Implementation Details FIFO

Anubha Verma **Shikha Kapoor**

Preliminary Report: CS316 under Prof M Balakrishnan
Department of Computer Science & Engineering
Indian Institute of Technology Delhi

1 FIFO Description

A FIFO forms the basic module in many hardware and software designs. The functionality is captured by the name, First in First Out, essentially a queue. The FIFO unit that we are designing has add on features, like sorting facility and priority based queuing in hardware.

2 Operation

The operations of the machine are described in Figure 1. As shown, the FIFO machine is equipped with a synchronous interface for accepting data.

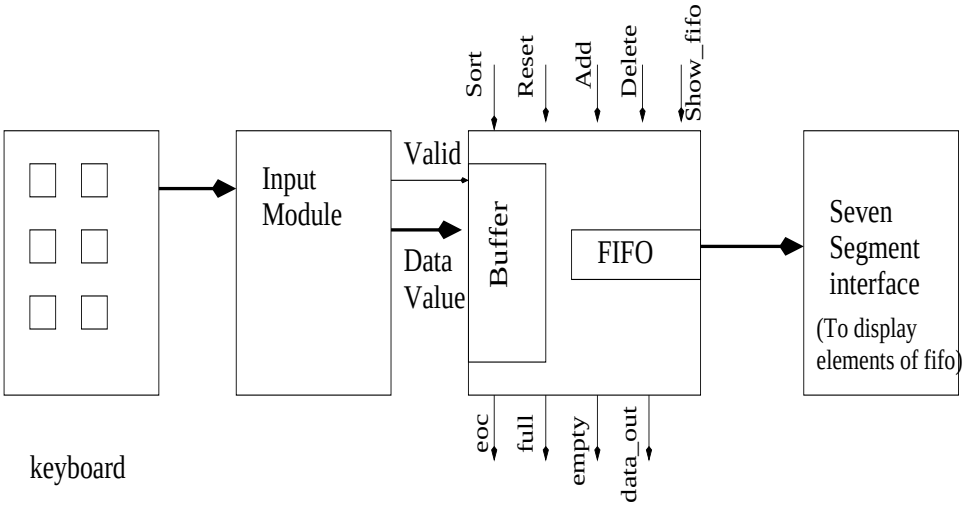

Figure 1: FIFO Overall Design

The I/O specifications are as follows:
Inputs:

1. DATA_IN: It is a 10-bit data bus that contains the data(8bit data + 2bit priority) to be added to the FIFO.
2. ADD: This signal is asserted whenever data is to be added to the FIFO.
3. DELETE: This signal is asserted whenever data is to be deleted from the FIFO. The data that was inserted earliest is the one that will get deleted.

4. RESET: This signal should initially be kept high and should be de-asserted in the next clock cycle in order to start the state machine control.
5. CLK: This is the clock signal that controls the operation of the module and is provided through an OSC4.
6. SORT: This signal is asserted to sort the queue with respect to data values within the same priority.
7. SHOW_QUEUE: This signal is asserted to see all the elements of the queue turn wise starting from head.

Outputs:

1. DATA_OUT: It is a 10BIT data bus that is always available as the head of the queue.
2. FULL: This signal is high whenever the FIFO is full, that is all the 8 numbers have been filled. The data entered hereafter are stored in a buffer
3. EMPTY: This signal is high when the FIFO is empty, that is, no value is present in it.
4. EOC: This signal is high when after the assertion of SORT signal the machine is finished with the computation and is a signal for the user to lower the SORT signal. After SORT becomes low, EOC also becomes low.

FIFO Machine:

- A set of 8 registers(10 bit wide) is used to implement a circular queue.
- Each register's input is taken from a 4 input mux(corresponding to maintain same data, swap with left neighbor, swap with right neighbor, and take data from DATA_IN respectively)
- The elements of the queue have a user defined priority associated with them and is used as explained below
- The mux selects obtained from some combination al logic over state machine outputs.
- Sort function sorts the elements of same priority with respect to data among themselves.
- SHOW_QUEUE displays all elements of queue turn wise.
- Any data entered when the queue is full is buffered in a memory and is fed to the queue when a delete is performed hence.

Priority: Priorities take values 0 to 3. A data with priority 3 is given highest priority, any new data inserted is given place after the last element of equal or greater priority. An element of greater priority is given preference over that of a lower priority only so far as the position in the queue is concerned. It cannot displace an element of lower priority from a full queue.

As of now the buffering is not implemented and any insert when queue is full is ignored, reflecting no change on the queue.

3 Design Steps

The first basic construct of FIFO implemented was a simple FIFO with an insert, delete. Every time INSERT is asserted for a non full queue the data is written to the register currently represented by tail and tail is then incremented. Every time DELETE is asserted for a non-empty queue the head is incremented and so the data is logically removed from the FIFO.

