Clock Domain Crossing

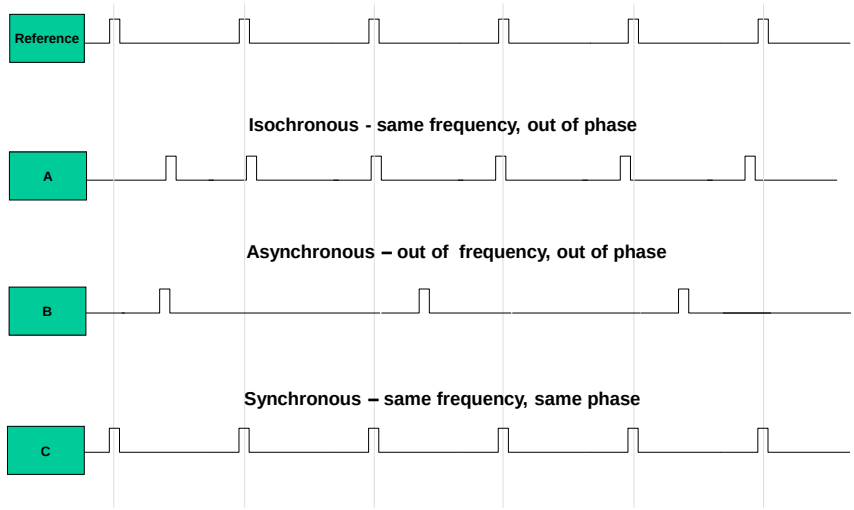
2014 - 2020

Ando Ki, Ph.D.
(adki@future-ds.com)

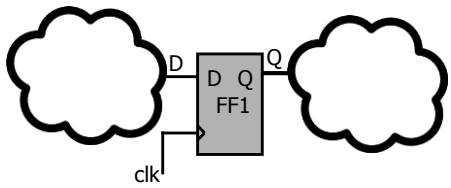
Agenda

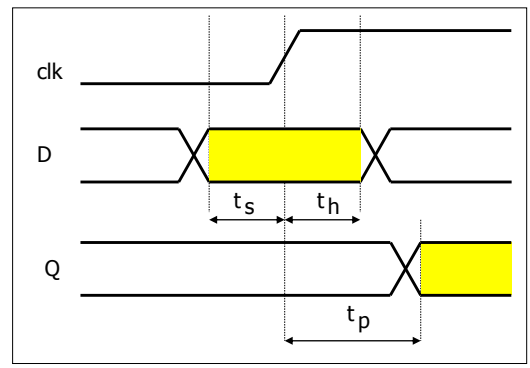
- Clock frequency and phase
- Setup and hold time
- Slack
- Metastability
- Clock domain crossing

Clock frequency and phase



Setup and hold time





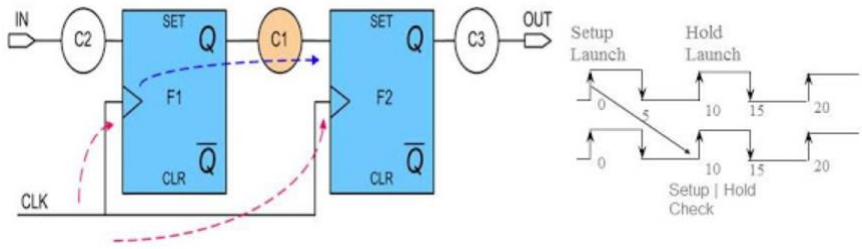
Setup time: The time data must remain stable before clock transitions on a flip-flop.

Hold time: The length of time of a flip-flop's input data must remain stable after the each clock transition.

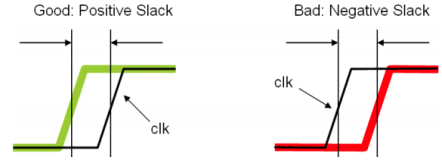
Violating the setup and hold time will create a metastable state, which is a condition giving a random data.

		min	typ	max
Setup		0ns		
Hold		0.2ns		
propagation	PLH	2ns	6ns	10ns
	PHL	2ns	6ns	10ns

Slack

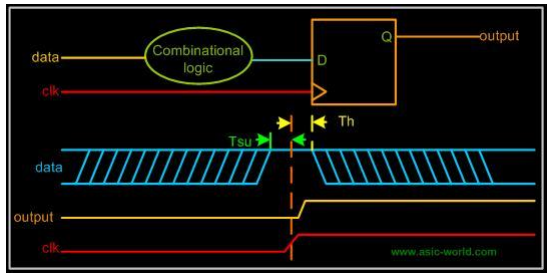


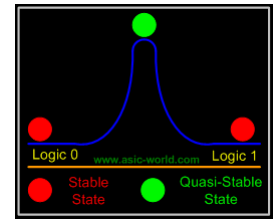
- Setup slack = (data required time) – (data arrival time)
- Hold slack = (data arrival time) – (data required time)
  - (data required time for setup) = (clock period) – (setup) ➔ signal should be valid before rising edge of clock
  - (data required time for hold) = (clock period) + (hold)
- Negative setup slack implies that design doesn't achieve the constrained frequency and timing. This is called as setup violation



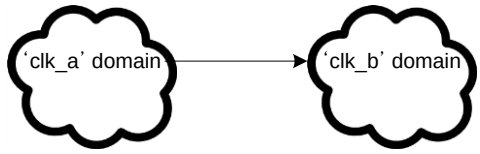
Metastability

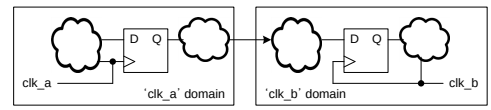
- Whenever the input signal D does not meet the Tsu (Setup) and Th (Hold) of the given D flip-flop, metastability occurs.

