CS 755 - Spring 1999
Project Description
Design and VLSI Implementation of a 4-port Crossbar-Switch based on Virtual-Output-Queueing

1 Functional Specification and Constraints

You are to design and implement down to the layout level a 4x4 crossbar switch for fixed-size packets. The switch will use input-queueing, but in order to avoid the Head-Of-Line (HOL) blocking effect it will support Virtual-Output-Queueing, i.e., a separate packet queue for each output port at each input port. Two packet priorities (low and high) will be supported.

The 12-bit packet format is shown in Figure 1. The packet fields are:

- **R:** The packet's route, which for simplicity, is assumed to always be three hops. The route is determined at the packet's source (what is referred to as *source-routing*), and it consists of a sequence of three 2-bit tags, one for each hop. Each tag specifies an output port (0 to 3) for the corresponding hop. After each switching point, the field R is rotated to the left by two bits, so that the output port appears always at the first two bits of the header.
- **P:** The priority of the packet (0 for low, 1 for high). In the basic specification of the project, a high-priority packet should be always transmitted before a low-priority packet, if both packets compete for the same output port.
- **C:** The check bit of the packet, which results from an even-parity generator of the 12 packet bits. The switch should verify the correctness of the check bit upon the arrival of the packet, and if this bit is incorrect, the packet should be dropped.
- **Data:** The packet payload.

The basic switch model is shown in Figure 2. Note that each input port has 8 FIFO queues, denoted as *O0-L, O0-H, O1-L, O1-H, O2-L, O2-H, O3-L, O3-H*. The notation 'Ox-P' means:

0	1	2	3	4	5	6	7	8	9	10	11
		R				P	C	DATA			

Figure 1: The packet format.

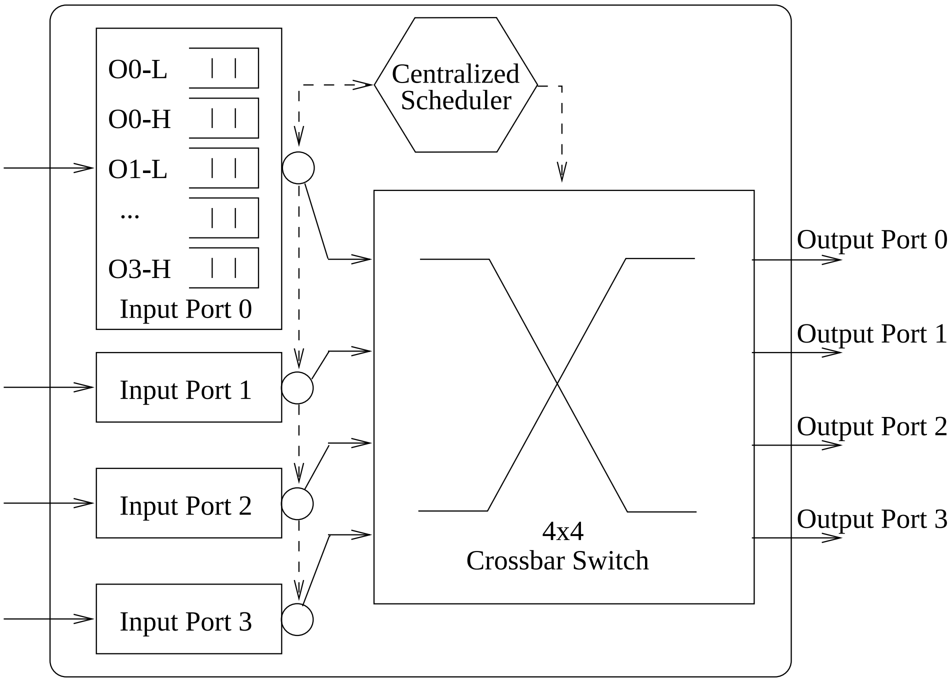

Figure 2: The basic switch model. The notation 'Ox-P' means: Output port x, priority class P.

Output port x, priority class P. Consequently, the switch will totally have 32 queues, 8 for each input port.

Note that the input queues are only logical queues; in the actual implementation you have to design the 8 queues of each input port as a shared pool of packet buffers. Such a pool of buffers is better than dedicated buffers for each logical queue, because it better utilizes the port memory resources. Each input port should provide 16 packet buffers only, in order to keep the transistor-budget of the project low.

2 Performance Specification and Constraints

The clock frequency will not be specified, although the designers should attempt to maximize it. You can achieve this with any combination of architectural, logic-design, circuit-design, or layout techniques you want.

A performance constraint, though, is that the switch should be able to achieve the peak throughput of 4 packets per cycle whenever possible, and whenever this does not conflict with the priority constraint. In other words, if there are four packets in the four input ports, respectively, destined to the four different output ports, the switch should be able to deliver all these packets in the same cycle. This constraint, however, is weaker than the constraint of servicing a high priority packet before servicing any low priority packets.

