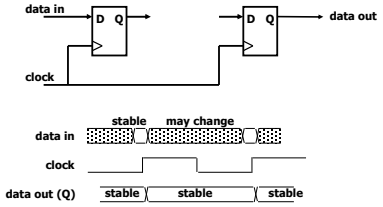
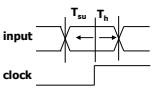
Registers

- Sample data using clock
- Hold data between clock cycles
- Computation (and delay) occurs between registers

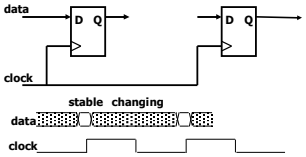


Timing Methodologies (cont'd)

- Definition of terms
  - setup time: minimum time before the clocking event by which the input must be stable (T_{su})
  - hold time:minimum time after the clocking event until which the input must remain stable (T_h)

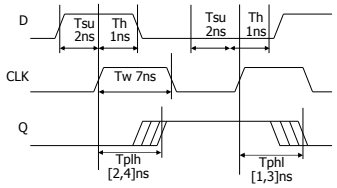


there is a timing "window" around the clocking event during which the input must remain stable and unchanged in order to be recognized



Typical timing specifications

- Positive edge-triggered D flip-flop
  - setup and hold times
  - minimum clock width
  - propagation delays (low to high, high to low, max and typical)



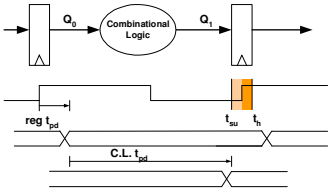
all measurements are made from the clocking event that is, the rising edge of the clock

Synchronous System Model

- Register-to-register operation
- Perform operations during transfer
- Many transfers/operations occur simultaneously

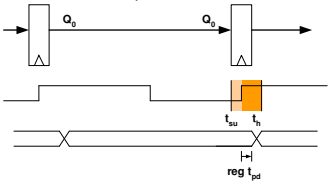
System Clock Frequency

- Register transfer must fit into one clock cycle
  - reg t_{pd} + C.L. t_{pd} + reg t_{su} < T_{clk}
  - Use maximum delays
  - Find the "critical path"
    - Longest register-register delay



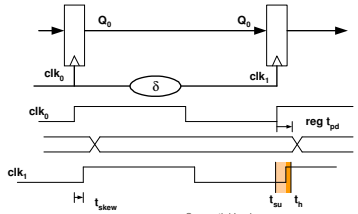
Short Paths

- Can a path have too little delay?
  - Yes: Hold time can be violated
  - $t_{pd} > t_h$
  - Use min delay (contamination delay)
- Fortunately, most registers have hold time = 0
  - But there can still be a problem! Clock skew…



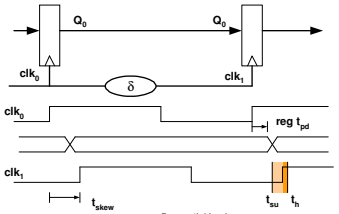
Clock Skew

- Cannot make clock arrive at registers at the same time
- If skew > 0:
  - $t_{pd} > t_h + t_{skew}$
  - Clock skew can cause system failure
    - Can you fix this after you've fabbed the chip?



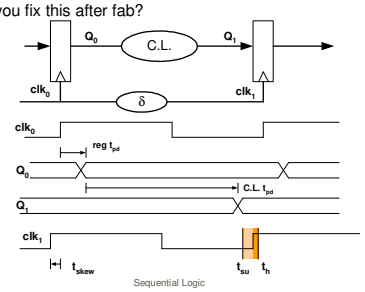
Clock Skew

- Cannot make clock arrive at registers at the same time
- If skew > 0:
  - $t_{pd} > t_h + t_{skew}$
  - Clock skew can cause system failure
    - Can you fix this after you've fabbed the chip?



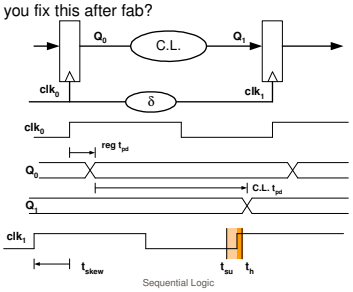
Clock Skew

- If skew < 0:
  - $t_{clk} > \text{reg } t_{pd} + \text{CL } t_{pd} + \text{ reg } t_{SU} + |t_{skew}|$
  - Can you fix this after fab?



Clock Skew

- If skew < 0:
  - $t_{clk} > \text{reg } t_{pd} + \text{CL } t_{pd} + \text{ reg } t_{SU} + |t_{skew}|$
  - Can you fix this after fab?

