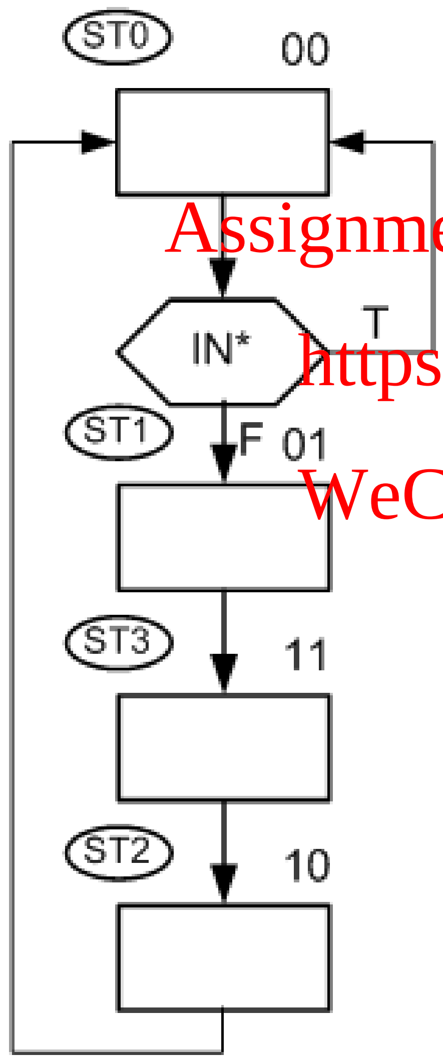
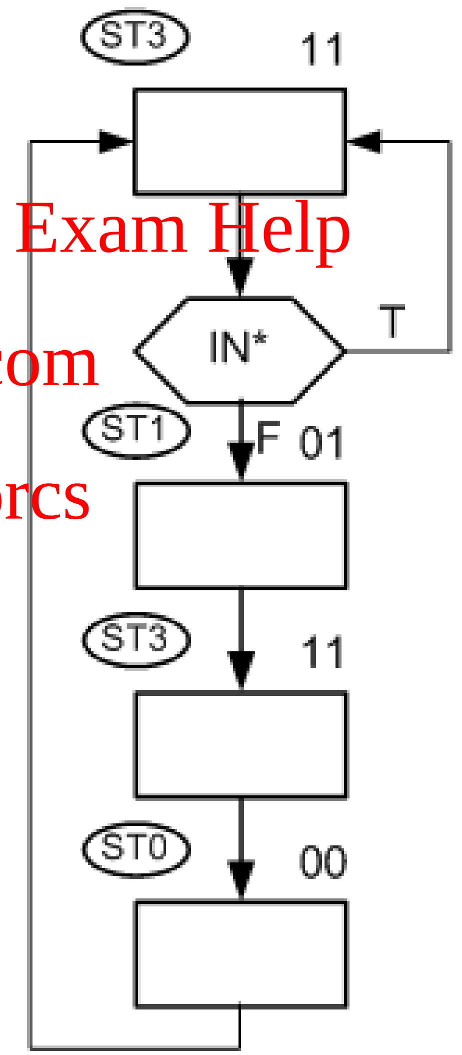
UNIVERSITY OF LIVERPOOL

Digital Systems Design

Metastability

Do the Asynchronous inputs cause problems in these ASMs?



