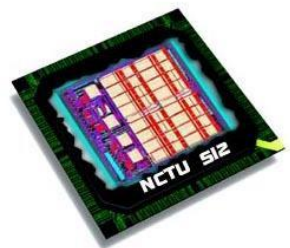


STATIC TIMING ANALYSIS

NYCU-EE IC LAB FALL-2021

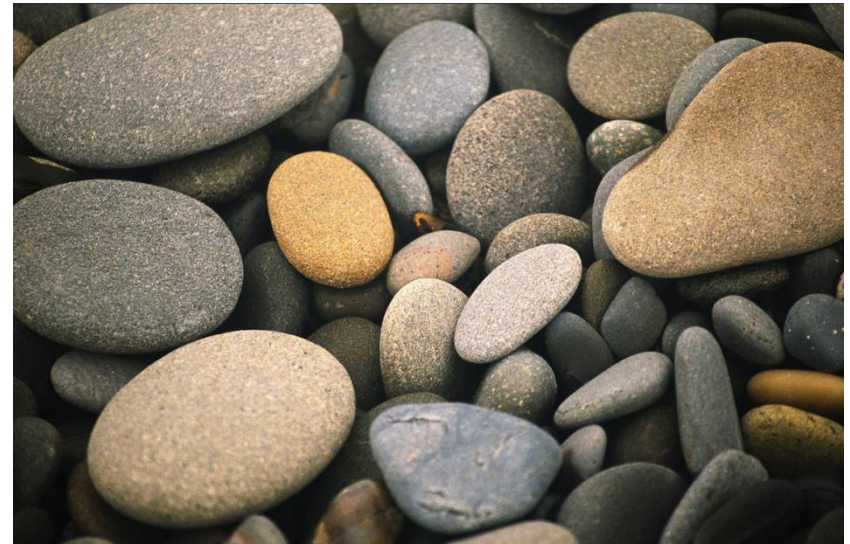
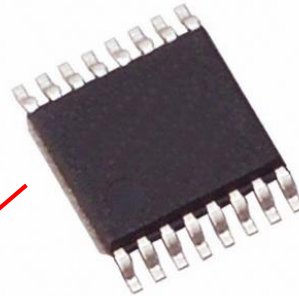


Lecturer: Hai-Feng Wang

- ✓ Timing analysis is integral part of ASIC/VLSI design flow. Anything else can be compromised but not timing!

Timing correct !

Timing error !



Outline

✓ **Timing Analysis Considering Clk Latency**

- STA vs DTA
- CLK latency
- CLK uncertainty

✓ **Multicycle Path Specification**

- Specify Multicycle Path
- Single clock system
- Multiple clocks system

✓ **Clock Domain Crossing (CDC)**

✓ **Appendix**



✓ **Timing Analysis Considering Clk Latency**

- STA vs DTA
- CLK latency
- CLK uncertainty

✓ **Multicycle Path Specification**

- Specify Multicycle Path
- Single clock system
- Multiple clocks system

✓ **Clock Domain Crossing (CDC)**

✓ **Appendix**



STA vs DTA

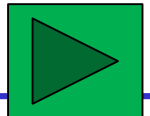
✓ STA(Static Timing Analysis)

- Verify the timing of a digital design based on a **statistic** method without requiring a simulation of the full circuit
- STA is a **complete** and **exhaustive** verification of all timing checks of a design because it **analyzes every timing paths in design according to constraints.**
- check that if there is any violation on setup/hold time in the timing path, glitches, critical path and clock skew.
- STA aims at different corner to analysis, so the analysis time is faster than DTA.

✓ DTA (Dynamic Timing Analysis)

- Quality(Coverage) of the Dynamic Timing Analysis (DTA) increases with the increase of **input test vectors.**
- The growing size of test vectors increase simulation time, especially in post-sim.

Review setup/hold time



Clock Latency

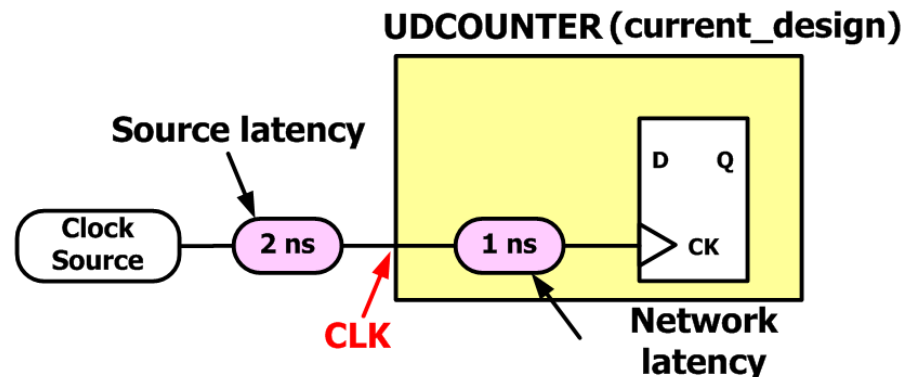
✓ Two types of clock latency

– Clock source latency

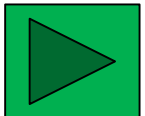
- Clock source latency is the delay from the **clock source** to the **clock definition point(CLK)**.
- Source latency could represent either on-chip or off-chip latency.

