CS294-7: Reconfigurable Computing

Day 2: Thursday, January 23

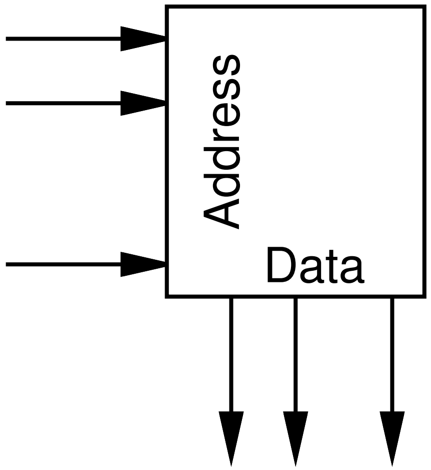
- Computing Requirements
- VLSI Scaling

Computing Requirements

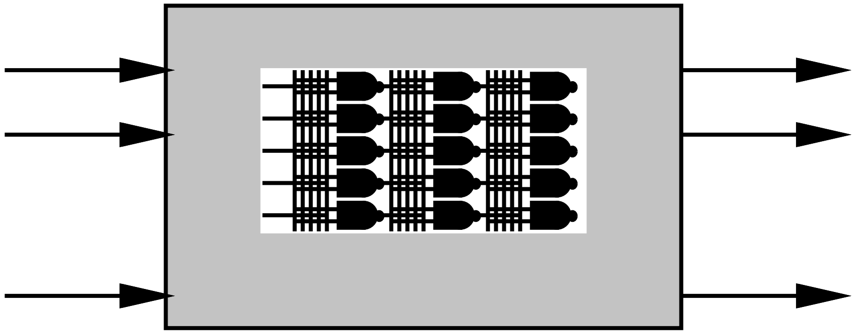
What basic elements do I need to compute?
(configure/program a late-bound computation)

What's wrong with?

One big lookup table?



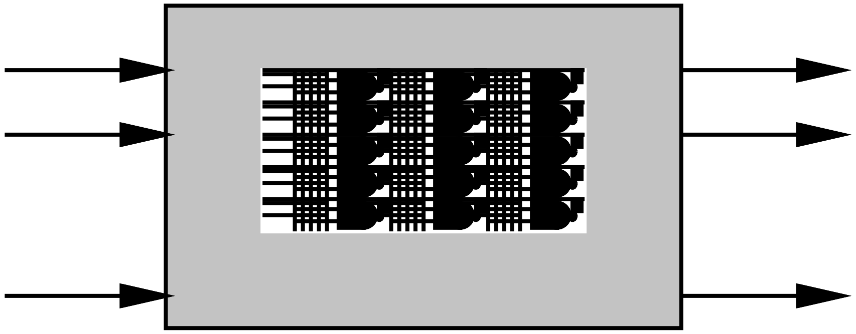
One big block of programmable, combinational logic?

