California Institute of Technology
Department of Computer Science
Computer Architecture

CS184a, Winter 2005 — Assignment 4: Compute — Monday, January 24

Due: Wednesday, February 2, 9:00AM

For this assignment, you will implement one simple logic function on computing arrays with three different compute blocks.

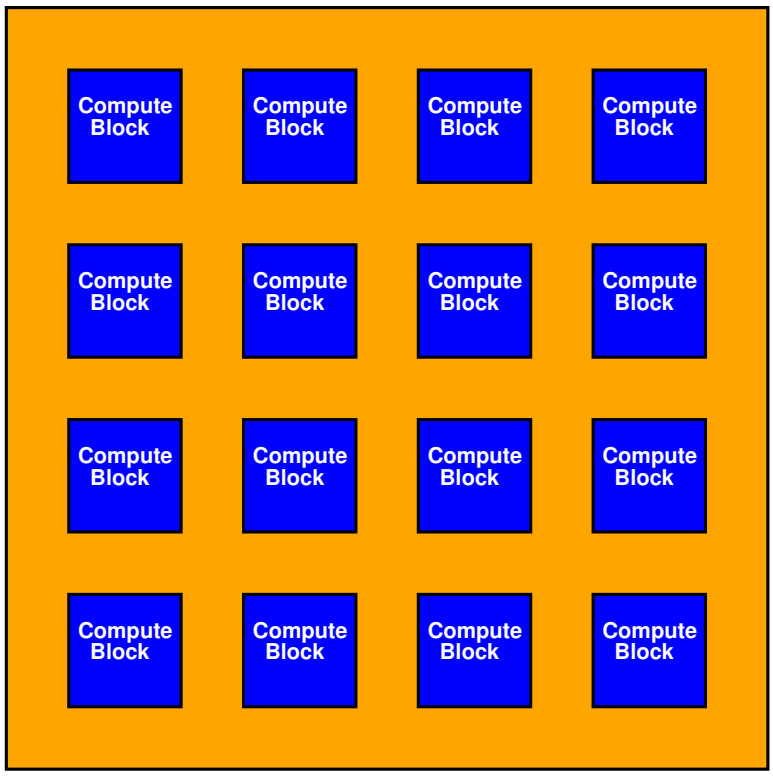
Problem: Decode a Huffman coded input stream. You get one bit per cycle through input b_{in}. Gather up the bits. Each time you recognize a code word, place the decoded word on D_{out}[2:0] and assert the D_{valid} signal for one cycle. Decoding runs infinitely (You may have a reset signal; otherwise, don't worry about beginning or ending the computation.)

(Input) **Code Sequence**	(Output) **Number**
1	0
011	1
010	2
00001	3
0011	4
00000	5
0010	6
0001	7

Starting Exercise: Consider the following: (These are my hints; ignore them at your own risk. A different style of solution may match different compute structures.)

- How would you solve this with an FSM?
- How would you solve this as μcode on a processor?
  - Without using main memory (only register file state, no indirection)
  - Exploiting main memory (possibly using indirect addressing into memory)

For all of the designs, the compute blocks live in an $n \times m$ array as shown:



Please measure all interconnect distances as the *Manhattan distance* between the source and the sink. That is:

$$\text{distance} = |X_{src} - X_{sink}| + |Y_{src} - Y_{sink}| \quad (1)$$

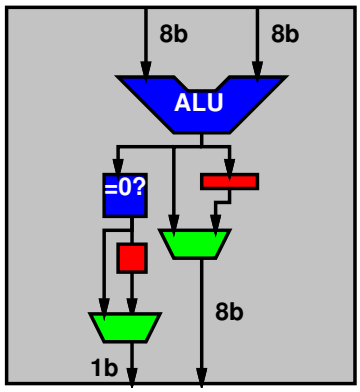
You may assume the interconnect provides pipelining should you need it. (*e.g.* if you are running at a cycle time of 1 delay unit and a wire transit takes 2 delay units, you may assume it can be pipelined within the network.)

For each design, please detail the configuration of each cell used and show layout diagram. For each LUT, you may give the logic equation it implements. Similarly, for a PLA, you may give the equations implemented by each product term and sum term.