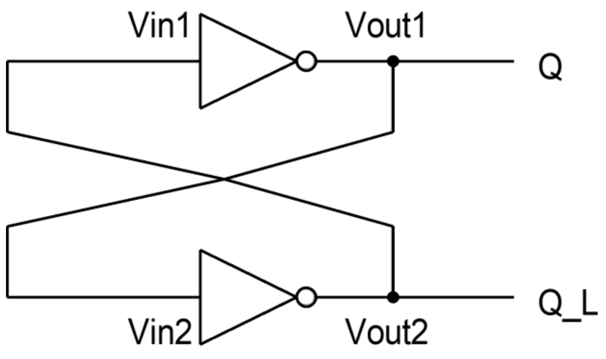


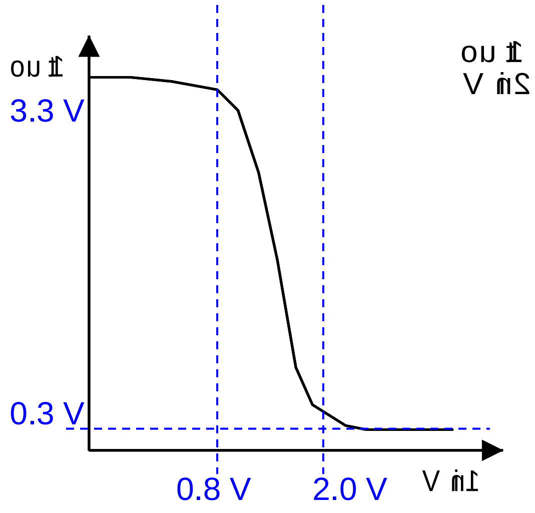
Metastability Transfer Functions

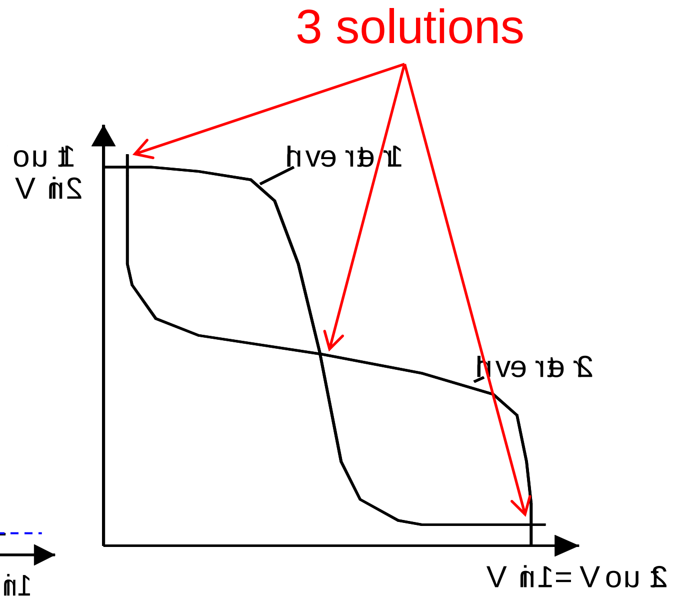
- Consider a simple bistable element with a standard transfer function (T).
- Consider only the steady state.
- For equilibrium:

  Vin1 = Vout2 = T(Vin2) = T(Vout1) = T(T(Vin1)) also

  Vin2 = T(T(Vin2))

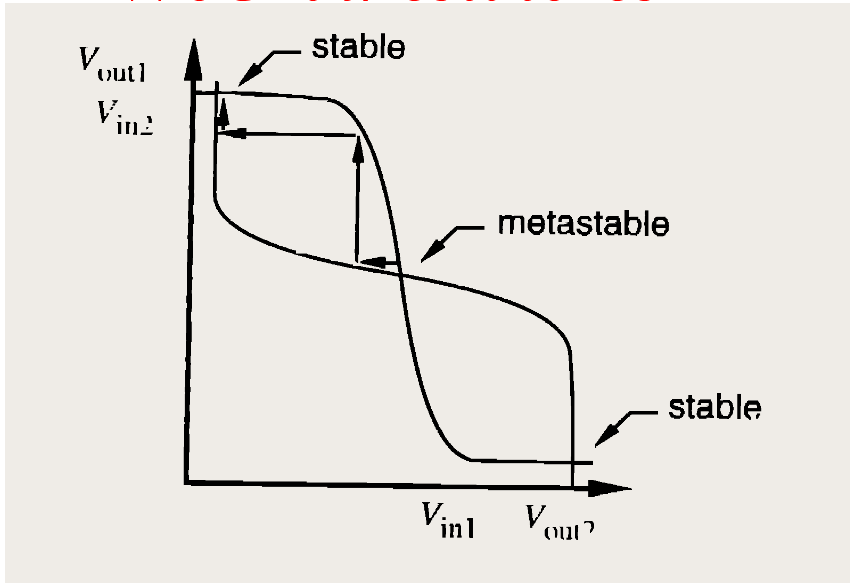






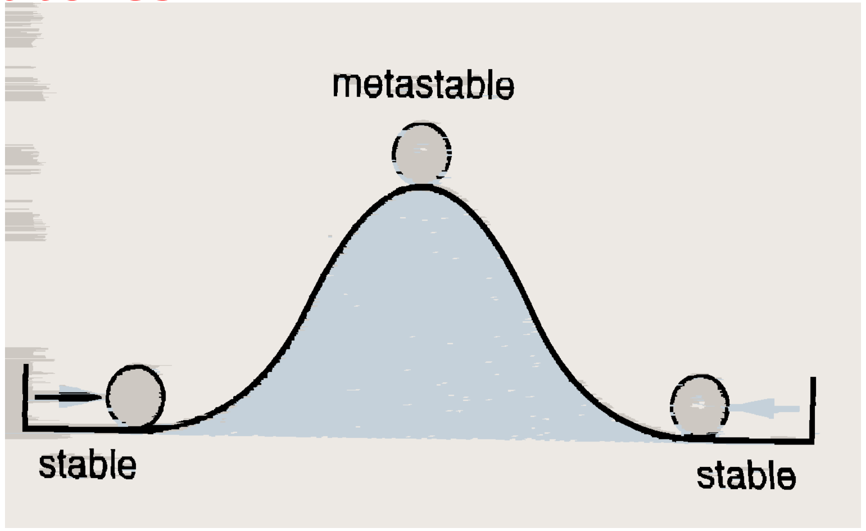
Metastable Behaviour (1)