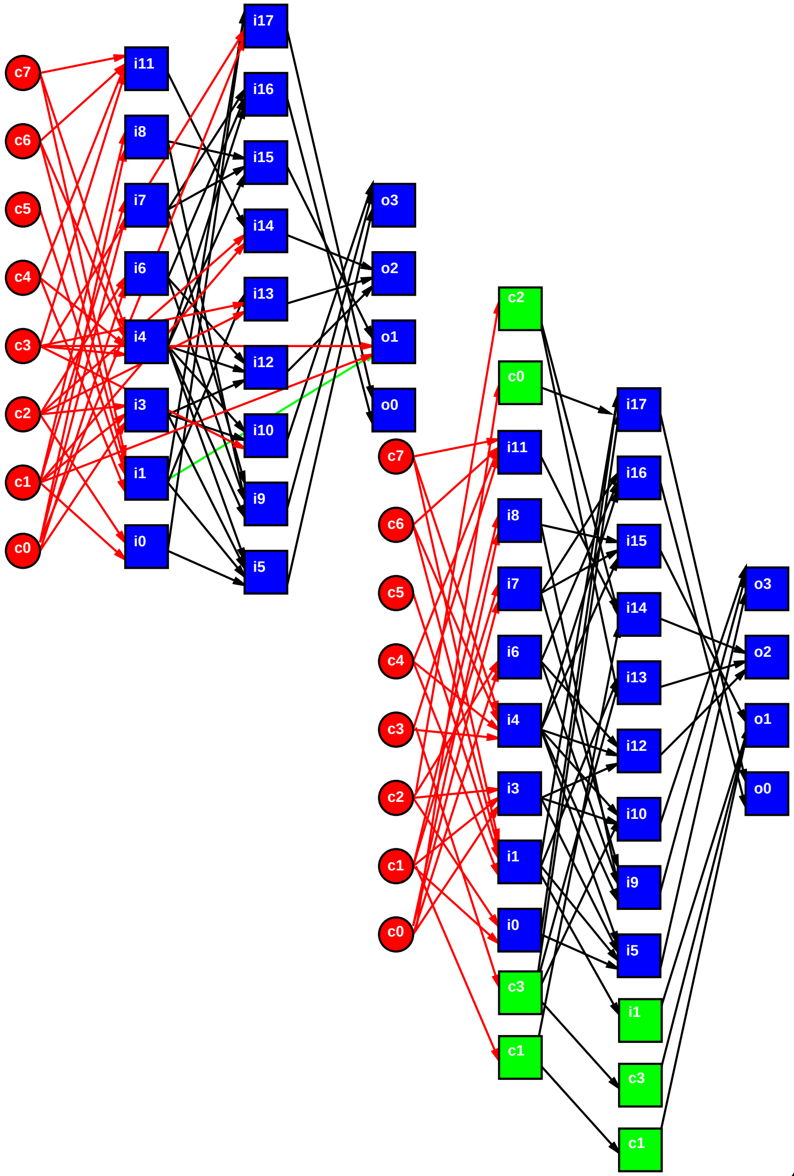


Probably want to reuse circuitry...

Add registers for:

- pipelining
- communicate data between cycles/iterations

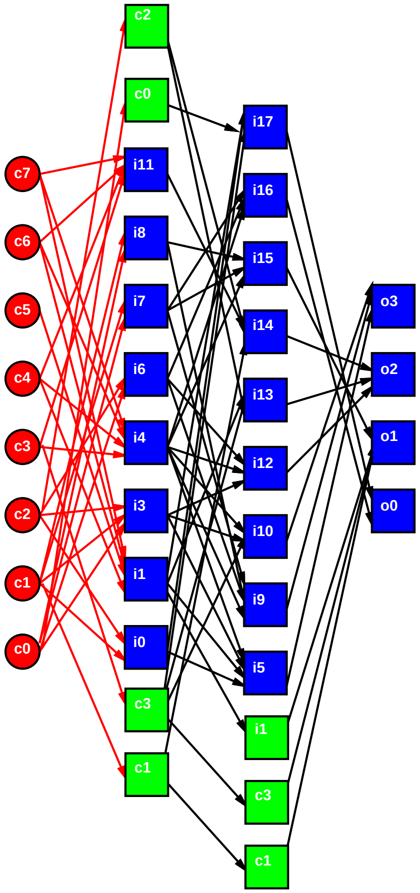




Run at gate+interconnect?

Why not?

- Critical path cycles force sequentialization
- Limited throughput requirement



Consequences:

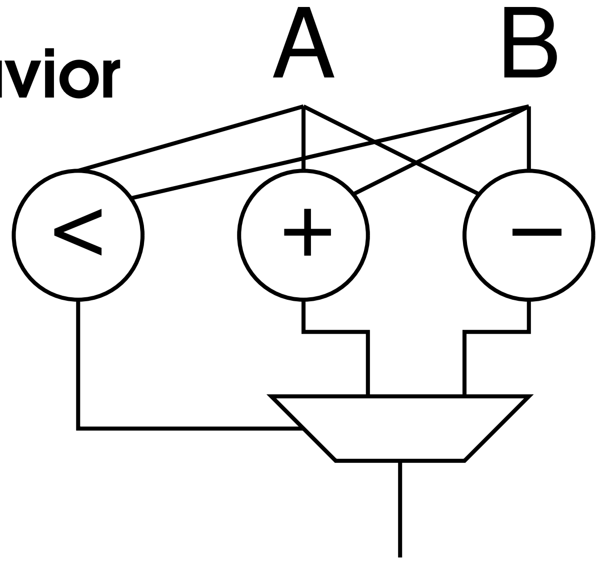
- can reuse hardware
  (fold in time)
- need to be able to change behavior
  (change instruction)
- must retime data to arrive
  when computation active
  (registers, memory)
- less area because:
  A(instruction+retimed data) $<$ A(interconnect+gate)

More Sharing

Mutually exclusive, data dependent computations:

Consequences:

- can share hardware
- need to be able to change behavior (change instruction)
- selection by data (control)



Examples: FSM, FSMD

Folding extreme

"One-bit" processor is extreme of folding

- one gate
- RF+memory for data retiming
- throughput $\propto$ (#gates)$^{-1}$

Reconfigurable Computing Requirements (1 of 2)

Basic Needs:

- Active compute
- Active interconnect

Application gives rise to (relative) throughput rate

To use computational area efficiently:

- reuse primitive elements on different data
- reuse primitives for different computations

Reconfigurable Computing Requirements (2 of 2)

Reuse gives rise to need to:

- retime data so appropriate set arrives when needed
- change instructions

We can store data and instructions more compactly in memory than active computations – two uses for memory:

- Hold non-active instructions
- Long term data retiming

What's Problematic about Gate Counts?