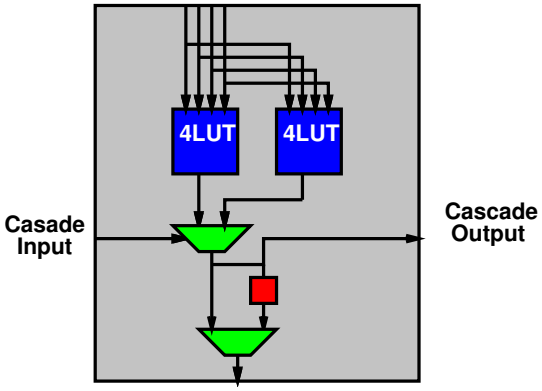
Assume the input comes from one side (you choose where) and the output exits an orthogonal, adjacent side (again, you choose where).

Compute Block Cases:

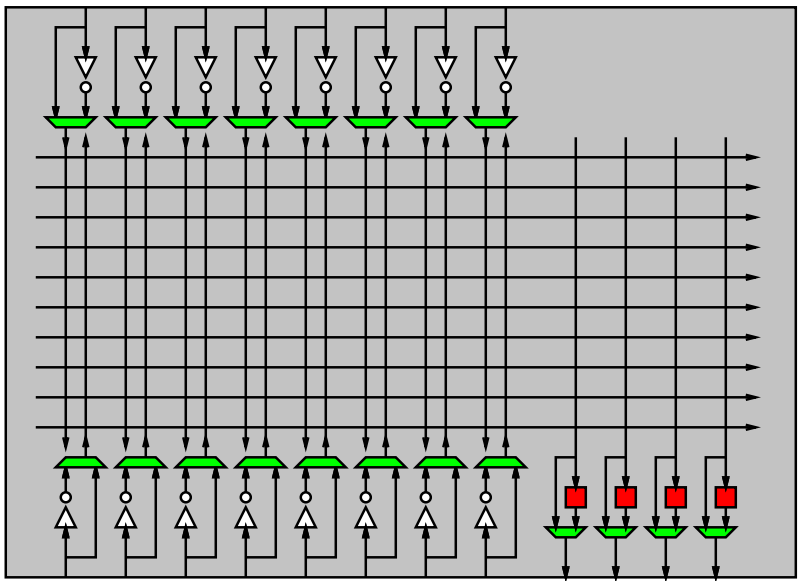
1. **8b-ALU**: The compute block is an 8b-ALU (with opcode functionality as on previous assignments). We add comparison to zero that produces a single bit of output that indicates whether the output of the ALU is a zero. The ALU output and zero-comparison output can be optionally registered. The network is a bit-level network and can route all 9 outputs and 16 inputs independently. In one delay unit, it is possible to perform an ALU operation and travel a Manhattan distance of two or to travel a Manhattan distance of 5 (*e.g.* if the source and sink are Manhattan distance 3 apart it will take 2 delay units; if they are Manhattan distance 7 apart it will still take only 2 delay units).



2. **Cascaded 4-LUT**: The compute block is a 4-LUT with a cascaded fifth input. The output may be a registered or unregistered version of the output of the cascade mux. The cascade travels along the X direction, such that the input comes from the left (X-1) and the output goes to the right (X+1). In one delay unit, it is possible to perform a 4-LUT lookup, a cascade up to length 3, and travel a Manhattan distance of 4. With no logic, it is possible to travel a Manhattan distance of 10 in one delay unit. Programming the two 4-LUTs identically suffices to segment (start/restart) the cascade.



3. **PLA 16×10×4**: The compute block is a 16 input, 10 product term, 4 output AND-OR PLA. Each of the inputs may be optionally inverted. Each of the outputs may be optionally registered. In one delay unit, it is possible to perform a PLA evaluation and travel a Manhattan distance of 2 or to travel a Manhattan distance of 5.



N.B. I think of the 8b-ALU logic block and PLA as being roughly the same size. The 4-LUT cascade logic block is about one-fourth the size of the PLA or 8b-ALU. This is the reason for the different distance metrics.

Summarize Results: Please fill in a table like the following to summarize your results. (Measure latency from arrival of last bit to output of associated code word.)

Compute Block	Clock Cycle	Latency	Blocks Used	Minimum Rectangle Enclosing Design
8b ALU				
4-LUT Cascade				
PLA 16×10×4				

e.g.

Compute Block	Clock Cycle	Latency	Blocks Used	Minimum Rectangle Enclosing Design
4-LUT Cascade	1	3	21	5×5