– Clock network latency

- from the **clock definition point(CLK)** to the **clock pin of a flip-flop**.



Appendix

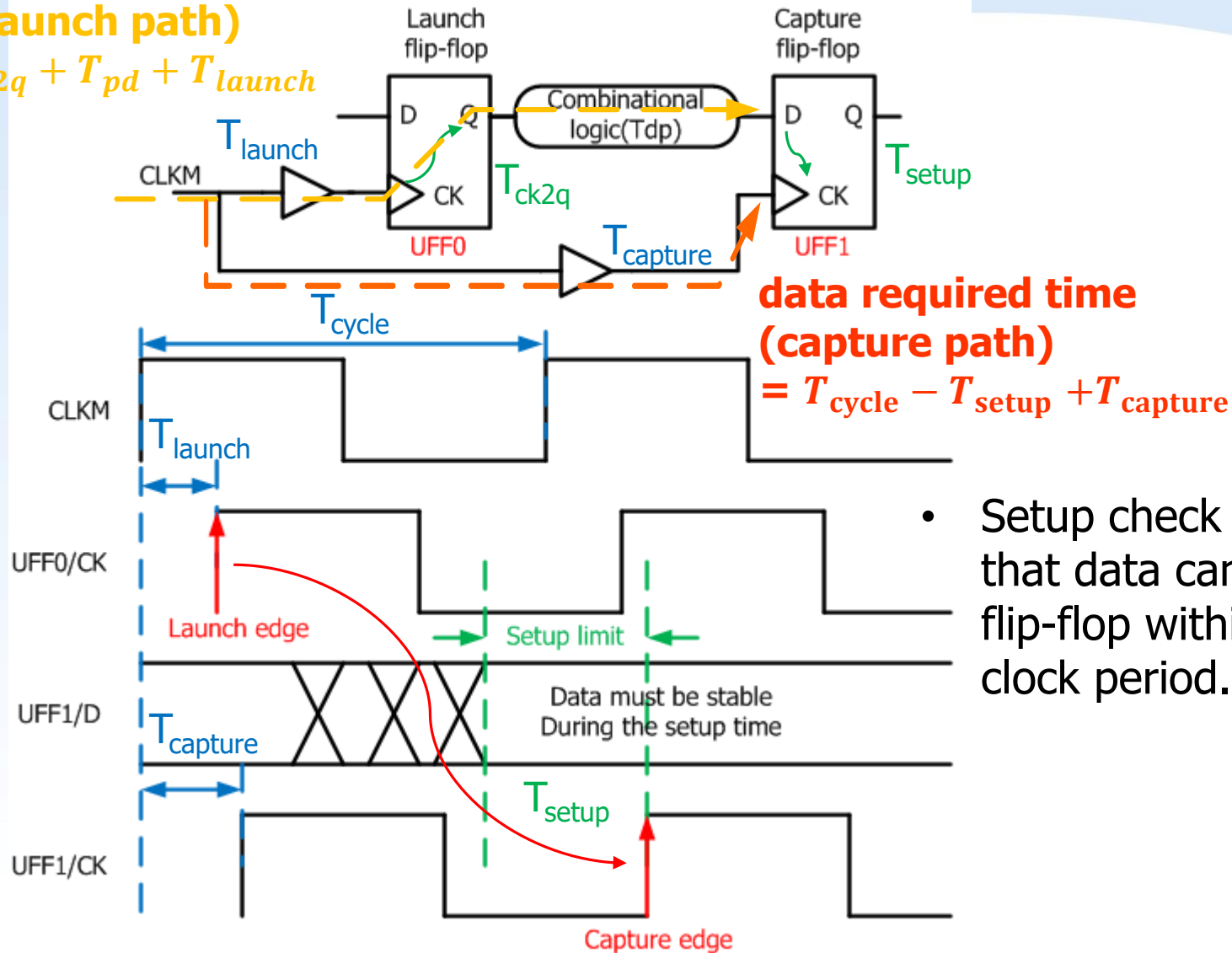


Setup time issue

data arrival time

(Launch path)

