Logic Design IV

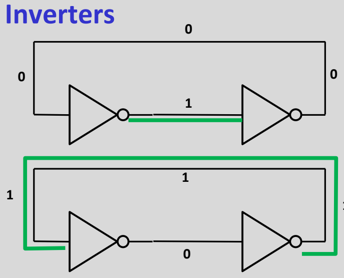
CSci 2021: Machine Architecture and Organization
Lecture #39, May 1st, 2015

Your instructor: Stephen McCamant

Sequential Circuits

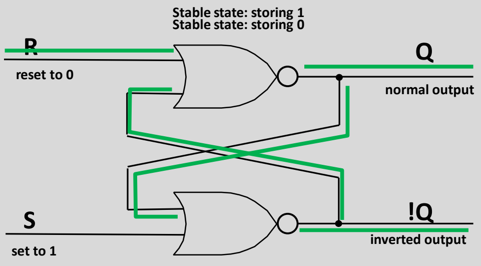
- **Introduce elements that keep state**
- **Cyclic connections between gates**
- **Makes more interesting computations possible**
  - Processing changing inputs over time
  - E.g., CPU
- **Raises more issues related to timing**
  - Coordinating timing of operations
  - Time margins for reliable operation
  - Avoiding transient incorrect results

Paired Inverters



- **Good: maintains a particular state**
- **Bad: no way to set**

S-R Latch

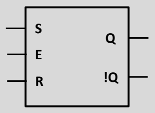


Coordination and Clock Signals

- **In sequential design, must control when events occur**
- **Standard approach: clock signal**
  - Alternates between 0 and 1
  - Same signal used throughout circuit
    - Challenge in high-speed designs: propagation speed
  - Rate controls speed of entire circuit
  - Design circuit to allow highest possible clock speed
  - Example: 3.0 GHz CPU
- **Use clock to control when sequential devices "read"**

Level-sensitivity

- **First approach:**
  - Value updated only when an enable signal (E) is high
  - Called a "level-sensitive" or "gated" device
- **Example: gated S-R latch**



- **Implementation: AND E with S and R inputs**

Transparency

- **Definition:**
  - A device is transparent if input changes immediately propagate to the output
  - S-R latch is an example
- **Transparent devices in series**
  - Connect output of one latch to input of another
  - Input causes both devices to change at the same time
  - Undesirable in many situations
    - E.g., remember Y86 pipeline stages

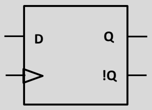
7

Edge-triggered Devices

- **Idea: update only on a clock "edge":**
  - Positive/rising edge: 0 to 1
  - Negative/falling edge: 1 to 0
  - One update per clock cycle
- **An edge-triggered bit-storage device is a "flip-flop"**
- **Flip-flops in series:**
  - Previous output changes only after next input is "read"
  - Leads to lock-step propagation, one flip-flop per cycle

8

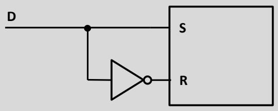
D Flip-Flop



- **Triangle indicates clock input**
  - No bubble → rising edge triggered
- **On edge, store the value of D ("data")**
- **This was our main building block for Y86 registers**
- **!Q is often unused, but available for free**

9

S-R to D

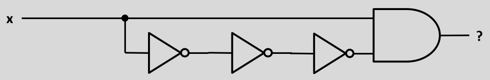


- **D = 1: S = 1, R = 0**
- **D = 0: S = 0, R = 1**
- **Avoids ever having S and R together**

- **Trickier: how to build edge-triggering?**

10

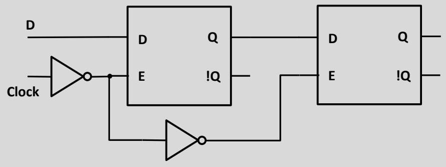
Transient Timing



- **What does this circuit do?**
- **Functional perspective:**
  - (x & !!!x) = (x & !x) = 0, useless?
- **Actually, rising edge of x causes a brief output pulse**
  - Fast path goes to 1 before delayed path goes to 0

11

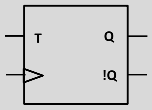
Master-slave D Flip-Flop



- **Make flip-flop out of two gated latches**
- **Updates only on rising clock edge**
  - Master freezes first
  - Then slave is enabled

