Kairi Kozuma
GTID: 903050898
ECE 3056
04/01/2016

Assignment 4 Design and Verification Report

Design: Modifications to Datapath

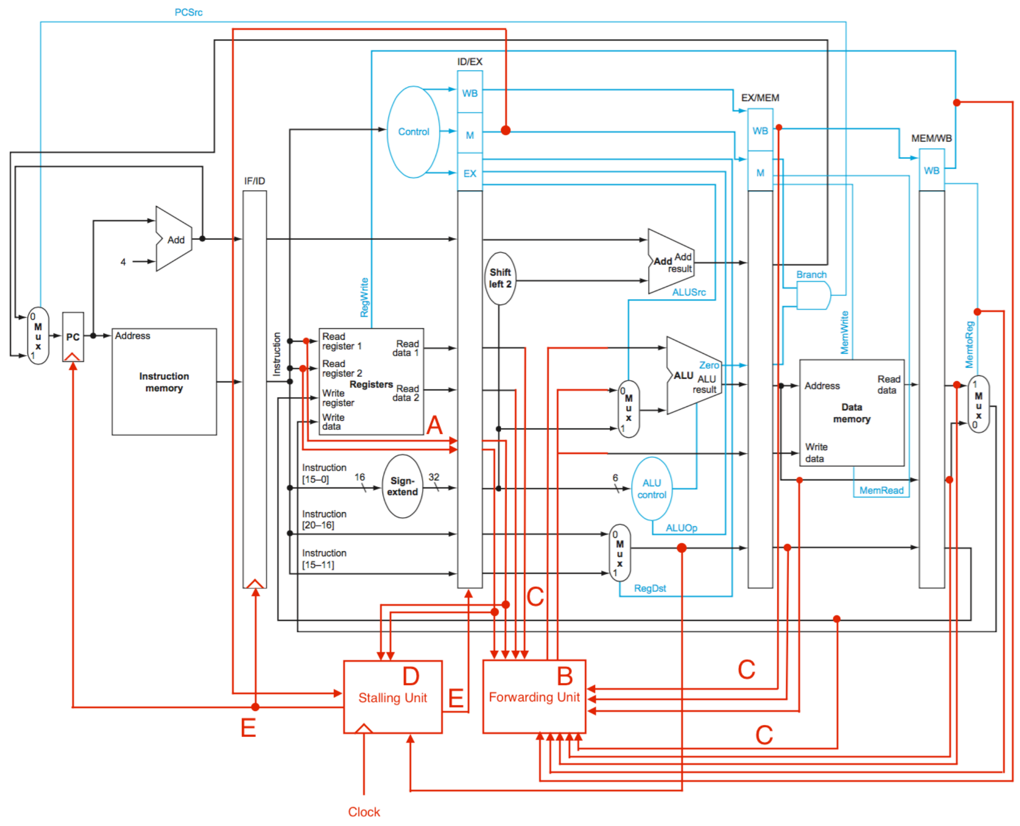

Figure 1. Modified pipeline datapath with forwarding to EX stage and load to use hazard detection and stall insertion.

I. Modified VHDL Modules

A. Add register Rs address and register Rt address in pipe_reg2.vhd, which is the ID/EX pipeline register.

B. Create a forwarding unit module ps_forwarding.vhd that outputs the appropriate Rs and Rt value to the EX stage. Check if forwarding is necessary by using Rs and Rt register addresses from EX, write register address from MEM and WB, regwrite signal from MEM and WB, and memory related signals from WB. Use a multiplexor to output the correct Rs and Rt values to EX.

C. Instantiate the forwarding unit in spim_pipe.vhd and connect necessary signals from the EX, MEM, and WB stages to implement forwarding.

D. Create a stalling unit module ps_stalling.vhd that detects load to use hazards and inserts a stall if necessary. Latch the inputs on falling edge to generate a load_use_stall internal signal half a clock behind, in order to prevent resetting on a pipeline registers on a rising edge. Output the clock_if_id signal, which is a clock signal that is turned off when a load to use stall is detected. Output a stall_or_reset signal, which is asserted when load_use_stall or reset is asserted.

E. Connect the clock_if_id signal to the IF stage and IF/ID pipeline register. Connect the stall_or_reset signal to ID/EX pipeline register.

II. Implementation Steps

Forwarding to EX Stage Implementation

1. Set internal signal forwardA to:
    - "10" if mem_RegWrite is asserted, mem_wreg_addr is not "0" and equal to ex_rs_address. Represents forwarding from MEM stage.
    - "01" if wb_RegWrite is asserted, wb_wreg_addr is not "0" and equal to ex_rs_address. Represents forwarding from WB stage, only if no forwarding from the MEM stage.
    - "00" otherwise, representing no forwarding.
2. Use a multiplexor to choose between the mem_alu_result, wb_alu_result, wb_memory_data, and ex_register_rs (value fetched from register). The forwardA signal is used to choose between the values. Another signal, wb_memory_data, is used to determine which WB stage value to forward.
3. Do steps 1 through 4 for register Rt, with internal signal forwardB.
4. Output the value chosen by multiplexor to use in the EX stage.

Load to Use Hazard Detection and Stall Insertion Implementation

1. Latch id_rs_address, id_rt_address, ex_memread, and ex_wreg_addr on the falling edge.
2. Generate an internal load_use_stall signal, asserted when ex_memread is "1", ex_wreg_addr is not "0", and ex_wreg_addr is equal to id_rs_address or id_rt_address.
3. Output a clock_if_id signal that is deasserted when load_use_stall is asserted, turning off the clock for the IF and ID stage.
4. Output a stall_or_reset signal, which is asserted when load_use_stall or reset signal is asserted. Flushes the ID/EX pipeline register to insert a stall.