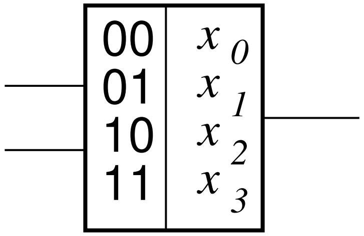
FPGAs are typically rated by a gate count.

...but gates aren't the only commodity in a computation.

...and gate counting itself is tricky.

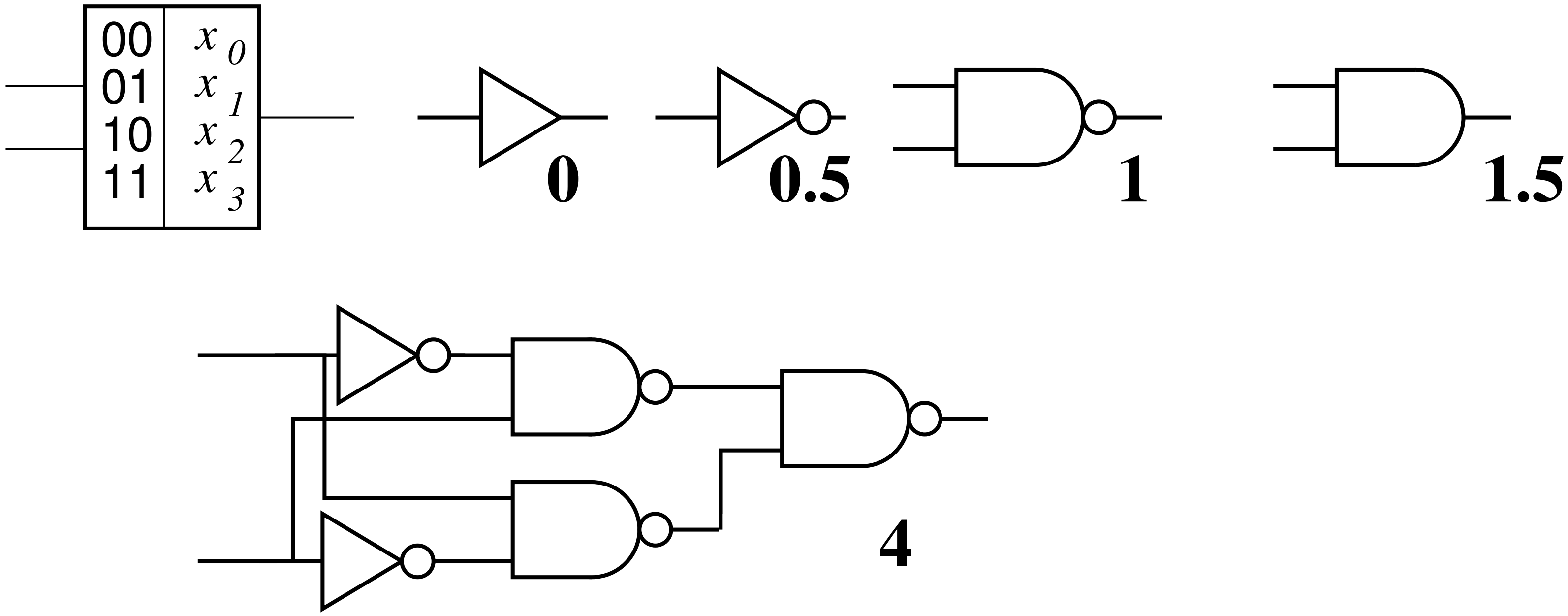
Gates per Module

How many gates in a 2-LUT?



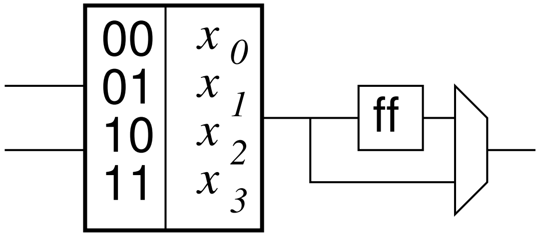
Gates per Module

How many gates in a 2-LUT?



Gates per Module

How many gates in a 2-LUT + Flip-flop?

