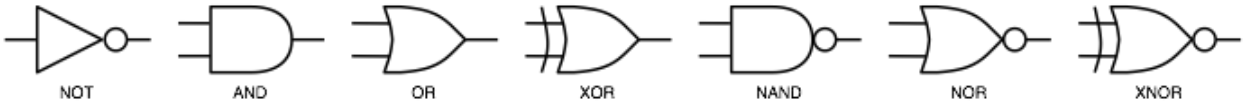
CS 61C Spring 2010 — Combinational Logic & State

Michael Greenbaum (cs61c-tf)
adopted from Scott Beamer

Week 9 (3/16)

Logic Gates



- Looking at what XNOR does, can you think of another name for it?
- How many different two-input logic gates are possible? How many n-input logic gates?
- Build NOT, AND, OR, and XOR using only NAND. To save yourself writing, once you have built a gate, you can re-use it.

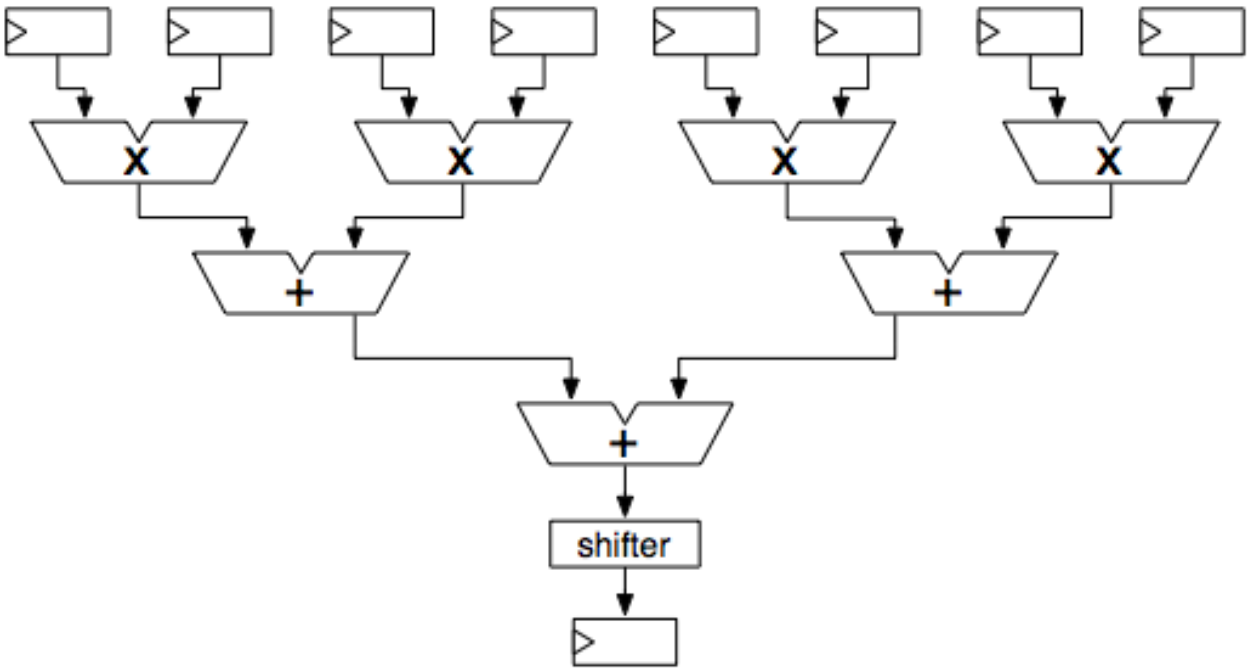
Pipelining Review

. *Recall*: Minimum clock period = $t_{clk\text{-}to\text{-}q} + t_{CL} + t_{setup}$
- Usually t_{CL} dominates, but it is only the combinational delay between registers
- If we place registers in the critical path, we can shorten the delay by reducing the amount of logic between registers

Pipelining Problem

- The circuit below computes the weighted average of 4 values
- Logic Delays - t_{mult} = 55ns, t_{add} = 19ns, t_{shift} = 2ns
- Register Parameters - t_{setup} = 2ns, t_{hold} = 1ns, $t_{clk\text{-}to\text{-}q}$ = 3ns
- What is the critical path delay and the maximum frequency this circuit can operate at?

- If you add one stage of registers (pipelining), what is the highest frequency you can get?



Boolean Simplification Practice

- Minimize the following boolean expressions:

$(A + B)(A + \bar{B})C$

$\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C} + A\bar{B}\bar{C} + ABC + A\bar{B}C$

$\bar{A}B + A\bar{B}$

Finite State Machine Practice

- *Goal:* A system that can output a value between 0 - 3 with the ability to increment and decrement. This system will have two 1-bit inputs: increment and decrement (as well as clock), and a 2-bit output (the count). If increment is high, the count should increase by one for the next cycle (wrap around if necessary). If decrement is high, the count should decrease by one for the next cycle (wrap around if necessary). If neither is high the system should stay at the same value, and they will never both be high at the same time.
- Draw a finite state machine for this system.

- Assign states binary encodings and complete a truth table for your FSM.

- Starting from sum-of-product expressions from the truth table, derive simplified expressions for next state as well as the output.