Last time: Extremely Clever Subtractor

The Honeynet project uses "honeypots" PCs & monitors how long it takes (sec-min) for them to be hacked and what happens. 1 Mebi "botnets" are used for spam, viruses, DDoS, Google AdSense, hacking gambling, id theft!

Review...

- Use muxes to select among input
  - S input bits selects 2^S inputs
  - Each input can be n-bits wide, indep of S
- Implement muxes hierarchically
- ALU can be implemented using a mux
  - Coupled with basic block elements
- N-bit adder-subtractor done using N 1-bit adders with XOR gates on input
  - XOR serves as conditional inverter

Combinational Logic from 10 miles up

- CL circuits simply compute a binary function (e.g., from truthtable)
- Once the inputs go away, the outputs go away, nothing is saved, no STATE
  - Similar to a function in Scheme with no set! or define to save anything
- How does the computer remember data? [e.g., for registers]
A simpler state circuit: two inverters

- When started up, it's internally stable.
- Provide an OR gate for coordination:

What's the result? How do we set to 0?

An R-S latch (cross-coupled NOR gates)

- S means “set” (to 1).
- R means “reset” (to 0).

- Adding Q' gives standard RS-latch:

Truth table

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q(t+Δt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>unstable</td>
</tr>
</tbody>
</table>

A B NOR

0 0 1
0 1 0
1 0 0
1 1 0

Q Hold!

An R-S latch (in detail)

Truth table

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q(t+Δt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>set</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>unstable</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>unstable</td>
</tr>
</tbody>
</table>

R-S latch in scheme

It’s really just recursion!

(define (rs-latch r s)
  (define (rsl-h q qbar)
    (rsl-h (nor r qbar)
    (nor q s)))
  (rsl-h #t #f))

State diagram

- States represent possible output values.
- Transitions represent changes between states based on inputs.

What does it mean to “clobber” midterm?

- You STILL have to take the final even if you aced the midterm!
- The final will contain midterm-material Qs and new, post-midterm Qs
- They will be graded separately
- If you do “better” on the midterm-material, we will clobber your midterm with the “new” score! If you do worse, midterm unchanged.

- What does “better” mean?
  - Better w.r.t. Standard Deviations around mean
- What does “new” mean?
  - Score based on remapping St. Dev. score on final midterm-material to midterm score St. Dev.
“Clobber the midterm” example

- Midterm
  - Mean: 47
  - Standard Deviation: 14
  - You got a 33, one σ below
- Final Midterm-Material Questions
  - Mean: 40
  - Standard Deviation: 20
  - You got a 60, one σ above
  - Your new midterm score is now mean + σ = 47 + 14 = 61 (~ double your old score)!

Controlling R-S latch with a clock

- Can’t change R and S while clock is active.
- Clocked latches are called flip-flops.

D flip-flop are what we really use

- Inputs C (clock) and D.
- When C is 1, latch open, output = D (even if it changes, “transparent latch”)
- When C is 0, latch closed, output = stored value.

D flip-flop details

- We don’t like transparent latches
- We can build them so that the latch is only open for an instant, on the rising edge of a clock (as it goes from 0⇒1)

But do you really understand NORs?

- If one input is 1, what is a NOR?
- If one input is 0, what is a NOR?

But do you really understand NANDs?

- If one input is 1, what is a NAND?
- If one input is 0, what is a NAND?
Peer Instruction

Pick the truth table that results from substituting NAND gates for the NOR gates in the R-S latch:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>undef</td>
</tr>
</tbody>
</table>

A B NOR  A B NAND
0 0 1    0 0 1
0 1 1    0 1 1
1 0 1    1 0 1
1 1 undef undef 1 1 1 1

A B Q  A B Q  A B Q  A B Q
0 0 hold 0 0 hold 0 0 undef 0 0 undef
0 1 0 0 1 0 0 1 0 0
1 0 1 1 0 0 1 0 1 0 0
1 1 undef 1 1 undef 1 1 hold 1 1 hold

Peer Instruction

A. (a+b)·(\overline{a}+b) = b
B. N-input gates can be thought of cascaded 2-input gates. I.e.,
\((a \Delta bc \Delta d \Delta e) = a \Delta (bc \Delta (d \Delta e))\)
where \(\Delta\) is one of AND, OR, XOR, NAND
C. You can use NOR(s) with clever wiring to simulate AND, OR, & NOT

Peer Instruction

A. Truth table for mux with 4-bits of signals has \(2^4\) rows
B. We could cascade N 1-bit shifters to make 1 N-bit shifter for sll, srl
C. If 1-bit adder delay is T, the N-bit adder delay would also be T

And In conclusion...

• We use feedback to maintain state
• RS-Latch the simplest memory element
  • We’re not allowed to assert both R and S
• Clocks tell us when latches change
• D-FlipFlops used to build Register files