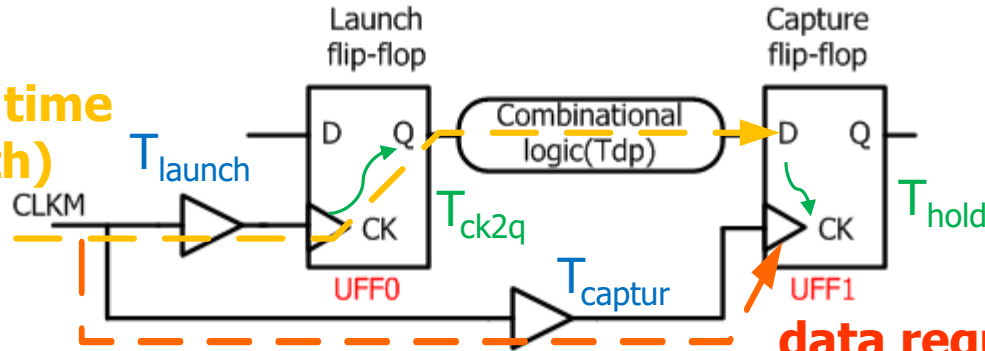
$$= T_{clk2q} + T_{pd} + T_{launch}$$



- Setup check ensures that data can arrive at a flip-flop within the given clock period.

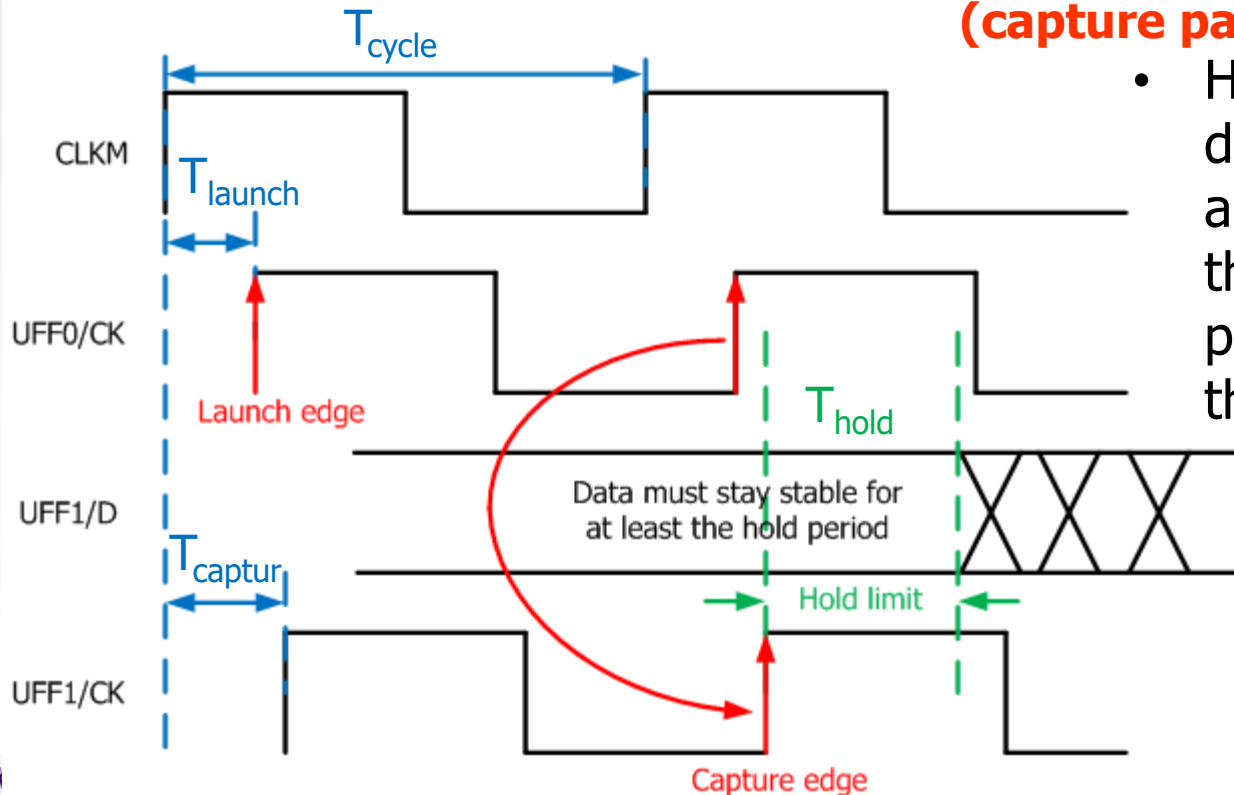
Hold time issue

data arrival time
(Launch path)



data required time
(capture path)

- Hold check ensures that data is held for at least a minimum time so that there is no unexpected pass-through of data through a flip-flop.