III. Screen Trace of Sample Program

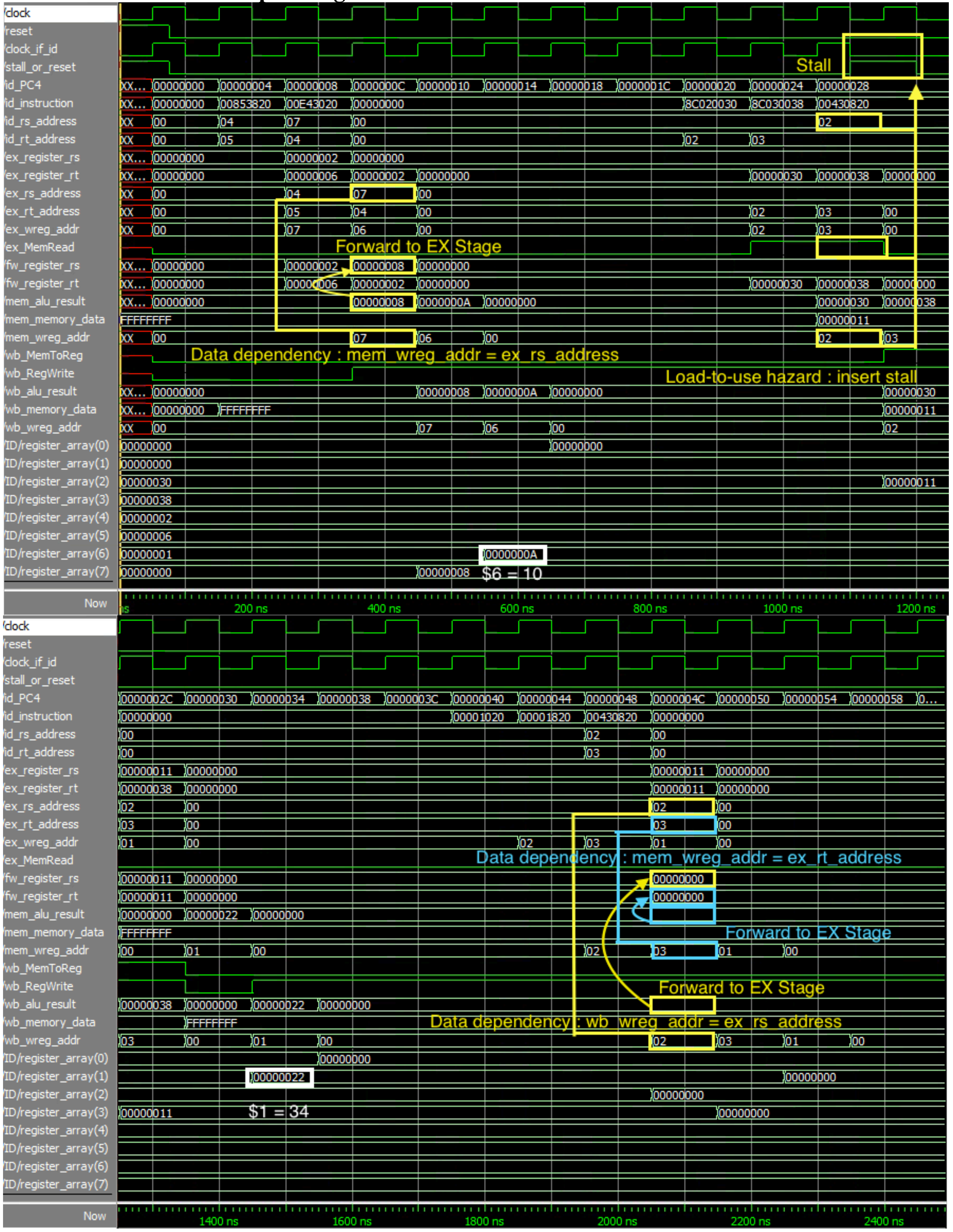
Stall
Forward to EX Stage
Data dependency : mem_wreg_addr = ex_rs_address
Load-to-use hazard : insert stall
$6 = 10
Data dependency : mem_wreg_addr = ex_rt_address
Forward to EX Stage
Forward to EX Stage
Data dependency : wb_wreg_addr = ex_rs_address
$1 = 34

Figure 2. Screen trace of sample program, showing forwarding to EX stage and load to use hazard detection with stall insertion. Expected register values are shown in the white box.

IV. CPI of Sample Program

Unmodified datapath: CPI = 29 cycles / 23 Instructions = 29/23 = 1.2609

```
add $7, $4, $5
nop
nop
add $6, $7, $4
nop
nop
nop
nop
nop
lw $2, 12($0)
lw $3, 14($0)
nop
nop
add $1, $2, $3
nop
nop
nop
nop
nop
add $2, $0, $0
add $3, $0, $0
nop
nop
add $1, $2, $3
nop
nop
nop
nop
nop
```

Modified datapath: CPI = 24 cycles / 23 Instructions = 24/8 = 1.0435. Reduction of .2147 CPI.

```
add $7, $4, $5
add $6, $7, $4
nop
nop
nop
nop
nop
lw $2, 12($0)
lw $3, 14($0)
<stall>
add $1, $2, $3
nop
nop
nop
nop
nop
add $2, $0, $0
add $3, $0, $0
add $1, $2, $3
nop
nop
nop
nop
nop
```