZE  : natural := 32;
56                     DELAY : time := 0 ns
57                 );
58         port (
59                  en     : in std_logic;
60                  data   : in std_logic_vector((SIZE-1) downto 0);
61                  output : out std_logic_vector((SIZE-1) downto 0)
62              );
63     end component reg;
64
65     component reg_file
66         generic (
67                     SIZE  : natural :=32;
68                     DELAY : time := 0 ns
69                 );
70         port (
71                  clk        : in std_logic;
72                  write_en   : in std_logic;
73                  read_r1    : in std_logic_vector(4 downto 0);
74                  read_r2    : in std_logic_vector(4 downto 0);
```

```
75                  write_r     : in std_logic_vector(4 downto 0);
76                  write_data  : in std_logic_vector((SIZE-1) downto 0);
77                  data_r1     : out std_logic_vector((SIZE-1) downto 0);
78                  data_r2     : out std_logic_vector((SIZE-1) downto 0)
79              );
80      end component reg_file;
81
82      component alu
83          port(
84                  op_A      : in std_logic_vector(31 downto 0);
85                  op_B      : in std_logic_vector(31 downto 0);
86                  shamt     : in std_logic_vector(4 downto 0);
87                  alu_ctrl  : in t_aluOp;
88                  f         : out std_logic_vector(31 downto 0);
89                  zero      : out std_logic;
90                  overflow  : out std_logic
91              );
92      end component alu;
93
94      signal PCDATA_int      : std_logic_vector((DATA_WIDTH - 1) downto 0);
95      signal PCOUT_int       : std_logic_vector((DATA_WIDTH - 1) downto 0);
96      signal instruction     : std_logic_vector((DATA_WIDTH - 1) downto 0);
97      signal mdreg           : std_logic_vector((DATA_WIDTH - 1) downto 0);
98      signal rf_write_data   : std_logic_vector((DATA_WIDTH - 1) downto 0);
99      signal rf_write_reg    : std_logic_vector(4 downto 0);
100     signal alu_a           : std_logic_vector((DATA_WIDTH - 1) downto 0);
101     signal alu_b           : std_logic_vector((DATA_WIDTH - 1) downto 0);
102     signal alu_reg_in      : std_logic_vector((DATA_WIDTH - 1) downto 0);
103     signal alu_reg_out     : std_logic_vector((DATA_WIDTH - 1) downto 0);
104     signal rega_in         : std_logic_vector((DATA_WIDTH - 1) downto 0);
105     signal regb_in         : std_logic_vector((DATA_WIDTH - 1) downto 0);
106     signal rega_out        : std_logic_vector((DATA_WIDTH - 1) downto 0);
107     signal regb_out        : std_logic_vector((DATA_WIDTH - 1) downto 0);
108     signal epc_out         : std_logic_vector((DATA_WIDTH - 1) downto 0);
109     signal cause_out       : std_logic_vector((DATA_WIDTH - 1) downto 0);
110     signal cause_data      : std_logic_vector((DATA_WIDTH - 1) downto 0);
111     signal instr_sext      : std_logic_vector((DATA_WIDTH - 1) downto 0);
112     signal instr_sexts     : std_logic_vector((DATA_WIDTH - 1) downto 0);
113     signal zero            : std_logic;
114     signal overflow        : std_logic;
115     signal do_cause        : std_logic;
116     signal PCWrite_int     : std_logic;
117 begin
118     OverflowEx <= overflow;
119     RF: reg_file
120     port map(
121                 clk => clk,
122                 write_en => RegWrite,
123                 read_r1 => instruction(25 downto 21),
124                 read_r2 => instruction(20 downto 16),
125                 write_r => rf_write_reg,
126                 write_data => rf_write_data,
127                 data_r1 => rega_in,
128                 data_r2 => regb_in
129             );
130
131     PC: reg
132     port map(
```

```
133                 en      => PCWRITE_int,
134                 data    => PCDATA_int,
135                 output => PCOUT_int
136             );
137
138     IR: reg
139     port map(
140                 en      => IRWrite,
141                 data    => mem_data_out,
142                 output => instruction
143             );
144     AOR: reg
145     port map(
146                 en => clk,
147                 data => alu_reg_in,
148                 output => alu_reg_out
149             );
150
151     MDR: reg
152     port map(
153                 en      => clk,
154                 data    => mem_data_out,
155                 output => mdreg
156             );
157
158     RRA: reg
159     port map(
160                 en      => clk,
161                 data    => rega_in,
162                 output => rega_out
163             );
164
165     RRB: reg
166     port map(
167                 en      => clk,
168                 data    => regb_in,
169                 output => regb_out
170             );
171
172     ALUU: alu
173     port map(
174                 op_A      => alu_a,
175                 op_B      => alu_b,
176                 alu_ctrl => ALUOp,
177                 f         => alu_reg_in,
178                 zero      => zero,
179                 shamt     => instruction(10 downto 6),
180                 overflow => overflow
181             );
182
183     EPC: reg
184     port map(
185                 en      => Exception,
186                 data    => PCOUT_int,
187                 output => epc_out
188             );
189
190     do_cause <= UndefInstrEx or overflow;
```

```
191     CAUSE: reg
192     port map(
193                 en      => do_cause,
194                 data    => cause_data,
195                 output => cause_out
196             );
197
198     mem_write_data <= regb_out;
199
200     CAUSEPROC: process(do_cause)
201     begin
202         if(do_cause='1') then
203             if(UndefInstrEx = '1') then
204                 cause_data <= UDEXP;
205             else
206                 cause_data <= OVFEXP;
207             end if;
208         end if;
209     end process;
210     ---- PC Write/Branch MUX ----
211     PCFinal : process(PCWriteCondEq, PCWriteCondNEq, PCWrite)
212     begin
213         if((PCWriteCondEq='1' and zero='1') or (PCWriteCondNEq='1' and zero
                ='0') or PCWrite='1') then
214             PCWRITE_int <= '1';
215         else
216             PCWRITE_int <= '0';
217         end if;
218     end process;
219
220     -- PCSource Mux --
221     PCSMux: process(PCSource, alu_reg_out, alu_reg_in)
222     begin
223         if(PCSource = PS_PCINC) then
224             PCDATA_int <= alu_reg_in;
225         elsif(PCSource = PS_ALUOUT) then
226             PCDATA_int <= alu_reg_out;
227         elsif(PCSource = PS_JMP) then
228             PCDATA_int <= PCOUT_int(31 downto 28) & instruction(25 downto
                    0) & "00";
229         elsif(PCSource = PS_FOUR) then
230             PCDATA_int <= x"00000004";
231         end if;
232     end process;
233
234     -- IorD Mux
235     IorDMux : process (PCOUT_int, alu_reg_out, IorD)
236     begin
237         if(IorD = IOD_PC) then
238             addr_bus <= PCOUT_int;
239         else
240             addr_bus <= alu_reg_out;
241         end if;
242     end process;
243
244     -- MemToReg Mux --
245     MTRMux: process (MemToReg, mdreg, alu_reg_out)
246     begin
```

```
247         if(MemToReg = MTR_ALUOUT) then
248             rf_write_data <= alu_reg_out;
249         elsif(MemToReg = MTR_MDR) then
250             rf_write_data <= mdreg;
251         else
252             rf_write_data <= PCOUT_int;
253         end if;
254     end process;
255
256     -- RegDst Mux --
257     RDMux : process (instruction, RegDst)
258     begin
259         if(RegDst = RD_RT) then
260             rf_write_reg <= instruction(20 downto 16);
261         elsif(RegDst = RD_RD) then
262             rf_write_reg <= instruction(15 downto 11);
263         else
264             rf_write_reg <= "11111";
265         end if;
266     end process;
267
268     -- ALUSrcA Mux --
269     ALUSA: process (ALUSrcA, PCOUT_int, rega_out)
270     begin
271         if(ALUSrcA = ASA_PC) then
272             alu_a <= PCOUT_int;
273         else
274             alu_a <= rega_out;
275         end if;
276     end process;
277
278     -- ALUSrcB Mux --
279     ALUSB: process (ALUSrcB, regb_out, instr_sext, instr_sexts)
280     begin
281         if(ALUSrcB = ASB_REGB) then
282             alu_b <= regb_out;
283         elsif(ALUSrcB = ASB_FOUR) then
284             alu_b <= "00000000000000000000000000000100";
285         elsif(ALUSrcB = ASB_SEXT) then
286             alu_b <= instr_sext;
287         else
288             alu_b <= instr_sexts;
289         end if;
290     end process;
291
292     -- Sign Extend and Shift --
293     SEXTS: process(instruction)
294     begin
295         instr_sext <= (31 downto 16 => instruction(15)) & instruction(15
                downto 0);
296         instr_sexts <= (31 downto 18 => instruction(15)) & instruction(15
                downto 0) & "00";
297     end process;
298
299 end architecture datapath_1;
```

Listing 5: Datapath

```
1 ----------------------------------------------------------------------------
```

```
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File           : ADDI_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity ADDI_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture ADDI_Testbench_1 of ADDI_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq : in std_logic;
31             PCWriteCondNEq : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
35             MemWrite        : in std_logic;
36             MemToReg        : in t_memToReg;
37             IRWrite         : in std_logic;
38             RegWrite        : in std_logic;
39             RegDst          : in t_regDst;
40             ALUSrcA         : in t_aluSrcA;
41             ALUSrcB         : in t_aluSrcB;
42             PCSource        : in t_pcSrc;
43             ALUOp           : in t_aluOp;
44
45             UndefInstrEx    : in std_logic;
46             OverflowEx      : out std_logic;
47             Exception       : in std_logic;
48
49     --Memory
50             mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
51             mem_read        : out std_logic;
52             addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
53             mem_write_data  : out std_logic_vector((DATA_WIDTH-1) downto 0)
54         );
55     end component datapath;
56
57     signal PCWriteCondEq : std_logic;
58     signal PCWriteCondNEq : std_logic;
59     signal PCWrite        : std_logic;
```

