University of Pennsylvania
Department of Electrical and System Engineering
Computer Organization

ESE534, Spring 2012

Assignment 6: Compute

February 20

Due: Problem 1 and 2: Friday, March 2, 12:00PM
Due: Problem 3 and 4: Wednesday, March 14, 12:00PM

For this assignment, you will implement one simple logic function on computing arrays with three different compute blocks. The goal of this assignment is to give you a feel for the strengths and weaknesses of these different forms of programmable compute blocks.

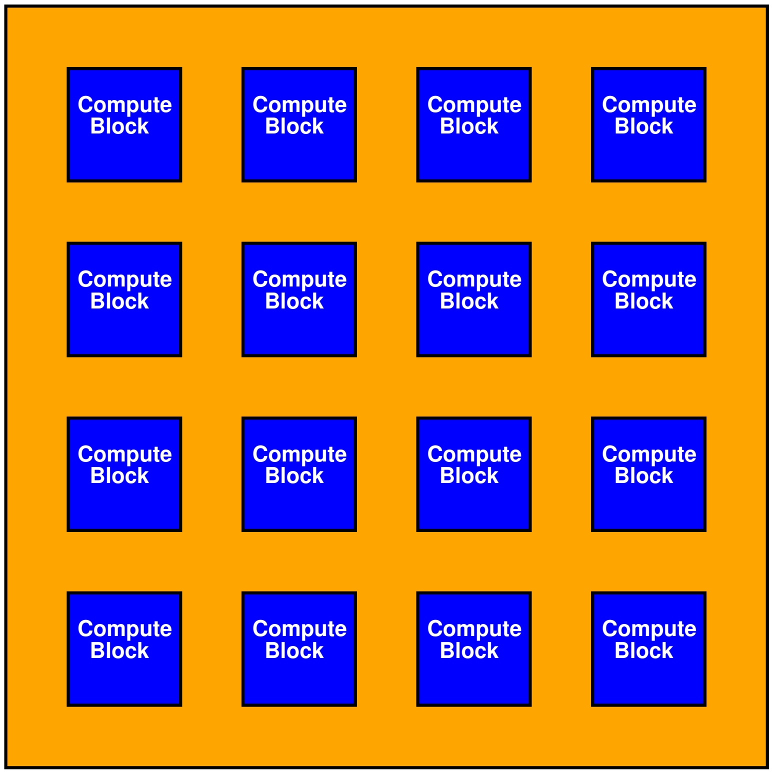
Problem: Perform a saturated addition on a pair of 5b-signed (two's complement – value range is [15:-16]) integers producing a 5b-signed result. Take the saturation bounds (maxval, minval) as inputs.

```
SATADD(A,B,maxval,minval)
  tmp=A+B // tmp will be a 6b-signed integer
  if ((A+B)>maxval)
    sum=maxval
  else if ((A+B)<minval)
    sum=minval
  else
    sum=A+B
```

One reason to use saturated arithmetic is to avoid the aliasing that occurs when the result is outside of the range supported by a limited datapath width. For example, if this were simply a 5b-ALU datapath and you added 10 to 10 (each encoded as 01010), the sum would be 10100, or -12. If, instead, we took maxval=15 and performed saturated addition, then SATADD(10,10,15,-16)=15. In some applications (particularly DSP applications), it is more useful to simply saturate the sum to a chosen maximum value like this than for the result to wrap around to a negative number.

- A different style of solution may match different compute structures.
- Try to minimize compute block count and evaluation latency in your implementations.

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:

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

For each design, please detail the configuration of each cell used and provide a 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.

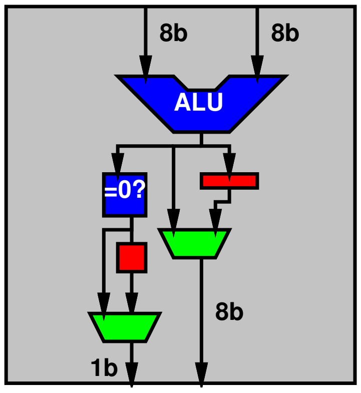
Assume the input comes from one side (you choose where) and the output exits the opposite side.

Compute Block Cases:

1. **8b ALU**: The compute block is an 8b ALU (with opcode functionality as on HW4). 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. For example, this allows you to route bit 5 from your inputs to bits 5, 6, 7, and 8 of the ALU to perform sign extension. The ALU is statically configured to perform a single operation.

