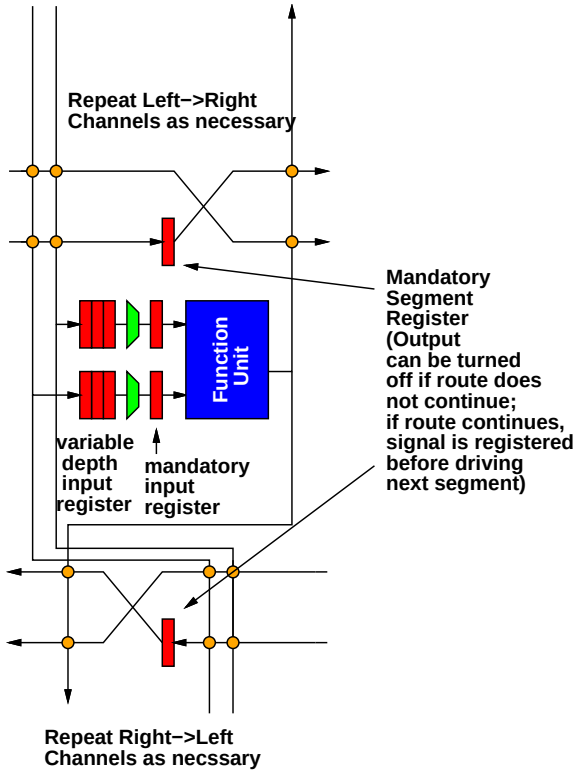
California Institute of Technology
Department of Computer Science
Computer Architecture

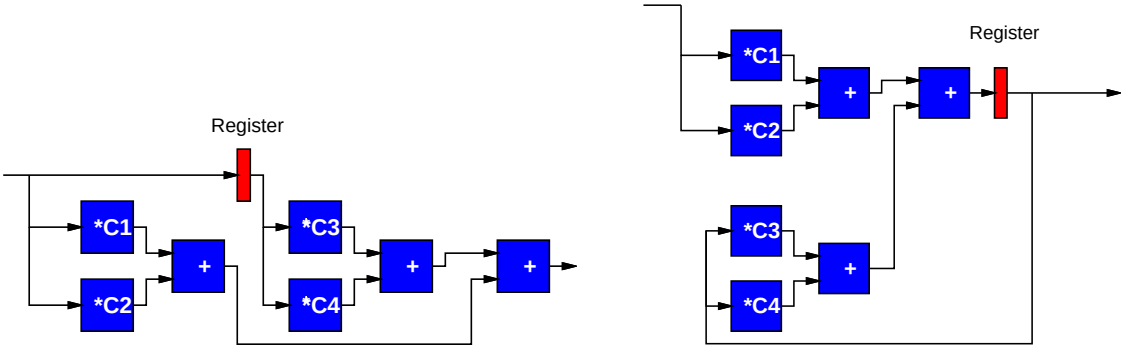
CS184a, Fall 2000 — Assignment 8: Retiming — Monday, November 13

Due: Monday, November 20, 10:30AM

1. Shown below is a single slice of a one-dimensional datapath architecture (tile this slice left to right to realize the 1D datapath). Notably, every functional unit has a mandatory register on its input which is preceded by a variable delay input register. The network consists of length-2 lines and there is a mandatory register at the programmable switch between segments (the funniness with crossover at the bottom maintains the invariant that outputs are driven after the segment register and inputs are consumed before the segment register so that we maintain the length-2 property suggested above).



Consider the two computational graphs shown here:



The following tables gives a mapping of the functions in each graph to the array (a placement):

Position	**Function**	**Comment**
0	input x	assume this is some io operation
1	$\times$C1	
2	$\times$C2	
3	$+$	$x_i \times$C1$+x_i \times$C2
4	$+$	final sum (output is y)
5	$\times$C3	
6	$+$	$x_{i-1} \times$C3$+x_{i-1} \times$C4 OR $y_{i-1} \times$C3$+y_{i-1} \times$C4
7	$\times$C4	
8	output x	also a special io operation

For each graph:

- What is the input retiming depth of each of the inputs to each (used) functional block in order to assure correct operation?
- If the first input arrives on cycle 0, on what cycle should the i-th input arrive for correct operation?
- If the first input arrives on cycle 0, on what cycle should a consumer expect the i-th output?

I expect you might approach this by:

- Generate a modified graph which adds delay blocks to model the mandatory retiming in the placed design.
- Determine what it will take to retime the resulting graph so that it is fully pipelined (pipeline, C-slow?).
- Slide registers to match the existing, mandatory registers and place the balance on the inputs of functional units.
- Read off/summarize the registers per input from your retimed design.

2. Consider a 4-LUT based design for an accumulator and a saturating accumulator. Naively, at least, both of these have cycles of length $c \cdot w + k$, where w is the accumulator width and c and k are constants you could determine for each design.

   - Show that the accumulator can be organized so that it can be fully pipelined at the single LUT delay level. [hint: this can use the same "redundant" representation idea as your low-latency multiplier—if it's not clear after 10 minutes of thinking about it, ask me.]
   - How far can you pipeline (retime) the saturating accumulator? (explain) [A standard accumulator says: acc_i=in_i+acc_{i-1}; A saturating accumulator says: {tmp=in_i+acc_{i-1}; if (tmp>MAX) {acc_i=MAX;} else {acc_i=tmp;}}; A typical saturating accumulator will have a MIN and a MAX saturation, but you don't need to add that complication here. You may assume MAX is a constant, including 2^w-1; handling any MAX is not the point of this exercise, just thinking about the implications of having to perform the saturation.]

3. Consider an architecture which uses a variable delay shift register on the output of each physical LUT (logic block) to support retiming. This variable delay shift register will have some maximum programmable length. Note that you can always use a LUT as a buffer when you need to provide more register depth than you can get out of a single logic block. Using the designs from problem 2, how long should the shift registers be to minimize area? Assume that the register area is one tenth the area of the logic block (this number is artificially large to simplify the problem).