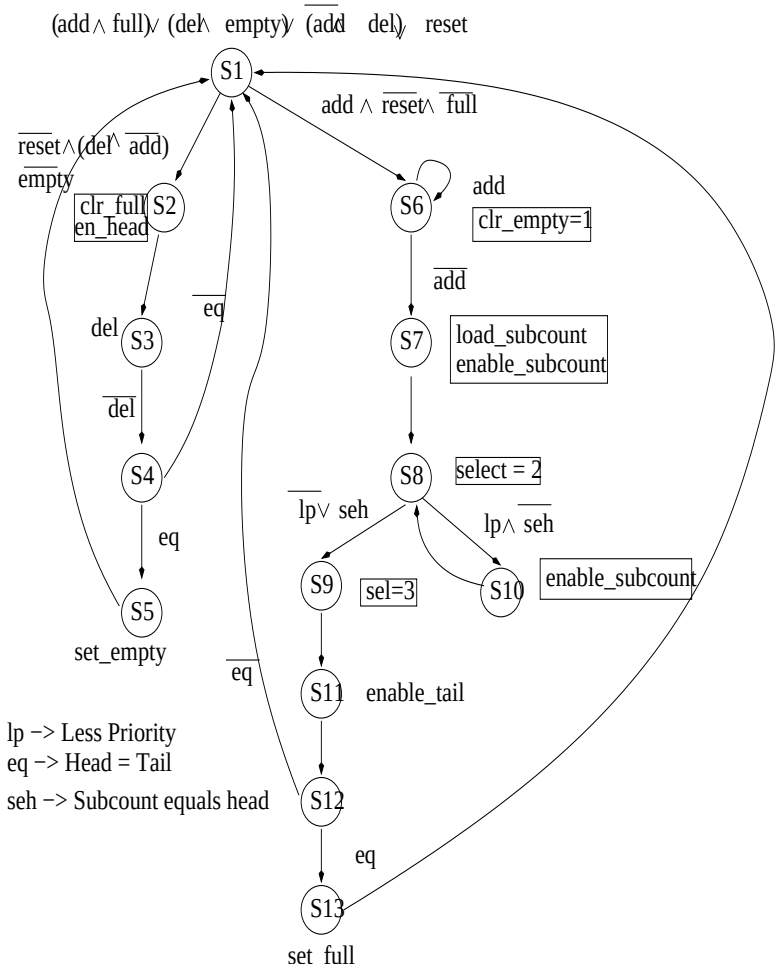

Figure 2: Main State Machine

3.1 Support for Priority

The augmentation to the above machine, a FIFO with support for priority, was implemented. INSERT: Here a counter is initialized to the tail. It is decremented and the elements met in the process are shifted to the right until-

- It reaches an element of equal or higher priority than of the input data or
- If it reaches the head.

  It then inserts the data at that position and tail is incremented.

DELETE: As before the head is incremented. The complete state machine implementing the priority based FIFO is shown in figure 2

Figure 3 shows the simulation for the priority queue.

3.2 Support for Sorting

Head-Tail generation

We are currently working on the sorting module. Sorting on a simple queue means that the data are sorted on the basis of some parameter of the data. Although this is not in accordance with the queue property, but sorting on a data set is a commonly required function. Consider an airport queue, which is generally first in first out, but in case of need, it is sorted in accordance with the time left for the respective flights to depart !

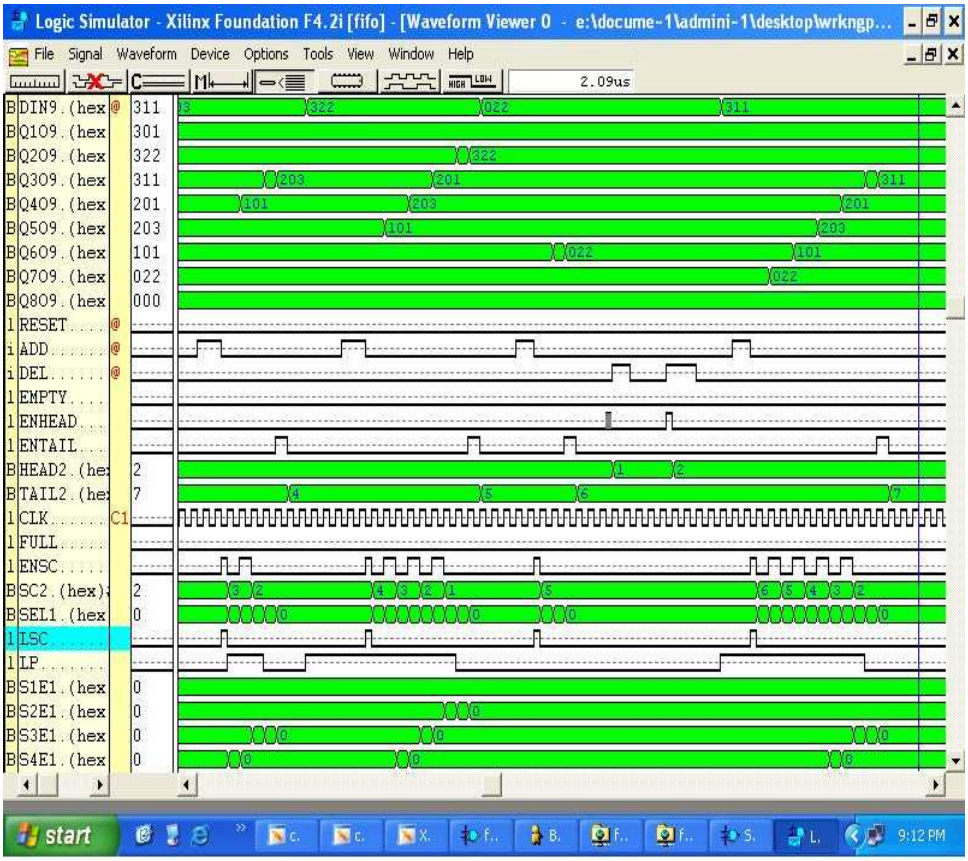

Figure 3: Simulation of Priority Queue

However, with the priority concept, sorting is not straightforward. Since the queue consists of elements in decreasing order of priority, each set of priority can be thought of as a queue and sorted. This is what we will implement.

4 Progress so far

So far priority queues are functioning. Figure 3 shows the simulation results for the same. We are still working on implementation and testing of sorting. We plan to provide support for a buffer which is placed before the FIFO and stores the incoming elements for the case when the FIFO is full.
Some additional features like consideration of age-based priority, for the elements which have been waiting in the buffer, will be supported if time permits.