ECE575 Computer Architecture **Student:** *Mihir Kavishwar*

Lab 4: Out-of-Order RISCV Processor – Reorder Buffer

April 9, 2023

1 Abstract

In this lab I implemented a Reorder Buffer (ROB) for a single issue, three wide, Out-of-Order superscalar processor. We were given the code for a I2O2 processor and had to convert this into an I2OI processor by building a ROB. I also added a test which failed with on the I2O2 processor but passed on the I2OI processor. Finally I evaluated the performance of the processor on the benchmark programs and compared it with the long processor.

2 Design

I designed the ROB as per the table 1 given in the lab handout - we have 16 entries and each entry has a valid bit, a pending bit and a physical register address. Thus three register arrays were defined to implement this table. I defined the head and tail pointers to keep track of the oldest entry ready to commit and newest entry ready to be allocated. I defined an additional signal called rob_empty to indicate whether the ROB is empty. If the tail pointer and the head pointer were ever equal and the ROB was not empty, that indicated that the ROB was full and the rob_alloc_req_rdy is then set to 0 to stall the processor.

3 Testing Methodology

Let MUL pipe be X0, X1, X2, X3; MEM pipe be Y0, Y1; ALU pipe be Z0.

3.1 Test 1

1	li x1, 1	F	D	Y0	Y1	W	c											
2	li x2, 2		F	D	Y0	Y1	W	c										
3	li x4, 3			F	D	Y0	Y1	W	c									
4	li x5, 4				F	D	Y0	Y1	W	c								
5	mul x3, x1, x2					F	D	X0	X1	X2	X3	W	c					
6	add x3, x4, x5						F	D	Z0	W	r	r	r	c				
7	add x7, x1, x0							F	D	Z0	W	r	r	r	c			
8	add x8, x2, x0								F	D	D	Z0	W	r	r	c		
9	add x9, x4, x0									F	F	D	Z0	W	r	r	c	
10	add x6, x1, x3											F	D	Z0	W	r	r	c

Table 1: TEST_CHECK_EQ(x6, 8)

This test:

1. Fails when values commit in the wrong order but passes when ROB is present because of WAW hazard between instructions 5 and 6 and RAW hazard between instructions 10 and 6
2. Has a value which has to be bypassed out of the ROB because instruction 6 has not committed when x3 is read by instruction 10

3.2 Test 2

1	li x1, 1	F	D	Y0	Y1	W	c											
2	li x2, 2		F	D	Y0	Y1	W	c										
3	li x4, 3			F	D	Y0	Y1	W	c									
4	li x5, 4				F	D	Y0	Y1	W	c								
5	mul x3, x1, x2					F	D	X0	X1	X2	X3	W	c					
6	add x3, x4, x5						F	D	Z0	W	r	r	r	c				
7	add x7, x1, x0							F	D	Z0	W	r	r	r	c			
8	add x8, x2, x0								F	D	D	Z0	W	r	r	c		
9	add x6, x1, x3									F	F	D	Z0	W	r	r	c	

Table 2: TEST_CHECK_EQ(x6, 8)

This test has the same WAW scenario as test 1 but nevertheless executes correctly on both the original and finished processor because instruction 5 hasn't yet written back when instruction 9 is in execute.

4 Evaluation

	riscvlong		riscvooo	
	# cycles	IPC	# cycles	IPC
bin-search	1678	0.709774	1709	0.696899
cmplx-mult	2550	0.734510	2600	0.720385
masked-filter	4916	0.833605	5849	0.700633
vvadd	471	0.961783	511	0.886497

Table 3: Simulation results for benchmark programs

5 Discussion

The long processor has slightly higher IPC than the out-of-order processor for every benchmark. This is because the out-of-order processor still has in-order issue and in-order commit. Thus the processor stalls every time there are RAW hazards and the execution of the previous instruction has not to bypass values, just like the long processor. The slightly additional number of cycles can be due to the extra commit stage.

6 Figures

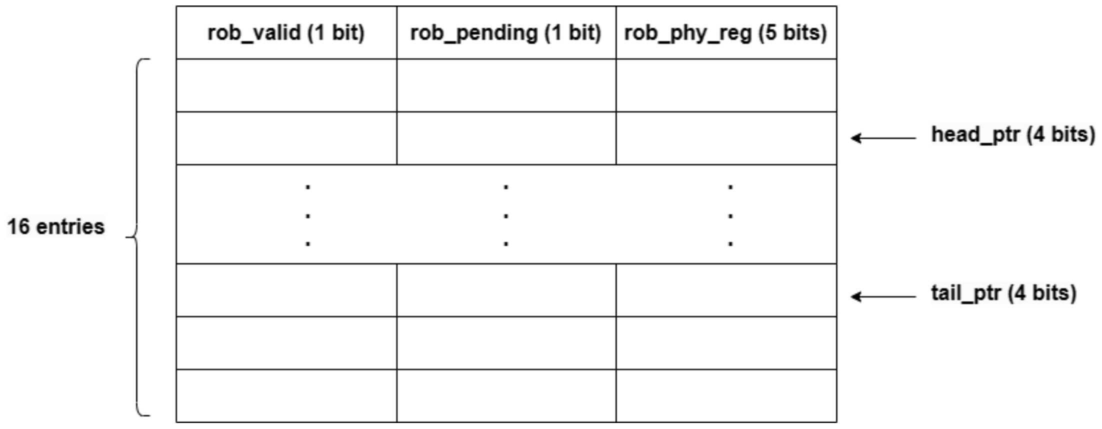

Figure 1: ROB