- Closer analysis of the situation at the metastable point shows that it is suitably named.
- It is not truly stable, because random noise (perhaps small changes in PSU voltage) will tend to drive a circuit that is operating at the metastable point towards one of the stable operating points.



Metastable Behaviour (2)

- **Ball and Hill Analogy**
  - If the ball lands on the very top of the hill it may unsteadily sit there until random forces start it rolling down the hill.
  - Depending on the force applied, three events may occur when trying to move the ball from one side to another.
  - Apply a strong force and the ball goes right over the top and lands in a stable place on the other side.
  - Apply a weak force and the ball falls back to its original starting place.
  - Apply an intermediate force and the ball lands on the top of the hill, teeters, and eventually falls down one side or the other.



Metastable Behaviour (3)

- This behaviour is completely analogous to what happens to latches and flip-flops under marginal triggering conditions.
- For example, in SR bistables where a pulse on the S input forces the latch from the 0 state to the 1 state.
  - A minimum pulse width is specified for the S input. Apply a pulse of this width or longer, and the latch immediately goes to the 1 state.
  - Apply a very short pulse, and the latch stays in the 0 state.
  - Apply a pulse just under the minimum width, and the latch may go into the metastable state. Once the latch is in the metastable state, its operation depends on "the shape of its hill".
- Latches and flip-flops built from high-gain and fast technologies tend to come out of metastability faster than ones built from low-performance technologies.

Synchronizer Failure and Metastability

- Synchronizers are used to synchronize an asynchronous input to a synchronous circuit. Normally a D type flip-flop is used.

- When the setup and hold times of a flip-flop are not met, the flip-flop may go into a third, metastable state halfway between 0 and 1.

- Worse, the length of time it may stay in this state before falling back into a legitimate 0 or 1 state is theoretically unbounded.

Synchronizer Failure and Metastability – cont.

- When other gates and flip-flops are presented with a metastable input signal, some may interpret it as a 0 and others as a 1, creating inconsistent behaviour that may cause transition races.
- Or the other gates and flip-flops may produce metastable outputs themselves (after all, they are now operating in the linear part of their operating range).
- Luckily, the probability of a flip-flop output remaining in the metastable state decreases exponentially with time, though never all the way to zero.

Synchroniser Failure

- Synchronizer failure is said to occur if a system uses a synchronizer output while the output is still in the metastable state.
- The way to avoid synchronizer failure is to ensure that the system waits "long enough" before using a synchronizer's output, "long enough" so that the "Mean Time Between Failures (MTBF)" is several orders of magnitude longer than the designer's expected length of employment.
- Metastability is more than an academic problem. More than a few experienced designers of high-speed digital systems have built (and released to production) circuits that suffer from intermittent synchronizer failures.

Synchroniser Failure – cont.

- There are two ways to get a flip-flop out of the metastable state:
  1. Force the flip-flop into a valid logic state using input signals that meet the published specifications for minimum pulse width, setup time, and so on.
  2. Wait "long enough," so the flip-flop comes out of metastability on its own.

- The only way to make synchronizers reliable is to wait long enough for any metastable outputs to resolve.

Metastability Resolution Time

- If the setup and hold times of a D flip-flop are met, the flip-flop output settles to a new value within time t_{pd} after the clock edge.

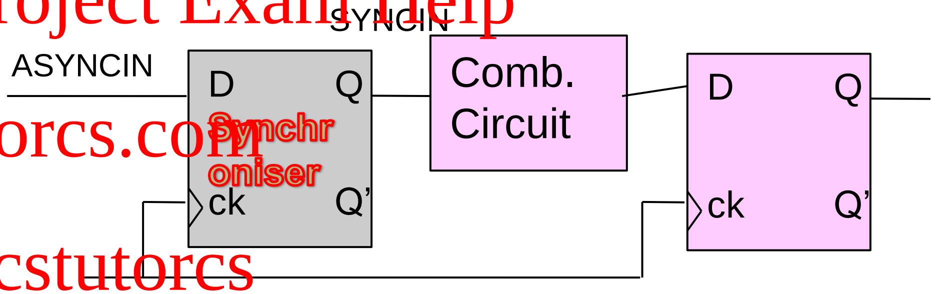
- If they are violated, the flip-flop output may be metastable for an arbitrary length of time.