```
60      signal IorD           : t_iord;
61      signal MemRead        : std_logic;
62      signal MemWrite       : std_logic;
63      signal MemToReg       : t_memToReg;
64      signal IRWrite        : std_logic;
65      signal RegWrite       : std_logic;
66      signal RegDst         : t_regDst;
67      signal ALUSrcA        : t_aluSrcA;
68      signal ALUSrcB        : t_aluSrcB;
69      signal PCSource       : t_pcSrc;
70      signal ALUOp          : t_aluOp;
71
72      signal UndefInstrEx   : std_logic;
73      signal OverflowEx     : std_logic;
74      signal Exception      : std_logic;
75
76  --Memory
77      signal mem_data_out    : std_logic_vector((DATA_WIDTH-1) downto 0);
78      signal mem_read        : std_logic;
79      signal addr_bus        : std_logic_vector((DATA_WIDTH-1) downto 0);
80      signal mem_write_data  : std_logic_vector((DATA_WIDTH-1) downto 0);
81
82  begin
83    --Create a clock.
84      PROCESS
85      BEGIN
86          Clk <= '0';
87          WAIT FOR T/2;
88          Clk <= '1';
89          WAIT FOR T/2;
90      END PROCESS;
91
92      DUT: entity work.datapath
93      port map(
94                  clk            => clk,
95                  PCWriteCondEq  => PCWriteCondEq,
96                  PCWriteCondNEq => PCWriteCondNEq,
97                  PCWrite        => PCWrite,
98                  IorD           => IorD,
99                  MemRead        => MemRead,
100                 MemWrite       => MemWrite,
101                 MemToReg       => MemToReg,
102                 IRWrite        => IRWrite,
103                 RegWrite       => RegWrite,
104                 RegDst         => RegDst,
105                 ALUSrcA        => ALUSrcA,
106                 ALUSrcB        => ALUSrcB,
107                 PCSource       => PCSource,
108                 ALUOp          => ALUOp,
109                 UndefInstrEx   => UndefInstrEx,
110                 OverflowEx     => OverflowEx,
111                 Exception      => Exception,
112
113                 mem_data_out   => mem_data_out,
114                 mem_read       => mem_read,
115                 addr_bus       => addr_bus,
116                 mem_write_data => mem_write_data
117             );
```

```
118
119     PROCESS
120     BEGIN
121         PCWriteCondEq <= '0';
122         PCWrite <= '0';
123         IorD <= IOD_PC;
124         MemRead <= '0';
125         MemWrite <= '0';
126         MemToReg <= MTR_ALUOUT;
127         IRWrite <= '0';
128         RegWrite <= '0';
129         RegDst <= RD_RT;
130         ALUSrcA <= ASA_PC;
131         ALUSrcB <= ASB_REGB;
132         PCSource <= PS_PCINC;
133         ALUOp <= AOP_AND;
134
135         -- R[1] <= R[0] + 8
136         mem_data_out <= "001000" & "00000" & "00001" & "0000000000001000";
137         wait for T;
138
139         -- IF
140         MemRead <= '1';
141         ALUSrcA <= ASA_PC;
142         IorD <= IOD_PC;
143         IRWrite <= '1';
144         ALUSrcB <= ASB_FOUR;
145         ALUOp <= AOP_ADD;
146         PCWrite <= '0';
147         PCSource <= PS_PCINC;
148         wait for T/2;
149         PCWrite <= '1';
150         wait for T/2;
151
152         -- ID/Bra Calc
153         PCWrite <= '0';
154         MemRead <= '0';
155         IRWrite <= '0';
156         ALUSrcA <= ASA_PC;
157         ALUSrcB <= ASB_SEXTS;
158         PCSource <= PS_PCINC;
159         ALUOp <= AOP_ADD;
160         wait for T;
161
162         -- Execute
163         ALUSrcA <= ASA_REG_A;
164         ALUSrcB <= ASB_SEXT;
165         ALUOp <= AOP_ADD;
166         wait for T;
167
168
169         --Write Back
170         MemToReg <= MTR_ALUOUT;
171         RegDst <= RD_RT;
172         wait for T/2;
173         RegWrite <= '1';
174         wait for T/2;
175
```

```
176
177     END PROCESS;
178
179 end architecture ADDI_Testbench_1;
```

Listing 6: ADDI_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : ADD_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity ADD_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture ADD_Testbench_1 of ADD_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq   : in std_logic;
31             PCWriteCondNEq  : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
35             MemWrite        : in std_logic;
36             MemToReg        : in t_memToReg;
37             IRWrite         : in std_logic;
38             RegWrite        : in std_logic;
39             RegDst          : in t_regDst;
40             ALUSrcA         : in t_aluSrcA;
41             ALUSrcB         : in t_aluSrcB;
42             PCSource        : in t_pcSrc;
43             ALUOp           : in t_aluOp;
44
45             UndefInstrEx    : in std_logic;
46             OverflowEx      : out std_logic;
47             Exception       : in std_logic;
48
49     --Memory
50             mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
51             mem_read        : out std_logic;
52             addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
```

```
53              mem_write_data : out std_logic_vector((DATA_WIDTH-1) downto 0)
54          );
55      end component datapath;
56
57      signal PCWriteCondEq : std_logic;
58      signal PCWriteCondNEq : std_logic;
59      signal PCWrite       : std_logic;
60      signal IorD          : t_iord;
61      signal MemRead       : std_logic;
62      signal MemWrite      : std_logic;
63      signal MemToReg      : t_memToReg;
64      signal IRWrite       : std_logic;
65      signal RegWrite      : std_logic;
66      signal RegDst        : t_regDst;
67      signal ALUSrcA       : t_aluSrcA;
68      signal ALUSrcB       : t_aluSrcB;
69      signal PCSource      : t_pcSrc;
70      signal ALUOp         : t_aluOp;
71
72      signal UndefInstrEx  : std_logic;
73      signal OverflowEx    : std_logic;
74      signal Exception     : std_logic;
75
76  --Memory
77      signal mem_data_out   : std_logic_vector((DATA_WIDTH-1) downto 0);
78      signal mem_read       : std_logic;
79      signal addr_bus       : std_logic_vector((DATA_WIDTH-1) downto 0);
80      signal mem_write_data : std_logic_vector((DATA_WIDTH-1) downto 0);
81
82  begin
83    --Create a clock.
84      PROCESS
85      BEGIN
86          Clk <= '0';
87          WAIT FOR T/2;
88          Clk <= '1';
89          WAIT FOR T/2;
90      END PROCESS;
91
92      DUT: entity work.datapath
93      port map(
94                  clk           => clk,
95                  PCWriteCondEq => PCWriteCondEq,
96                  PCWriteCondNEq => PCWriteCondNEq,
97                  PCWrite       => PCWrite,
98                  IorD          => IorD,
99                  MemRead       => MemRead,
100                 MemWrite      => MemWrite,
101                 MemToReg      => MemToReg,
102                 IRWrite       => IRWrite,
103                 RegWrite      => RegWrite,
104                 RegDst        => RegDst,
105                 ALUSrcA       => ALUSrcA,
106                 ALUSrcB       => ALUSrcB,
107                 PCSource      => PCSource,
108                 ALUOp         => ALUOp,
109                 UndefInstrEx  => UndefInstrEx,
110                 OverflowEx    => OverflowEx,
```

```
111                 Exception      => Exception,
112
113                 mem_data_out   => mem_data_out,
114                 mem_read       => mem_read,
115                 addr_bus       => addr_bus,
116                 mem_write_data => mem_write_data
117             );
118
119     PROCESS
120     BEGIN
121         PCWriteCondEq <= '0';
122         PCWrite <= '0';
123         IorD <= IOD_PC;
124         MemRead <= '0';
125         MemWrite <= '0';
126         MemToReg <= MTR_ALUOUT;
127         IRWrite <= '0';
128         RegWrite <= '0';
129         RegDst <= RD_RT;
130         ALUSrcA <= ASA_PC;
131         ALUSrcB <= ASB_REGB;
132         PCSource <= PS_PCINC;
133         ALUOp <= AOP_AND;
134
135         -- R[2] <= R[0] + R[1]
136         mem_data_out <= "000000" & "00000" & "00001" & "00010" & "00000" &
                "100000";
137         wait for T;
138
139         -- IF
140         MemRead <= '1';
141         ALUSrcA <= ASA_PC;
142         IorD <= IOD_PC;
143         IRWrite <= '1';
144         ALUSrcB <= ASB_FOUR;
145         ALUOp <= AOP_ADD;
146         PCWrite <= '0';
147         PCSource <= PS_PCINC;
148         wait for T/2;
149         PCWrite <= '1';
150         wait for T/2;
151
152         -- ID/Bra Calc
153         PCWrite <= '0';
154         MemRead <= '0';
155         IRWrite <= '0';
156         ALUSrcA <= ASA_PC;
157         ALUSrcB <= ASB_SEXTS;
158         PCSource <= PS_PCINC;
159         ALUOp <= AOP_ADD;
160         wait for T;
161
162         -- Execute
163         ALUSrcA <= ASA_REG_A;
164         ALUSrcB <= ASB_REGB;
165         ALUOp <= AOP_ADD;
166         wait for T;
167
```

```
168
169             --Write Back
170             MemToReg <= MTR_ALUOUT;
171             RegDst <= RD_RD;
172             wait for T/2;
173             RegWrite <= '1';
174             wait for T/2;
175
176
177         END PROCESS;
178
179     end architecture ADD_Testbench_1;
```

Listing 7: ADD_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : AND_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity AND_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture AND_Testbench_1 of AND_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq   : in std_logic;
31             PCWriteCondNEq  : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
35             MemWrite        : in std_logic;
36             MemToReg        : in t_memToReg;
37             IRWrite         : in std_logic;
38             RegWrite        : in std_logic;
39             RegDst          : in t_regDst;
40             ALUSrcA         : in t_aluSrcA;
41             ALUSrcB         : in t_aluSrcB;
42             PCSource        : in t_pcSrc;
43             ALUOp           : in t_aluOp;
44
```

```
45              UndefInstrEx   : in std_logic;
46              OverflowEx     : out std_logic;
47              Exception      : in std_logic;
48
49      --Memory
50              mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
51              mem_read        : out std_logic;
52              addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
53              mem_write_data  : out std_logic_vector((DATA_WIDTH-1) downto 0)
54          );
55      end component datapath;
56
57      signal PCWriteCondEq : std_logic;
58      signal PCWriteCondNEq : std_logic;
59      signal PCWrite        : std_logic;
60      signal IorD           : t_iord;
61      signal MemRead        : std_logic;
62      signal MemWrite       : std_logic;
63      signal MemToReg       : t_memToReg;
64      signal IRWrite        : std_logic;
65      signal RegWrite       : std_logic;
66      signal RegDst         : t_regDst;
67      signal ALUSrcA        : t_aluSrcA;
68      signal ALUSrcB        : t_aluSrcB;
69      signal PCSource       : t_pcSrc;
70      signal ALUOp          : t_aluOp;
71
72      signal UndefInstrEx   : std_logic;
73      signal OverflowEx     : std_logic;
74      signal Exception      : std_logic;
75
76  --Memory
77      signal mem_data_out    : std_logic_vector((DATA_WIDTH-1) downto 0);
78      signal mem_read        : std_logic;
79      signal addr_bus        : std_logic_vector((DATA_WIDTH-1) downto 0);
80      signal mem_write_data  : std_logic_vector((DATA_WIDTH-1) downto 0);
81
82  begin
83    --Create a clock.
84      PROCESS
85      BEGIN
86          Clk <= '0';
87          WAIT FOR T/2;
88          Clk <= '1';
89          WAIT FOR T/2;
90      END PROCESS;
91
92      DUT: entity work.datapath
93      port map(
94                  clk            => clk,
95                  PCWriteCondEq  => PCWriteCondEq,
96                  PCWriteCondNEq => PCWriteCondNEq,
97                  PCWrite        => PCWrite,
98                  IorD           => IorD,
99                  MemRead        => MemRead,
100                 MemWrite       => MemWrite,
101                 MemToReg       => MemToReg,
102                 IRWrite        => IRWrite,
```

