ECE/CS 752
PROJECT-1 REPORT
ISSUE QUEUE

Abhishek Desai

Bhavesh Mehta

Fall 2003
Prof Mikko Lipasti

Issue Queue for a Superscalar Processor

Introduction:

In a superscalar processor, pipelined execution takes place out-of-order also. Until we issue the instructions the flow is in-order. The actual execution of instructions occurs out-of-order. And of course to maintain the program order, the retiring of the instructions must happen in in-order. Issue Queue is the data-structure which facilitates that out-of-order execution. Issue Queue represents the dispatch stage of the pipeline. It gets decoded instructions as inputs from the ID stage and issues them to the EX unit(s). We assume here that before IQ (Issue Queue) gets the decoded instructions, the operands are renamed to avoid the WAR and WAW hazards. So the only type of data hazard remaining to be checked is the RAW hazard (the TRUE dependence). We make sure that we do not issue the instructions before we resolve the true dependence. As soon as the true dependence is resolved the instruction is ready to be fired. It is this stage where we exploit out-of-order paradigm. If at a particular instance of time more than one instruction is ready to be fired, we issue them (depending upon the availability of the functional units) in any order. The IQ can have multiple read ports (to issue to multiple functional units) and multiple write ports (to get new instructions). How many instructions to store in IQ is really a question of the kind of workload we are considering. Not all the times the IQ will have enough instructions to issue, in which case it just outputs NOPs to the functional units.

Issue Queue Specification:

In this project, our IQ holds up to 16 instructions at a time. It has 2 read ports and 2 write ports. The pipeline is assumed to have 2 execution units with the following constraints. The two issue ports R0 and R1 lead to asymmetric pipelines: all instructions issued from R0 are executed on a pipeline with full bypass to itself and the R1 pipeline. Hence, all instructions issued from R0 are followed in the very next cycle by a dependent instruction in either pipeline. The issue queue logic meets this requirement of back-to-back issue of dependent instructions, as long as the producer is issued through port R0. However, instructions issued through port R1 issue to a pipeline that does not have bypass paths (assume that due to physical design constraints, only one set of bypass paths could be provided for the execute stage to stay within cycle time requirements). Hence, an instruction that issues through port R1 does not allow a dependent instruction to issue from either port until 3 cycles later (2 pipeline bubbles). This allows enough time for the producer instruction to execute and write back its result to the register file before the consumer instruction issues and reads from the register file**.** The issue queue logic meets his requirement of delaying instructions dependent on an instruction issued through R1 by three cycles.

Input Specification:

Figure 1 shows a detailed view of the data written through each write port: the RobID specifies the reorder buffer slot the instruction resides in, DestOp specifies the physical register number of the register written by this instruction, and LeftOp and RightOp specify the left and right source operands of this instruction. In addition, each Op field (DestOp, LeftOp, and RightOp) has an associated used bit which indicates whether or not this field is used in the instruction. For example, if one of the right operands was an immediate value, then the used bit for this value would be cleared.

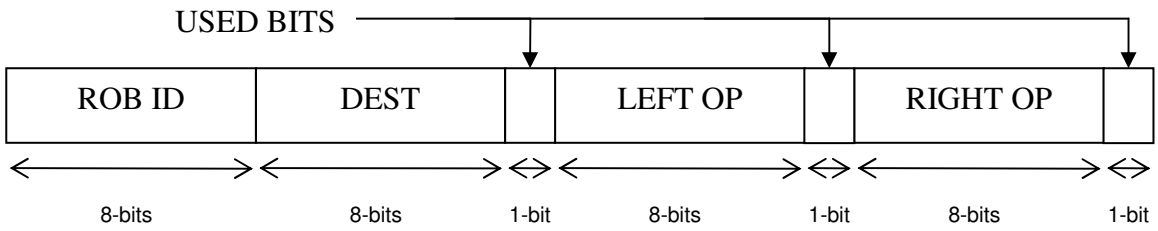


Figure 1: Input Specification

Line in the IQ:

To this input we add a valid bit and store it in the line of the issue queue as shown in Figure 2. This bit gives us information if the particular line in the IQ is empty or valid. If V = 1 then the line is full and valid. If V= 0 then the line is assumed to be empty. The valid bit also aids us in designing the collapse logic, ready logic and request logic. Also the used bits act as ready bits.

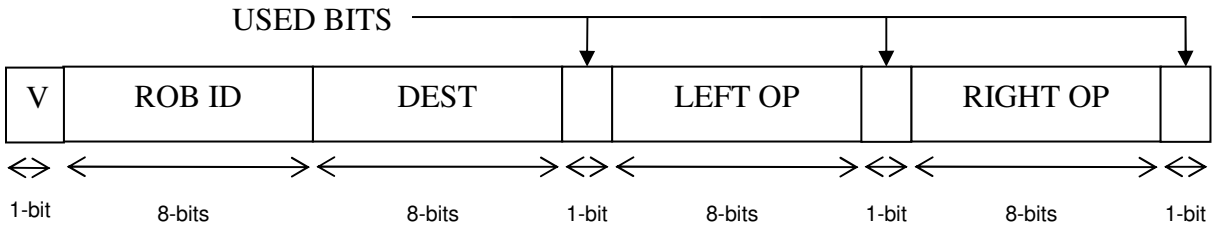


Figure 2: Line in the IQ

Module Descriptions:

Our design has the following 11 modules and they constitute the issue logic. **issue** is the name of the top module and all other modules are in the 2nd level of hierarchy under this top module.

