ECE 341, Fall 2013,
Siddharth Dedhia (sdedhia)
Project 2 NOC Report

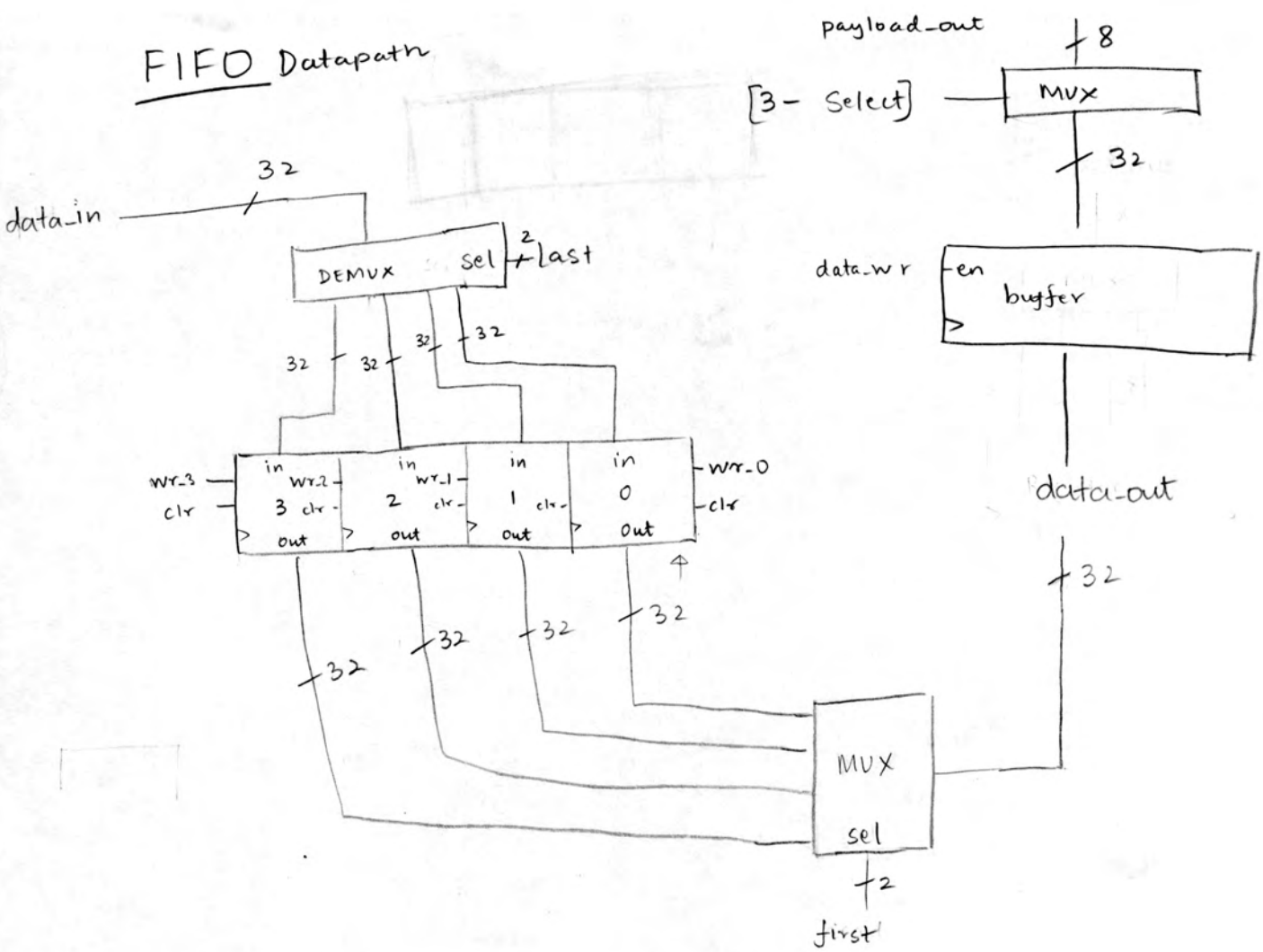
Description

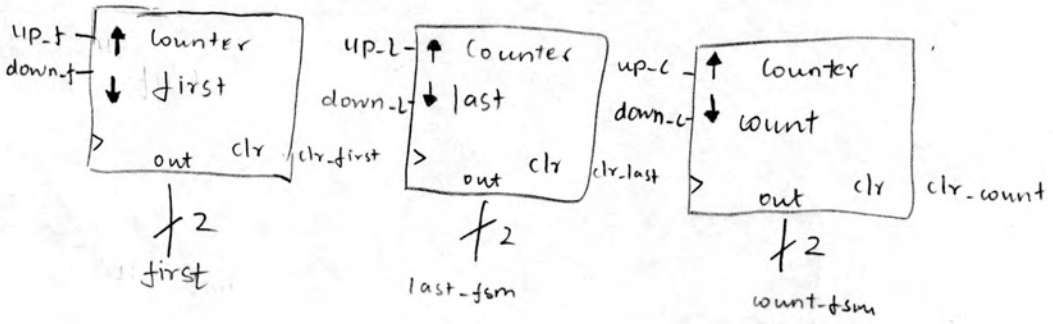
There are several components to the NoC systemverilog implementation. The most complex one being the organization of the router, who's job is to route the packets to the appropriate ports in the least possible time, and at the same time being fair to all the ports. To avoid designing a very complex FSM for the router there is a queue at each of ports' output buffer that can hold at most 4 items and can pipeline the data before it is sent to the node. There is further logic in the router to maintain a round robin priority so as to be fair to each node if data appears on them at the **same time** and wants to go to the **same place**. Priority is not invoked if at the same time multiple nodes want to send data to different destinations, in which case there is a concurrent transfer of data from the input buffers of the ports to the appropriate queues.