```
103                 RegWrite        => RegWrite,
104                 RegDst          => RegDst,
105                 ALUSrcA         => ALUSrcA,
106                 ALUSrcB         => ALUSrcB,
107                 PCSource        => PCSource,
108                 ALUOp           => ALUOp,
109                 UndefInstrEx    => UndefInstrEx,
110                 OverflowEx      => OverflowEx,
111                 Exception       => Exception,
112
113                 mem_data_out    => mem_data_out,
114                 mem_read        => mem_read,
115                 addr_bus        => addr_bus,
116                 mem_write_data  => mem_write_data
117             );
118
119     PROCESS
120     BEGIN
121         PCWriteCondEq <= '0';
122         PCWrite <= '0';
123         IorD <= IOD_PC;
124         MemRead <= '0';
125         MemWrite <= '0';
126         MemToReg <= MTR_ALUOUT;
127         IRWrite <= '0';
128         RegWrite <= '0';
129         RegDst <= RD_RT;
130         ALUSrcA <= ASA_PC;
131         ALUSrcB <= ASB_REGB;
132         PCSource <= PS_PCINC;
133         ALUOp <= AOP_AND;
134
135
136
137         -- R[2] <= R[2] & R[1]
138         mem_data_out <= "000000" & "00010" & "00001" & "00010" & "00000" &
                "100100";
139         wait for T;
140
141         -- IF
142         MemRead <= '1';
143         ALUSrcA <= ASA_PC;
144         IorD <= IOD_PC;
145         IRWrite <= '1';
146         ALUSrcB <= ASB_FOUR;
147         ALUOp <= AOP_ADD;
148         PCWrite <= '0';
149         PCSource <= PS_PCINC;
150         wait for T/2;
151         PCWrite <= '1';
152         wait for T/2;
153
154         -- ID/Bra Calc
155         PCWrite <= '0';
156         MemRead <= '0';
157         IRWrite <= '0';
158         ALUSrcA <= ASA_PC;
159         ALUSrcB <= ASB_SEXTS;
```

```
160             PCSource <= PS_PCINC;
161             ALUOp <= AOP_ADD;
162             wait for T;
163
164             -- Execute
165             ALUSrcA <= ASA_REG_A;
166             ALUSrcB <= ASB_REGB;
167             ALUOp <= AOP_AND;
168             wait for T;
169
170
171             --Write Back
172             MemToReg <= MTR_ALUOUT;
173             RegDst <= RD_RD;
174             wait for T/2;
175             RegWrite <= '1';
176             wait for T/2;
177
178
179         END PROCESS;
180
181     end architecture AND_Testbench_1;
```

Listing 8: AND_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : BEQ_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity BEQ_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture BEQ_Testbench_1 of BEQ_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq : in std_logic;
31             PCWriteCondNEq : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
```

```
35              MemWrite       : in std_logic;
36              MemToReg       : in t_memToReg;
37              IRWrite        : in std_logic;
38              RegWrite       : in std_logic;
39              RegDst         : in t_regDst;
40              ALUSrcA        : in t_aluSrcA;
41              ALUSrcB        : in t_aluSrcB;
42              PCSource       : in t_pcSrc;
43              ALUOp          : in t_aluOp;
44              UndefInstrEx   : in std_logic;
45              OverflowEx     : out std_logic;
46              Exception      : in std_logic;
47
48      --Memory
49              mem_data_out   : in  std_logic_vector((DATA_WIDTH-1) downto 0);
50              mem_read       : out std_logic;
51              addr_bus       : out std_logic_vector((DATA_WIDTH-1) downto 0);
52              mem_write_data : out std_logic_vector((DATA_WIDTH-1) downto 0)
53          );
54      end component datapath;
55
56      signal PCWriteCondEq : std_logic;
57      signal PCWriteCondNEq : std_logic;
58      signal PCWrite       : std_logic;
59      signal IorD          : t_iord;
60      signal MemRead       : std_logic;
61      signal MemWrite      : std_logic;
62      signal MemToReg      : t_memToReg;
63      signal IRWrite       : std_logic;
64      signal RegWrite      : std_logic;
65      signal RegDst        : t_regDst;
66      signal ALUSrcA       : t_aluSrcA;
67      signal ALUSrcB       : t_aluSrcB;
68      signal PCSource      : t_pcSrc;
69      signal ALUOp         : t_aluOp;
70      signal UndefInstrEx  : std_logic;
71      signal OverflowEx    : std_logic;
72      signal Exception     : std_logic;
73
74  --Memory
75      signal mem_data_out   : std_logic_vector((DATA_WIDTH-1) downto 0);
76      signal mem_read       : std_logic;
77      signal addr_bus       : std_logic_vector((DATA_WIDTH-1) downto 0);
78      signal mem_write_data : std_logic_vector((DATA_WIDTH-1) downto 0);
79
80  begin
81    --Create a clock.
82      PROCESS
83      BEGIN
84          Clk <= '0';
85          WAIT FOR T/2;
86          Clk <= '1';
87          WAIT FOR T/2;
88      END PROCESS;
89
90      DUT: entity work.datapath
91      port map(
92                  clk          => clk,
```

```
93                  PCWriteCondEq => PCWriteCondEq,
94                  PCWriteCondNEq => PCWriteCondNEq,
95                  PCWrite       => PCWrite,
96                  IorD          => IorD,
97                  MemRead       => MemRead,
98                  MemWrite      => MemWrite,
99                  MemToReg      => MemToReg,
100                 IRWrite       => IRWrite,
101                 RegWrite      => RegWrite,
102                 RegDst        => RegDst,
103                 ALUSrcA       => ALUSrcA,
104                 ALUSrcB       => ALUSrcB,
105                 PCSource      => PCSource,
106                 ALUOp         => ALUOp,
107                 UndefInstrEx  => UndefInstrEx,
108                 OverflowEx    => OverflowEx,
109                 Exception     => Exception,
110
111                 mem_data_out   => mem_data_out,
112                 mem_read       => mem_read,
113                 addr_bus       => addr_bus,
114                 mem_write_data => mem_write_data
115             );
116
117     PROCESS
118     BEGIN
119         PCWriteCondEq <= '0';
120         PCWrite <= '0';
121         IorD <= IOD_PC;
122         MemRead <= '0';
123         MemWrite <= '0';
124         MemToReg <= MTR_ALUOUT;
125         IRWrite <= '0';
126         RegWrite <= '0';
127         RegDst <= RD_RT;
128         ALUSrcA <= ASA_PC;
129         ALUSrcB <= ASB_REGB;
130         PCSource <= PS_PCINC;
131         ALUOp <= AOP_AND;
132
133         -- BEQ R1, R0, 8
134         mem_data_out <= "000100" & "00000" & "00001" & "0000000000000010";
135         wait for T;
136
137         -- IF
138         MemRead <= '1';
139         ALUSrcA <= ASA_PC;
140         IorD <= IOD_PC;
141         IRWrite <= '1';
142         ALUSrcB <= ASB_FOUR;
143         ALUOp <= AOP_ADD;
144         PCWrite <= '0';
145         PCSource <= PS_PCINC;
146         wait for T/2;
147         PCWrite <= '1';
148         wait for T/2;
149
150         -- ID/Bra Calc
```

```
151             PCWrite <= '0';
152             MemRead <= '0';
153             IRWrite <= '0';
154             ALUSrcA <= ASA_PC;
155             ALUSrcB <= ASB_SEXTS;
156             PCSource <= PS_PCINC;
157             ALUOp <= AOP_ADD;
158             wait for T;
159
160             -- Branch Completion
161             ALUSrcA <= ASA_REG_A;
162             ALUSrcB <= ASB_REGB;
163             ALUOp <= AOP_SUB;
164             PCWriteCondEq <= '0';
165             PCSource <= PS_ALUOUT;
166             wait for T/2;
167             PCWriteCondEq <= '1';
168             wait for T/2;
169
170         END PROCESS;
171
172     end architecture BEQ_Testbench_1;
```

Listing 9: BEQ_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : BNEQ_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity BNEQ_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture BNEQ_Testbench_1 of BNEQ_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk          : in std_logic;
29     -- Control Unit
30             PCWriteCondEq : in std_logic;
31             PCWrite       : in std_logic;
32             IorD          : in t_iord;
33             MemRead       : in std_logic;
34             MemWrite      : in std_logic;
```

```
35              MemToReg       : in t_memToReg;
36              IRWrite        : in std_logic;
37              RegWrite       : in std_logic;
38              RegDst         : in t_regDst;
39              ALUSrcA        : in t_aluSrcA;
40              ALUSrcB        : in t_aluSrcB;
41              PCSource       : in t_pcSrc;
42              ALUOp          : in t_aluOp;
43              UndefInstrEx   : in std_logic;
44              OverflowEx     : out std_logic;
45              Exception      : in std_logic;
46
47      --Memory
48              mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
49              mem_read        : out std_logic;
50              addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
51              mem_write_data  : out std_logic_vector((DATA_WIDTH-1) downto 0)
52          );
53      end component datapath;
54
55      signal PCWriteCondEq : std_logic;
56      signal PCWriteCondNEq : std_logic;
57      signal PCWrite        : std_logic;
58      signal IorD           : t_iord;
59      signal MemRead        : std_logic;
60      signal MemWrite       : std_logic;
61      signal MemToReg       : t_memToReg;
62      signal IRWrite        : std_logic;
63      signal RegWrite       : std_logic;
64      signal RegDst         : t_regDst;
65      signal ALUSrcA        : t_aluSrcA;
66      signal ALUSrcB        : t_aluSrcB;
67      signal PCSource       : t_pcSrc;
68      signal ALUOp          : t_aluOp;
69      signal UndefInstrEx   : std_logic;
70      signal OverflowEx     : std_logic;
71      signal Exception      : std_logic;
72  --Memory
73      signal mem_data_out    : std_logic_vector((DATA_WIDTH-1) downto 0);
74      signal mem_read        : std_logic;
75      signal addr_bus        : std_logic_vector((DATA_WIDTH-1) downto 0);
76      signal mem_write_data  : std_logic_vector((DATA_WIDTH-1) downto 0);
77
78  begin
79    --Create a clock.
80      PROCESS
81      BEGIN
82          Clk <= '0';
83          WAIT FOR T/2;
84          Clk <= '1';
85          WAIT FOR T/2;
86      END PROCESS;
87
88      DUT: entity work.datapath
89      port map(
90                  clk            => clk,
91                  PCWriteCondEq  => PCWriteCondEq,
92                  PCWriteCondNEq => PCWriteCondNEq,
```