1) input_logic
2) input_mux
3) scoreboard
4) wakeup_logic
5) requestlogic
6) selectlogic
7) execR1
8) readylogic
9) output_mux
10) collapse_controllogic
11) issue

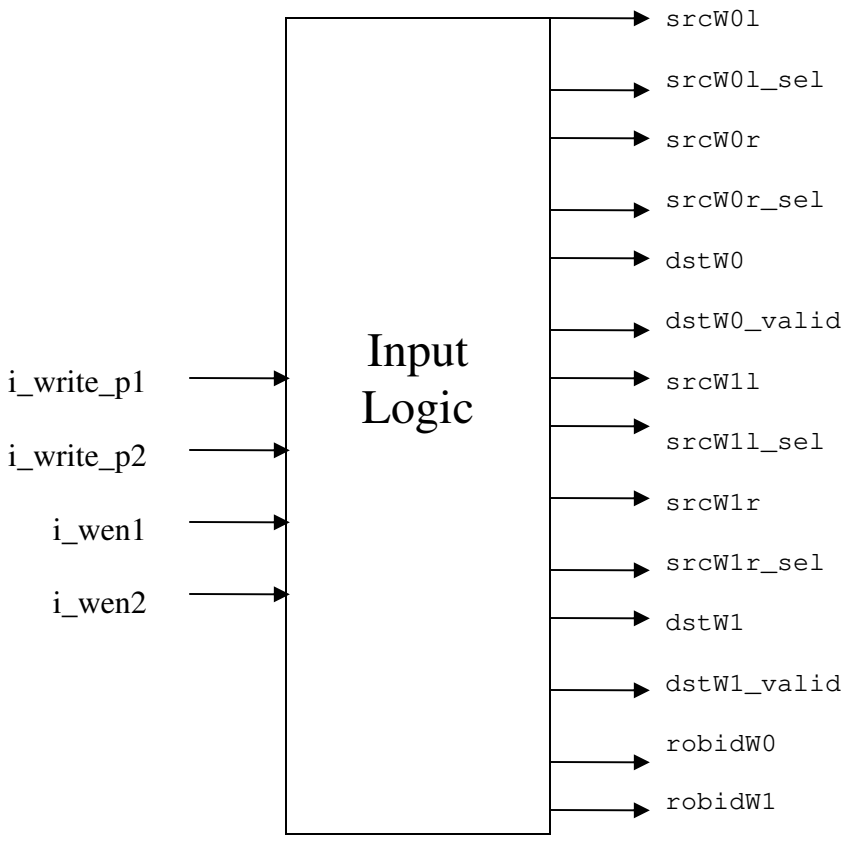
We explain below the functionality of each of the modules with the bottom-up approach.

Module: input_logic
File: input_logic.v
Type: combinational
Functionality:

It prepares proper operands (LeftOp, RightOp, DestOp) for the input_mux and the issue queue depending upon the validity of the input. i.e. depending upon whether WEN for the corresponding input is high or not. If it is not high, operands are set to 0s and the robid is set to 128. It also generates the select signals to be used by InputMux depending upon whether the source operands are used or not.
This module is functioning properly as seen from the simulation.
The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
i_write_pn (input) (2 instances)	Robid	Reorder buffer ID, 8 bits, 0-127 correspond to real ROB entries, 8'b10000000 indicates NOP
	Dest Op	Destination physical register, 8 bits
	Dest_valid	Indicates if destination is valid, 1 bit
	Left Op	Left source physical register, 8 bits
	Left_Op used	Indicates if left operand is used, 1 bit
	Right Op	Right source physical register, 8 bits
	Right Op used	Indicates if right operand is used, 1 bit
i_wen1 (input)	N/A	Writes first write port into issue queue, 1bit
i_wen2 (input)	N/A	Writes second write port into issue queue, 1bit

srcW0l (output)	N/A	Left Op at port W0, 8 bits
srcW0l_sel (output)	N/A	selects the input of input_mux module for left operand, 1 bit
srcW0r (output)	N/A	Right Op at port W0, 8 bits
srcW0r_sel (output)	N/A	selects the input of input_mux module for right operand, 1 bit
dstW0 (output)	N/A	Destination Op at port W0, 8 bits
dstW0_valid (output)	N/A	Indicates validity of destination bit at W0 port, 1 bit
srcW1l (output)	N/A	Left Op at port W1, 8 bits
srcW1l_sel (output)	N/A	selects the input of input_mux module for left operand, 1 bit
srcW1r (output)	N/A	Right Op at port W1, 8 bits
srcW1r_sel (output)	N/A	selects the input of input_mux module for right operand, 1 bit
dstW1 (output)	N/A	Destination Op at port W1, 8 bits
dstW1_valid (output)	N/A	Indicates validity of destination bit at W1 port, 1 bit
robidW0 (output)	N/A	Robid at port W0, 8 bits
robidW1 (output)	N/A	Robid at port W1, 8 bits



