California Institute of Technology
Department of Computer Science
Computer Architecture

CS184b, Spring 2005 # Assignment P2: Switching Primitives Monday, April 11

Due: Monday, April 18, 9:00AM

Goals:

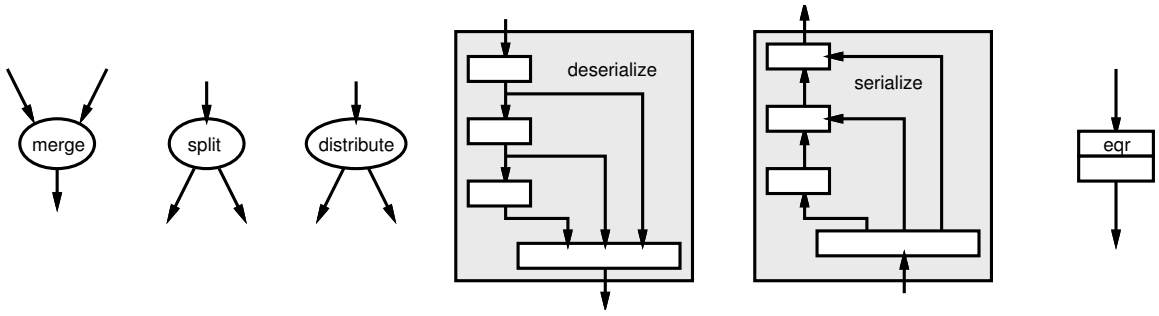
- Build switch primitives which underly switching blocks
- Start collecting data for area model

Collaboration: This assignment is a group assignment. You will need to build 2 flavors of 5 kinds of primitives and 1 flavor of 1 unit for a total of 11 designs. Divide the primitives among team members. Each team member should have primary responsibility for at least two of the primitives. You may turnin multiple implementations of a primitive, but all 10 primitives should be covered by your group.

Team: The class makes up 1 team = {hbarnor, nmehta, nachiket, mwilson, ychao}

Target: All designs should target a Virtex2-6000-4.

Packet Width: We have reduced the packet width to 15b. This is an odd compromise to deal with IO signal limitations on the Virtex2-6000 and the Dishoom board.



Tasks:

1. Build Time Multiplexed and Packet Switched versions of the following primitives:
   - split
   - merge
   - distribute
   - serialize
   - deserialize
   - eqr (expandable queue register) – packet switch case only

   All units should be designed to run at 200MHz.
   For each unit:

   (a) Write Debug/VHDL (turnin your VHDL code)

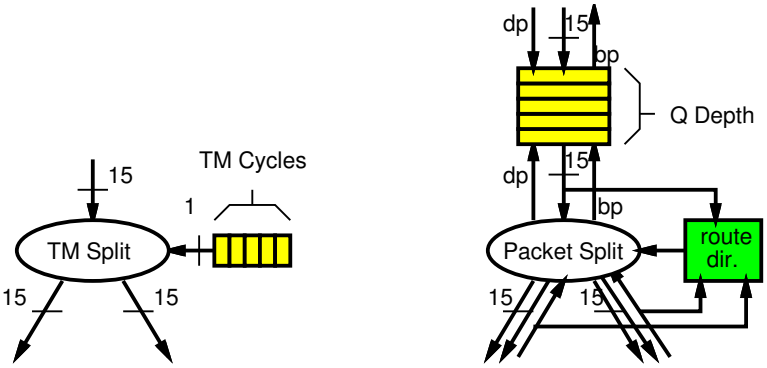
   (b) Verify correct operation (turnin your testing script)
   Use 16b queue depths, 4 word packets, and 16 cycle time-multiplexed memories for testing. Make sure your units stall properly when necessary output locations are full or necessary inputs are not present.

   (c) Synthesize/place/route

   (d) Provide a function for the area of the switching primitive as a function Q-Depth, TM-Cycles, or parallel and serial widths as appropriate. (Include input queues in your design, but not output queues.) For Q-Depth and TM-Cycles it is acceptable to give depths/cycles quantized to depth 16 multiples.

Turnin: We have created a directory: `/scratch/ic/cs184b/project/p2/` Please put the files requested above in that directory (but keep your a master copy elsewhere, that directory is not backed up). Create a master file `p2.html` in that directory which points to all the constituent files and provides any necessary explanations. An html template is at: `/cs/courses/cs184/spring2005/assign/p2/p2.html`.
Email `mdel@cs.caltech.edu` when the materials are in the directory and ready for review.

Split



Input: One 15b data stream
Outputs: Two 15b data streams
Operation:

- Based on the Time-Multiplexed Instruction Control or the Packet Routing Information, send the packet to one of the two output port.

The Time Multiplexed version will simply use a local memory to tell it which way to switch on each cycle. Include the local memory in your design. The memory contents should be a ROM which you should instantiate and set using the ROM16X1 Xilinx component. This is described in the Xilinx manual:
`/cs/research/ic/software/xilinx6_1/doc/usenglish/books/manuals.pdf`
Look on page 3 of the manual for a link to the Libraries Guide whose table of contents lists Xilinx components. The value of the ROM is set literally in the VHDL file, which should be set reasonably for testing thoroughness.

The Packet Switched version will look at the lead word of any present incoming packets and decide which way to route the packet. Once the header goes a given direction, the packet payload follows. Packet lengths will be static for an application, but vary between applications. (*e.g.* SMVM packet lengths may be 6 15b words, while ConceptNet packets may be 4 15b words, both including the header/destination word.) In some cases, the packet may be able to go either direction; here the route decision may want to look at the queue depth (or, fullness, or some threshold) along each outgoing direction.

