Discussion 11 Richard Guo
SDS, Logic, and Truth 03/11/09

1. SDS

Review: SDS has two types of circuits: CL (adders, muxers) and State (registers, flip flops)
CL updates its output as soon as (or after a short delay) any of its inputs change.
CL typically performs some bitwise task on the inputs (adding, subtracting, xor-ing, etc.)
State does not sample the input until it is triggered to do so by a *load signal* or the *clock*.
State contains the value of the previous input it sampled and constantly outputs its contents.
A *clock* keeps time and makes sure all registers sample at the same time. Only used for state circuits.

Delays

Delays associated with CL circuits are called *CL delay*, which is how long it takes to update the output. *Registers* require that the input be constant and stable for a certain time before the clock trigger (the *setup time*) and after the clock (*hold time*). If it is not, the register will not sample the correct value with 100% certainty. Registers also have a delay in updating their contents and output called *clock-to-q delay*. As covered in lecture, the hold time must be less than the clock-to-q delay.

Clock Speed

How fast the clock can cycle depends on the longest path within the circuit. The clock must make sure all registers have sampled and updated the inputs before it can cycle again. Thus, we are interested in the maximum amount of time it takes for a signal to travel between 2 of the 3 following nodes: overall input, a state element, overall output

The delay between two nodes is dependent on:
CL delays + setup time + clk-to-q delay

Therefore, for the overall circuit, if we assume that clk-to-q delay is constant for all registers
Total delay = MAX(total CL delays in all nodes) + setup time + clk-to-q delay

2. Some Logic Circuits

The basics are: And, Or, Not, Nor, Xor, Nand, Xnor
Know their symbols and be able to write a truth table for 2 inputs from memory.
Question: What would Xand and Xnand be?

The Adder

Bitwise output is determined by xor-ing inputs with the carry bit (if any) and the carry bit output is determined by taking the majority of inputs and carry bit (1 if any pairwise AND yields 1)

Subtraction is done by inverting the second input according to 2's complement before feeding into the adder. Inverting is easy – xor the input with 1. Adding 1 is done by a carry bit into the first flip-flop.

The Latch

Review notes. Leaves output unchanged if both inputs are 0. Sets to 0 or 1 otherwise. Building block of flip-flop. Input of 1 and 1 is unstable.

3. Truth Table → Logic Expression

It's possible to interconvert between all 3: truth table ↔ boolean logic expression ↔ logic circuit

Truth Table → Boolean Expression: Treat each product at a time. 2 ways to do this, but we'll just teach sum of products:

Find all the inputs where the output is 1. Add up (OR) all the particular cases in which that combination of inputs exist (using AND and NOT). Then simplify with boolean algebra.

Boolean Expression → Logic Circuit: Simplify and then draw wires linking CL blocks and inputs.

Logic Circuit → Truth Table: Test all combinations of input(s) and record the output(s).

4. Boolean Algebra

Just like regular algebra, except the following table contain the rules that are used to simplify:

$x \cdot \overline{x} = 0$	$x + \overline{x} = 1$	complementarity
$x \cdot 0 = 0$	$x + 1 = 1$	laws of 0's and 1's
$x \cdot 1 = x$	$x + 0 = x$	identities
$x \cdot x = x$	$x + x = x$	idempotent law
$x \cdot y = y \cdot x$	$x + y = y + x$	commutativity
$(xy)z = x(yz)$	$(x + y) + z = x + (y + z)$	associativity
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	distribution
$xy + x = x$	$(x + y)x = x$	uniting theorem
$\overline{x \cdot y} = \overline{x} + \overline{y}$	$\overline{(x + y)} = \overline{x} \cdot \overline{y}$	DeMorgan's Law

5. Solved Problems

1. I want to build a circuit that takes 4 bits as its input and outputs four bits. The top two bits of the output stores the number of the first bit that has a nonzero value (0-3). The bottom two bits of the output stores the total number of 1s in the input bits (0-3). Assume 0000 and 1111 are invalid inputs. Write out the truth table for this circuit.

2. Write out the simplify boolean expression using just 2 bit AND, OR, and NOT.

3. How could you simplify things if you can also use 2 bit XOR?