- In a particular system design, we use the parameter t_r, called the metastability resolution time, to denote the maximum time that the output can remain metastable without causing synchronizer (and system) failure.

Metastability Resolution Time

- For example, consider a standard state machine.
- The available metastability resolution time, t_r ,is

$$t_r = t_{clk} - t_{comb} - t_{setup}$$



Where:

- t_{clk} is the clock period,
- t_{comb} is the propagation delay of the combinational logic,
- t_{setup} is the setup time of the flip-flops used in the state memory.

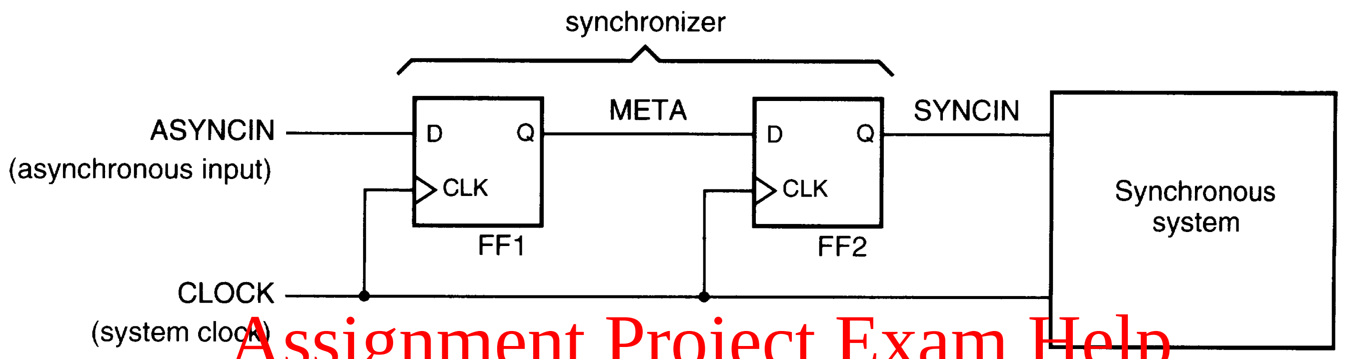
Reliable Synchroniser Design

- The most reliable synchronizer is one that allows the maximum amount of time for metastability resolution.
- However, we often need synchronizers that work reliably with very short clock periods.
- We showed previously that a state machine with an asynchronous input, has

$$t_r = t_{clk} - t_{comb} - t_{setup}$$

- In order to maximize t_r, for a given clock period, we should minimize t_{comb} and t_{setup}. The value of t_{setup} depends on the type of flip-flops used in the state memory; in general, faster flip-flops have shorter setup times.
- The minimum value of t_{comb} is zero and is achieved by the synchronizer design shown in the next slide.

Reliable Synchroniser Design- cont.

