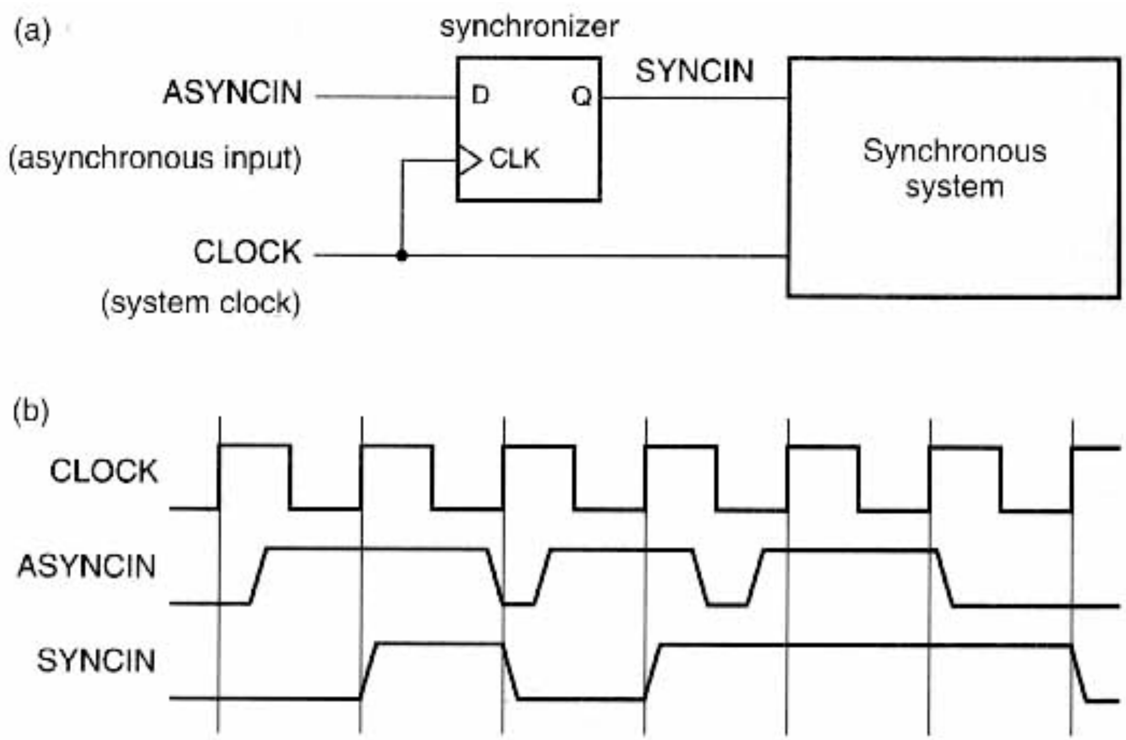
Asynchronous Inputs to Synchronous Systems

- Many synchronous systems need to interface to asynchronous input signals:
  - Consider a computer system running at some clock frequency, say 1GHz with:
    - Interrupts from I/O devices, keystrokes, etc.
    - Data transfers from devices with their own clocks
      - Ethernet has its own 100MHz clock
      - PCI bus transfers, 66MHz standard clock.
  - These signals could have no known timing relationship with the system clock of the CPU.
  - (On FPGAs we can use FIFOs - separate clocks for input and output - as the interface. In general, this is overkill - and too expensive).

"Synchronizer" Circuit

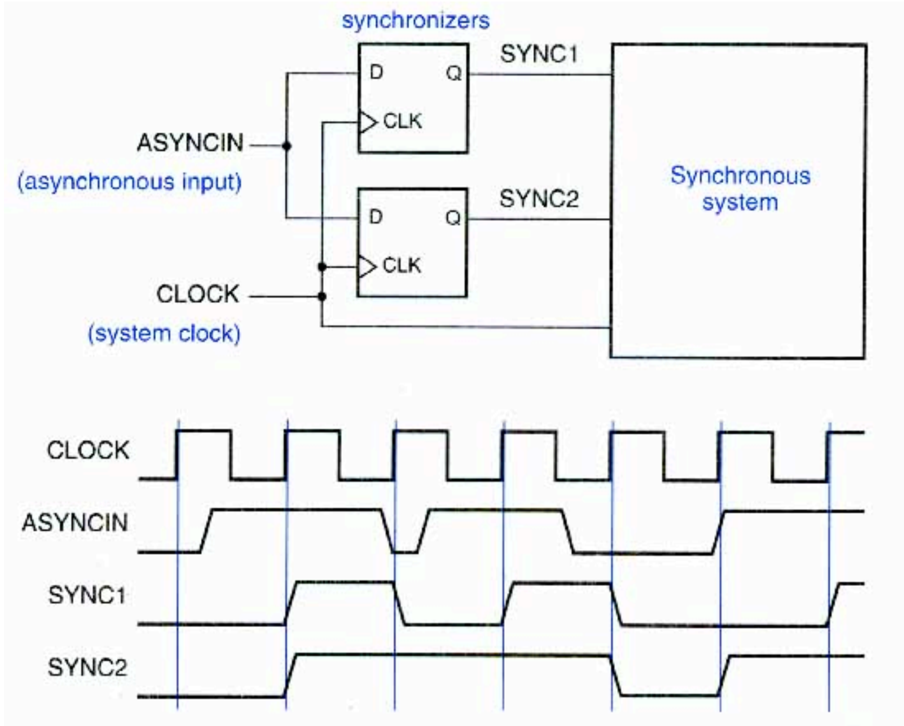
- For a single asynchronous input, we use a simple flip-flop to bring the external input signal into the timing domain of the system clock:



- The D flip-flop samples the asynchronous input at each cycle and produces a synchronous output that meets the setup time of the next stage.

"Synchronizer" Circuit

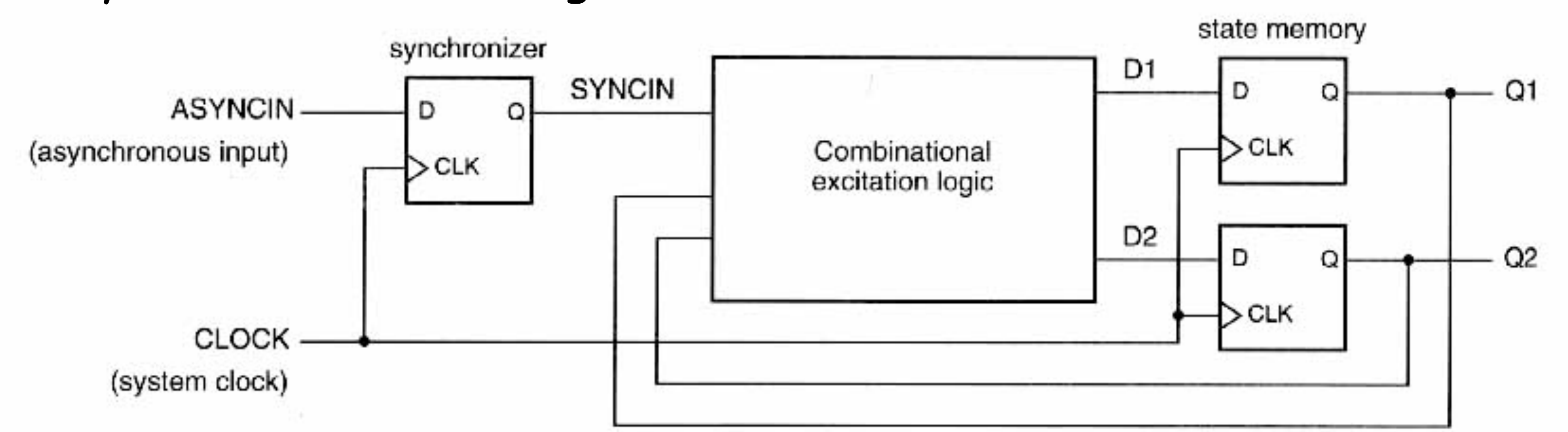
- It is essential for asynchronous inputs to be synchronized at only one place.



- Two flip-flops may not receive the clock and input signals at precisely the same time (clock *and* data skew).
- When the asynchronous changes near the clock edge, one flip-flop may sample input as 1 and the other as 0.

“Synchronizer” Circuit

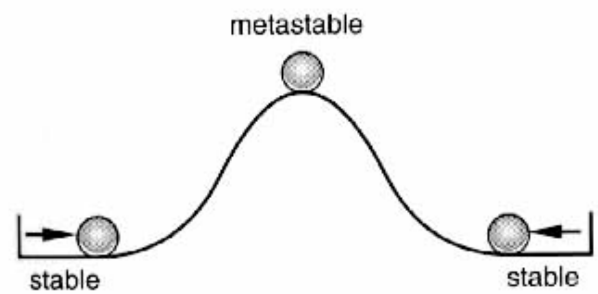
- Single point of synchronization is even more important when input goes to a combinational logic block (ex. FSM)
- The CL block can accidentally hide the fact that the signal is synchronized at multiple points.
- The CL magnifies the chance of the multiple points of synchronization seeing different values.