Datapath

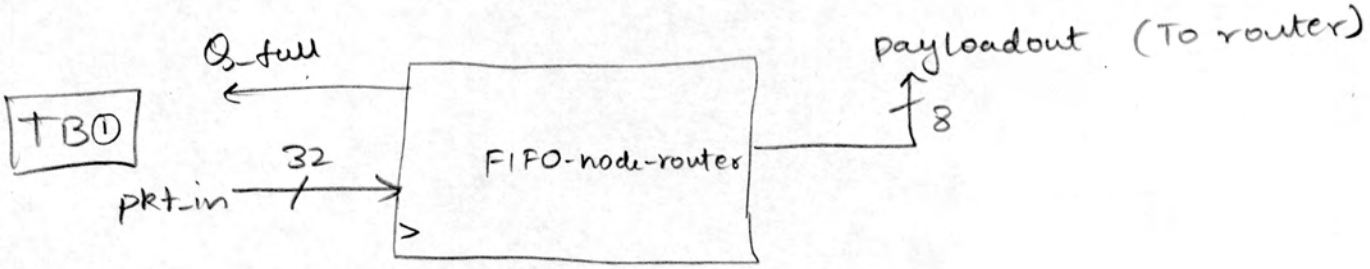
FIFO
Node
Router

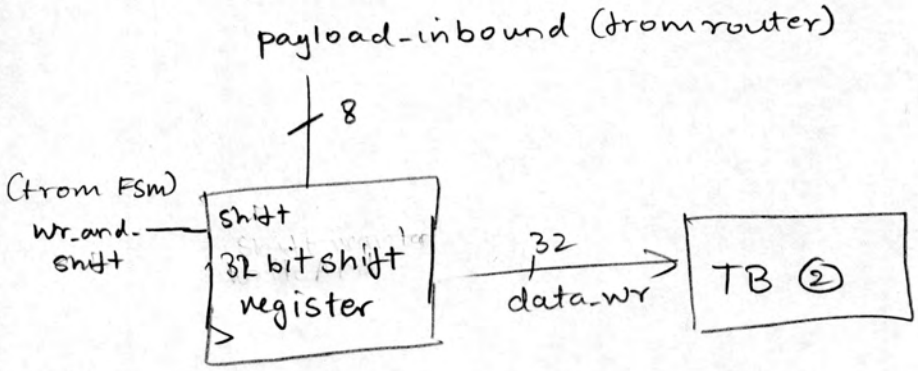