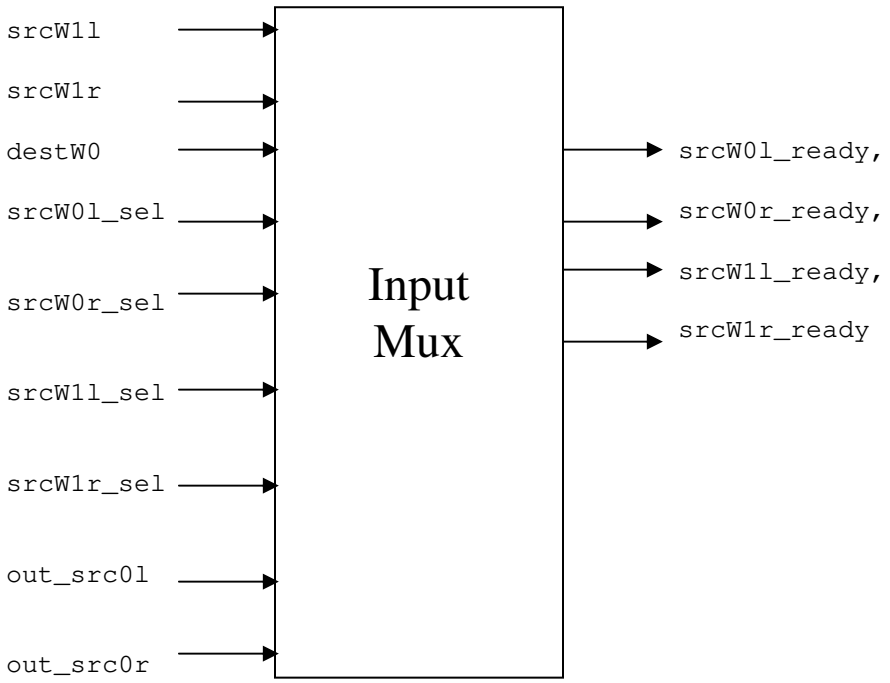
Module : input_mux
File: input_mux.v
Type: combinational
Functionality:

This module selects proper ready bits from the scoreboard entries. If the source operands are not used then it sets the ready bits to zeros. Further it also checks for RAW hazard between the two inputs at W0 and W1. If a RAW hazard is detected then the corresponding ready bits are reset.

This module is functioning properly as seen from the simulations.

The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
srcW1l (input)	N/A	Left Op at port W1, 8 bits (for RAW detection)
srcW1r (input)	N/A	Right Op at port W1, 8 bits (for RAW detection)
destW0 (input)	N/A	Destination Op at port W0, 8 bits (for RAW detection)
srcW0l_sel (input)	N/A	selects the input of input_mux module for left operand, 1 bit
srcW0r_sel (input)	N/A	selects the input of input_mux module for right operand, 1 bit
srcW1l_sel (input)	N/A	selects the input of input_mux module for left operand, 1 bit
srcW1r_sel (input)	N/A	selects the input of input_mux module for right operand, 1 bit
out_src0l (input)	N/A	readiness of left source Op from scoreboard, 1 bit
out_src0r (input)	N/A	readiness of right source Op from scoreboard, 1 bit
out_src1l (input)	N/A	readiness of left source Op from scoreboard, 1 bit
out_src1r (input)	N/A	readiness of right source Op from scoreboard, 1 bit
srcW0l_ready (output)	N/A	Ready bit of left Op, 1 bit (for instruction at W0)
srcW0r_ready (output)	N/A	Ready bit of Right Op, 1 bit (for instruction at W0)
srcW1l_ready (output)	N/A	Ready bit of left Op, 1 bit (for instruction at W1)
srcW1r_ready (output)	N/A	Ready bit of Right Op, 1 bit (for instruction at W1)



Module : scoreboard
File: scoreboard.v
Type: sequential
Functionality:

This module is a memory of 256 flip flops which stores the status of the destination registers for each instruction. If the register is busy the corresponding flip flop is set, else it is reset. In the beginning we clear all the flip flops with the reset signal.

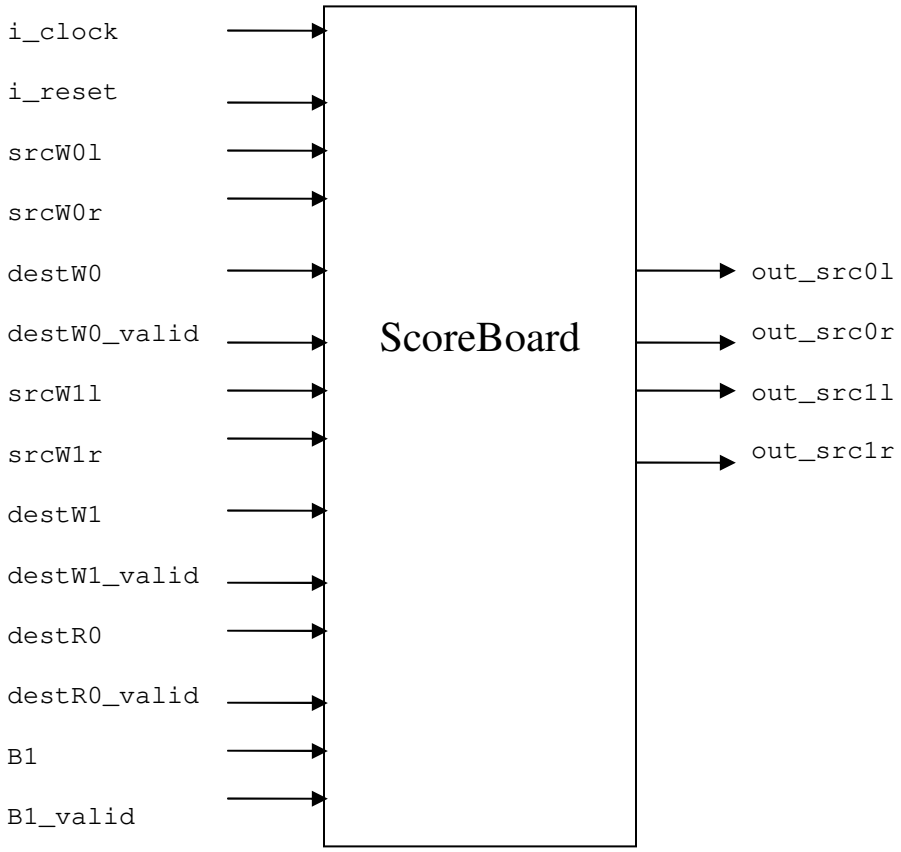
The scoreboard is updated by the broadcast buses at every positive edge of the clock so that the destination registers of the instructions which have finished execution will be cleared. The new instructions entering the system will set the corresponding bit of the destination operand at the positive edge of the clock.