3 Scheduler Specification and Constraints

Note that in each clock cycle only one input port can be connected to an output port, and also, in each clock cycle only one input queue of an input port can be serviced. The switch scheduler determines at each clock cycle the configuration of the switch (which input port is connected to each output port), and also it determines which queue of each input port will transmit a packet during that cycle.

In order to simplify the project, the scheduler can be designed as centralized, i.e., it knows the current status of all input queues (keep this simplification in mind when you read related research papers).

The exact algorithm and architecture of the switch scheduler is part of your work in this project. The constraint that you have to follow, however, is that the scheduler should be *fair*, i.e., it should not let some input queues (of the same priority level) starve, because some other queues are always serviced first. The exact interpretation of the fairness constraint, however, is not specified here.

4 Input and Output Interfaces

To simplify the project, you can assume that packets arrive at each input port through an 12-bit parallel interface, which is synchronous with the rest of the switch. In other words, the clock signals for both the input interfaces and the switch core are the same, without any phase difference.

The output ports can also be assumed to be 12-bit parallel interfaces, without providing latching of the output data.

An additional control signal at each input and output port identifies if there is a packet received/transmitted during that cycle.

5 Appetizers and Desserts

In the following we give some ideas for possible project extensions. You may need to do something like this if your group is of larger size:

1. **Spatial Priorities:** Occassionally all packet buffers may be full, and so some packets will need to be dropped. In the simplest scheme you can just drop the incoming packet. Interesting alternatives would be to drop low-priority packets first, or to have drop-thresholds for each priority class.

2. **Dynamic Priorities:** In order to avoid the long-term starvation of the low-priority class, you can implement a time-varying priority scheme in which the priority of a packet increases with its waiting time. This should only occur, though, locally in each hop; in other words the packet will keep its original priority marking as it traverses switches.
3. **Multicasting:** You can include one-to-many (multicasting), or one-to-all (broadcasting) switching. In that case, however, we will need to restructure the packet header for your project, in order to include the additional possibilities for the packet destinations.
4. Whatever else you think it will be challenging and fun. Please talk with the instructors first, though.

6 Group Size, Milestones, Deadlines, and Report

The project should be done in groups of 3 or 4 people, with the understanding that the 4-member groups will have to provide additional functionality, or to strive for higher performance.

There will be three phases in the project design:

- **Milestone-1: April 14**
  1. Determine the scheduler's algorithm, and write a functional simulator in C or C++ which will demonstrate the operation of the switch. The simulator should be written in a way that can be easily used from the class instructors when evaluating the project. The simulator will be used later for testing the correctness of your design, by comparing the output sequences that it produces from arbitrary input packet sequences, with the answers that your design gives.
  2. Complete the logic design of the switch using the library of standard cells *scn08hp*, and verify its correctness using QuickSim simulations.
  3. Identify the design components that limit the switch performance and that you may attempt to optimize in the next phases.
  4. Write a short report, explaining the scheduler algorithm and your logic design.
- **Milestone-2: April 23**
  1. Create the floorplan of the final layout. This includes determining the components of the design that will be made in full-custom (or SDL), and the components that will be made in standard-cells. Also, decide the relative position of the chip ports, and the relative structure of the global interconnects.
  2. If you have decided to optimize some design components with faster circuits, you should have completed the circuit designs and Accusim simulations by this point.
  3. Complete the full-custom layout of a single packet buffer, taking into account that this layout cell will be later used hierarchically to create the shared pool of packet buffers.

4. Write a short report, showing the progress of your work in the phase.

- **Final Deadline: May 5**

  1. Complete the layout of the project. The layout should pass both DRC and LVS checks.
  2. Determine the maximum possible clock frequency, using Accusim simulations on the extracted netlist. Due to the large size of the circuit you may not be able to run many simulations. Consequently, attempt to identify the critical path of the design, and simulate input sequences that excite that path.
  3. Write a report including the results of this last phase of the project development.

A final demonstration of the project will be scheduled for the week starting on May 9. All the group members should be present in the demo.

7 Readings

For a survey of the basics about switches see [1]. To find more on virtual-output-queueing see [2]. For an implementation of such a switch see [3]. For a centralized scheduler in input-queued switches see [4]. Additional references may be given later.

References

[1] H.Ahmadi and W.E.Denzel, "A survey of modern high-performance switching techniques," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 1091–1103, Sept. 1989.

[2] N.McKeown, V.Anantharam, and J.Walrand, "Achieving 100% throughput in an input-queued switch," in *Proceedings IEEE INFOCOM*, Mar. 1996.

[3] N.McKeown, M.Izzard, A.Mekkittikul, W.Ellersick, and M.Horowitz, "The tiny tera: A packet switch core," *IEEE Micro*, pp. 26–33, Jan. 1997.

[4] R.O.LaMaire and D.N.Serpanos, "Two-dimensional round-robin schedulers for packet switches with multiple input queues," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 471–482, Oct. 1994.