CS61c Summer 2014 Discussion 11 – Clocking and finite state machines

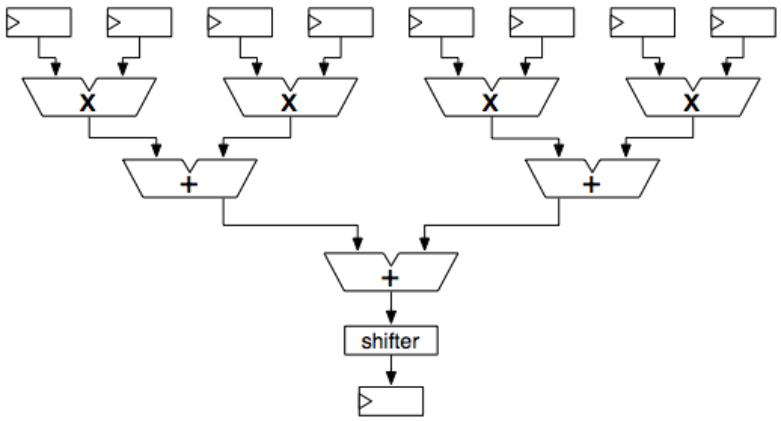
1 Clocking

1.1 Clocking Methodology

- The input signal to each state element must stabilize before each rising edge.
- Critical path: Longest delay path between state elements in the circuit.
- Min clock period = $t_{clk-to-q} + t_{CL} + t_{setup}$, where t_{CL} is the critical path in the comnibational logic.
- If we place registers in the critical path, we can shorten the period by reducing the amount of logic between registers.

1.2 Clocking 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, $t_{clk-to-q}$ = 3ns



Exercise 1. What is the critical path delay and the maximum clock rate this circuit can operate at?

Critical path - path from a top register to the bottom register.

Clk-to-q + mult + add + add + shift + setup = 3 + 55 + 19 + 19 + 2 + 2 = 100 ns.

Max frequency = 1/Min period = 10 MHz

Exercise 2. If you add one stage of registers (pipelining), what is the highest clock rate you can get?

Best to add registers in a place that minimizes the longest path through the circuit. This would be right after the multiplication.

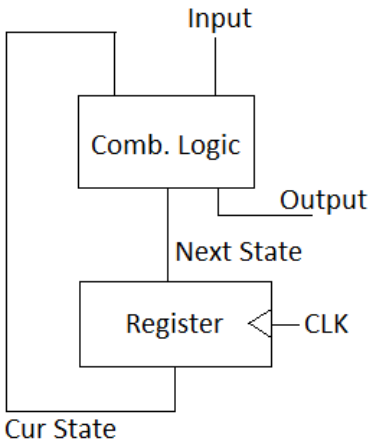
Path from top registers to middle registers is clk-to-q + mult + setup = 3 + 55 + 2 = 60 ns.

Path from middle to bottom registers is clk-to-q + add + add + shift + setup = 3 + 19 + 19 + 2 + 2 = 45 ns.

So our new critical path has Min period of 60 ns, so our Max frequency = 16.67 MHz.

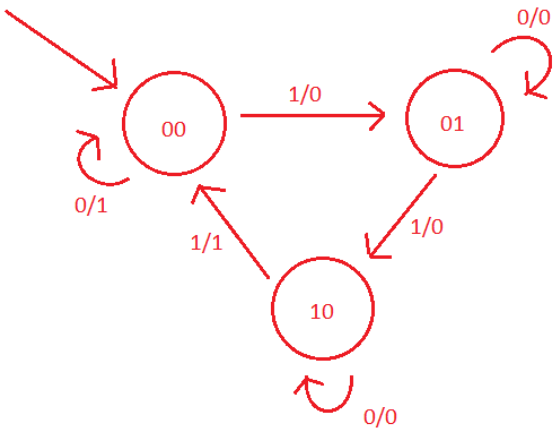
2 Finite State Machines

FSMs can be an incredibly useful computational tool. They have a straightforward implementation in hardware:



The register holds the current state (encoded as a particular combination of bits), and the combinational logic block maps from {current state, input} to {next state, output}.

Exercise 3. Draw a transition diagram for an FSM that can take in an input sequence one bit at a time, and after each input is received, output whether the number of 1s is divisible by 3.



The states each correspond to the number of 1s seen so far, mod 3. When this quantity is 0, 1s seen so far is divisible by 3, and we output 1.

Exercise 4. Write out the truth table that the combinational logic block must implement (remember to assign each state a binary encoding).

We assign our three states the encodings 00, 01, 10. Note that these encodings are completely arbitrary – but other encodings (i.e. 01, 10, 11) will lead to a different truth table and final combinational logic.

cur. state	input	next state	output
00	0	00	1
00	1	01	0
01	0	01	0
01	1	10	0
10	0	10	0
10	1	00	1
11	0	XX	X
11	1	XX	X

Behavior for current state 11 is "undefined" since we don't expect our machine to ever reach that state. We represent this in the truth table above using an 'X' to stand for "don't care." We can use this to our advantage by choosing values (0 or 1 for each X) that will simplify our combinational logic.

Exercise 5. Finally, write the Boolean algebra expressions that implement the FSM's truth table.

Boolean logic expressions (CS[1],CS[0] – bits of current state, I – input, NS[1], NS[0] – next state, Out – output). Overbar means complement.

NS[1]:

Direct method: $\overline{\text{CS}[1]} \cdot \text{CS}[0] \cdot \text{I} + \text{CS}[1] \cdot \overline{\text{CS}[0]} \cdot \bar{\text{I}}$

Simplest possible: $\text{CS}[0] \cdot \text{I} + \text{CS}[1] \cdot \bar{\text{I}}$

(Simplest version assumes NS[1] = 1 for both $\text{CS}[1] \cdot \text{CS}[0]$)

NS[0]:

Direct method: $\overline{\text{CS}[1]} \cdot \overline{\text{CS}[0]} \cdot \text{I} + \overline{\text{CS}[1]} \cdot \text{CS}[0] \cdot \bar{\text{I}}$

Simplest possible: $\overline{\text{CS}[1]} \cdot \overline{\text{CS}[0]} \cdot \text{I} + \text{CS}[0] \cdot \bar{\text{I}}$

(Simplest version assumes NS[0] = 1 for $\text{CS}[1] \cdot \text{CS}[0] \cdot \bar{\text{I}}$ and NS[0] = 0 for $\text{CS}[1] \cdot \text{CS}[0] \cdot \text{I}$)

Out:

Direct method: $\overline{\text{CS}[1]} \cdot \overline{\text{CS}[0]} \cdot \bar{\text{I}} + \text{CS}[1] \cdot \overline{\text{CS}[0]} \cdot \text{I}$

Simplest possible: $\overline{\text{CS}[1]} \cdot \overline{\text{CS}[0]} \cdot \bar{\text{I}} + \text{CS}[1] \cdot \text{I}$

(Simplest version assumes Out = 0 for $\text{CS}[1] \cdot \text{CS}[0] \cdot \bar{\text{I}}$ and Out = 1 for $\text{CS}[1] \cdot \text{CS}[0] \cdot \text{I}$)