Every time a new instruction enters the system the source ops will check if that particular register is busy or not and read in the corresponding bits. These bits also represent the ready bits for the source ops. They also check the broadcast buses to check for their status. If there is a conflict with the status bits indicated by the scoreboard and the broadcast buses, precedence is given to the status indicated by the broadcast buses.

This module is functioning properly as seen from the simulations.

The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
i_clock (input)	N/A	Synchronizes the storage elements in the scoreboard, 1 bit
i_reset (input)	N/A	Clears all entries in the scoreboard (synchronous reset), 1 bit
srcW0l (input)	N/A	Left source physical register (at W0) checks its ready bit, 8 bits
srcW0r (input)	N/A	Right source physical register (at W0) checks its ready bit, 8 bits
destW0 (input)	N/A	Destination Op at port W0 marks its bit as busy, 8 bits
destW0_valid (input)	N/A	Indicates validity of destination Op at W0, 1 bit
srcW1l (input)	N/A	Left source physical register (at W1) checks its ready bit, 8 bits
srcW1r (input)	N/A	Right source physical register (at W1) checks its ready bit, 8 bits
destW1 (input)	N/A	Destination Op at port W1 marks its bit as busy, 8 bits
destW1_valid (input)	N/A	Indicates validity of destination Op at W1, 1 bit
destR0 (input)	N/A	Broadcast destination Op from R0, 8 bits
destR0_valid (input)	N/A	Indicates validity of broadcast bus B0, 1 bit
B1 (input)	N/A	Broadcast destination Op from R1, 8 bits
B1_valid (input)	N/A	Indicates validity of broadcast bus B1, 1 bit
out_src0l (output)	N/A	Ready bit of left Op, 1 bit (for instruction at W0)
out_src0r (output)	N/A	Ready bit of right Op, 1 bit (for instruction at W0)
out_src1l (output)	N/A	Ready bit of left Op, 1 bit (for instruction at W1)
out_src1r (output)	N/A	Ready bit of right Op, 1 bit (for instruction at W1)



Module: wakeup_logic
File: wakeup_logic.v
Type: combinational
Functionality:

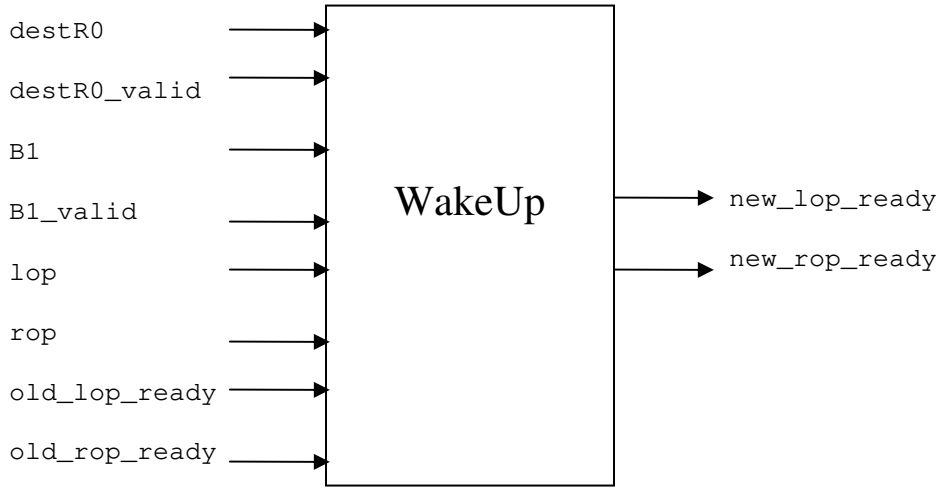
This module continuously monitors the broadcast buses and as soon as the destination op broadcasted matches the source ops the corresponding ready bit for those source operands is made 0. We have 16 of these modules one for each of the 16 lines in the issue queue. If there is no match then the earlier ready bits are maintained as it is.

This module is functioning properly as seen from the simulations.

The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
destR0 (input)	N/A	Broadcast value from R0 to compare Source Ops in the queue, 8 bits
destR0_valid (input)	N/A	Indicates validity of broadcast B0, 1 bit
Bl (input)	N/A	Broadcast value from R1 to compare Source Ops in the queue, 8 bits

B1_valid (input)	N/A	Indicates validity of broadcast B1, 1 bit
lop (input)	N/A	Left Source Op to compare with broadcast, 8 bits
rop (input)	N/A	Right Source Op to compare with broadcast, 8 bits
old_lop_ready (input)	N/A	Left Source Op old (current) ready bit, 1 bit
old_rop_ready (input)	N/A	Right Source Op old (current) ready bit, 1 bit
new_lop_ready (output)	N/A	Left Source Op updated ready bit, 1 bit
new_rop_ready (output)	N/A	Right Source Op updated ready bit, 1 bit



