Lecture 15

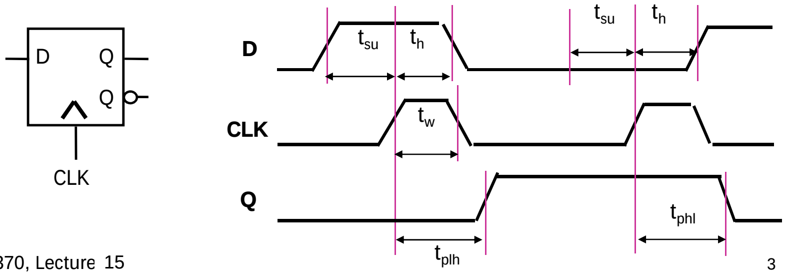
- Logistics
  - HW6 is out, due Wednesday
- Last lecture
  - Continuing on basic building blocks for sequential logic
    - Latches and flip-flops
    - Clear/Preset
    - State Diagram
    - Asynchronous inputs
- Today
  - Continue more on building blocks for sequential logic
    - Timing issues with asynchronous inputs and some solutions
    - Registers
    - Summary of sequential logic building blocks

Asynchronous inputs

- Clocked circuits are synchronous
- Unclocked circuits or signals are asynchronous
- Synchronous circuits have asynchronous inputs
  - Reset signal, memory wait, user input, etc.
  - Inputs can change at any time
    - We must synchronize the input to our clock
    - Inputs can violate flip-flop setup/hold times

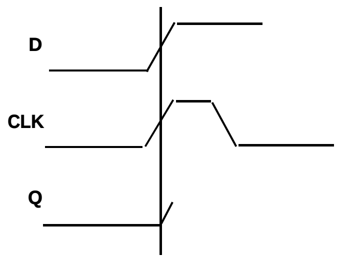
Timing terminology and constraints

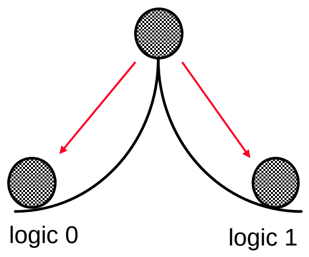
- Setup time t_{su}: Amount of time the input must be stable before the clock transitions high (or low for negative-edge triggered FF)
- Hold time t_h: Amount of time the input must be stable after the clock transitions high (or low for negative-edge triggered FF)
- Clock width t_w : Clock width that must be met
- Propagation delays t_{plh} and t_{phl}: Propagation delay (high to low, low to high)



Synchronizer failure

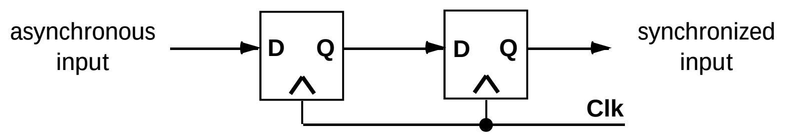
- Occurs when input changes near clock edge
  - Input is neither 1 or 0 when clock goes high
  - Output may be neither 0 or 1
    - May stay undefined for a long time
  - Undefined state is called metastability





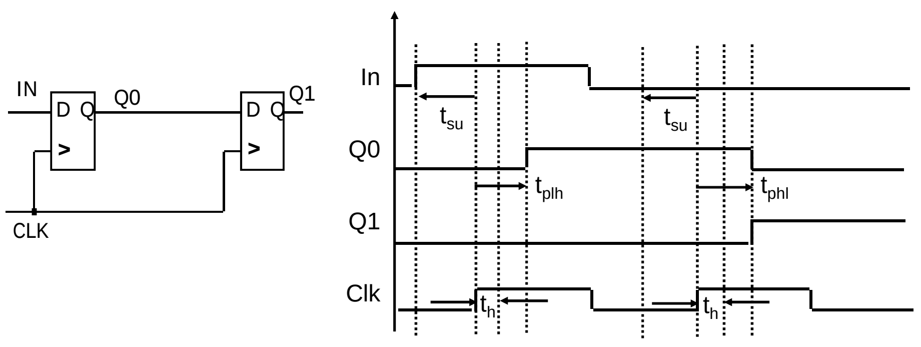
Minimizing synchronizer failures

- Failure probability can never be zero
  - Cascade two (or more) flip-flops
    - Effectively synchronizes twice
    - Both would have to fail for system to fail