FIFO Datapath

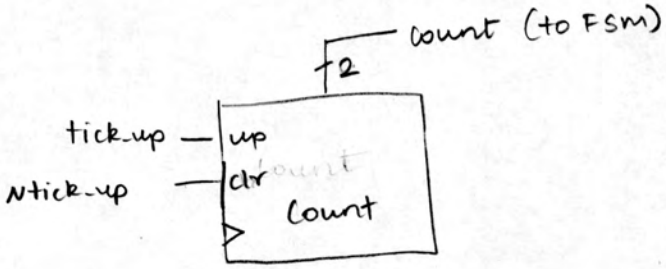




NODE Datapath

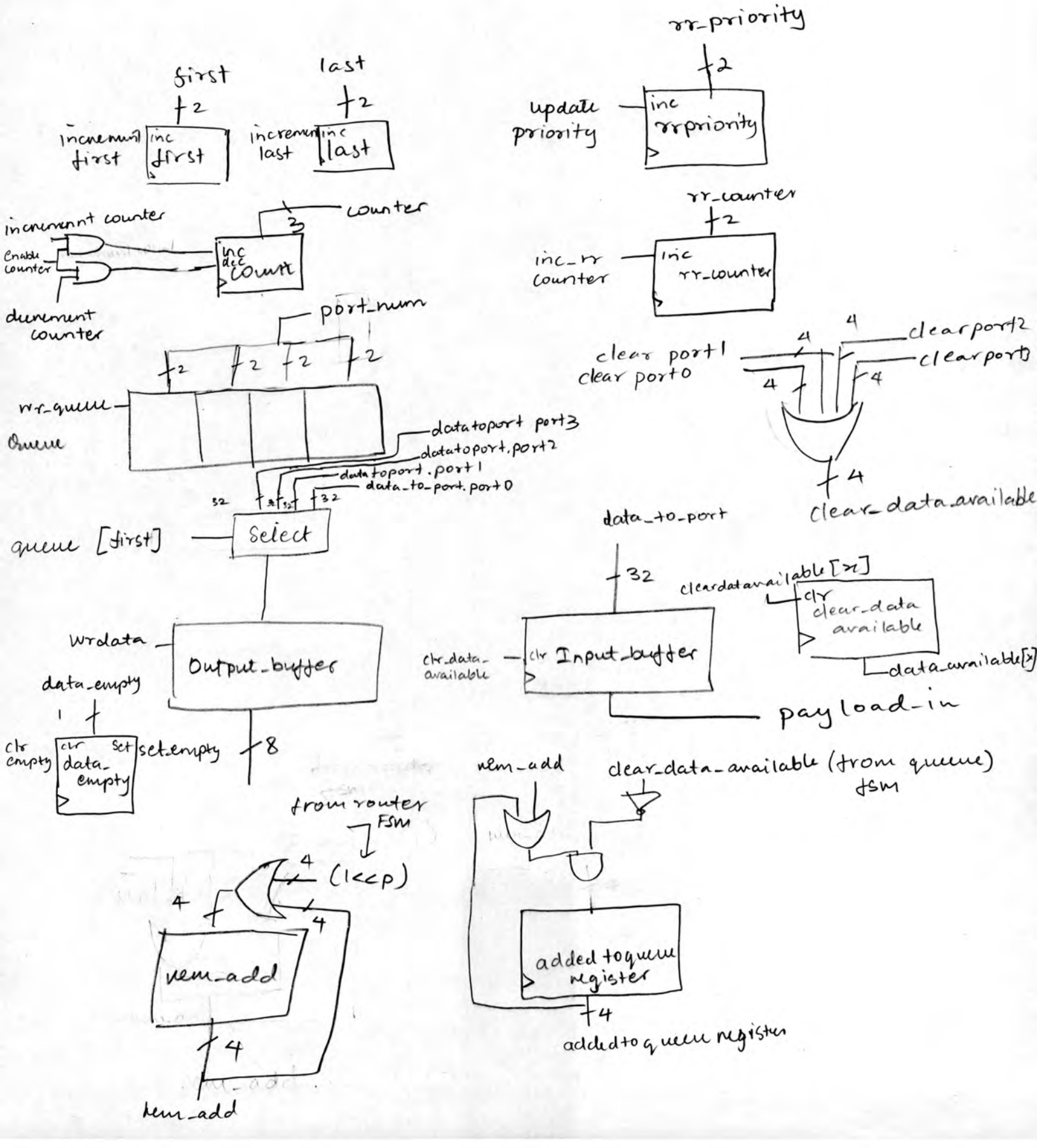