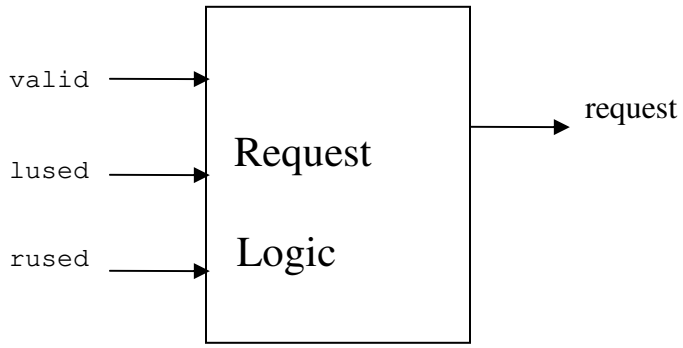
Module: requestlogic
File: requestlogic.v
Type: combinational
Functionality:

This module continuously monitors the valid bits in the issue queue and the ready bits for both the source operands. When both the ready bits go to zero it indicates that there are no more dependences for that particular instruction and so it makes the corresponding request line high to specify that the instruction is ready to fire. We have 16 of these modules one for each of the 16 lines in the issue queue.

This module is functioning properly as seen from the simulations.

The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
valid (input)	N/A	Indicates validity of the line in the issue queue, 1 bit
lused (input)	N/A	Used (ready) bit of left source Op, 1 bit
rused (input)	N/A	Used (ready) bit of right source Op, 1 bit
request (output)	N/A	Request for issue for this line of issue queue, 1 bit



Module: selectlogic
File: selectlogic.v
Type: combinational
Functionality:

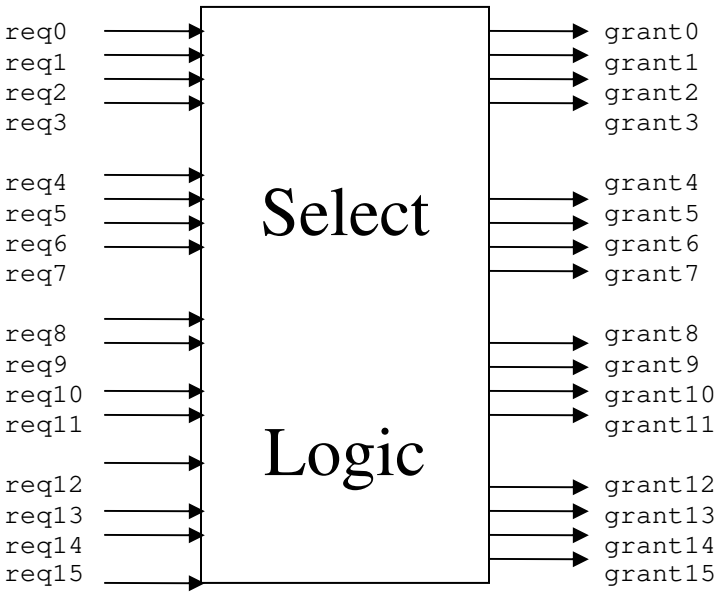
This module continuously monitors the requests from all the 16 lines of the issue queue and depending on which instructions are the oldest in the queue it grants permission to issue utmost 2 instructions during the next positive edge of the clock. The oldest instructions are given precedence over the newer instructions. We keep track about the new and old instructions by collapsing the issue queue every clock cycle so that the oldest instructions are always on top of the queue and the newer instructions are at the bottom of the queue.

This module is functioning properly as seen from the simulations.

The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
req0 (input)	N/A	Request from line0 to issue, 1 bit
req1 (input)	N/A	Request from line1 to issue, 1 bit

req2 (input)	N/A	Request from line2 to issue, 1 bit
. . .	. . .	. . .
req14 (input)	N/A	Request from line14 to issue, 1 bit
req15 (input)	N/A	Request from line15 to issue, 1 bit
grant0 (output)	N/A	Granting permission to issue line0, 1 bit
grant1 (output)	N/A	Granting permission to issue line1, 1 bit
grant2 (output)	N/A	Granting permission to issue line2, 1 bit
. . .	. . .	. . .
grant14 (output)	N/A	Granting permission to issue line14, 1 bit
grant15 (output)	N/A	Granting permission to issue line15, 1 bit



Module: execR1
File: execR1.v
Type: sequential

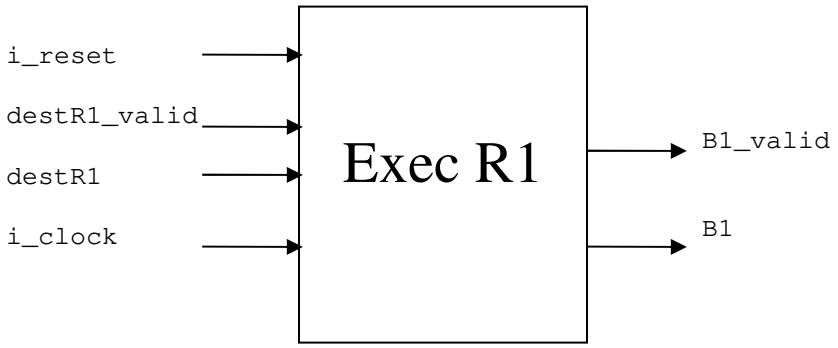
Functionality:

This module will latch a specific output of the output_mux at every positive edge of the clock. It latches the Destination Op and its corresponding valid bit for the instruction which is issued at R1. We need to latch this information because we need to delay its broadcast by 3 cycles. This is in accordance with the project specification. The latched value is transferred to another register in the next clock cycle and then this value is available as the broadcast on bus B1.