```
93                  PCWrite         => PCWrite,
94                  IorD            => IorD,
95                  MemRead         => MemRead,
96                  MemWrite        => MemWrite,
97                  MemToReg        => MemToReg,
98                  IRWrite         => IRWrite,
99                  RegWrite        => RegWrite,
100                 RegDst          => RegDst,
101                 ALUSrcA         => ALUSrcA,
102                 ALUSrcB         => ALUSrcB,
103                 PCSource        => PCSource,
104                 ALUOp           => ALUOp,
105
106                 UndefInstrEx    => UndefInstrEx,
107                 OverflowEx      => OverflowEx,
108                 Exception       => Exception,
109
110                 mem_data_out    => mem_data_out,
111                 mem_read        => mem_read,
112                 addr_bus        => addr_bus,
113                 mem_write_data  => mem_write_data
114             );
115
116     PROCESS
117     BEGIN
118         PCWriteCondEq <= '0';
119         PCWriteCondNEq <= '0';
120         PCWrite <= '0';
121         IorD <= IOD_PC;
122         MemRead <= '0';
123         MemWrite <= '0';
124         MemToReg <= MTR_ALUOUT;
125         IRWrite <= '0';
126         RegWrite <= '0';
127         RegDst <= RD_RT;
128         ALUSrcA <= ASA_PC;
129         ALUSrcB <= ASB_REGB;
130         PCSource <= PS_PCINC;
131         ALUOp <= AOP_AND;
132
133         -- BEQ R1, R0, 8
134         mem_data_out <= "000101" & "00000" & "00001" & "0000000000000010";
135         wait for T;
136
137         -- IF
138         MemRead <= '1';
139         ALUSrcA <= ASA_PC;
140         IorD <= IOD_PC;
141         IRWrite <= '1';
142         ALUSrcB <= ASB_FOUR;
143         ALUOp <= AOP_ADD;
144         PCWrite <= '0';
145         PCSource <= PS_PCINC;
146         wait for T/2;
147         PCWrite <= '1';
148         wait for T/2;
149
150         -- ID/Bra Calc
```

```
151             PCWrite <= '0';
152             MemRead <= '0';
153             IRWrite <= '0';
154             ALUSrcA <= ASA_PC;
155             ALUSrcB <= ASB_SEXTS;
156             PCSource <= PS_PCINC;
157             ALUOp <= AOP_ADD;
158             wait for T;
159
160             -- Branch Completion
161             ALUSrcA <= ASA_REG_A;
162             ALUSrcB <= ASB_REGB;
163             ALUOp <= AOP_SUB;
164             PCWriteCondNEq <= '0';
165             PCSource <= PS_ALUOUT;
166             wait for T/2;
167             PCWriteCondNEq <= '1';
168             wait for T/2;
169
170         END PROCESS;
171
172     end architecture BNEQ_Testbench_1;
```

Listing 10: BNEQ_Testbench

```
1  ------------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : IF_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ------------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ------------------------------------------------------------------------------
16 Entity IF_Testbench is
17 ------------------------------------------------------------------------------
18 end entity;
19
20
21 ------------------------------------------------------------------------------
22 Architecture IF_Testbench_1 of IF_Testbench is
23 ------------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq   : in std_logic;
31             PCWriteCondNEq  : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
```

```
35              MemWrite       : in std_logic;
36              MemToReg       : in t_memToReg;
37              IRWrite        : in std_logic;
38              RegWrite       : in std_logic;
39              RegDst         : in t_regDst;
40              ALUSrcA        : in t_aluSrcA;
41              ALUSrcB        : in t_aluSrcB;
42              PCSource       : in t_pcSrc;
43              ALUOp          : in t_aluOp;
44              UndefInstrEx   : in std_logic;
45              OverflowEx     : out std_logic;
46              Exception      : in std_logic;
47
48      --Memory
49              mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
50              mem_read        : out std_logic;
51              addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
52              mem_write_data  : out std_logic_vector((DATA_WIDTH-1) downto 0)
53          );
54      end component datapath;
55
56      signal PCWriteCondEq : std_logic;
57      signal PCWriteCondNEq : std_logic;
58      signal PCWrite       : std_logic;
59      signal IorD          : t_iord;
60      signal MemRead       : std_logic;
61      signal MemWrite      : std_logic;
62      signal MemToReg      : t_memToReg;
63      signal IRWrite       : std_logic;
64      signal RegWrite      : std_logic;
65      signal RegDst        : t_regDst;
66      signal ALUSrcA       : t_aluSrcA;
67      signal ALUSrcB       : t_aluSrcB;
68      signal PCSource      : t_pcSrc;
69      signal ALUOp         : t_aluOp;
70
71      signal UndefInstrEx  : std_logic;
72      signal OverflowEx    : std_logic;
73      signal Exception     : std_logic;
74
75
76  --Memory
77      signal mem_data_out    : std_logic_vector((DATA_WIDTH-1) downto 0);
78      signal mem_read        : std_logic;
79      signal addr_bus        : std_logic_vector((DATA_WIDTH-1) downto 0);
80      signal mem_write_data  : std_logic_vector((DATA_WIDTH-1) downto 0);
81
82  begin
83    --Create a clock.
84      PROCESS
85      BEGIN
86          Clk <= '0';
87          WAIT FOR T/2;
88          Clk <= '1';
89          WAIT FOR T/2;
90      END PROCESS;
91
92      DUT: entity work.datapath
```

```
93      port map(
94                  clk             => clk,
95                  PCWriteCondEq   => PCWriteCondEq,
96                  PCWriteCondNEq  => PCWriteCondNEq,
97                  PCWrite         => PCWrite,
98                  IorD            => IorD,
99                  MemRead         => MemRead,
100                 MemWrite        => MemWrite,
101                 MemToReg        => MemToReg,
102                 IRWrite         => IRWrite,
103                 RegWrite        => RegWrite,
104                 RegDst          => RegDst,
105                 ALUSrcA         => ALUSrcA,
106                 ALUSrcB         => ALUSrcB,
107                 PCSource        => PCSource,
108                 ALUOp           => ALUOp,
109                 UndefInstrEx    => UndefInstrEx,
110                 OverflowEx      => OverflowEx,
111                 Exception       => Exception,
112
113                 mem_data_out    => mem_data_out,
114                 mem_read        => mem_read,
115                 addr_bus        => addr_bus,
116                 mem_write_data  => mem_write_data
117         );
118
119     PROCESS
120     BEGIN
121         PCWriteCondEq <= '0';
122         PCWrite <= '0';
123         IorD <= IOD_PC;
124         MemRead <= '0';
125         MemWrite <= '0';
126         MemToReg <= MTR_ALUOUT;
127         IRWrite <= '0';
128         RegWrite <= '0';
129         RegDst <= RD_RT;
130         ALUSrcA <= ASA_PC;
131         ALUSrcB <= ASB_REGB;
132         PCSource <= PS_PCINC;
133         ALUOp <= AOP_AND;
134
135         --mem_data_out <= x"000000F0";
136         mem_data_out <= "00010000000000010000000000000010";
137         wait for T;
138
139         MemRead <= '1';
140         ALUSrcA <= ASA_PC;
141         IorD <= IOD_PC;
142         IRWrite <= '1';
143         ALUSrcB <= ASB_FOUR;
144         ALUOp <= AOP_ADD;
145         PCWrite <= '0';
146         PCSource <= PS_PCINC;
147         wait for T/2;
148         PCWrite <= '1';
149         wait for T/2;
150         PCWrite <= '0';
```