Clock Latency_(cont.)

✓ Example

```
dc_shell>current_design UDCOUNTER
```

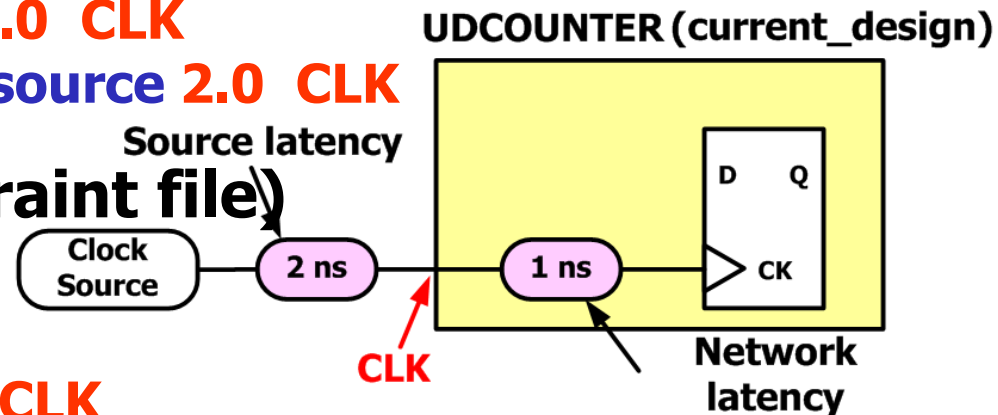
```
dc_shell>set_clock_latency 1.0 CLK
```

```
dc_shell>set_clock_latency -source 2.0 CLK
```

✓ .sdc file (design constraint file)

✓ Check clock latency

```
dc_shell>report_clock -skew CLK
```



Report : clock_skew

Design : UDCOUNTER

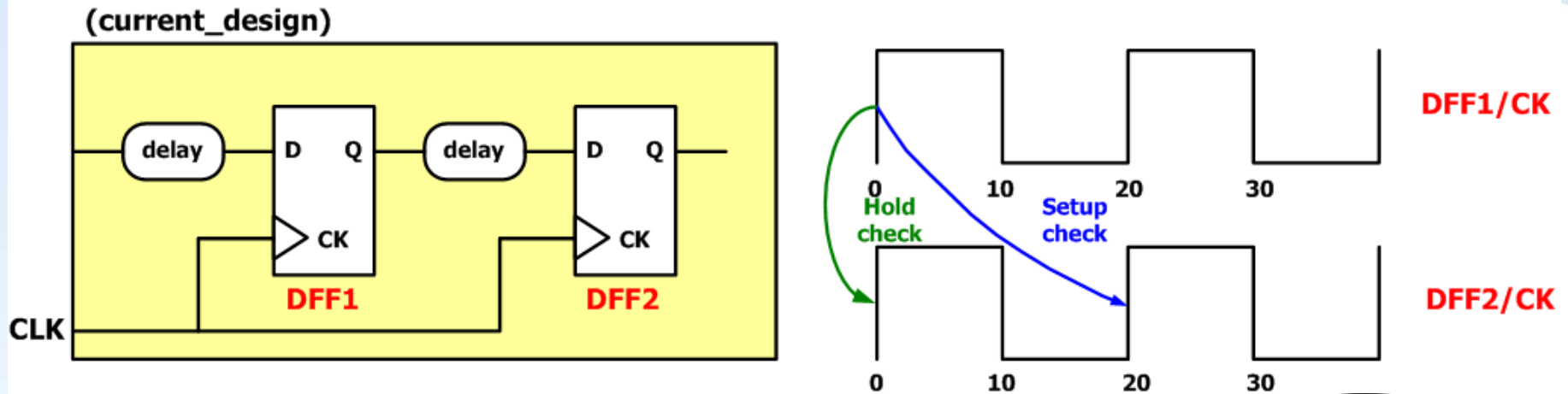
| Object | Rise | Fall | Min Rise | Min fall | Uncertainty | |
|--------|-------|-------|----------|----------|-------------|-------|
| | Delay | Delay | Delay | Delay | Plus | Minus |
| CLK | 1.0 | 1.0 | 1.0 | 1.0 | - | - |

| Object | Max Source Latency | | | | Min Source Latency | | | |
|--------|--------------------|------------|-----------|-----------|--------------------|------------|-----------|-----------|
| | Early Rise | Early Fall | Late Rise | Late Fall | Early Rise | Early Fall | Late Rise | Late Fall |
| CLK | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |

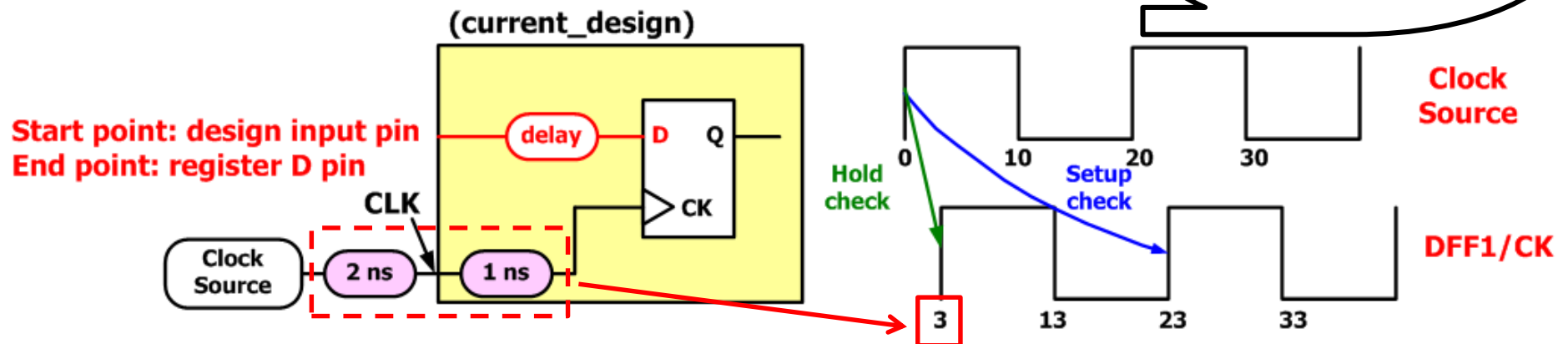


Clock Latency_(cont.)

– Timing checks for ideal case



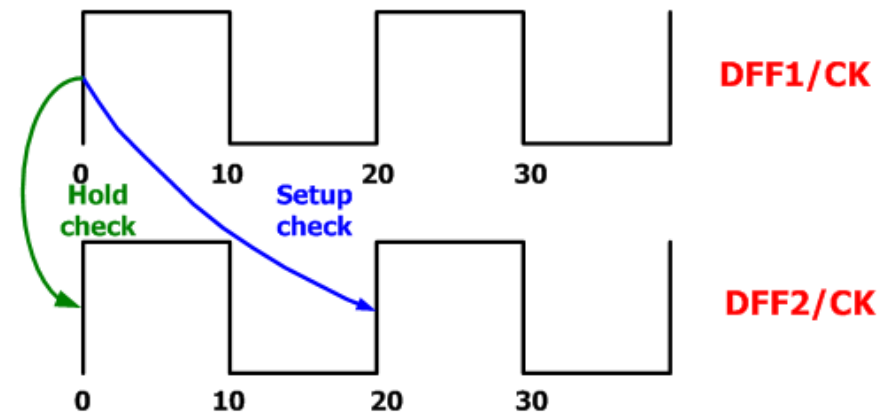
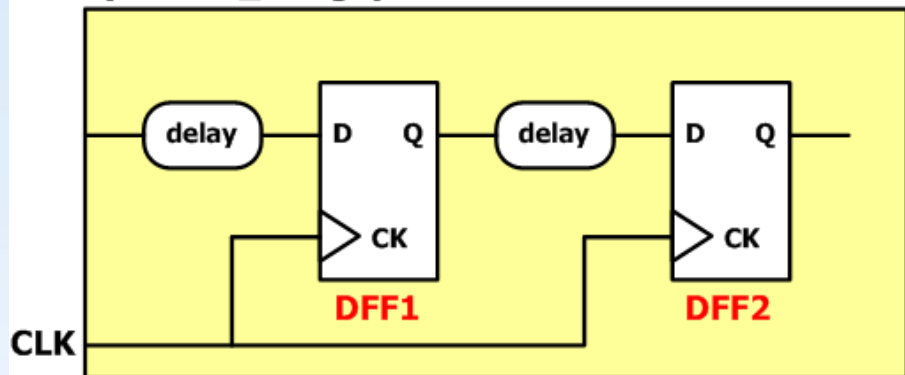
– Timing checks with clock network latency



Clock Latency_(cont.)