This module is functioning properly as seen from the simulations.

The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
i_reset (input)	N/A	Clears all flip flops in the dummy execution unit, 1 bit (synchronous)
destR1_valid (input)	N/A	Indicates validity of the destination Op for instruction at R1, 1 bit
destR1 (input)	N/A	Destination Op for instruction issued at R1, 8 bits
i_clock (input)	N/A	Synchronizes the storage elements in this dummy execution unit, 1bit
B1_valid (output)	N/A	Indicates validity of Broadcast B1, 1 bit
B1 (output)	N/A	Broadcast Bus B1 carrying Destination Op for R1 instruction, 8 bits



Module: readylogic
File: readylogic.v
Type: combinational
Functionality:

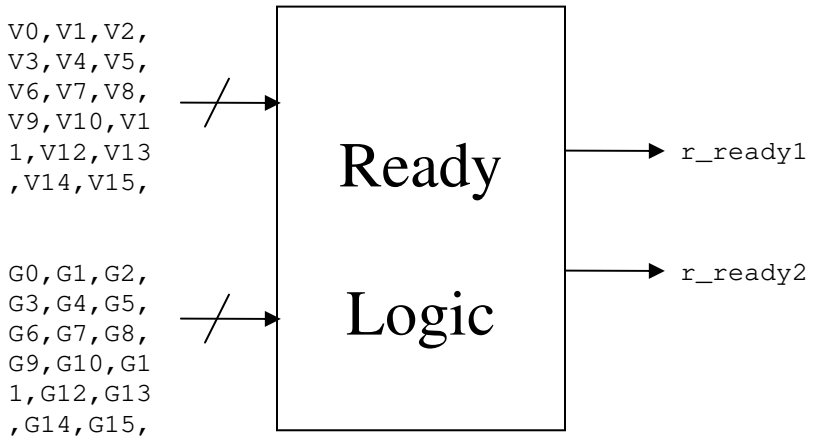
The ready0 signal is associated with the W0 port and the ready1 signal is associated with the W1 port. This module continuously monitors the valid bits of the issue lines and the grant signals from the select logic. If the entire issue queue is full (all valid bits are asserted) and no

grant signals are high then both the ready signals (at the output) go low indicating that the decode unit must stall giving instructions to the issue unit because there is no space left. If the issue queue is full but two grant lines have been asserted then both the ready signals continue to remain high because the IQ can accept two instructions in the next clock cycle since two instructions will get issued. If only one grant line is high and the issue queue is full then ready0 remains high but ready1 goes low. Thus when only one instruction can be accepted the instruction at W0 port is taken in. This situation also occurs when the last line of the IQ is empty and none of the lines are granted permission to fire.

This module is functioning properly as seen from the simulations.

The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
V0 (input)	N/A	Indicates validity of line0 of issue queue, 1 bit
V1 (input)	N/A	Indicates validity of line1 of issue queue, 1 bit
V2 (input)	N/A	Indicates validity of line2 of issue queue, 1 bit
. . .	. . .	. . .
V14 (input)	N/A	Indicates validity of line14 of issue queue, 1 bit
V15 (input)	N/A	Indicates validity of line15 of issue queue, 1 bit
G0 (input)	N/A	Grant permission to issue line0, 1 bit
G1 (input)	N/A	Grant permission to issue line1, 1 bit
G2 (input)	N/A	Grant permission to issue line2, 1 bit
. . .	. . .	. . .
G14 (input)	N/A	Grant permission to issue line14, 1 bit
G15 (input)	N/A	Grant permission to issue line15, 1 bit
r_ready1 (output)	N/A	Indicates that the issue queue can accept instructions at W0, 1 bit
r_ready2 (output)	N/A	Indicates that the issue queue can accept instructions at W1, 1 bit



Module: output_mux
File: output_mux.v
Type: combinational
Functionality:

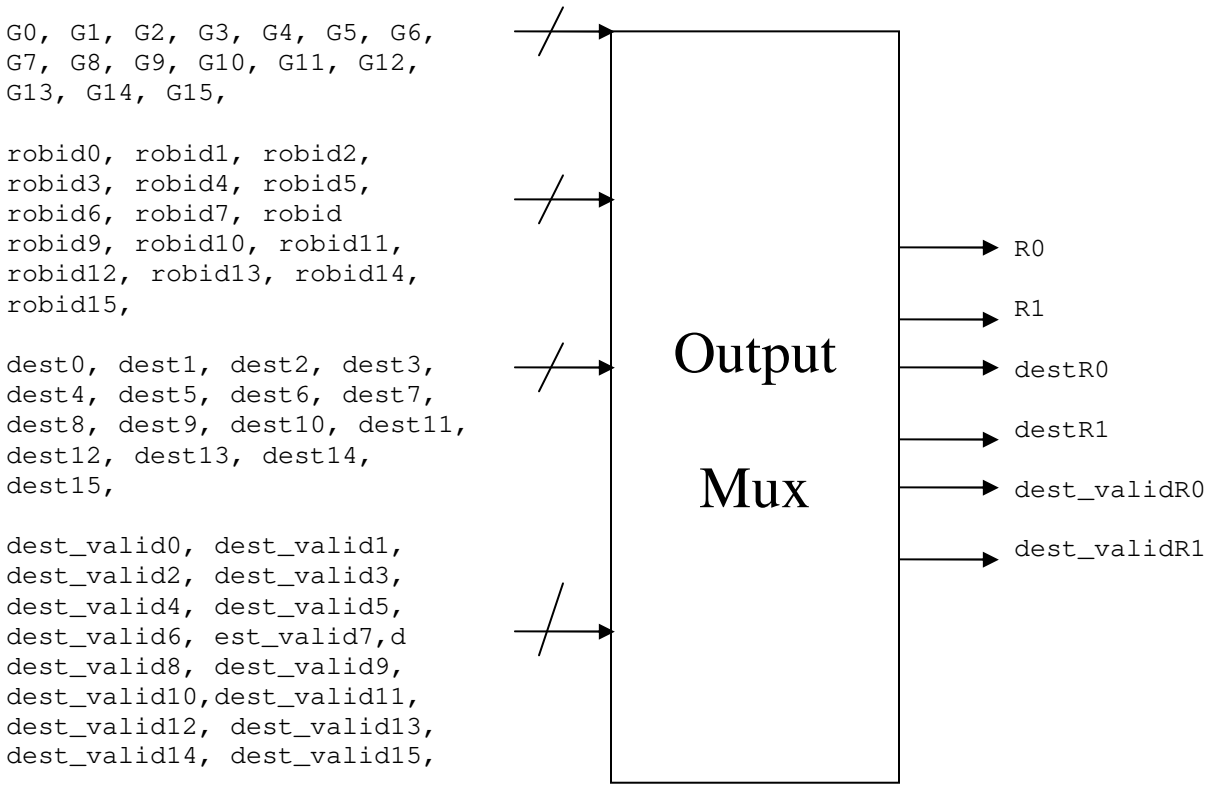
This module basically selects which robid has to be routed to the output of R0 or R1 depending on the grant lines. If two grant lines are high then the two robids will be routed to R0 and R1 the oldest instruction being routed to R0. If only one grant line is high then that robid is routed to R0 and NOPs are issued at R1. When no lines are granted then NOPs are issued at both R0 and R1. Along with routing the robids this logic also routs the appropriate destination ops and corresponding valid bits so that broadcasts can occur.

This module is functioning properly as seen from the simulations.

The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
G0 (input)	N/A	Grant permission to issue line0, 1 bit
G1 (input)	N/A	Grant permission to issue line1, 1 bit
. . .	. . .	. . .
G15 (input)	N/A	Grant permission to issue line15, 1 bit
robid0 (input)	N/A	Indicates Robid of line0 of issue queue, 8 bits
robid1 (input)	N/A	Indicates Robid of line1 of issue queue, 8 bits
. . .	. . .	. . .
robid15 (input)	N/A	Indicates Robid of line15 of issue queue, 8 bits
dest0 (input)	N/A	Indicates destination Op at line0, 8 bits

dest1 (input)	N/A	Indicates destination Op at line1, 8 bits
. . .	. . .	. . .
dest15 (input)	N/A	Indicates destination Op at line15, 8 bits
dest_valid0 (input)	N/A	Indicates validity of destination Op at line0, 1 bit
dest_valid1 (input)	N/A	Indicates validity of destination Op at line1, 1 bit
. . .	. . .	. . .
dest_valid15 (input)	N/A	Indicates validity of destination Op at line15, 1 bit
R0 (output)	N/A	Robid of instruction that is issued, 8 bits (at port R0)
R1 (output)	N/A	Robid of instruction that is issued, 8 bits (at port R1)
destR0 (output)	N/A	Destination Op of instruction issued at R0, 8 bits
destR1 (output)	N/A	Destination Op of instruction issued at R1, 8 bits
dest_validR0 (output)	N/A	Indicates validity of destination Op of instruction issued at R0, 1 bit
dest_validR1 (output)	N/A	Indicates validity of destination Op of instruction issued at R1, 1 bit



Module: collapse_controllogic
File: collapse_controllogic.v
Type: combinational
Functionality:

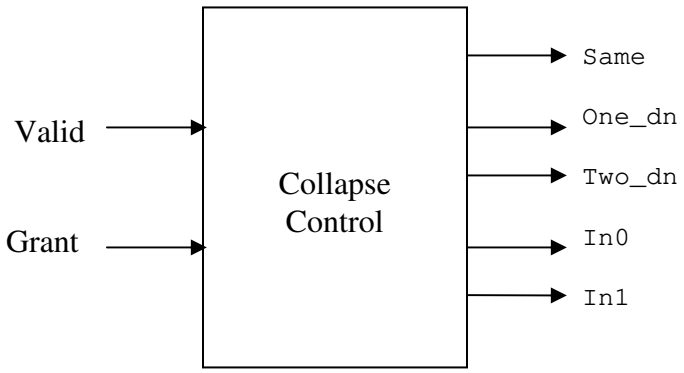
This module continuously monitors the grant signals and the valid bits from the issue lines and generates four signals for each line of the issue queue. The 'Same' signal indicates that the particular line will remain unchanged after the collapse has occurred. The 'One_dn' signal indicates that the particular line of the issue queue will be overwritten by the line immediately below it. The 'Two_dn' signal indicates that the particular line of the issue queue will be overwritten by the line which is two positions below it. The 'In0' signal indicates that the particular line will be overwritten by the contents at port W0. The 'In1' signal indicates that the particular line will be overwritten by the contents at port W1.

This module is functioning properly as seen from the simulations.