```
151             MemRead <= '0';
152             IRWrite <= '0';
153             ALUSrcA <= ASA_PC;
154             ALUSrcB <= ASB_SEXTS;
155             PCSource <= PS_PCINC;
156             ALUOp <= AOP_ADD;
157             wait for T;
158
159         END PROCESS;
160
161     end architecture IF_Testbench_1;
```

Listing 11: IF_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : JAL_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity JAL_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture JAL_Testbench_1 of JAL_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq : in std_logic;
31             PCWriteCondNEq : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
35             MemWrite        : in std_logic;
36             MemToReg        : in t_memToReg;
37             IRWrite         : in std_logic;
38             RegWrite        : in std_logic;
39             RegDst          : in t_regDst;
40             ALUSrcA         : in t_aluSrcA;
41             ALUSrcB         : in t_aluSrcB;
42             PCSource        : in t_pcSrc;
43             ALUOp           : in t_aluOp;
44             UndefInstrEx    : in std_logic;
45             OverflowEx      : out std_logic;
```

```
46              Exception      : in std_logic;
47
48      --Memory
49              mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
50              mem_read        : out std_logic;
51              addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
52              mem_write_data  : out std_logic_vector((DATA_WIDTH-1) downto 0)
53          );
54      end component datapath;
55
56      signal PCWriteCondEq : std_logic;
57      signal PCWriteCondNEq : std_logic;
58      signal PCWrite        : std_logic;
59      signal IorD           : t_iord;
60      signal MemRead        : std_logic;
61      signal MemWrite       : std_logic;
62      signal MemToReg       : t_memToReg;
63      signal IRWrite        : std_logic;
64      signal RegWrite       : std_logic;
65      signal RegDst         : t_regDst;
66      signal ALUSrcA        : t_aluSrcA;
67      signal ALUSrcB        : t_aluSrcB;
68      signal PCSource       : t_pcSrc;
69      signal ALUOp          : t_aluOp;
70      signal UndefInstrEx   : std_logic;
71      signal OverflowEx     : std_logic;
72      signal Exception      : std_logic;
73  --Memory
74      signal mem_data_out    : std_logic_vector((DATA_WIDTH-1) downto 0);
75      signal mem_read        : std_logic;
76      signal addr_bus        : std_logic_vector((DATA_WIDTH-1) downto 0);
77      signal mem_write_data  : std_logic_vector((DATA_WIDTH-1) downto 0);
78
79  begin
80    --Create a clock.
81      PROCESS
82      BEGIN
83          Clk <= '0';
84          WAIT FOR T/2;
85          Clk <= '1';
86          WAIT FOR T/2;
87      END PROCESS;
88
89      DUT: entity work.datapath
90      port map(
91                  clk             => clk,
92                  PCWriteCondEq   => PCWriteCondEq,
93                  PCWriteCondNEq  => PCWriteCondNEq,
94                  PCWrite         => PCWrite,
95                  IorD            => IorD,
96                  MemRead         => MemRead,
97                  MemWrite        => MemWrite,
98                  MemToReg        => MemToReg,
99                  IRWrite         => IRWrite,
100                 RegWrite        => RegWrite,
101                 RegDst          => RegDst,
102                 ALUSrcA         => ALUSrcA,
103                 ALUSrcB         => ALUSrcB,
```

```
104                 PCSource        => PCSource,
105                 ALUOp           => ALUOp,
106                 UndefInstrEx    => UndefInstrEx,
107                 OverflowEx      => OverflowEx,
108                 Exception       => Exception,
109
110                 mem_data_out    => mem_data_out,
111                 mem_read        => mem_read,
112                 addr_bus        => addr_bus,
113                 mem_write_data  => mem_write_data
114             );
115
116     PROCESS
117     BEGIN
118         PCWriteCondEq <= '0';
119         PCWriteCondNEq <= '0';
120         PCWrite <= '0';
121         IorD <= IOD_PC;
122         MemRead <= '0';
123         MemWrite <= '0';
124         MemToReg <= MTR_ALUOUT;
125         IRWrite <= '0';
126         RegWrite <= '0';
127         RegDst <= RD_RT;
128         ALUSrcA <= ASA_PC;
129         ALUSrcB <= ASB_REGB;
130         PCSource <= PS_PCINC;
131         ALUOp <= AOP_AND;
132
133         -- BEQ R1, R0, 8
134         mem_data_out <= "000011" & "00000000000000000000001000";
135         wait for T;
136
137         -- IF
138         MemRead <= '1';
139         ALUSrcA <= ASA_PC;
140         IorD <= IOD_PC;
141         IRWrite <= '1';
142         ALUSrcB <= ASB_FOUR;
143         ALUOp <= AOP_ADD;
144         PCWrite <= '0';
145         PCSource <= PS_PCINC;
146         wait for T/2;
147         PCWrite <= '1';
148         wait for T/2;
149
150         -- ID/Bra Calc
151         PCWrite <= '0';
152         MemRead <= '0';
153         IRWrite <= '0';
154         ALUSrcA <= ASA_PC;
155         ALUSrcB <= ASB_SEXTS;
156         PCSource <= PS_PCINC;
157         ALUOp <= AOP_ADD;
158         wait for T;
159
160         -- Branch Completion
161         MemToReg <= MTR_PC;
```

```
162             RegDst <= RD_RA;
163             RegWrite <= '0';
164             PCWrite <= '0';
165             PCSource <= PS_JMP;
166             wait for T/2;
167             PCWrite <= '1';
168             RegWrite <= '1';
169             wait for T/2;
170
171         END PROCESS;
172
173     end architecture JAL_Testbench_1;
```

Listing 12: JAL_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : J_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity J_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture J_Testbench_1 of J_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq   : in std_logic;
31             PCWriteCondNEq  : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
35             MemWrite        : in std_logic;
36             MemToReg        : in t_memToReg;
37             IRWrite         : in std_logic;
38             RegWrite        : in std_logic;
39             RegDst          : in t_regDst;
40             ALUSrcA         : in t_aluSrcA;
41             ALUSrcB         : in t_aluSrcB;
42             PCSource        : in t_pcSrc;
43             ALUOp           : in t_aluOp;
44             UndefInstrEx    : in std_logic;
```

```
45              OverflowEx     : out std_logic;
46              Exception      : in std_logic;
47      --Memory
48              mem_data_out   : in  std_logic_vector((DATA_WIDTH-1) downto 0);
49              mem_read       : out std_logic;
50              addr_bus       : out std_logic_vector((DATA_WIDTH-1) downto 0);
51              mem_write_data : out std_logic_vector((DATA_WIDTH-1) downto 0)
52          );
53      end component datapath;
54
55      signal PCWriteCondEq : std_logic;
56      signal PCWriteCondNEq : std_logic;
57      signal PCWrite       : std_logic;
58      signal IorD          : t_iord;
59      signal MemRead       : std_logic;
60      signal MemWrite      : std_logic;
61      signal MemToReg      : t_memToReg;
62      signal IRWrite       : std_logic;
63      signal RegWrite      : std_logic;
64      signal RegDst        : t_regDst;
65      signal ALUSrcA       : t_aluSrcA;
66      signal ALUSrcB       : t_aluSrcB;
67      signal PCSource      : t_pcSrc;
68      signal ALUOp         : t_aluOp;
69      signal UndefInstrEx  : std_logic;
70      signal OverflowEx    : std_logic;
71      signal Exception     : std_logic;
72  --Memory
73      signal mem_data_out   : std_logic_vector((DATA_WIDTH-1) downto 0);
74      signal mem_read       : std_logic;
75      signal addr_bus       : std_logic_vector((DATA_WIDTH-1) downto 0);
76      signal mem_write_data : std_logic_vector((DATA_WIDTH-1) downto 0);
77
78  begin
79    --Create a clock.
80      PROCESS
81      BEGIN
82          Clk <= '0';
83          WAIT FOR T/2;
84          Clk <= '1';
85          WAIT FOR T/2;
86      END PROCESS;
87
88      DUT: entity work.datapath
89      port map(
90                  clk            => clk,
91                  PCWriteCondEq  => PCWriteCondEq,
92                  PCWriteCondNEq => PCWriteCondNEq,
93                  PCWrite        => PCWrite,
94                  IorD           => IorD,
95                  MemRead        => MemRead,
96                  MemWrite       => MemWrite,
97                  MemToReg       => MemToReg,
98                  IRWrite        => IRWrite,
99                  RegWrite       => RegWrite,
100                 RegDst         => RegDst,
101                 ALUSrcA        => ALUSrcA,
102                 ALUSrcB        => ALUSrcB,
```

```
103                 PCSource        => PCSource,
104                 ALUOp           => ALUOp,
105                 UndefInstrEx    => UndefInstrEx,
106                 OverflowEx      => OverflowEx,
107                 Exception       => Exception,
108
109                 mem_data_out    => mem_data_out,
110                 mem_read        => mem_read,
111                 addr_bus        => addr_bus,
112                 mem_write_data  => mem_write_data
113             );
114
115     PROCESS
116     BEGIN
117         PCWriteCondEq <= '0';
118         PCWriteCondNEq <= '0';
119         PCWrite <= '0';
120         IorD <= IOD_PC;
121         MemRead <= '0';
122         MemWrite <= '0';
123         MemToReg <= MTR_ALUOUT;
124         IRWrite <= '0';
125         RegWrite <= '0';
126         RegDst <= RD_RT;
127         ALUSrcA <= ASA_PC;
128         ALUSrcB <= ASB_REGB;
129         PCSource <= PS_PCINC;
130         ALUOp <= AOP_AND;
131
132         -- BEQ R1, R0, 8
133         mem_data_out <= "000010" & "00000000000000000000001000";
134         wait for T;
135
136         -- IF
137         MemRead <= '1';
138         ALUSrcA <= ASA_PC;
139         IorD <= IOD_PC;
140         IRWrite <= '1';
141         ALUSrcB <= ASB_FOUR;
142         ALUOp <= AOP_ADD;
143         PCWrite <= '0';
144         PCSource <= PS_PCINC;
145         wait for T/2;
146         PCWrite <= '1';
147         wait for T/2;
148
149         -- ID/Bra Calc
150         PCWrite <= '0';
151         MemRead <= '0';
152         IRWrite <= '0';
153         ALUSrcA <= ASA_PC;
154         ALUSrcB <= ASB_SEXTS;
155         PCSource <= PS_PCINC;
156         ALUOp <= AOP_ADD;
157         wait for T;
158
159         -- Branch Completion
160         PCWrite <= '0';
```

