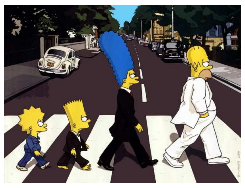
ECE4100/6100 Advanced Computer Architecture
Programming Assignment 3

Assigned: October 1, 2010 **Due 12:35pm October 29 (Fri), 2010**

Score: ____________ Date:___________ Name:___________________________________
TA Signoff: ____



In this assignment, you will implement a simplified version of **Tomasulo Algorithm** using Verilog **"behavioral model"** programming. ModelSim will be used for this project. Similar to previous project, this project must be done individually.

<u>Problem Statement</u>
Design and implement a variation of Tomasulo Algorithm using Verilog behavioral model, to execute the instructions in the data-flow order, rather than sequential order.

The instruction flow
The instruction flow in Tomasulo algorithm is shown in Figure 1. Each instruction on arrival fetches its two operands from the register file that has 16 registers. Each register in this file holds either an actual value, or a tag id indicating the reservation station that will produce the register value when it completes. The instruction and its operands (either values or tag ids) are stored in a reservation station. The reservation station watches the results returning from the execution units, and when the tag of a result matches one of its input operands, it records the value in the place of the tag. When the reservation station has both values of its operands, it may issue its instruction to an execution unit if it is free. When the tagged result returns from the pipeline, the reservation station is cleared, and the result value is stored in the destination register. There is also a stall output indicating that an instruction cannot currently be received. A stall can happen when all the reservation stations are full. In the whole system, the instructions are read from during the positive edge, and written to during the negative edge of the clock cycle.

Instruction Fetch
The instruction stream comes from the instruction fetch unit called GT_IFU. The GT_IFU module that is provided is in the **GT_IFU_PR3.v** Verilog file. At every positive edge of the clock cycle, the GT_IFU sends one instruction address to the dynamic scheduling logic. Each instruction address is 32-bits broken down as follows:

instr_addr [31:28]	-	opcode	(4 bits)
instr_addr [27:24]	-	destination reg	(4 bits)
instr_addr [23:20]	-	source reg 1	(4 bits)
instr_addr [19:16]	-	source reg 2	(4 bits)

The lower order 16 bits of the instruction is ignored.

Opcode 0000 – ADD
ADD R3, R1, R2 => R3 = R1 + R2
Latency – 1 cycle

Opcode 0001 – SUB
ADD R3, R1, R2 => R3 = R1 - R2
Latency – 1 cycle

Opcode 0010 – MULT
ADD R3, R1, R2 => R3 = R1 * R2
Latency – 2 cycles

Opcode 0011 – DIV
ADD R3, R1, R2 => R3 = R1/ R2
Latency – 3 cycles

Opcode 1111 – HLT

Execution units
The execution unit latencies are much lower than the realistic values for this experiment to reduce the overall simulation time. A new instruction can start in the execution unit only after the specified number of latency cycles. In other words, none of the execution unit is pipelined. The GT_IFU can be stalled when all the reservation stations are full and cannot accept any new instructions using the gated clock signal.

When the opcode is 1111, stop fetching the instructions by clock gating the GT_IFU and let the system complete the instructions that are in flight.

Register File
There are 16 registers in the register file. The width of each register is 32-bits. In addition to the 32-bits, each register also contains a bit that states if the value contained in the register is a tag id of the reservation station or the actual data value. A 1 in this bit location indicates the register contains a tag id (lower order 2 bits denoting the reservation station number), and a 0 indicates that the register contains the actual data value. The register values are read during the positive edge of the clock cycle. The results from the execution unit are written to the destination register during the negative edge of the clock cycle.

Initialize all the registers to contain decimal value 5, and also indicate it is an actual data value.

Reservation Stations
There are 4 reservation stations. Each instruction can go to any of the reservation station without any restriction. Each reservation station contains the instruction and its operands (either values or tags). The operands are read from the reservation station and sent to the execution unit during the

positive edge of the clock cycle. The results along with tag id from execution unit are written into the reservation station during the negative edge of the clock cycle.