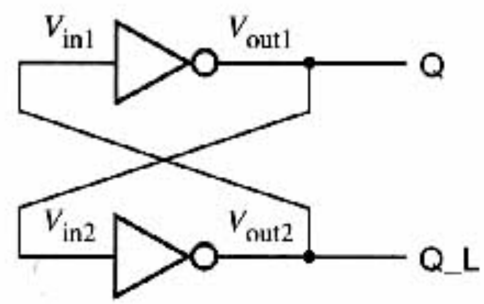


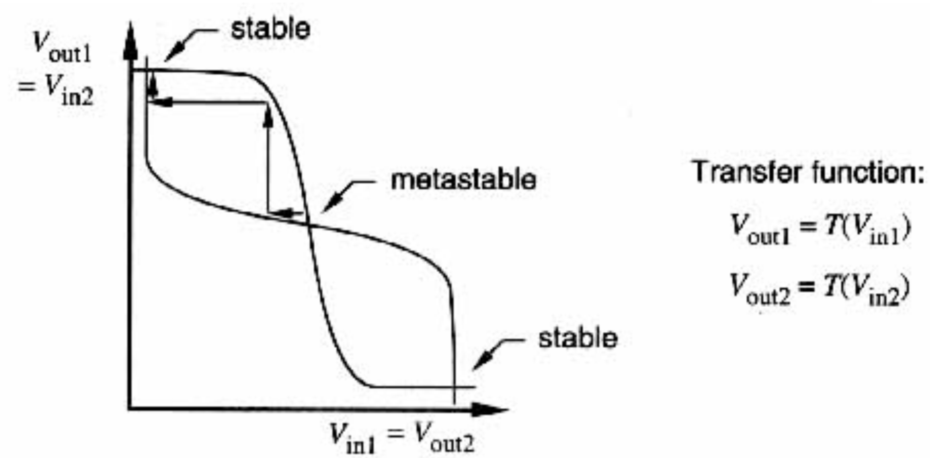
- Sounds simple, right?

Synchronizer Failure & Metastability

- We think of flip-flops having only two stable states - but all have a third *metastable* state halfway between 0 and 1.
- When the setup and hold times of a flip-flop are not met, the flip-flop could be put into the metastable state.
- Noise will be amplified and push the flip-flop one way or other.
- However, in theory, the time to transition to a legal state is unbounded.
- Does this really happen?
- The probability is low, but number of trials is high!





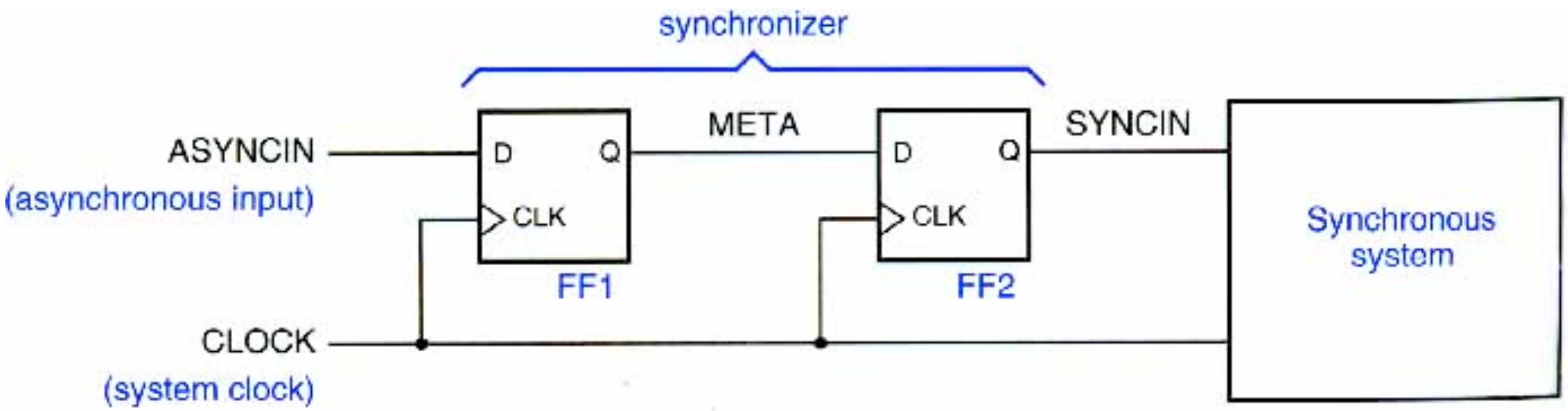


Synchronizer Failure & Metastability

- If the system uses a synchronizer output while the output is still in the metastable state ⇒ synchronizer failure.
- Initial versions of several commercial ICs have suffered from metastability problems - effectively synchronization failure:
  - AMD9513 system timing controller
  - AMD9519 interrupt controller
  - Zilog Z-80 Serial I/O interface
  - Intel 8048 microprocessor
  - AMD 29000 microprocessor
- To avoid synchronizer failure wait long enough before using a synchronizer's output. *"Long enough", according to Wakerly, is so that the mean time between synchronizer failures is several orders of magnitude longer than the designer's expected length of employment!*
- In practice all we can do is reduce the probability of failure to a vanishing small value.

Reliable Synchronizer Design

- **The probability that a flip-flop stays in the metastable state decreases exponentially with time.**
- Therefore, any scheme that delays using the signal can be used to decrease the probability of failure.
- In practice, delaying the signal by a cycle is usually sufficient:



- If the clock period is greater than metastability resolution time plus FF2 setup time, FF2 gets a synchronized version of ASYNCIN.
- Multi-cycle synchronizers (using counters or more cascaded flip-flops) are even better – but often overkill.