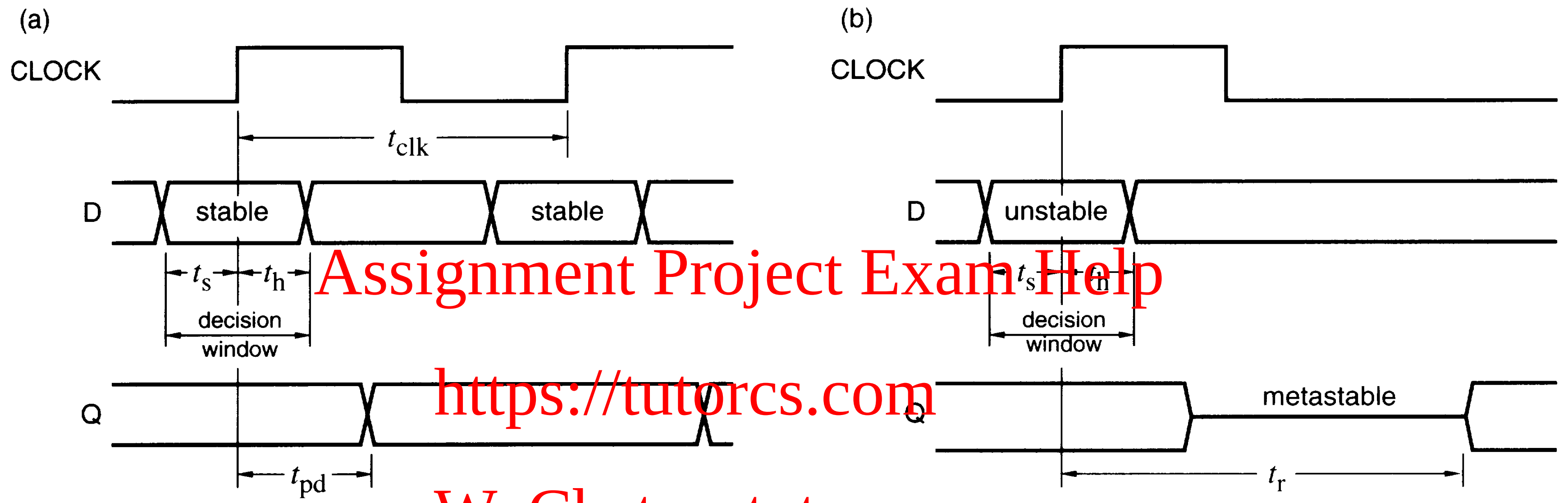


- Inputs to flip-flop FF1 are asynchronous with the clock and may violate the flip-flop's setup and hold times.

- When this happens, the META output may become metastable and remain in that state for an arbitrary time.

- However, we assume that the maximum duration of metastability after the clock edge is t_r .
- As long as the clock period is greater than t_r, plus the FF2's setup time, SYNCIN becomes a synchronized copy of the asynchronous input on the next clock tick without ever becoming metastable itself.
- The SYNCIN signal is distributed as required to the rest of the system.

Analysis of Metastable Timing



- As long as the D input changes outside the decision window, as in (a), the manufacturer guarantees that the output will change and settle to a valid logic state before time t_{pd}.
- If D changes inside the decision window, as in (b), metastability may occur and persist until time t_r.

Metastable Timing - cont.

- Theoretical research suggests, and experimental research has confirmed, that when asynchronous inputs change during the decision window, the duration of metastable outputs is governed by an exponential formula:

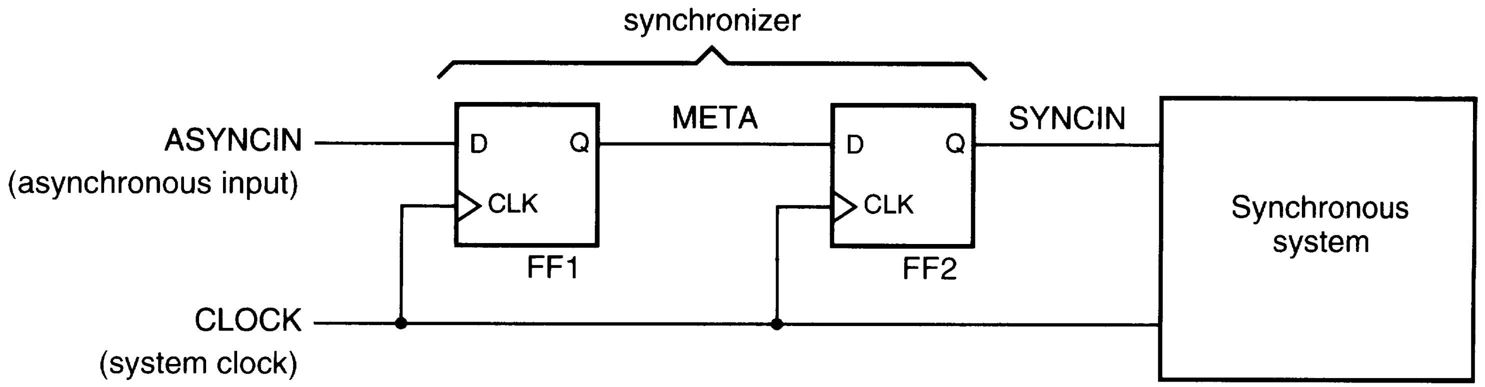
$$\mathrm{MTBF}(t_r) = \frac{\exp\left(t_r / \tau\right)}{T_0 \cdot f \cdot a}$$

- Here MTBF(t_r) is the mean time between synchronizer failures. A failure occurs if metastability persists beyond time t_r, after a clock edge. This MTBF depends on
  - *f,* the frequency of the flip-flop clock;
  - *a*, the number of asynchronous input changes per second applied to the flip-flop;
  - T_0 and τ *(Tau)*, are constants that depend on the electrical characteristics of the flip-flop.
- For a typical 74LS74, T_0 ~ 0.4 s and τ ~ 1.5 ns.