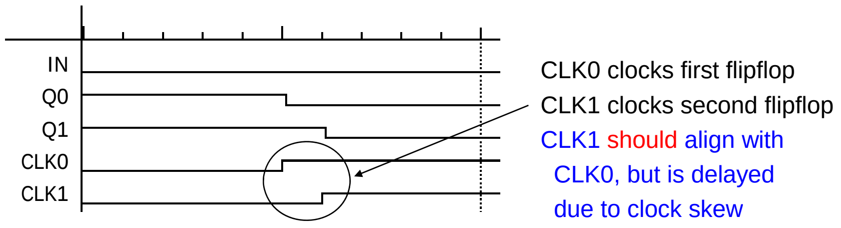
Cascading flip-flops

- Flip-flop propagation delays exceed hold times
  - Second stage commits its input before input changes



Side note: Clock skew

- Goal: Clock all flip-flops at the same time
  - Difficult to achieve in high-speed systems
    - Clock delays (wire, buffers) are comparable to logic delays
  - Problem is called clock skew

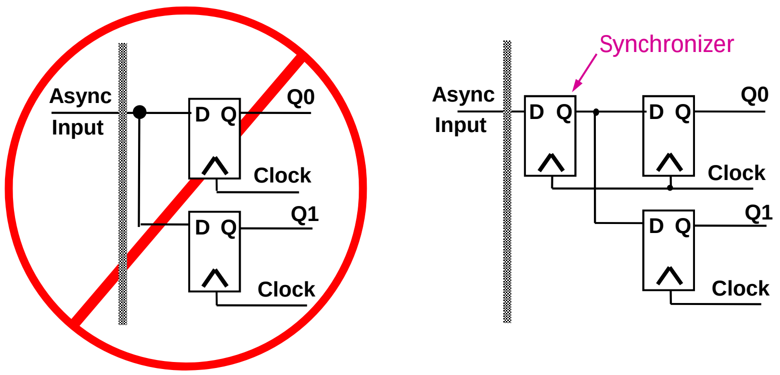


Original state: IN = 0, Q0 = 1, Q1 = 1
Next state: Q0 = 0, Q1 = 0 (should be Q1 = 1)

  - Avoiding clock skew: design identical delays

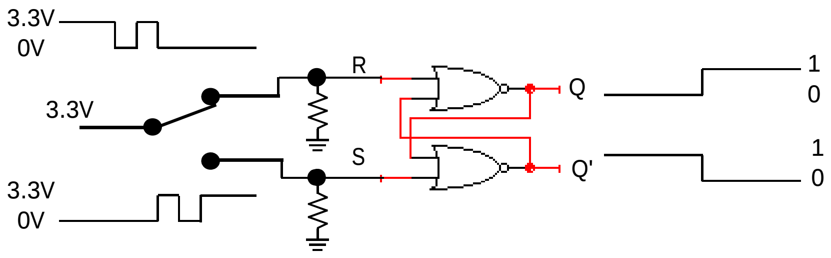
Handling asynchronous inputs

- Never fan-out asynchronous inputs
  - Synchronize at circuit boundary
  - Fan-out synchronized signal



One more important concept: Debouncing

- Switch inputs bounce
  - i. e. don't make clean transitions
- Can use SR latch for debouncing
  - Eliminates dynamic hazards
  - "Cleans-up" inputs