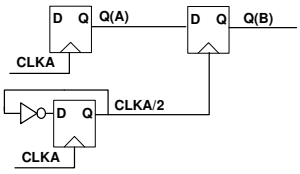


Clock Skew

- Correct behavior assumes that all storage elements sample at exactly the same time
- Not possible in real systems:
  - clock driven from some central location
  - different wire delay to different points in the circuit
- Problems arise if skew is of the same order as FF contamination delay
- Gets worse as systems get faster (wires don't improve as fast)
  - 1) distribute clock signals against the data flow
  - 2) wire carrying the clock between two communicating components should be as short as possible
  - 3) try to make all wires from the clock generator be the same length => clock tree

Nasty Example

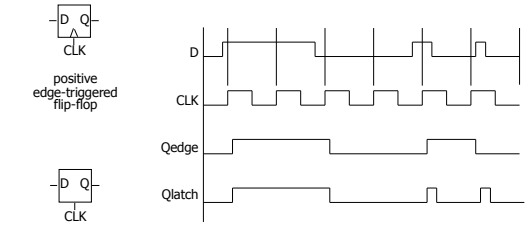
- What can go wrong?
- How can you fix it?



Other Types of Latches and Flip-Flops

- D-FF is ubiquitous
  - simplest design technique, minimizes number of wires
    preferred in PLDs and FPGAs
    good choice for data storage register
    edge-triggered has most straightforward timing constraints
- Historically J-K FF was popular
    versatile building block, often requires less total logic
    two inputs require more wiring and logic
    can always be implemented using D-FF
- Level-sensitive latches in special circumstances
    popular in VLSI because they can be made very small (4 T)
    fundamental building block of all other flip-flop types
      two latches make a D-FF
- Preset and clear inputs are highly desirable
  - System reset

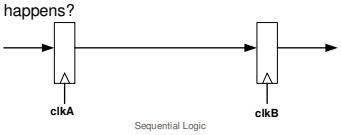
Comparison of latches and flip-flops



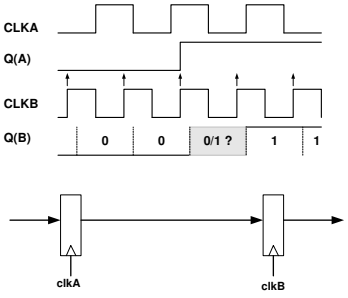
behavior is the same unless input changes while the clock is high

What About External Inputs?

- Internal signals are OK
  - Can only change when clock changes
- External signals can change at any time
  - Asynchronous inputs
  - Truly asynchronous
  - Produced by a different clock
- This means register may sample a signal that is changing
  - Violates setup/hold time
  - What happens?

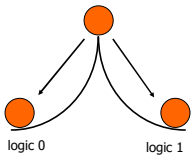


Sampling external inputs

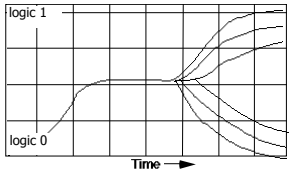


Synchronization failure

- Occurs when FF input changes close to clock edge
  - the FF may enter a metastable state – neither a logic 0 nor 1 –
  - it may stay in this state an indefinite amount of time
  - this is not likely in practice but has some probability



small, but non-zero probability that the FF output will get stuck in an in-between state



oscilloscope traces demonstrating synchronizer failure and eventual decay to steady state

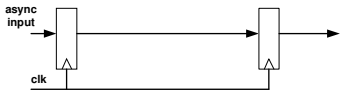
Calculating probability of failure

- For a single synchronizer

  Mean-Time Between Failure (MTBF) = $\exp(tr / \tau) / (T0 \times f \times a)$

  where a failure occurs if metastability persists beyond time tr
- tr is the resolution time - extra time in clock period for settling
  - Tclk - (t_{pd} + T_{CL} + t_{setup})
- f is the frequency of the FF clock
- a is the number of asynchronous input changes per second applied to the FF
- T0 and τ are constaints that depend on the FF's electrical characteristics (e.g., gain or steepness of curve)
  - example values are T0 = .4s and τ = 1.5ns (sensitive to temperature, voltage, cosmic rays, etc.).
- Must add probabilities from all synchronizers in system
  1/MTBFsystem = Σ 1/MTBFsynch

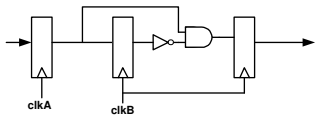
Metastability



- Example
  - input changes at 1 MHz
  - system clock of 10MHz, flipflop (t_{pd} + t_{setup}) = 5ns
    MTBF = exp(95ns / 1.5ns) / ($.4s \times 10^7 \times 10^6$) = 25 million years
  - if we go to 20MHz then:
    MTBF = exp(45ns / 1.5ns) / ($.4s \times 2 \times 10^7 \times 10^6$) = 1.33 seconds!
  - And we're not even doing any logic!
- Must do the calculations and allow enough time for synchronization

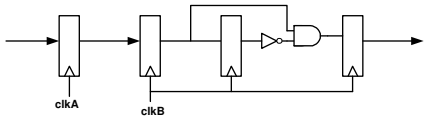
What does this circuit do?

- What's wrong with this?



What does this circuit do?

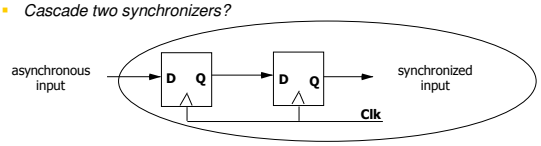
- How much better is this?