Example

- Suppose that we build a microprocessor system with a 10MHz clock and use this circuit to synchronize an asynchronous input.
- Let us assume that the D flip-flops are 74LS74s with the setup time of $t_{setup} = 20$ ns.
- $t_r = t_{clock} - t_{setup} = 100\text{-}20 = 80$ ns
- If the asynchronous input changes 100,000 times per second, then the synchronizer MTBF is:

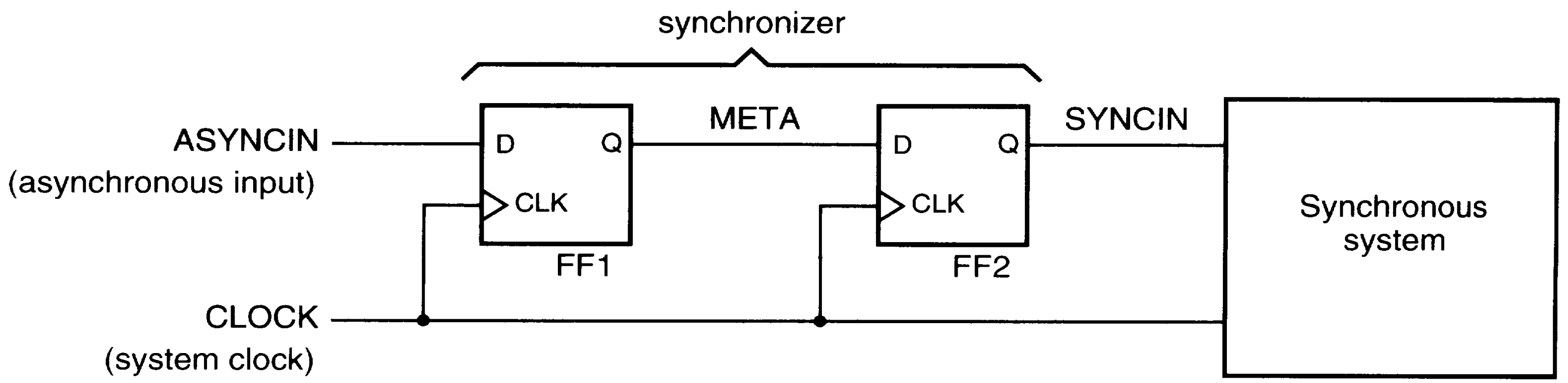
MTBF(80ns) =



Example

- Suppose that we build a microprocessor system with a 10MHz clock and use this circuit to synchronize an asynchronous input.
- Let us assume that the D flip-flops are 74LS74s with the setup time of $t_{setup} = 20$ ns.
- $t_r = t_{clock} - t_{setup} = 100\text{-}20 = 80$ ns
- If the asynchronous input changes 100,000 times per second, then the synchronizer MTBF is:

$$\mathrm{MTBF(80ns)} = \frac{\exp(80/1.5)}{0.4 \cdot 10^{7} \cdot 10^{5}} = 3.6 \cdot 10^{11}\, s$$



Example (cont.)

- That's not bad, about 100 centuries between failures! Of course, if we're lucky enough to sell 10,000 copies of our system, one of them will fail in this way every year.
- Suppose we upgrade our system to use a faster microprocessor chip with a clock speed of 16 MHz
- With a clock period of 62.5 ns, the new synchronizer MTBF is

MTBF(42.5ns) =

- we're likely to discover the problem in the engineering lab before the product ships!

Example (cont.)

- That's not bad, about 100 centuries between failures! Of course, if we're lucky enough to sell 10,000 copies of our system, one of them will fail in this way every year.
- Suppose we upgrade our system to use a faster microprocessor chip with a clock speed of 16 MHz
- With a clock period of 62.5 ns, the new synchronizer MTBF is

$$\mathrm{MTBF}(42.5\mathrm{ns}) = \frac{\exp(42.5/1.5)}{0.4 \cdot 1.6 \cdot 10^{7} \cdot 10^{5}} = 3.1s$$

- we're likely to discover the problem in the engineering lab before the product ships!

MTBF for designs with multiple synchronisers

- The overall design MTBF can be determined by the MTBF of each synchronizer chain in the design.
- The failure rate for a synchronizer is 1/MTBF, and the failure rate for the entire design is calculated by adding the failure rates for each synchronizer chain:

$$failure\_rate_{design} = \frac{1}{MTBF_{design}} = \sum_{i=1}^{number\_of\_chains} \frac{1}{MTBF_i}$$

$$\text{The design metasbilit y MTBF is then} = \frac{1}{failure\_rate_{design}}$$

Multiple chains – Example 1

- This has 10 chains with the same MTBF of 10,000 years.
- The failure rate for the design is the sum of the failure rates for each chain, where the failure rate is 1/MTBF
  - Metastability failure rate of 10 chains × 1/10,000 years = 0.001
  - Therefore the design MTBF is 1000 years