12

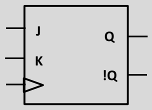
T Flip-Flop



- **Another input style, T = “toggle”**
- **Flip-flop behavior summarized by state update formula**
  - Here, Q’ = (Q ^ T)
  - T = 0: unchanged; T = 1: value is negated
- **Would this make sense without a clock?**

13

J-K Flip-Flop



- **More powerful input type**
  - Like combination of S-R and T
- **Cases:**
  - J = K = 0: no change
  - J = 1: set; K = 1: reset
  - J = K = 1: toggle
- **Q’ = (Q & !K) | (!Q & J)**

14

Propagation Delay and the Clock

- **Combinational circuits take time**
  - Most important: gate input to gate output propagation
  - Less significant: signal propagation on wires
- **Worst-case circuit timing**
  - Conceptually, consider all input to output paths
  - For each path, sum maximum component times
  - Worst case depends on slowest path
- **Timing determines maximum clock speed**
  - Consider slowest flip-flop to flip-flop path
  - Too-fast clock causes incorrect results

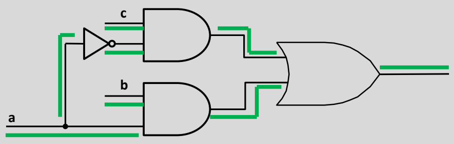
15

Hazards and Glitches

- **Glitch:**
  - A transient circuit output that differs from the ideal output
- **(Logic) hazard:**
  - Circuit feature that makes glitches possible
- **When?**
  - Unexpected values appear while changes are propagating
- **Easiest option:**
  - Use clocks and flip-flops (registers) so that circuit outputs matter only after they have settled
- **Sometimes needed:**
  - Redesign circuit or add gates to prevent hazards

16

Logic Hazard Example



- **Static-1 hazard:**
  - (a, b, c) = (1, 1, 1) and (0, 1, 1) should both output 1
  - Depending on delay, a 0 glitch may occur

17

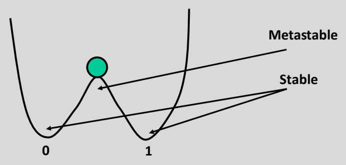
Timing for Flip-Flops

- **Input must be steady around clock edge for reliable operation**
  - Setup time: amount of time before clock input must be right
  - Hold time: amount of time after clock that input must right
- **Delay before output changes**
  - Clock-to-output time: delay between clock edge and output edge
- **Important fact:**
  - (hold time) < (clock-to-output time)
  - If true of two flip-flops, it is safe to connect output of one to input of another, on the same clock

18

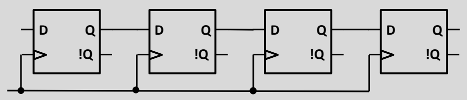
Metastability

- **Analogy:**



- **Flip-flop tries to return to a stable state**
  - Self-correcting feedback
- **But, if close to metastable point:**
  - Might take a long time to "decide"
  - Must avoid this situation for a reliable circuit

Shift Register



- **Flip-flops connected in series**
- **Behavior:**
  - Sequence of bits each move one stage per clock cycle
- **Variations:**
  - Serial or parallel input
  - Series or parallel output
  - Shift only one some cycles

Counters

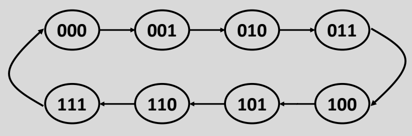
- **Simple kind of time-varying digital system**
  - Produces a single sequence of states, repeating
  - Changes every cycle or on a count pulse
- **Example: 3-bit binary up-counter**
  - Produces 000, 001, 010, 011, 100, 101, 110, 111, 000, 001, …
- **Variations:**
  - Down-counters
  - Decade counters (for decimal): 0 through 9
  - Gray code: sequence where only one bit changes at a time
  - Ring counter: circular shift register producing one-hot outputs

State Machines

- **State machine perspective:**
  - If a device has limited memory, we can enumerate its possible *states*
  - *n* bits of memory -> at most 2^n states
  - Inputs cause state transitions
  - Outputs depend on state and possibly inputs
- **General approach for designing sequential circuits**
  - Enumerate states
  - Determine state transitions and I/O
    - Last two summarized in *state transition diagram*
  - Use flip-flops to remember state
  - Use combinational logic to implement update and output functions

Counters as State Machines

- **Simplified special case**
  - Output is generally identical to state
  - No input other than a clock
  - State transition diagram is a single cycle
- **Three-bit up counter:**


000
001
010
011
100
101
110
111