```
161             PCSource <= PS_JMP;
162             wait for T/2;
163             PCWrite <= '1';
164             wait for T/2;
165
166         END PROCESS;
167
168     end architecture J_Testbench_1;
```

Listing 13: J_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : LW_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity LW_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture LW_Testbench_1 of LW_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq : in std_logic;
31             PCWriteCondNEq : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
35             MemWrite        : in std_logic;
36             MemToReg        : in t_memToReg;
37             IRWrite         : in std_logic;
38             RegWrite        : in std_logic;
39             RegDst          : in t_regDst;
40             ALUSrcA         : in t_aluSrcA;
41             ALUSrcB         : in t_aluSrcB;
42             PCSource        : in t_pcSrc;
43             ALUOp           : in t_aluOp;
44             UndefInstrEx    : in std_logic;
45             OverflowEx      : out std_logic;
46             Exception       : in std_logic;
47     --Memory
48             mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
```

```
49              mem_read        : out std_logic;
50              addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
51              mem_write_data  : out std_logic_vector((DATA_WIDTH-1) downto 0)
52          );
53      end component datapath;
54
55      signal PCWriteCondEq : std_logic;
56      signal PCWriteCondNEq : std_logic;
57      signal PCWrite       : std_logic;
58      signal IorD          : t_iord;
59      signal MemRead       : std_logic;
60      signal MemWrite      : std_logic;
61      signal MemToReg      : t_memToReg;
62      signal IRWrite       : std_logic;
63      signal RegWrite      : std_logic;
64      signal RegDst        : t_regDst;
65      signal ALUSrcA       : t_aluSrcA;
66      signal ALUSrcB       : t_aluSrcB;
67      signal PCSource      : t_pcSrc;
68      signal ALUOp         : t_aluOp;
69      signal UndefInstrEx  : std_logic;
70      signal OverflowEx    : std_logic;
71      signal Exception     : std_logic;
72  --Memory
73      signal mem_data_out   : std_logic_vector((DATA_WIDTH-1) downto 0);
74      signal mem_read       : std_logic;
75      signal addr_bus       : std_logic_vector((DATA_WIDTH-1) downto 0);
76      signal mem_write_data : std_logic_vector((DATA_WIDTH-1) downto 0);
77
78  begin
79    --Create a clock.
80      PROCESS
81      BEGIN
82          Clk <= '0';
83          WAIT FOR T/2;
84          Clk <= '1';
85          WAIT FOR T/2;
86      END PROCESS;
87
88      DUT: entity work.datapath
89      port map(
90                  clk           => clk,
91                  PCWriteCondEq => PCWriteCondEq,
92                  PCWriteCondNEq => PCWriteCondNEq,
93                  PCWrite       => PCWrite,
94                  IorD          => IorD,
95                  MemRead       => MemRead,
96                  MemWrite      => MemWrite,
97                  MemToReg      => MemToReg,
98                  IRWrite       => IRWrite,
99                  RegWrite      => RegWrite,
100                 RegDst        => RegDst,
101                 ALUSrcA       => ALUSrcA,
102                 ALUSrcB       => ALUSrcB,
103                 PCSource      => PCSource,
104                 ALUOp         => ALUOp,
105                 UndefInstrEx  => UndefInstrEx,
106                 OverflowEx    => OverflowEx,
```

```
107                 Exception      => Exception,
108
109                 mem_data_out   => mem_data_out,
110                 mem_read       => mem_read,
111                 addr_bus       => addr_bus,
112                 mem_write_data => mem_write_data
113             );
114
115     PROCESS
116     BEGIN
117         PCWriteCondEq <= '0';
118         PCWriteCondNEq <= '0';
119         PCWrite <= '0';
120         IorD <= IOD_PC;
121         MemRead <= '0';
122         MemWrite <= '0';
123         MemToReg <= MTR_ALUOUT;
124         IRWrite <= '0';
125         RegWrite <= '0';
126         RegDst <= RD_RT;
127         ALUSrcA <= ASA_PC;
128         ALUSrcB <= ASB_REGB;
129         PCSource <= PS_PCINC;
130         ALUOp <= AOP_AND;
131
132         -- R[1] <= R0 + 4
133         mem_data_out <= "100011" & "00000" & "00001" & x"00FF";
134         wait for T;
135
136         -- IF
137         MemRead <= '1';
138         ALUSrcA <= ASA_PC;
139         IorD <= IOD_PC;
140         IRWrite <= '1';
141         ALUSrcB <= ASB_FOUR;
142         ALUOp <= AOP_ADD;
143         PCWrite <= '0';
144         PCSource <= PS_PCINC;
145         wait for T/2;
146         PCWrite <= '1';
147         wait for T/2;
148         -- ID/Bra Calc
149         PCWrite <= '0';
150         MemRead <= '0';
151         IRWrite <= '0';
152         ALUSrcA <= ASA_PC;
153         ALUSrcB <= ASB_SEXTS;
154         PCSource <= PS_PCINC;
155         ALUOp <= AOP_ADD;
156         wait for T;
157
158         -- Memory Address Comp
159         ALUSrcA <= ASA_REG_A;
160         ALUSrcB <= ASB_SEXT;
161         ALUOp <= AOP_ADD;
162         wait for T;
163
164         -- Memory Access
```

```
165             IorD <= IOD_ALUOUT;
166             MemRead <= '1';
167             wait for T/2;
168             MemRead <= '0';
169             wait for T/2;
170
171             -- Write Back
172             MemToReg <= MTR_MDR;
173             RegDst <= RD_RT;
174             wait for T/2;
175             RegWrite <= '1';
176             wait for T/2;
177
178
179         END PROCESS;
180
181     end architecture LW_Testbench_1;
```

Listing 14: LW_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : OR_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity OR_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture OR_Testbench_1 of OR_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq : in std_logic;
31             PCWriteCondNEq : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
35             MemWrite        : in std_logic;
36             MemToReg        : in t_memToReg;
37             IRWrite         : in std_logic;
38             RegWrite        : in std_logic;
39             RegDst          : in t_regDst;
```

```
40              ALUSrcA        : in t_aluSrcA;
41              ALUSrcB        : in t_aluSrcB;
42              PCSource       : in t_pcSrc;
43              ALUOp          : in t_aluOp;
44              UndefInstrEx   : in std_logic;
45              OverflowEx     : out std_logic;
46              Exception      : in std_logic;
47      --Memory
48              mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
49              mem_read        : out std_logic;
50              addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
51              mem_write_data : out std_logic_vector((DATA_WIDTH-1) downto 0)
52          );
53      end component datapath;
54
55      signal PCWriteCondEq : std_logic;
56      signal PCWriteCondNEq : std_logic;
57      signal PCWrite        : std_logic;
58      signal IorD           : t_iord;
59      signal MemRead        : std_logic;
60      signal MemWrite       : std_logic;
61      signal MemToReg       : t_memToReg;
62      signal IRWrite        : std_logic;
63      signal RegWrite       : std_logic;
64      signal RegDst         : t_regDst;
65      signal ALUSrcA        : t_aluSrcA;
66      signal ALUSrcB        : t_aluSrcB;
67      signal PCSource       : t_pcSrc;
68      signal ALUOp          : t_aluOp;
69
70      signal UndefInstrEx   : std_logic;
71      signal OverflowEx     : std_logic;
72      signal Exception      : std_logic;
73  --Memory
74      signal mem_data_out    : std_logic_vector((DATA_WIDTH-1) downto 0);
75      signal mem_read        : std_logic;
76      signal addr_bus        : std_logic_vector((DATA_WIDTH-1) downto 0);
77      signal mem_write_data : std_logic_vector((DATA_WIDTH-1) downto 0);
78
79  begin
80    --Create a clock.
81      PROCESS
82      BEGIN
83          Clk <= '0';
84          WAIT FOR T/2;
85          Clk <= '1';
86          WAIT FOR T/2;
87      END PROCESS;
88
89      DUT: entity work.datapath
90      port map(
91                  clk            => clk,
92                  PCWriteCondEq  => PCWriteCondEq,
93                  PCWriteCondNEq => PCWriteCondNEq,
94                  PCWrite        => PCWrite,
95                  IorD           => IorD,
96                  MemRead        => MemRead,
97                  MemWrite       => MemWrite,
```

```
98                  MemToReg        => MemToReg,
99                  IRWrite         => IRWrite,
100                 RegWrite        => RegWrite,
101                 RegDst          => RegDst,
102                 ALUSrcA         => ALUSrcA,
103                 ALUSrcB         => ALUSrcB,
104                 PCSource        => PCSource,
105                 ALUOp           => ALUOp,
106                 UndefInstrEx    => UndefInstrEx,
107                 OverflowEx      => OverflowEx,
108                 Exception       => Exception,
109
110                 mem_data_out    => mem_data_out,
111                 mem_read        => mem_read,
112                 addr_bus        => addr_bus,
113                 mem_write_data  => mem_write_data
114             );
115
116     PROCESS
117     BEGIN
118         PCWriteCondEq <= '0';
119         PCWrite <= '0';
120         IorD <= IOD_PC;
121         MemRead <= '0';
122         MemWrite <= '0';
123         MemToReg <= MTR_ALUOUT;
124         IRWrite <= '0';
125         RegWrite <= '0';
126         RegDst <= RD_RT;
127         ALUSrcA <= ASA_PC;
128         ALUSrcB <= ASB_REGB;
129         PCSource <= PS_PCINC;
130         ALUOp <= AOP_AND;
131
132         -- R[2] <= R[2] | R[1]
133         mem_data_out <= "000000" & "00010" & "00001" & "00010" & "00000" &
                "100101";
134         wait for T;
135
136         -- IF
137         MemRead <= '1';
138         ALUSrcA <= ASA_PC;
139         IorD <= IOD_PC;
140         IRWrite <= '1';
141         ALUSrcB <= ASB_FOUR;
142         ALUOp <= AOP_ADD;
143         PCWrite <= '0';
144         PCSource <= PS_PCINC;
145         wait for T/2;
146         PCWrite <= '1';
147         wait for T/2;
148
149         -- ID/Bra Calc
150         PCWrite <= '0';
151         MemRead <= '0';
152         IRWrite <= '0';
153         ALUSrcA <= ASA_PC;
154         ALUSrcB <= ASB_SEXTS;
```