In a mesh-type route structure, the route decision may compare the destination X and Y address in the packet header to determine suitable direction(s). In a tree-type route structure, the route decision may look at particular bits of the input. *For this assignment, test with a splitter that looks at bits 14:12 of the packet header and sends the packet left if those bits represent a value greater than 3 and right otherwise.* This will need to be parameterized; later versions may need to look at multiple fields in the packet header and make more complex decisions.

Distribute

Input: One 15b data stream
Outputs: Two 15b data streams
Operation:

- Based on the Time-Multiplexed Instruction Control or the Output Queue Fullness, send the packet to one of the two output port.

The difference between Split and Distribute is that both outputs for the Distribute go in the same logical direction. Packets should be routed to an available output to minimize congestion. Preference may be given based on queue fullness and/or randomly. As in the Split, once the Packet Switch Distribute selects an output, the whole packet is routed to the output before it looks at the next packet.

For Distribute, Merge, and Split to determine queue fullnes, your queue will need to be modified to output its length; this may be as simple as making a (registered?) version of the internal queue length an output signal.

Merge

Input: Two 15b data streams
Outputs: One 15b data stream
Operation:

- Based on the Time-Multiplexed Instruction Control or the Input Queue Fullness, send one of the inputs to the output port.

This is elaborated into Time Multiplexed and Packet Switched versions similar to the Split unit. In the Packed Switched case, if both inputs have present data, the input preference may be random or look at the fullness of the two input queues. As in the Split, once the Packet Switch Merge selects an output, the whole packet is routed to the output before it looks at the next packet.

Deserialize

Input: One 15b data stream
Outputs: Wide data streams n times the width of the input **Operation:**

- Collect up n inputs and present them to the output.

The Time Multiplexed version does not need flow control.

The Packet Switched version should use data presence to properly clock the serial data into the wide word. It should maintain proper flow control on both sides of the interface.

Serialize

Input: Wide data stream n times the width of the output
Outputs: One 15b data stream **Operation:**

- Send the wide word input out over n cycles.

The Time Multiplexed version does not need flow control.

The Packet Switched version should use data presence to properly clock the wide word data into the serializer. It should maintain proper flow control on both sides of the interface.

EQR (Expandable Queue Register)

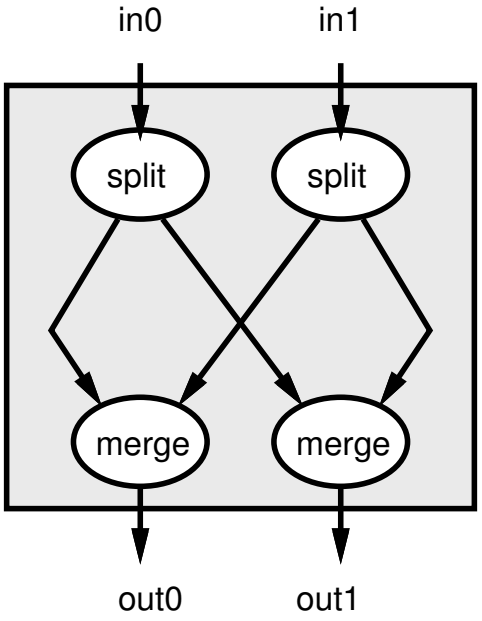
Input: One 15b data streams
Outputs: One 15b data stream
Operation:

- This is a depth 2 queue to allow pipeling of backpressure control (see Day 6 lecture).

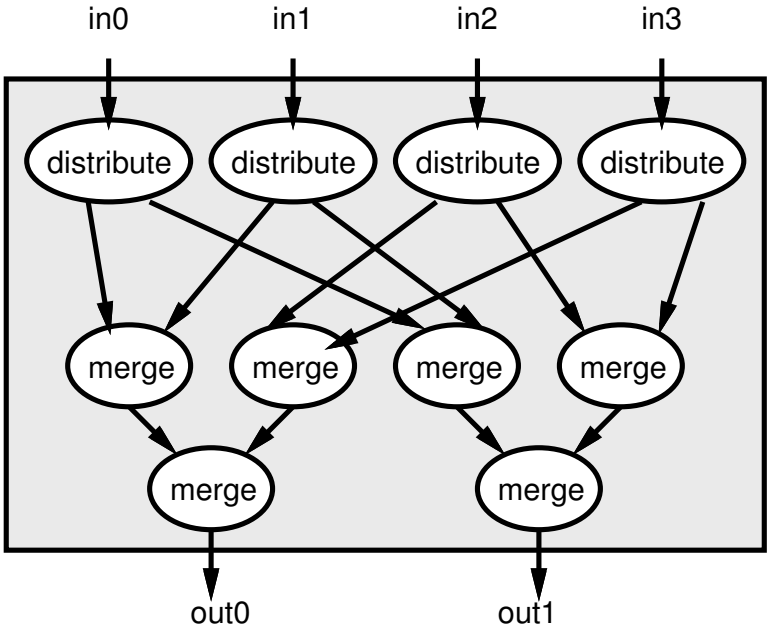
This design is only needed for the Packet Switched Case.

Application

Shown below are examples of building switches from these primitives.



2×2 Switch



(Two outputs in same logical direction)

4×1 dilation=2 Switch