The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
Valid (input)	Valid[0]	Indicates validity of lines in issue queue, 16 bits
	Valid[1]	
	. . .	
	Valid[15]	
Grant (input)	Grant[0]	Grants to lines in issue queue, 16 bits
	Grant[1]	
	. . .	
	Grant[15]	
Same (output)	Same[0]	Indicates that the line contents remain unchanged, 16 bits
	Same[1]	
	. . .	
	Same[15]	
One_dn (output)	One_dn[0]	Indicates that the line contents are replaced by the contents of the line which is 1 line below it, 16 bits
	One_dn[1]	
	. . .	
	One_dn[15]	
Two_dn (output)	Two_dn[0]	Indicates that the line contents are replaced by the contents of the line which is 2 lines below it, 16 bits
	Two_dn[1]	

	. . .	
	Two_dn[15]	
In0 (output)	In0[0]	Indicates that the line will be overwritten by new instruction from W0, 16 bits
	In0[1]	
	. . .	
	In0[15]	
In1 (output)	In1[0]	Indicates that the line will be overwritten by new instruction from W1, 16 bits
	In1[1]	
	. . .	
	In1[15]	



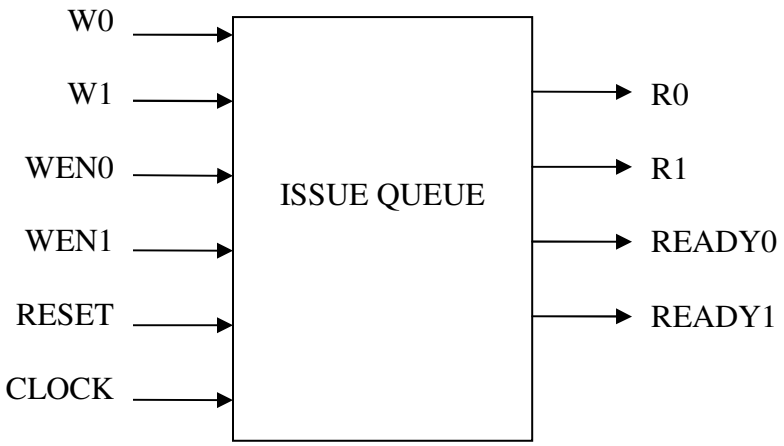
Module: issue
File: issue_logic.v
Type: sequential
Functionality:

This is the top level module of the issue queue system. It has 16 registers which are clocked at the positive edge. It monitors the signals for each line from the collapse_controllogic module and collapses the queue at every positive edge of the clock. It also chooses the appropriate ready bits and updates them after the collapse has occurred. The structure of each line has been shown in figure 2 earlier in the report.

This module is functioning properly as seen from the simulations.

The inputs and output to this module and their functionality is described below.

Input/Output	Field	Function, size
i_write_p1 (input)	Robid	Reorder buffer ID, 8 bits, 0-127 correspond to real ROB entries, 8'b10000000 indicates NOP
	Dest Op	Destination physical register, 8 bits
	Dest_valid	Indicates if destination is valid, 1 bit
	Left Op	Left source physical register, 8 bits
	Left_Op used	Indicates if left operand is used, 1 bit
	Right Op	Right source physical register, 8 bits
	Right Op used	Indicates if right operand is used, 1 bit
i_write_p2 (input)	Robid	Reorder buffer ID, 8 bits, 0-127 correspond to real ROB entries, 8'b10000000 indicates NOP
	Dest Op	Destination physical register, 8 bits
	Dest_valid	Indicates if destination is valid, 1 bit
	Left Op	Left source physical register, 8 bits
	Left_Op used	Indicates if left operand is used, 1 bit
	Right Op	Right source physical register, 8 bits
	Right Op used	Indicates if right operand is used, 1 bit
i_wen1 (input)	N/A	Writes first write port into issue port, 1 bit
i_wen2 (input)	N/A	Writes second write port into issue port, 1 bit
i_reset (output)	N/A	Clears all entries in issue queue (synchronous) , 1 bit
i_clock (output)	N/A	Synchronizes the storage elements in the issue queue, 1 bit
r_read_p1 (output)	N/A	Robid of instruction issuing in that cycle from R0, 8 bits
r_read_p2 (output)	N/A	Robid of instruction issuing in that cycle from R1, 8 bits
r_ready1 (output)	N/A	Indicates IQ ready to accept a new entry on the first write port, 1 bit
r_ready2 (output)	N/A	Indicates IQ ready to accept a new entry on the second write port, 1 bit



Overall Picture:

Thus every instruction that enters the issue queue system marks its destination bit as busy in the scoreboard and checks the scoreboard and the broadcast bits to see if its source ops are ready or not. If they are ready then the instruction gets issued immediately in the same cycle if it is granted permission to fire. If the source ops are not ready then it waits in the issue queue until the wakeup logic detects that the source ops are ready. At this time it will request permission to fire. When select logic grants permission to fire, its robid is routed through the output_mux to the port R0 or R1 and the destop of that instruction is broadcasted. While the instruction was waiting in the queue it might get displaced to different lines due to collapsing of the queue. Every time it is displaced its ready bits are appropriately updated. The ready0 and ready1 output signals of this system indicate if the issue queue is full or if it can accommodate more instructions. Instructions which are issued through R0 are broadcasted immediately but those issued through R1 are broadcasted after a 3 cycle delay.

Contribution of each team member:

Task	Abhishek	Bhavesh
Design of system	50%	50%
Coding the modules	50%	50%
Testing and Simulation	50%	50%
Synthesis	50%	50%
Report writing	50%	50%