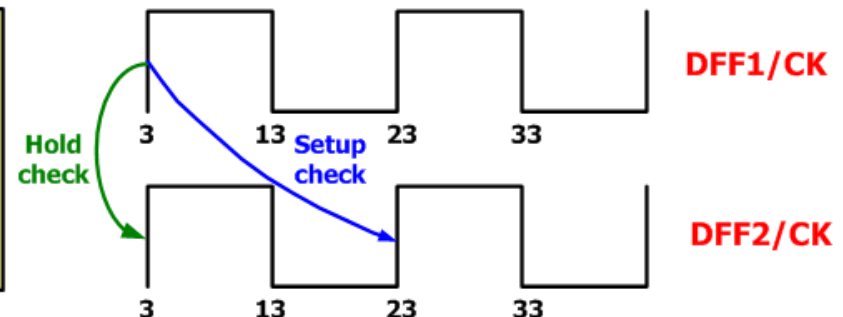
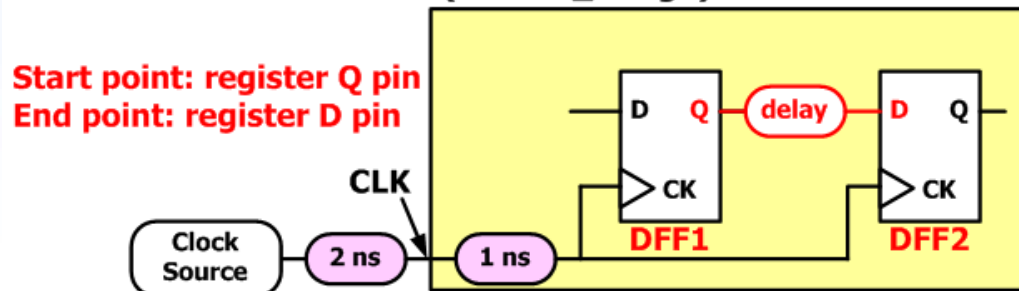
– Timing checks for ideal case

(current_design)



– Timing checks with clock network latency

(current_design)



Setup time check same
Hold time check same

Clock Latency(cont.)

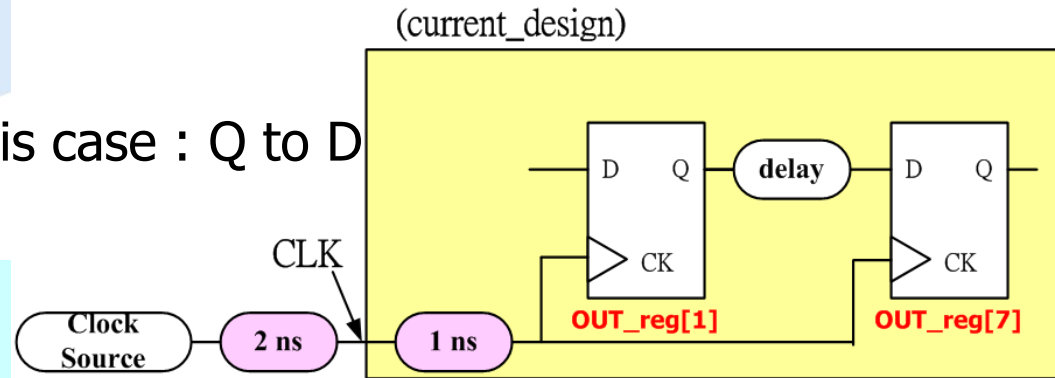
✓ Example

- Setup timing report for this case : Q to D

dc_shell>report_timing

```
*****
Report : timing -path full -delay max -max_paths 1
Design : UDCOUNTER
*****
```

| Point | Incr | Path |
|--------------------------------------|-------|---------|
| clock CLK (rise edge) | 0.00 | 0.00 |
| clock network delay (ideal) | 3.00 | 3.00 |
| OUT_reg[1]/CK (EDFFX1) | 0.00 | 3.00 r |
| OUT_reg[1]/Q (EDFFX1) | 0.36 | 3.36 f |
| add_70/A[1] (UDCOUNTER_DW01_add_8_0) | 0.00 | 3.36 f |
| ...(ignored) | | |
| U274/Y (DLY4X1) | 1.10 | 12.63 f |
| OUT_reg[7]/D (EDFFX1) | 0.00 | 12.63 f |
| data arrival time | | 12.63 |
| clock CLK (rise edge) | 20.00 | 20.00 |
| clock network delay (ideal) | 3.00 | 23.00 |
| OUT_reg[7]/CK (EDFFX1) | 0.00 | 23.00 r |
| library setup time | -0.89 | 22.11 |
| data required time | | 22.11 |
| data required time | | 22.11 |
| data arrival time | | -12.63 |
| slack (MET) | | 9.48 |