Summary: Timing issues with asynchronous inputs

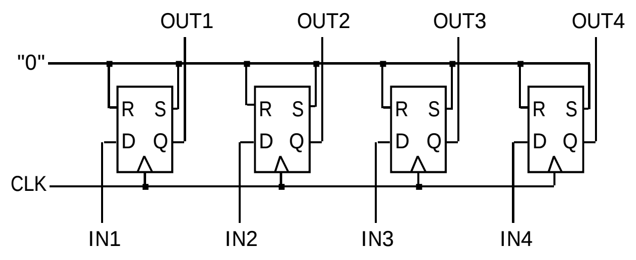
- For sequential logic circuits, timing issues have to be considered.
- Inputs are often asynchronous and can cause problems.
- Different amount of delay at different part of the circuit can cause problems also.
- Solutions:
  - Cascade flip flops in series
  - Incorporate RS latch for debouncing
  - Design to keep timing alignment in mind (length of cable, etc)

Registers

- Group of storage elements read/written as a unit.
  - Store related values (e.g. a binary word)
- Collection of flip-flops with common control
  - Share clock, reset, set lines
- Example:
  - Storage registers
  - Shift registers
  - Counters

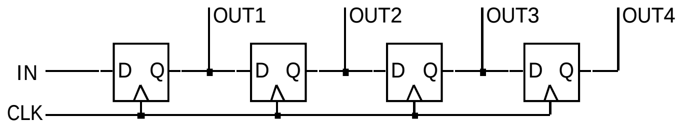
Storage registers

- Basic storage registers uses flip flops
- Example: 4 bit storage register



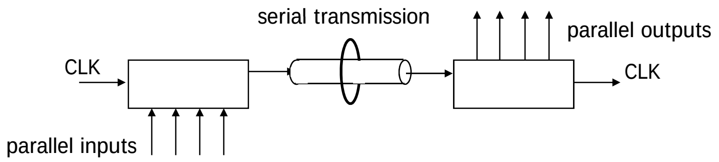
Shift registers

- Hold successively sampled input values
  - Delays values in time
  - Example: 4-bit shift register
    - Stores 4 input values in sequence

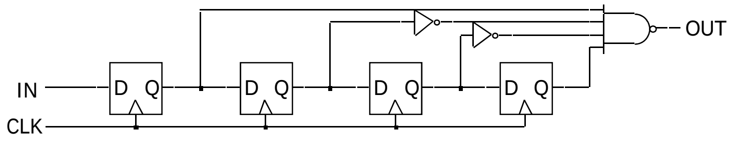


Shift-register applications

- Parallel-to-serial conversion for signal transmission

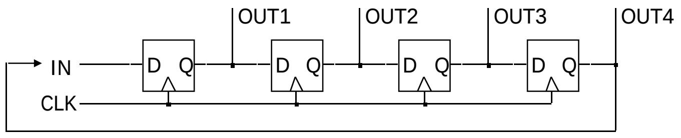


- Pattern recognition (circuit recognizes 1001)

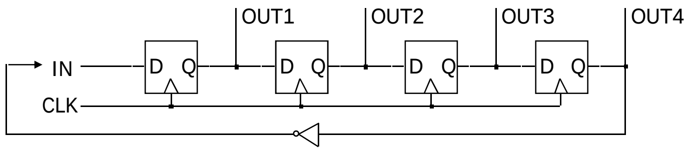


Counters

- Ring counter: Sequence is 1000, 0100, 0010, 0001
  - Assuming one of these patterns is the starting state

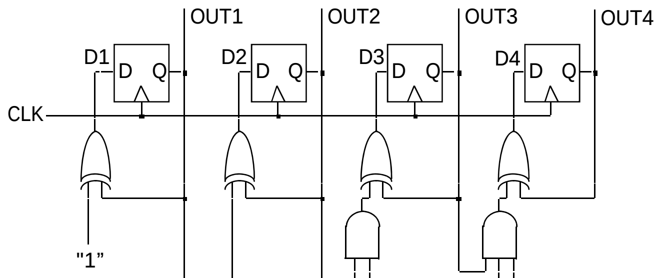


- Johnson counter: Sequence is 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000



A binary counter

- Has logic between flip-flops



Summary: Sequential-logic building blocks

- Know latches and flip-flops
- Know clocks, timing, timing diagrams
- Understand asynchronous inputs
- Know basic registers