Router Datapath

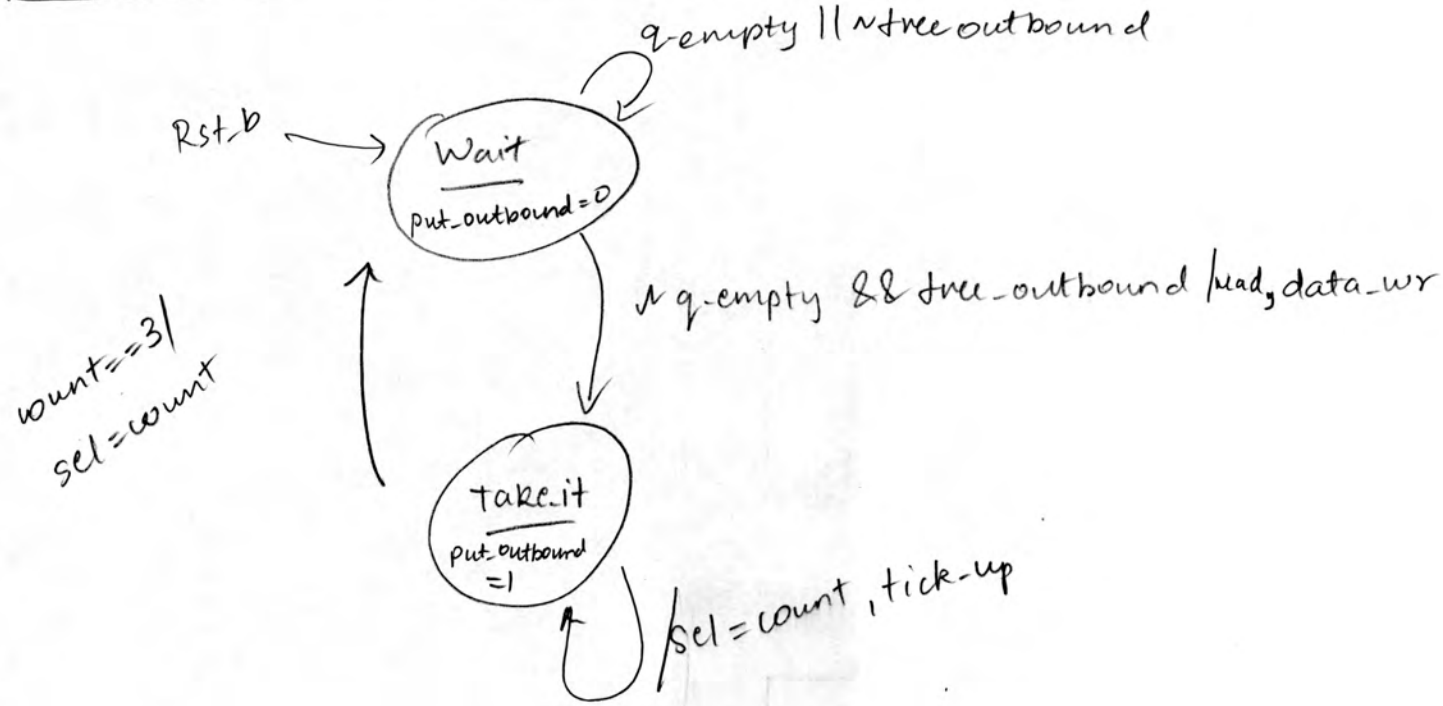
Note: rem_add is a latch (intended)