Delay model:

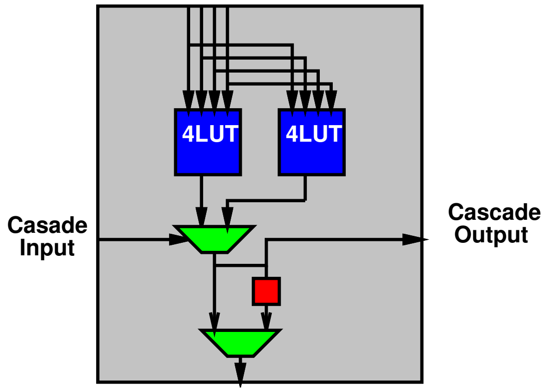
ALU evaluation (including zero compare)	0.5
Final (flip-flop) mux	0.1
Interconnect	0.2 × Manhattan distance



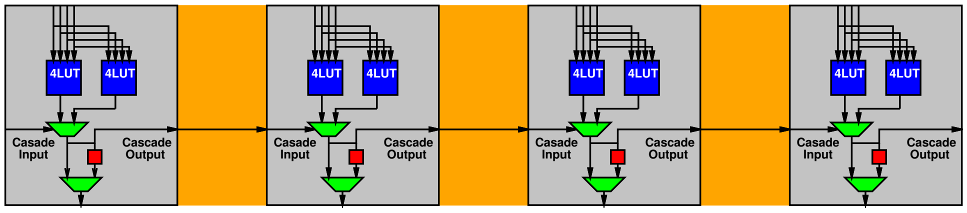
2. **Cascaded 4-LUT**: The compute block is a 4-LUT with a cascaded fifth input. The output may be a registered or unregistered version of the output of the cascade mux. The cascade travels along the X direction, such that the input comes from the left (X-1) and the output goes to the right (X+1).

Delay model:

4-LUT prior to 2-input cascade mux	0.2
2-input mux delay from cascade input	0.1
Final (flip-flop) mux	0.1
Interconnect	0.1 × Manhattan distance



The cascade input is hardwired so that it is generated only by the 4-LUT on the left:

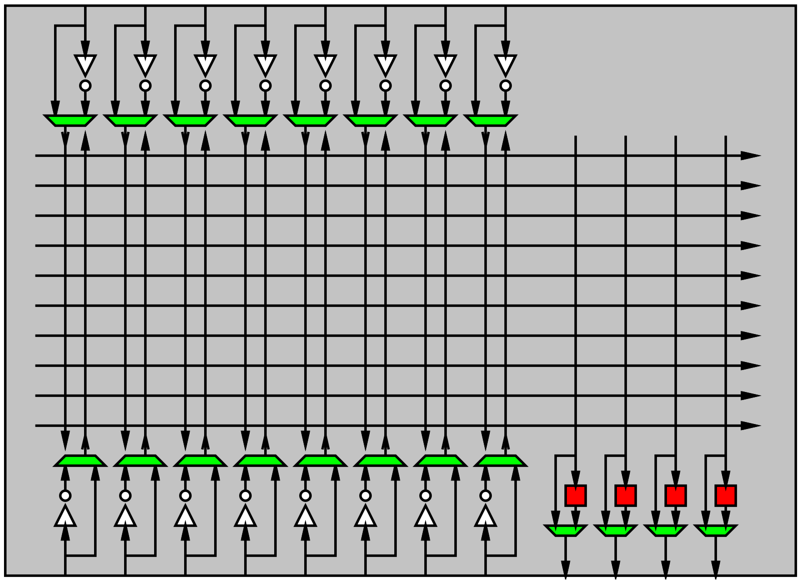


Since the cascade is hardwired, the only way to use the cascade input is as the output of the previous block—so if you just want to use the cascade to implement a 5-LUT, you will need to use two compute blocks.

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.

Delay model:

PLA Evaluation (both planes)	0.6
Final (flip-flop) mux	0.1
Interconnect	0.2 × Manhattan distance



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.

4. **Summarize Results:** Please fill in a table like the following to summarize your results.

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

e.g.

Compute Block	Latency	Blocks Used	Minimum Rectangle Enclosing Design
4-LUT Cascade	10	33	6×6