Broadcasting results and Selection logic

You only need to implement a location-based selection algorithm (based on the reservation station ID) and issue one instruction per cycle from the reservation stations to any of the three functional units. We also assume each FU has its own broadcast bus to broadcast their respective result back to the reservation stations for associative matching. With this, you will avoid conflicts when more than one functional unit intends to use the broadcast bus simultaneously.

Outputs from the design

After every clock cycle the system should update the values in the register file, reservation stations, selection logic, and the execution units appropriately for the given sequence of instructions. When you encounter HLT instruction, let the system run the simulation until all the instructions that are in flight are executed and the results written to the register file. In other words, at the end of the simulation the reservation station and the execution units should be empty. Note that TA will use a different instruction stream to test the functionality of your code.

Design Strategy

You are encouraged to form a study group with your class peers to discuss your design. However, you are not allowed to copy others' codes.

How to receive the credit

Check off by TA is needed. Similar to the previous assignments, you need to prepare a schematic diagram that illustrates a reasonable high-level logic design of the Tomasulo Algorithm to provide TA a better understanding of your design approach.

Honor Code

This assignment must be done **individually**. Due to its complexity, you are encouraged to form a study group with your peers to discuss your design approaches. However, you are **not** allowed to copy others' codes. For those who violate the rule, both the originator and the copier will receive zero credit and will be **immediately** reported to the Dean of Students' Affair for further action.

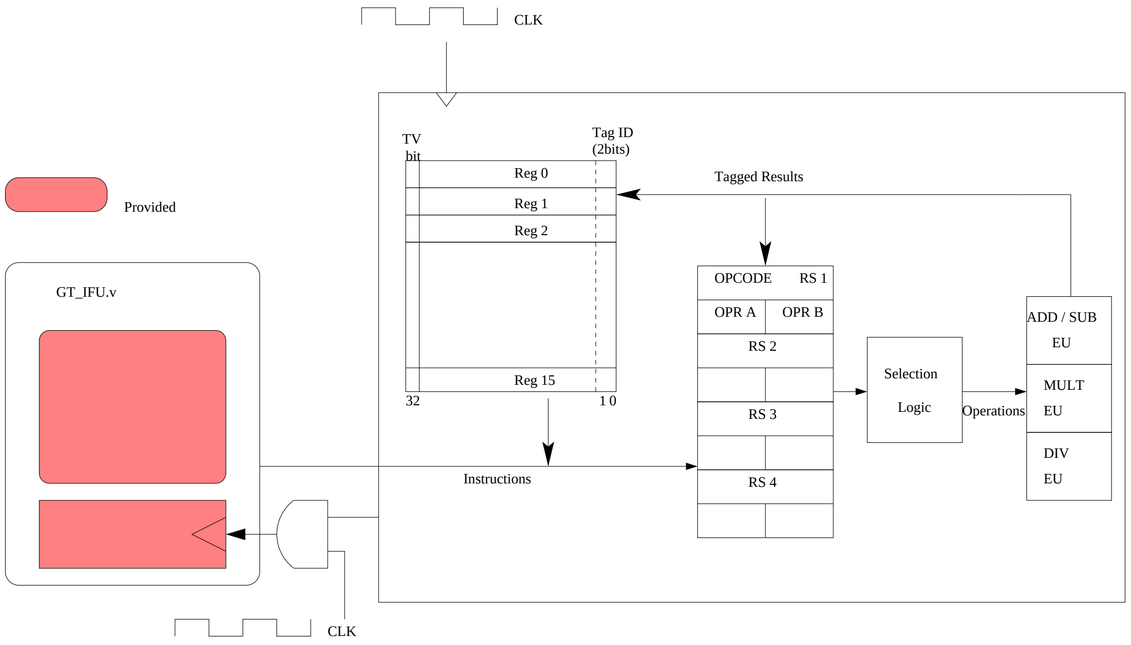


Figure 1: Instruction flow in Tomasulo Algorithm