```
155             PCSource <= PS_PCINC;
156             ALUOp <= AOP_ADD;
157             wait for T;
158
159             -- Execute
160             ALUSrcA <= ASA_REG_A;
161             ALUSrcB <= ASB_REGB;
162             ALUOp <= AOP_OR;
163             wait for T;
164
165
166             --Write Back
167             MemToReg <= MTR_ALUOUT;
168             RegDst <= RD_RD;
169             wait for T/2;
170             RegWrite <= '1';
171             wait for T/2;
172
173
174         END PROCESS;
175
176     end architecture OR_Testbench_1;
```

Listing 15: OR_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : SLL_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity SLL_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture SLL_Testbench_1 of SLL_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq : in std_logic;
31             PCWriteCondNEq : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
```

```
35              MemWrite       : in std_logic;
36              MemToReg       : in t_memToReg;
37              IRWrite        : in std_logic;
38              RegWrite       : in std_logic;
39              RegDst         : in t_regDst;
40              ALUSrcA        : in t_aluSrcA;
41              ALUSrcB        : in t_aluSrcB;
42              PCSource       : in t_pcSrc;
43              ALUOp          : in t_aluOp;
44              UndefInstrEx   : in std_logic;
45              OverflowEx     : out std_logic;
46              Exception      : in std_logic;
47      --Memory
48              mem_data_out   : in  std_logic_vector((DATA_WIDTH-1) downto 0);
49              mem_read       : out std_logic;
50              addr_bus       : out std_logic_vector((DATA_WIDTH-1) downto 0);
51              mem_write_data : out std_logic_vector((DATA_WIDTH-1) downto 0)
52          );
53      end component datapath;
54
55      signal PCWriteCondEq : std_logic;
56      signal PCWriteCondNEq : std_logic;
57      signal PCWrite       : std_logic;
58      signal IorD          : t_iord;
59      signal MemRead       : std_logic;
60      signal MemWrite      : std_logic;
61      signal MemToReg      : t_memToReg;
62      signal IRWrite       : std_logic;
63      signal RegWrite      : std_logic;
64      signal RegDst        : t_regDst;
65      signal ALUSrcA       : t_aluSrcA;
66      signal ALUSrcB       : t_aluSrcB;
67      signal PCSource      : t_pcSrc;
68      signal ALUOp         : t_aluOp;
69      signal UndefInstrEx  : std_logic;
70      signal OverflowEx    : std_logic;
71      signal Exception     : std_logic;
72  --Memory
73      signal mem_data_out   : std_logic_vector((DATA_WIDTH-1) downto 0);
74      signal mem_read       : std_logic;
75      signal addr_bus       : std_logic_vector((DATA_WIDTH-1) downto 0);
76      signal mem_write_data : std_logic_vector((DATA_WIDTH-1) downto 0);
77
78  begin
79    --Create a clock.
80      PROCESS
81      BEGIN
82          Clk <= '0';
83          WAIT FOR T/2;
84          Clk <= '1';
85          WAIT FOR T/2;
86      END PROCESS;
87
88      DUT: entity work.datapath
89      port map(
90                  clk           => clk,
91                  PCWriteCondEq => PCWriteCondEq,
92                  PCWriteCondNEq => PCWriteCondNEq,
```

```
93                  PCWrite         => PCWrite,
94                  IorD            => IorD,
95                  MemRead         => MemRead,
96                  MemWrite        => MemWrite,
97                  MemToReg        => MemToReg,
98                  IRWrite         => IRWrite,
99                  RegWrite        => RegWrite,
100                 RegDst          => RegDst,
101                 ALUSrcA         => ALUSrcA,
102                 ALUSrcB         => ALUSrcB,
103                 PCSource        => PCSource,
104                 ALUOp           => ALUOp,
105                 UndefInstrEx    => UndefInstrEx,
106                 OverflowEx      => OverflowEx,
107                 Exception       => Exception,
108
109                 mem_data_out    => mem_data_out,
110                 mem_read        => mem_read,
111                 addr_bus        => addr_bus,
112                 mem_write_data  => mem_write_data
113             );
114
115     PROCESS
116     BEGIN
117         PCWriteCondEq <= '0';
118         PCWriteCondNEq <= '0';
119         PCWrite <= '0';
120         IorD <= IOD_PC;
121         MemRead <= '0';
122         MemWrite <= '0';
123         MemToReg <= MTR_ALUOUT;
124         IRWrite <= '0';
125         RegWrite <= '0';
126         RegDst <= RD_RT;
127         ALUSrcA <= ASA_PC;
128         ALUSrcB <= ASB_REGB;
129         PCSource <= PS_PCINC;
130         ALUOp <= AOP_AND;
131
132         -- R[2] <= R[1] << 2
133         mem_data_out <= "000000" & "00000" & "00001" & "00010" & "00010" &
                "000000";
134         wait for T;
135
136         -- IF
137         MemRead <= '1';
138         ALUSrcA <= ASA_PC;
139         IorD <= IOD_PC;
140         IRWrite <= '1';
141         ALUSrcB <= ASB_FOUR;
142         ALUOp <= AOP_ADD;
143         PCWrite <= '0';
144         PCSource <= PS_PCINC;
145         wait for T/2;
146         PCWrite <= '1';
147         wait for T/2;
148
149         -- ID/Bra Calc
```

```
150             PCWrite <= '0';
151             MemRead <= '0';
152             IRWrite <= '0';
153             ALUSrcA <= ASA_PC;
154             ALUSrcB <= ASB_SEXTS;
155             PCSource <= PS_PCINC;
156             ALUOp <= AOP_ADD;
157             wait for T;
158
159             -- Execute
160             ALUSrcA <= ASA_REG_A;
161             ALUSrcB <= ASB_REGB;
162             ALUOp <= AOP_SLL;
163             wait for T;
164
165
166             --Write Back
167             MemToReg <= MTR_ALUOUT;
168             RegDst <= RD_RD;
169             wait for T/2;
170             RegWrite <= '1';
171             wait for T/2;
172
173
174         END PROCESS;
175
176     end architecture SLL_Testbench_1;
```

Listing 16: SLL_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File           : SLT_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity SLT_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
20
21 ----------------------------------------------------------------------------
22 Architecture SLT_Testbench_1 of SLT_Testbench is
23 ----------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
```

```
30              PCWriteCondEq : in std_logic;
31              PCWriteCondNEq : in std_logic;
32              PCWrite       : in std_logic;
33              IorD          : in t_iord;
34              MemRead       : in std_logic;
35              MemWrite      : in std_logic;
36              MemToReg      : in t_memToReg;
37              IRWrite       : in std_logic;
38              RegWrite      : in std_logic;
39              RegDst        : in t_regDst;
40              ALUSrcA       : in t_aluSrcA;
41              ALUSrcB       : in t_aluSrcB;
42              PCSource      : in t_pcSrc;
43              ALUOp         : in t_aluOp;
44
45              UndefInstrEx  : in std_logic;
46              OverflowEx    : out std_logic;
47              Exception     : in std_logic;
48      --Memory
49              mem_data_out   : in  std_logic_vector((DATA_WIDTH-1) downto 0);
50              mem_read       : out std_logic;
51              addr_bus       : out std_logic_vector((DATA_WIDTH-1) downto 0);
52              mem_write_data : out std_logic_vector((DATA_WIDTH-1) downto 0)
53          );
54      end component datapath;
55
56      signal PCWriteCondEq : std_logic;
57      signal PCWriteCondNEq : std_logic;
58      signal PCWrite       : std_logic;
59      signal IorD          : t_iord;
60      signal MemRead       : std_logic;
61      signal MemWrite      : std_logic;
62      signal MemToReg      : t_memToReg;
63      signal IRWrite       : std_logic;
64      signal RegWrite      : std_logic;
65      signal RegDst        : t_regDst;
66      signal ALUSrcA       : t_aluSrcA;
67      signal ALUSrcB       : t_aluSrcB;
68      signal PCSource      : t_pcSrc;
69      signal ALUOp         : t_aluOp;
70
71      signal UndefInstrEx  : std_logic;
72      signal OverflowEx    : std_logic;
73      signal Exception     : std_logic;
74  --Memory
75      signal mem_data_out   : std_logic_vector((DATA_WIDTH-1) downto 0);
76      signal mem_read       : std_logic;
77      signal addr_bus       : std_logic_vector((DATA_WIDTH-1) downto 0);
78      signal mem_write_data : std_logic_vector((DATA_WIDTH-1) downto 0);
79
80  begin
81    --Create a clock.
82      PROCESS
83      BEGIN
84          Clk <= '0';
85          WAIT FOR T/2;
86          Clk <= '1';
87          WAIT FOR T/2;
```

```
88      END PROCESS;
89
90      DUT: entity work.datapath
91      port map(
92                  clk             => clk,
93                  PCWriteCondEq   => PCWriteCondEq,
94                  PCWriteCondNEq  => PCWriteCondNEq,
95                  PCWrite         => PCWrite,
96                  IorD            => IorD,
97                  MemRead         => MemRead,
98                  MemWrite        => MemWrite,
99                  MemToReg        => MemToReg,
100                 IRWrite         => IRWrite,
101                 RegWrite        => RegWrite,
102                 RegDst          => RegDst,
103                 ALUSrcA         => ALUSrcA,
104                 ALUSrcB         => ALUSrcB,
105                 PCSource        => PCSource,
106                 ALUOp           => ALUOp,
107                 UndefInstrEx    => UndefInstrEx,
108                 OverflowEx      => OverflowEx,
109                 Exception       => Exception,
110
111                 mem_data_out    => mem_data_out,
112                 mem_read        => mem_read,
113                 addr_bus        => addr_bus,
114                 mem_write_data  => mem_write_data
115             );
116
117     PROCESS
118     BEGIN
119         PCWriteCondEq <= '0';
120         PCWrite <= '0';
121         IorD <= IOD_PC;
122         MemRead <= '0';
123         MemWrite <= '0';
124         MemToReg <= MTR_ALUOUT;
125         IRWrite <= '0';
126         RegWrite <= '0';
127         RegDst <= RD_RT;
128         ALUSrcA <= ASA_PC;
129         ALUSrcB <= ASB_REGB;
130         PCSource <= PS_PCINC;
131         ALUOp <= AOP_AND;
132
133         -- R[2] <= R[2] & R[1]
134         mem_data_out <= "000000" & "00010" & "00001" & "00010" & "00000" &
                "101010";
135         wait for T;
136
137         -- IF
138         MemRead <= '1';
139         ALUSrcA <= ASA_PC;
140         IorD <= IOD_PC;
141         IRWrite <= '1';
142         ALUSrcB <= ASB_FOUR;
143         ALUOp <= AOP_ADD;
144         PCWrite <= '0';
```

```
145             PCSource <= PS_PCINC;
146             wait for T/2;
147             PCWrite <= '1';
148             wait for T/2;
149
150             -- ID/Bra Calc
151             PCWrite <= '0';
152             MemRead <= '0';
153             IRWrite <= '0';
154             ALUSrcA <= ASA_PC;
155             ALUSrcB <= ASB_SEXTS;
156             PCSource <= PS_PCINC;
157             ALUOp <= AOP_ADD;
158             wait for T;
159
160             -- Execute
161             ALUSrcA <= ASA_REG_A;
162             ALUSrcB <= ASB_REGB;
163             ALUOp <= AOP_SLT;
164             wait for T;
165
166
167             --Write Back
168             MemToReg <= MTR_ALUOUT;
169             RegDst <= RD_RD;
170             wait for T/2;
171             RegWrite <= '1';
172             wait for T/2;
173
174
175     END PROCESS;
176
177 end architecture SLT_Testbench_1;
```

Listing 17: SLT_Testbench

```
1  ------------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : SUB_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ------------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ------------------------------------------------------------------------------
16 Entity SUB_Testbench is
17 ------------------------------------------------------------------------------
18 end entity;
19
20
21 ------------------------------------------------------------------------------
22 Architecture SUB_Testbench_1 of SUB_Testbench is
23 ------------------------------------------------------------------------------
```