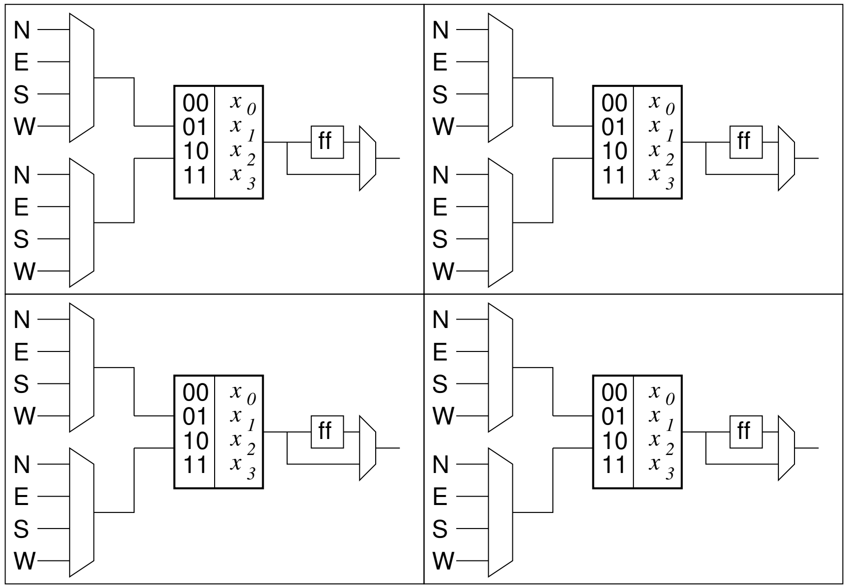


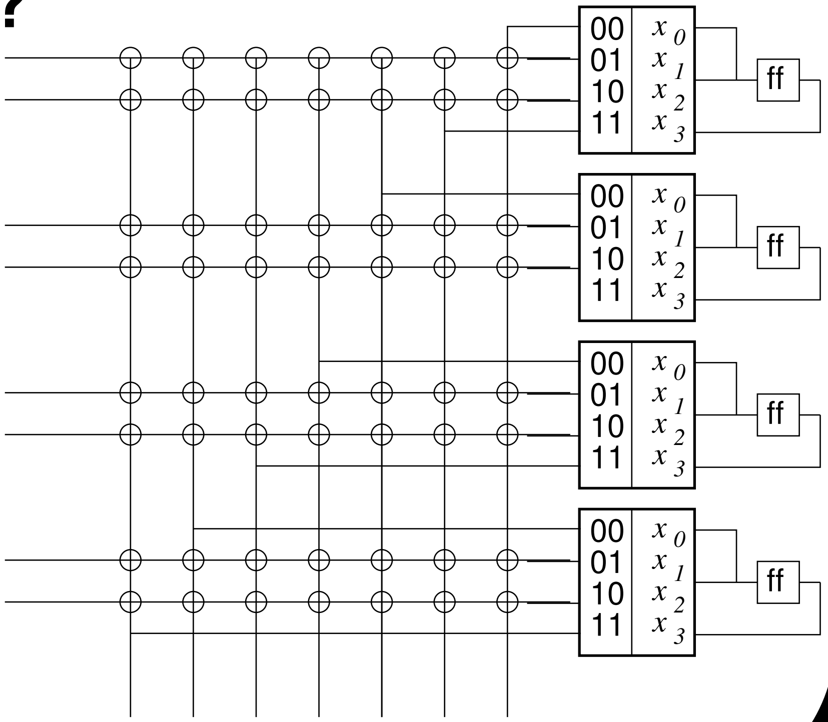
FF $\approx$ 6+ NAND gates

Interconnect

Which has:

- more gates/unit area?
- more yielded gates/unit area?
- more predictable gate yield?



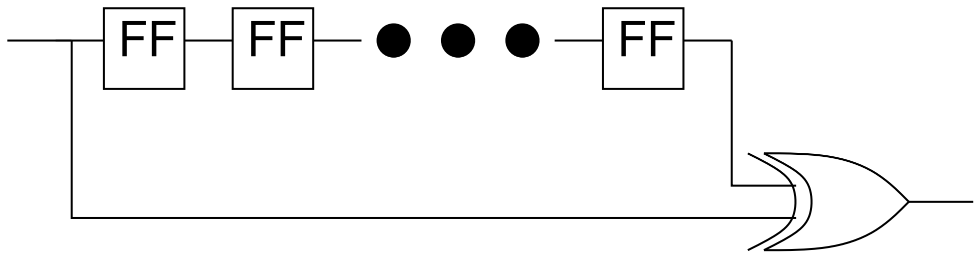


Retiming

$$B_i = A_i \oplus A_{i-3}$$

$$B_i = A_i \oplus A_{i-2000}$$

- How many gates?
- Gates consumed on an XC6200?
- If can use LUT memory for data storage (ala. XC4000)?
- How would you do it?



Gate Counting

Three basic components to computation:

1. compute element
2. interconnect
3. retiming (memory, ff)

Any RC architecture provides them in fixed ratio.

Rating one component can be misleading.
(unless that one component happens to be the limiting one for a design under consideration)