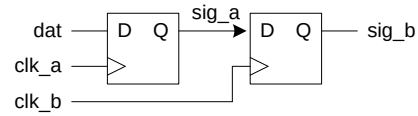




CDC: Clock-Domain Crossing



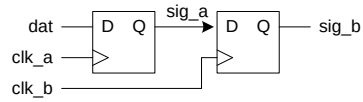


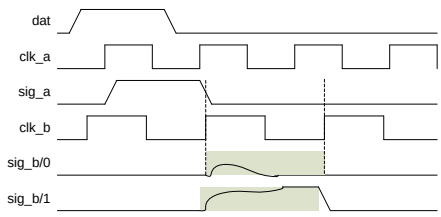


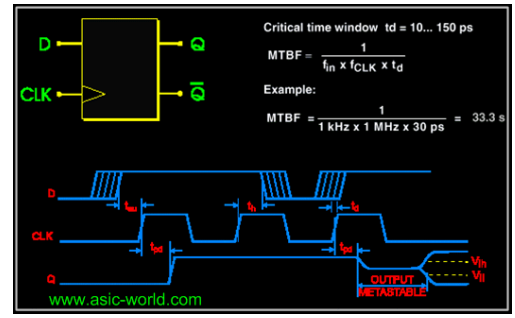

```
localparam STAGES=3;
reg [STAGES:0] sync={STAGES+1{INIT_VALUE[0]}};
  integer idx;
  always @ (posedge clk or negedge reset_n) begin
  if (reset_n==1'b0) begin
     sync <= {STAGES+1{INIT_VALUE[0]}};
  end else begin
     sync[0] <= sig_in;
     for (idx=1; idx<=STAGES; idx=idx+1) begin
        sync[idx] <= sync[idx-1];
     end
  end // if
  end // always
  assign sig_out = sync[STAGES];
```

Multi-stage synchronizer

CDC: Meta-stability



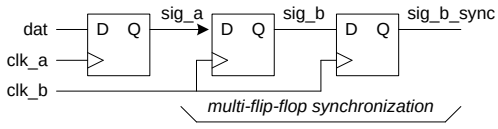




MTBF – Mean Time Between Failures

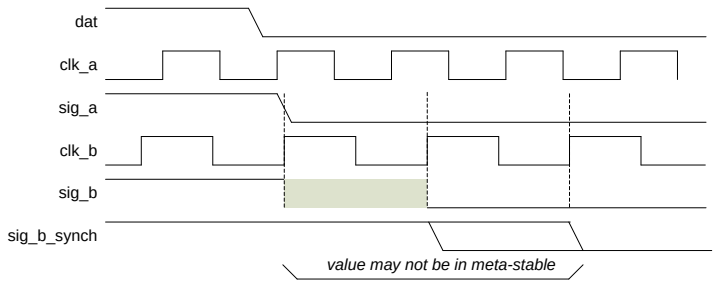
'sig_b' can be settle down to '0' or '1' depending on situation (fan-out, temperature, ...).

CDC: Multi-flip-flop synchronization



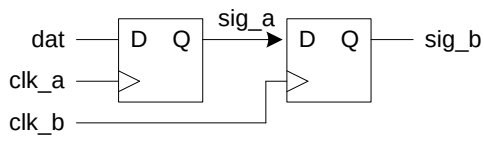
Multi-stage synchronizer

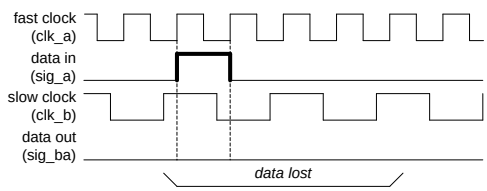
```
localparam STAGES=3;
reg [STAGES:0] sync={STAGES+1{INIT_VALUE[0]}};
  integer idx;
  always @ (posedge clk or negedge reset_n) begin
  if (reset_n==1'b0) begin
     sync <= {STAGES+1{INIT_VALUE[0]}};
  end else begin
     sync[0] <= sig_in;
     for (idx=1; idx<=STAGES; idx=idx+1) begin
        sync[idx] <= sync[idx-1];
     end
  end // if
  end // always
  assign sig_out = sync[STAGES];
```

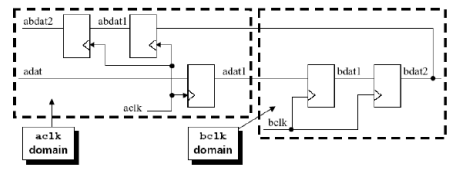


CDC: data loss problem

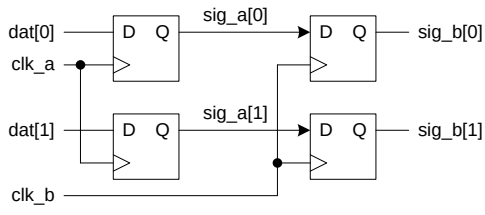
- Data can be lost with fast-to-slow clock domain
- Solutions
  - Edge detection
  - Feedback or handshake

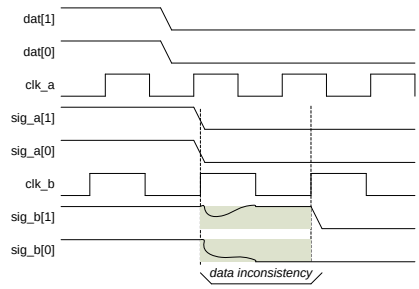






CDC: Data inconsistency





- Parallel data can be inconsistency due to meta-stability
- Solutions
  - Synchronization and ignore some intermediate cycles
  - MUX synchronization
  - Use single-bit changing code, such an grey code
    - But not applicable for all applications