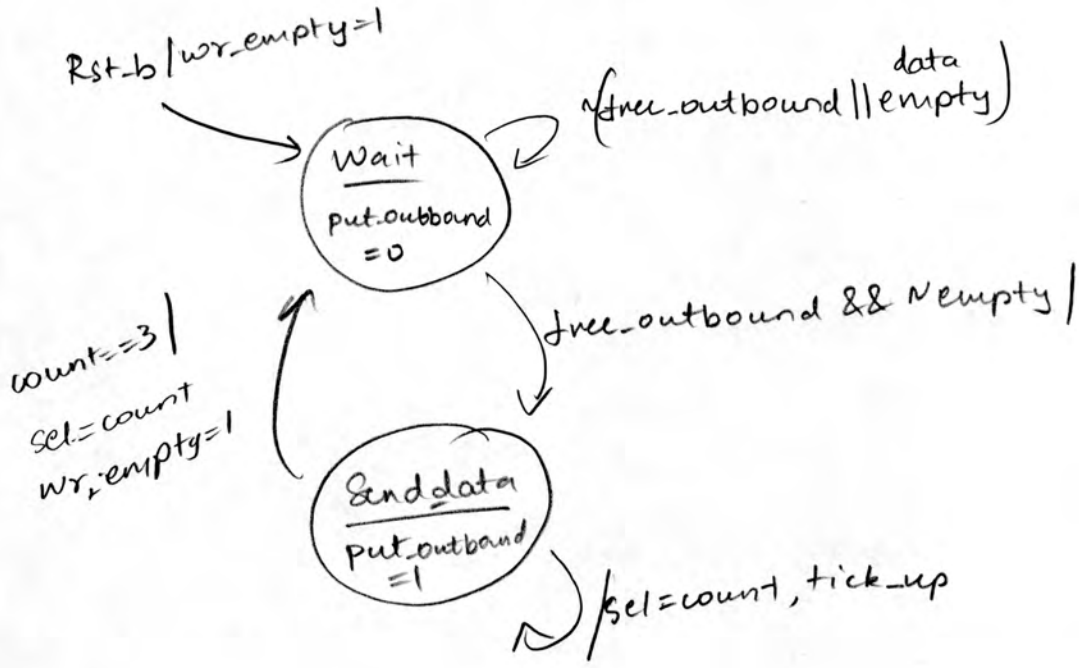
FSM's
Node –Handshakes
Router - Handshakes
Queue
Router