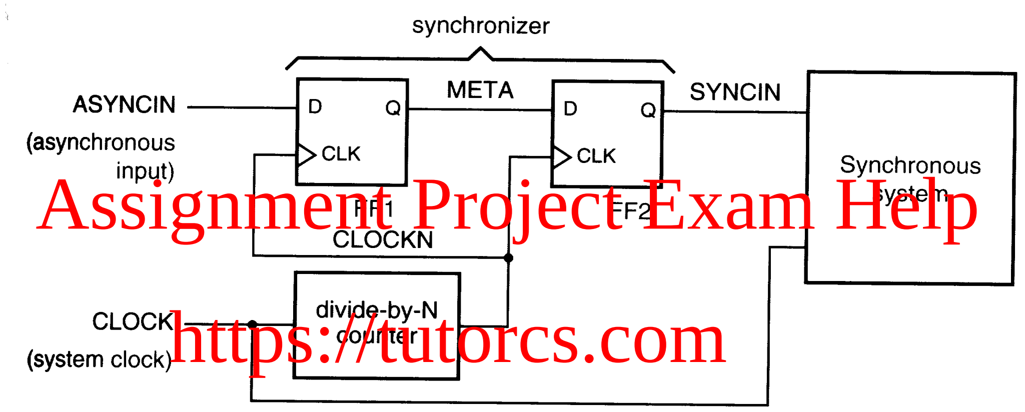
Multiple chains – Example 2

- This has 9 chains with the MTBF of 1,000,000 years and 1 chain with a MTBF of 100 years.
- The failure rate for the design is the sum of the failure rates for each chain, where the failure rate is 1/MTBF
- Metastability failure rate of 9 chains × 1/1,000,000 + 1/100 = 0.01009
- Therefore the design MTBF is about 99 years
  - Just slightly less than the MTBF of the worst chain

Other Synchronizer designs

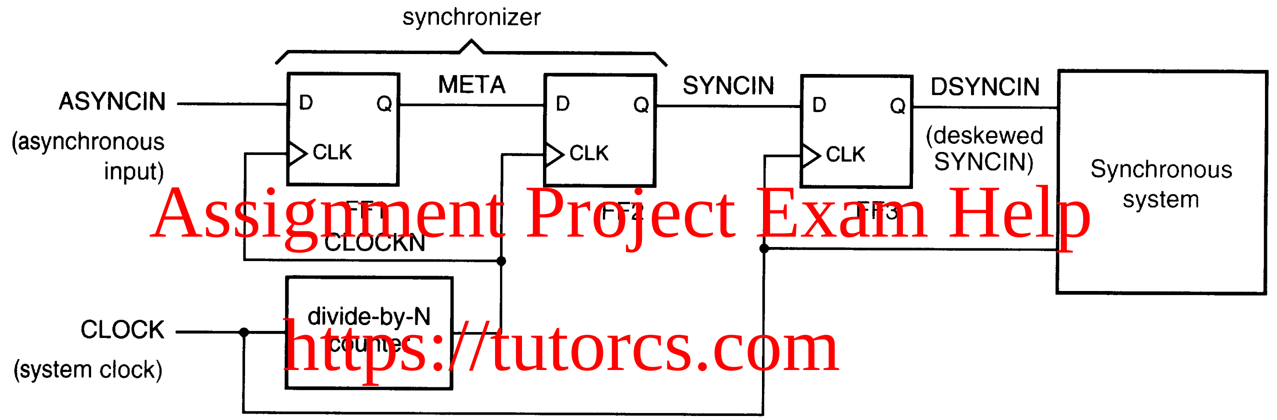
- The simplest solution, which works for most design requirements, is simply to use a flip-flop from a faster technology.
- Nowadays much faster technologies are available for flip-flops, whether discrete or embedded in PLDs, FPGAs, or ASICs.
- The second solution is to increase the value of t_r, in the MTBF equation.
  - For a given system clock, the best value we can obtain for t_r using the previous circuit is t_{clk}, if FF2 has a setup time of 0.
  - However, we can get values of t_r on the order of $n.t_{clk}$ by using a multiple cycle synchronizer circuit. Here we divide the system clock by n to obtain a slower synchronizer clock and longer $t_r = (n.t_{clk}) - t_{setup}$.
  - Usually a value of n = 2 or n = 3 gives adequate synchronizer reliability