Clock network delay
 = source latency + network latency
 = 2.0 + 1.0 = 3.0

critical path of **setup time** check

- start point : register Q pin
- end point : register D pin

dc_shell> report_timing -max_paths 5



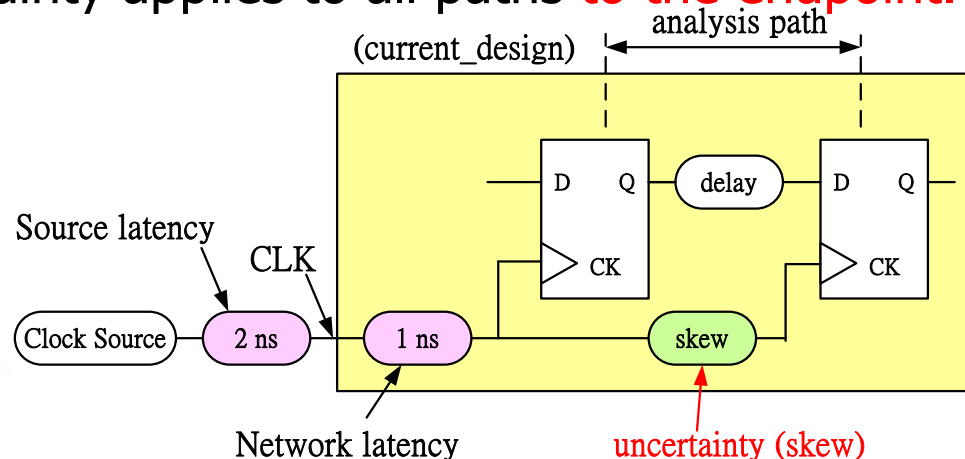
Set Clock Uncertainty

✓ Clock uncertainty (clock skew)

- This constraint can be used to model various factors that can influence the effective clock period.
- These factors may come from clock jitter and any other factors that one may want to include for timing analysis.
- Leave some **timing margin** for **APR routing** imperfection.

✓ Clock skew of **single** clock network

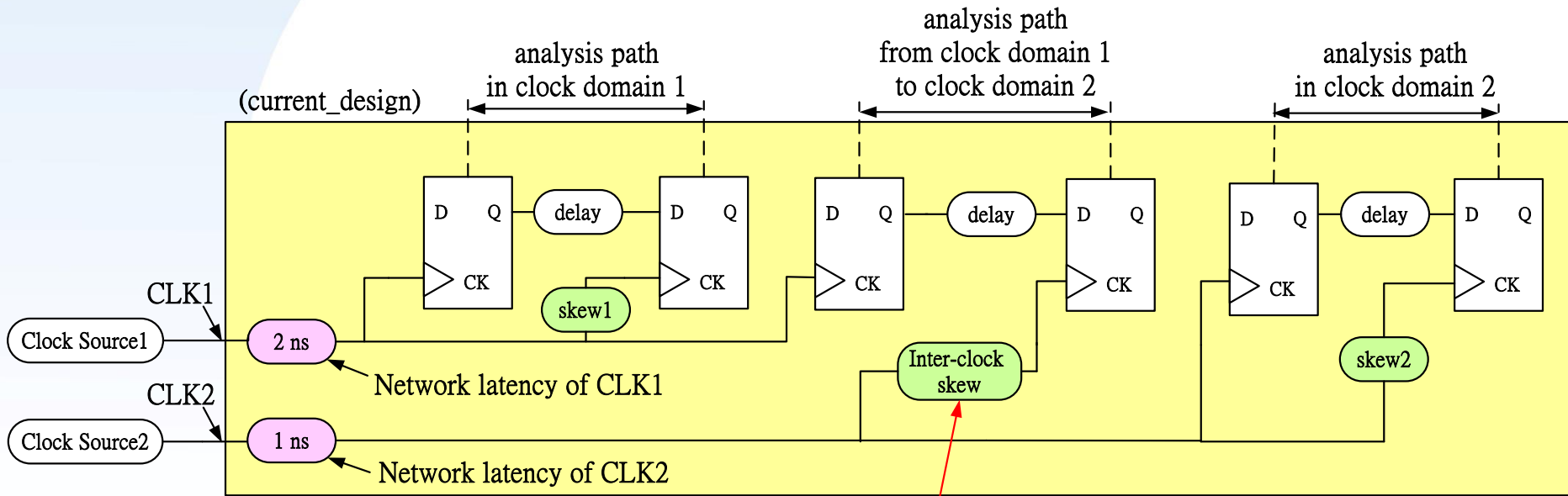
- **Simple uncertainty** means the setup uncertainty and hold uncertainty applies to all paths **to the endpoint**.