- Can you do better?

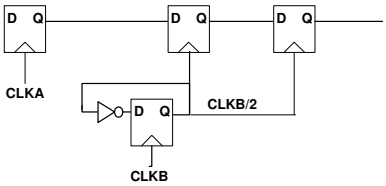
Guarding against synchronization failure

- Give the register time to decide
  - Probability of failure cannot be reduced to 0, but it can be reduced
- *Slow down the system clock?*
- *Use very fast technology for synchronizer -> quicker decision?*
- *Cascade two synchronizers?*



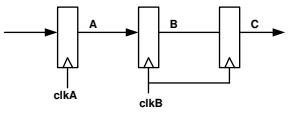
Stretching the Resolution Time

- Also slows the sample rate and transfer rate



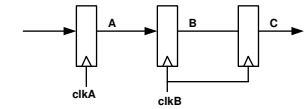
Sampling Rate

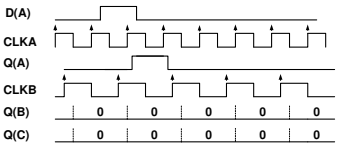
- How fast does your sample clock need to be?



Sampling Rate

- How fast does your sample clock need to be? f(clkB) > 2f(clkA)



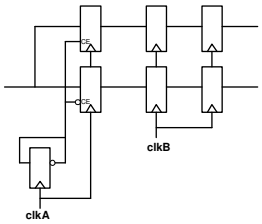


Sampling Rate

- What if sample clock can't go faster?
- If input clock is not available, no solution(?)
- If input clock is available (e.g. video codec)

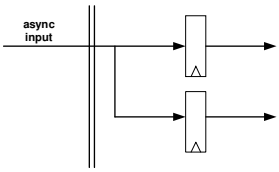
Increasing sample rate

- The problem is the **relative** sample rate
  - Slow down the input clock!



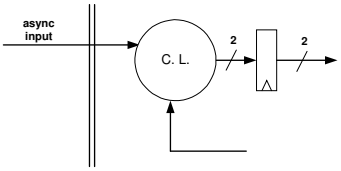
Another Problem with Asynchronous inputs

- What goes wrong here? (Hint: it's not a metastability thing)
- What is the fix?



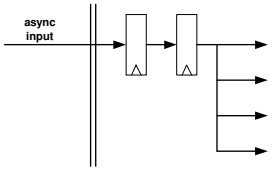
More Asynchronous inputs

- What is the problem?
- What is the fix?



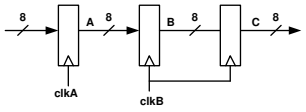
Important Rule!

- Exactly one register makes the synchronizing decision



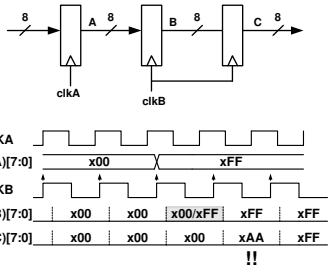
More Asynchronous inputs

- Can we input asynchronous data values with several bits?



More Asynchronous inputs

- How can we input asynchronous data values with several bits?

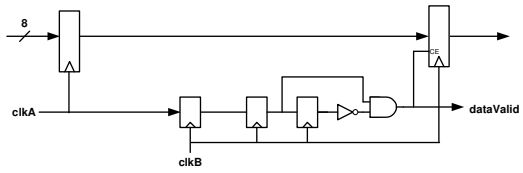


What Went Wrong?

- Each bit has a different delay
  - Wire lengths differ
  - Gate thresholds differ
  - Driver speeds are different
  - Register delays are different
    - Rise vs. Fall times
  - Clock skews to register bits
- Bottom line – "data skew" is inevitable
  - aka Bus Skew
  - Longer wires => More skew
- What is the solution??

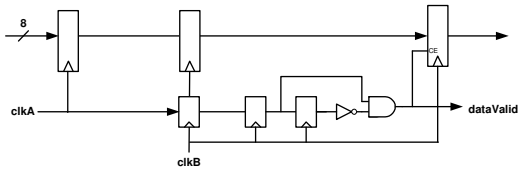
Sending Multiple Data Bits

- Must send a "clock" with the data
  - Waits until data is stable
  - De-skewing delay
- **f(clkB) > 2 f(clkA)**



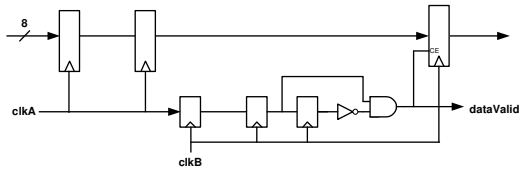
Sending Multiple Data Bits

- Balancing path delays . . .
- What's wrong with this solution?
- What's the right way to do it?



Sending Multiple Data Bits

- The right way to do it . . .



Sending Multiple Data Bits

- Slightly different alternative . . .