```
24      constant T: time := 100 ns;
25      signal Clk : std_logic := '0';
26      component datapath
27              port(
28              clk             : in std_logic;
29      -- Control Unit
30              PCWriteCondEq : in std_logic;
31              PCWriteCondNEq : in std_logic;
32              PCWrite         : in std_logic;
33              IorD            : in t_iord;
34              MemRead         : in std_logic;
35              MemWrite        : in std_logic;
36              MemToReg        : in t_memToReg;
37              IRWrite         : in std_logic;
38              RegWrite        : in std_logic;
39              RegDst          : in t_regDst;
40              ALUSrcA         : in t_aluSrcA;
41              ALUSrcB         : in t_aluSrcB;
42              PCSource        : in t_pcSrc;
43              ALUOp           : in t_aluOp;
44              UndefInstrEx    : in std_logic;
45              OverflowEx      : out std_logic;
46              Exception       : in std_logic;
47      --Memory
48              mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
49              mem_read        : out std_logic;
50              addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
51              mem_write_data : out std_logic_vector((DATA_WIDTH-1) downto 0)
52          );
53      end component datapath;
54
55      signal PCWriteCondEq : std_logic;
56      signal PCWriteCondNEq : std_logic;
57      signal PCWrite         : std_logic;
58      signal IorD            : t_iord;
59      signal MemRead         : std_logic;
60      signal MemWrite        : std_logic;
61      signal MemToReg        : t_memToReg;
62      signal IRWrite         : std_logic;
63      signal RegWrite        : std_logic;
64      signal RegDst          : t_regDst;
65      signal ALUSrcA         : t_aluSrcA;
66      signal ALUSrcB         : t_aluSrcB;
67      signal PCSource        : t_pcSrc;
68      signal ALUOp           : t_aluOp;
69      signal UndefInstrEx    : std_logic;
70      signal OverflowEx      : std_logic;
71      signal Exception       : std_logic;
72  --Memory
73      signal mem_data_out    : std_logic_vector((DATA_WIDTH-1) downto 0);
74      signal mem_read        : std_logic;
75      signal addr_bus        : std_logic_vector((DATA_WIDTH-1) downto 0);
76      signal mem_write_data : std_logic_vector((DATA_WIDTH-1) downto 0);
77
78  begin
79    --Create a clock.
80      PROCESS
81      BEGIN
```

```
82          Clk <= '0';
83          WAIT FOR T/2;
84          Clk <= '1';
85          WAIT FOR T/2;
86      END PROCESS;
87
88      DUT: entity work.datapath
89      port map(
90                  clk             => clk,
91                  PCWriteCondEq => PCWriteCondEq,
92                  PCWriteCondNEq => PCWriteCondNEq,
93                  PCWrite         => PCWrite,
94                  IorD            => IorD,
95                  MemRead         => MemRead,
96                  MemWrite        => MemWrite,
97                  MemToReg        => MemToReg,
98                  IRWrite         => IRWrite,
99                  RegWrite        => RegWrite,
100                 RegDst          => RegDst,
101                 ALUSrcA         => ALUSrcA,
102                 ALUSrcB         => ALUSrcB,
103                 PCSource        => PCSource,
104                 ALUOp           => ALUOp,
105                 UndefInstrEx    => UndefInstrEx,
106                 OverflowEx      => OverflowEx,
107                 Exception       => Exception,
108
109                 mem_data_out    => mem_data_out,
110                 mem_read        => mem_read,
111                 addr_bus        => addr_bus,
112                 mem_write_data => mem_write_data
113             );
114
115     PROCESS
116     BEGIN
117         PCWriteCondEq <= '0';
118         PCWrite <= '0';
119         IorD <= IOD_PC;
120         MemRead <= '0';
121         MemWrite <= '0';
122         MemToReg <= MTR_ALUOUT;
123         IRWrite <= '0';
124         RegWrite <= '0';
125         RegDst <= RD_RT;
126         ALUSrcA <= ASA_PC;
127         ALUSrcB <= ASB_REGB;
128         PCSource <= PS_PCINC;
129         ALUOp <= AOP_AND;
130
131         -- R[2] <= R[0] - R[1]
132         mem_data_out <= "000000" & "00000" & "00001" & "00010" & "00000" &
                "100010";
133         wait for T;
134
135         -- IF
136         MemRead <= '1';
137         ALUSrcA <= ASA_PC;
138         IorD <= IOD_PC;
```

```
139             IRWrite <= '1';
140             ALUSrcB <= ASB_FOUR;
141             ALUOp <= AOP_ADD;
142             PCWrite <= '0';
143             PCSource <= PS_PCINC;
144             wait for T/2;
145             PCWrite <= '1';
146             wait for T/2;
147
148             -- ID/Bra Calc
149             PCWrite <= '0';
150             MemRead <= '0';
151             IRWrite <= '0';
152             ALUSrcA <= ASA_PC;
153             ALUSrcB <= ASB_SEXTS;
154             PCSource <= PS_PCINC;
155             ALUOp <= AOP_ADD;
156             wait for T;
157
158             -- Execute
159             ALUSrcA <= ASA_REG_A;
160             ALUSrcB <= ASB_REGB;
161             ALUOp <= AOP_SUB;
162             wait for T;
163
164
165             --Write Back
166             MemToReg <= MTR_ALUOUT;
167             RegDst <= RD_RD;
168             wait for T/2;
169             RegWrite <= '1';
170             wait for T/2;
171
172
173         END PROCESS;
174
175     end architecture SUB_Testbench_1;
```

Listing 18: SUB_Testbench

```
1  ----------------------------------------------------------------------------
2  -- Author(s)    : Jay Mundrawala <mundra@ir.iit.edu>
3  --
4  -- File         : SW_Testbench.vhdl
5  -- Creation Date : 21/11/2009
6  -- Description:
7  --
8  ----------------------------------------------------------------------------
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.numeric_std.all;
13 use work.mips_lib.all;
14
15 ----------------------------------------------------------------------------
16 Entity SW_Testbench is
17 ----------------------------------------------------------------------------
18 end entity;
19
```

```
20
21 ------------------------------------------------------------------------
22 Architecture SW_Testbench_1 of SW_Testbench is
23 ------------------------------------------------------------------------
24     constant T: time := 100 ns;
25     signal Clk : std_logic := '0';
26     component datapath
27             port(
28             clk             : in std_logic;
29     -- Control Unit
30             PCWriteCondEq   : in std_logic;
31             PCWriteCondNEq  : in std_logic;
32             PCWrite         : in std_logic;
33             IorD            : in t_iord;
34             MemRead         : in std_logic;
35             MemWrite        : in std_logic;
36             MemToReg        : in t_memToReg;
37             IRWrite         : in std_logic;
38             RegWrite        : in std_logic;
39             RegDst          : in t_regDst;
40             ALUSrcA         : in t_aluSrcA;
41             ALUSrcB         : in t_aluSrcB;
42             PCSource        : in t_pcSrc;
43             ALUOp           : in t_aluOp;
44             UndefInstrEx    : in std_logic;
45             OverflowEx      : out std_logic;
46             Exception       : in std_logic;
47
48     --Memory
49             mem_data_out    : in  std_logic_vector((DATA_WIDTH-1) downto 0);
50             mem_read        : out std_logic;
51             addr_bus        : out std_logic_vector((DATA_WIDTH-1) downto 0);
52             mem_write_data  : out std_logic_vector((DATA_WIDTH-1) downto 0)
53         );
54     end component datapath;
55
56     signal PCWriteCondEq  : std_logic;
57     signal PCWriteCondNEq : std_logic;
58     signal PCWrite        : std_logic;
59     signal IorD           : t_iord;
60     signal MemRead        : std_logic;
61     signal MemWrite       : std_logic;
62     signal MemToReg       : t_memToReg;
63     signal IRWrite        : std_logic;
64     signal RegWrite       : std_logic;
65     signal RegDst         : t_regDst;
66     signal ALUSrcA        : t_aluSrcA;
67     signal ALUSrcB        : t_aluSrcB;
68     signal PCSource       : t_pcSrc;
69     signal ALUOp          : t_aluOp;
70
71     signal UndefInstrEx   : std_logic;
72     signal OverflowEx     : std_logic;
73     signal Exception      : std_logic;
74 --Memory
75     signal mem_data_out   : std_logic_vector((DATA_WIDTH-1) downto 0);
76     signal mem_read       : std_logic;
77     signal addr_bus       : std_logic_vector((DATA_WIDTH-1) downto 0);
```

```
78      signal mem_write_data : std_logic_vector((DATA_WIDTH-1) downto 0);
79
80  begin
81      --Create a clock.
82      PROCESS
83      BEGIN
84          Clk <= '0';
85          WAIT FOR T/2;
86          Clk <= '1';
87          WAIT FOR T/2;
88      END PROCESS;
89
90      DUT: entity work.datapath
91      port map(
92                  clk             => clk,
93                  PCWriteCondEq => PCWriteCondEq,
94                  PCWriteCondNEq => PCWriteCondNEq,
95                  PCWrite         => PCWrite,
96                  IorD            => IorD,
97                  MemRead         => MemRead,
98                  MemWrite        => MemWrite,
99                  MemToReg        => MemToReg,
100                 IRWrite         => IRWrite,
101                 RegWrite        => RegWrite,
102                 RegDst          => RegDst,
103                 ALUSrcA         => ALUSrcA,
104                 ALUSrcB         => ALUSrcB,
105                 PCSource        => PCSource,
106                 ALUOp           => ALUOp,
107                 UndefInstrEx    => UndefInstrEx,
108                 OverflowEx      => OverflowEx,
109                 Exception       => Exception,
110
111                 mem_data_out    => mem_data_out,
112                 mem_read        => mem_read,
113                 addr_bus        => addr_bus,
114                 mem_write_data => mem_write_data
115             );
116
117     PROCESS
118     BEGIN
119         PCWriteCondEq <= '0';
120         PCWrite <= '0';
121         IorD <= IOD_PC;
122         MemRead <= '0';
123         MemWrite <= '0';
124         MemToReg <= MTR_ALUOUT;
125         IRWrite <= '0';
126         RegWrite <= '0';
127         RegDst <= RD_RT;
128         ALUSrcA <= ASA_PC;
129         ALUSrcB <= ASB_REGB;
130         PCSource <= PS_PCINC;
131         ALUOp <= AOP_AND;
132
133         -- R[1] <= R0 + 4
134         mem_data_out <= "101011" & "00000" & "00001" & x"00FF";
135         wait for T;
```

```
136
137             -- IF
138             MemRead <= '1';
139             ALUSrcA <= ASA_PC;
140             IorD <= IOD_PC;
141             IRWrite <= '1';
142             ALUSrcB <= ASB_FOUR;
143             ALUOp <= AOP_ADD;
144             PCWrite <= '0';
145             PCSource <= PS_PCINC;
146             wait for T/2;
147             PCWrite <= '1';
148             wait for T/2;
149             -- ID/Bra Calc
150             PCWrite <= '0';
151             MemRead <= '0';
152             IRWrite <= '0';
153             ALUSrcA <= ASA_PC;
154             ALUSrcB <= ASB_SEXTS;
155             PCSource <= PS_PCINC;
156             ALUOp <= AOP_ADD;
157             wait for T;
158
159             -- Memory Address Comp
160             ALUSrcA <= ASA_REG_A;
161             ALUSrcB <= ASB_SEXT;
162             ALUOp <= AOP_ADD;
163             wait for T;
164
165             -- Memory Access
166             IorD <= IOD_ALUOUT;
167             wait for T/2;
168             MemWrite <= '1';
169             wait for T/2;
170
171
172         END PROCESS;
173
174 end architecture SW_Testbench_1;
```

Listing 19: SW_Testbench