FSM | Handshake router → node |

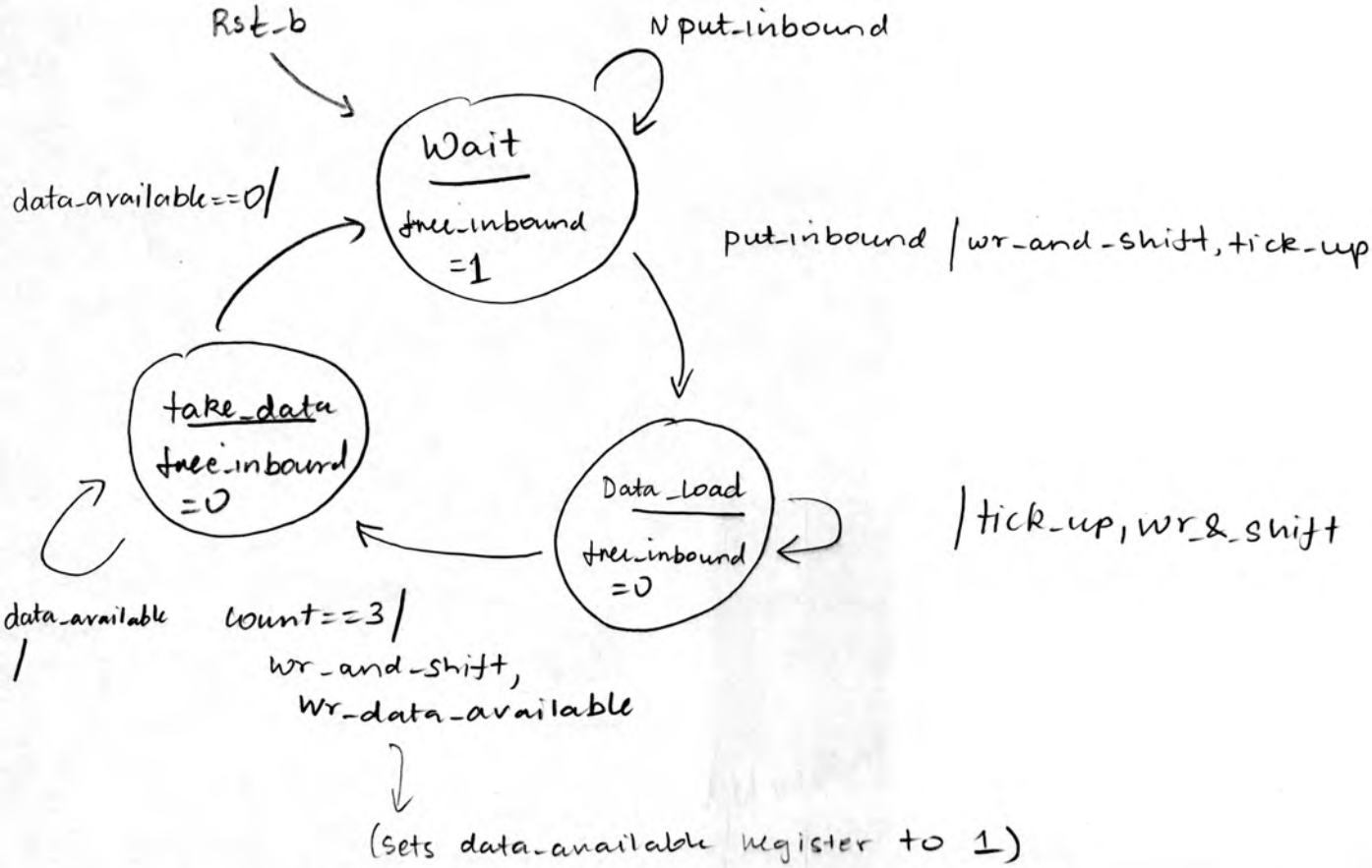
Handshake node → router



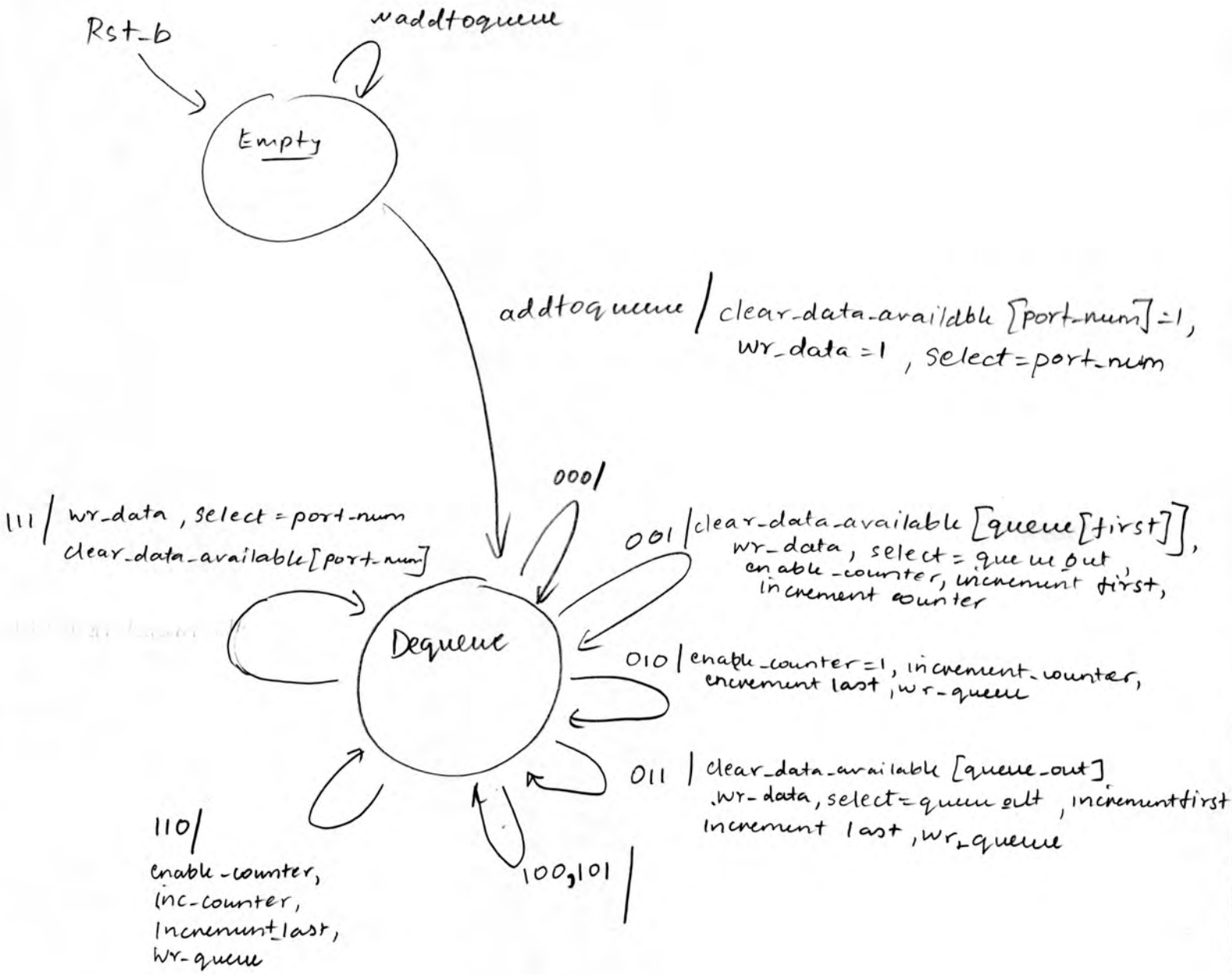
FSM Handshake router → node



FSM Handshake node → router



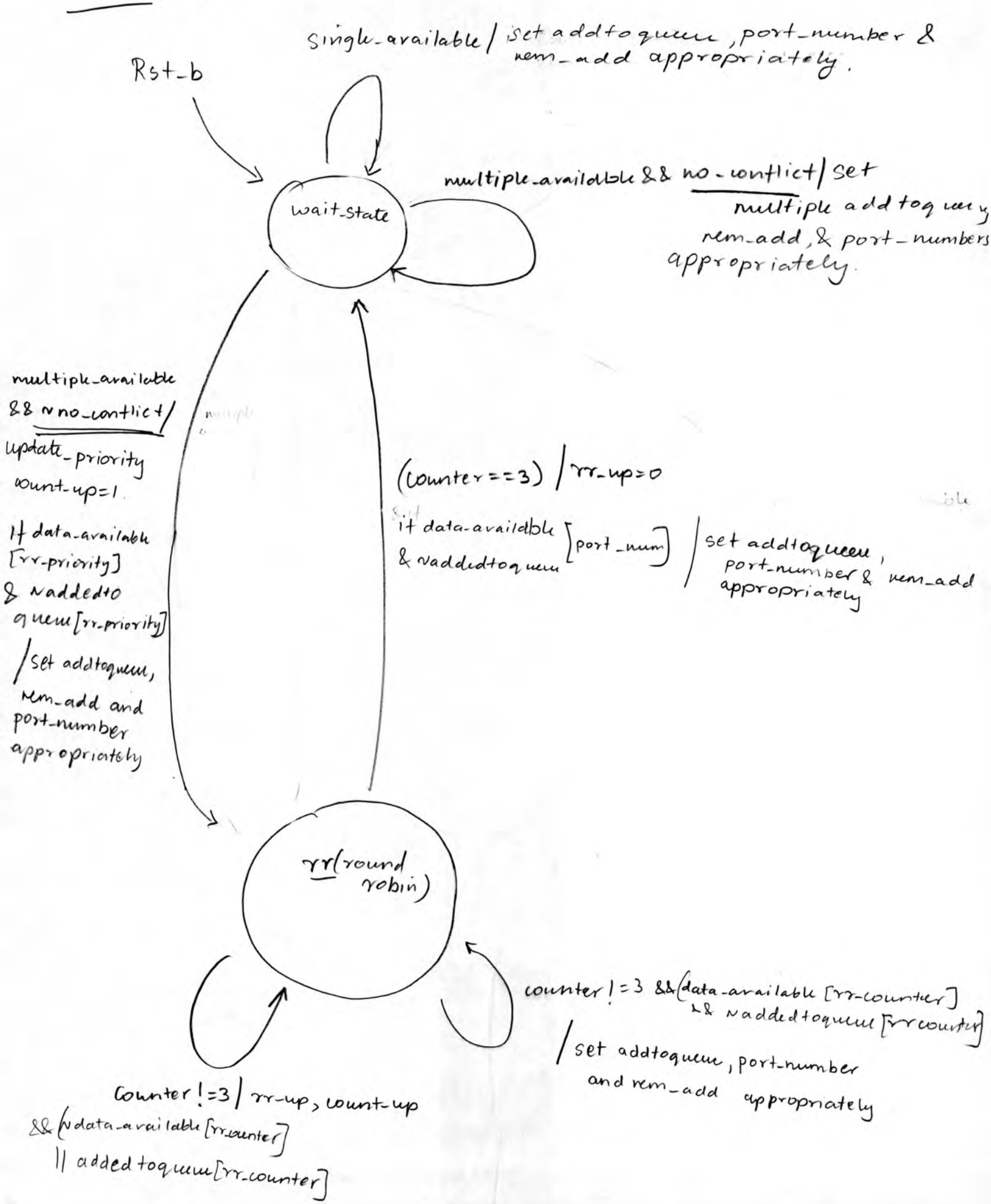
Queue FSM



	input	input
queue_empty	addto queue	data_empty
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

queueempty = (count == 0) ? 1 : 0;

Router FSM


single_available / set addtoqueue, port_number & rem_add appropriately.
Rst_b
wait_state
multiple_available && no_conflict / set multiple addtoqueue, rem_add, & port_numbers appropriately.
multiple_available && !no_conflict / update_priority count_up=1
If data_available [rr_priority] & !addedto queue[rr_priority] / set addtoqueue, rem_add and port_number appropriately
(counter==3) / rr_up=0
if data_available [port_num] & !addedtoqueue / set addtoqueue, port_number & rem_add appropriately
rr(round robin)
counter!=3 && (data_available [rr_counter] && !addedtoqueue [rr counter] / set addtoqueue, port_number and rem_add appropriately
Counter!=3 / rr_up, count_up && (!data_available [rr_counter] || added toqueue[rr_counter]

Testbench

The testbench I wrote first checked whether a packet was successfully sent within a router. Thereafter it checked whether a packet was correctly sent between routers. Then multiple packets were sent from different nodes to different destinations, to check for concurrency. The final check was to send multiple packets to the same destination and check whether no packet was dropped. The FIFO and handshakes were tested individually in the prelab testbench and the queue was tested separately with a different testbench.