Multi-cycle synchroniser



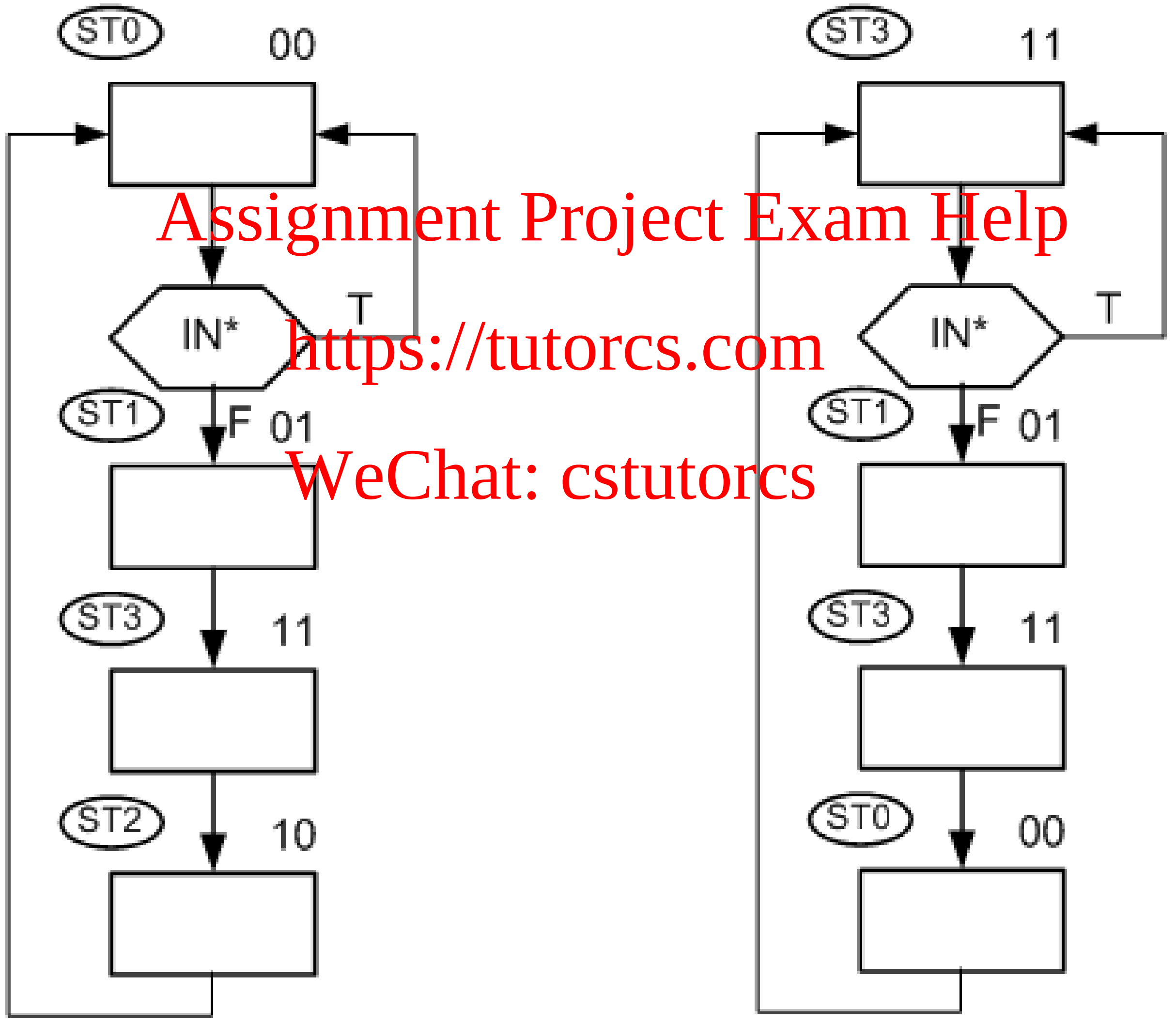
- Note that the edges of CLOCKN will lag the edges of CLOCK because CLOCKN comes from the Q output of a counter flip-flop that is clocked by CLOCK.
- This means that SYNCIN, in turn, will be delayed or skewed relative to other signals in the synchronous system that come directly from flip-flops clocked by CLOCK.
- If SYNCIN goes through additional combinational logic in the synchronous system before reaching its flip-flop inputs, their setup time may be inadequate.

De-skewed Multi-cycle Synchronizer



- Here, SYNCIN is re-clocked by CLOCK using FF3 to produce DSYNCIN, which will have the same timing as other flip-flop outputs in the synchronous system.
- Of course, the delay from CLOCK to CLOCKN must still be short enough that SYNCIN meets the setup time requirement of FF3.
- In an n-cycle synchronizer, the larger the value of n, the longer it takes for an asynchronous input change to be seen by the synchronous system.

Do the Asynchronous inputs cause problems in these circuits?


ST0
00
IN*
T
ST1
F 01
ST3
11
ST2
10
ST3
11
IN*
T
ST1
F 01
ST3
11
ST0
00