Set Clock Uncertainty_(cont.)

✓ Inter-clock skew of **multiple** clock networks

- **Inter-clock uncertainty** means the setup uncertainty and hold uncertainty applies to all paths **from start point of clock domain1 to the endpoint of clock domain2**
- skew1 and skew2 are simple uncertainty of clock domain1 and clock domain 2



Inter-clock uncertainty
(if not defined, it will be skew2)

Set Clock Uncertainty_(cont.)

✓ Setting simple clock uncertainty

```
set_clock_uncertainty [-setup] [-hold] value {object_list}
```

✓ Setting inter-clock uncertainty

```
set_clock_uncertainty [-from from_clock] [-to to_clock] [-rise] [-fall] [-setup] [-hold]  
value
```

— Options

| arguments | Usage | Default |
|--------------------|--|----------------------------|
| [-setup] | If specified, uncertainty is applied only for setup check | Both setup and hold checks |
| [-hold] | If specified, uncertainty is applied only for hold check | Both setup and hold check |
| [-from from_clock] | Specifies the source clock for inter-clock uncertainty | |
| [-to to_clock] | Specifies the destination clock for inter-clock uncertainty | |
| [-rise] | Specifies uncertainty applied to the rising edge of destination clock | Both |
| [-fall] | Specifies uncertainty applied to the falling edge of destination clock | Both |



Set Clock Uncertainty_(cont.)

✓ Example:

```
module MULTICLOCK
(
  // Output Port
  OUT,
  // Input Port
  CLK1, CLK2, IN1, IN2
);

output OUT;
input CLK1;
input CLK2;
input IN1;
input IN2;

reg temp1, temp2, OUT;
wire CLK1, CLK2, data;

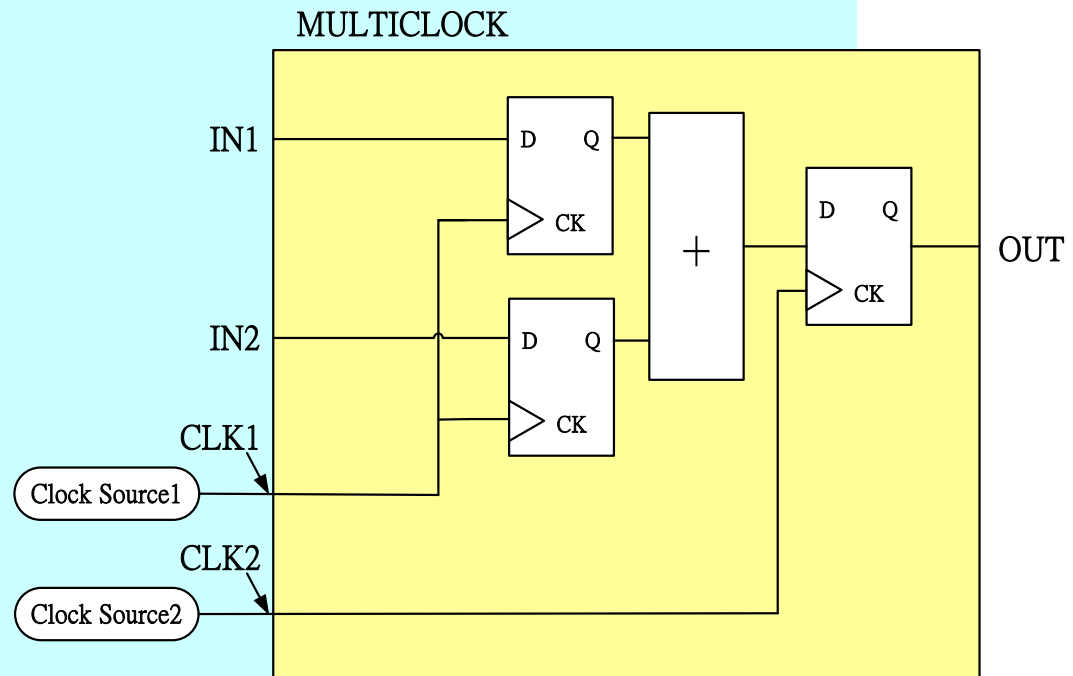
assign data = temp1 + temp2;

always @(posedge CLK1)
  temp1 <= IN1;

always @(posedge CLK1)
  temp2 <= IN2;

always @(posedge CLK2)
  OUT <= data;

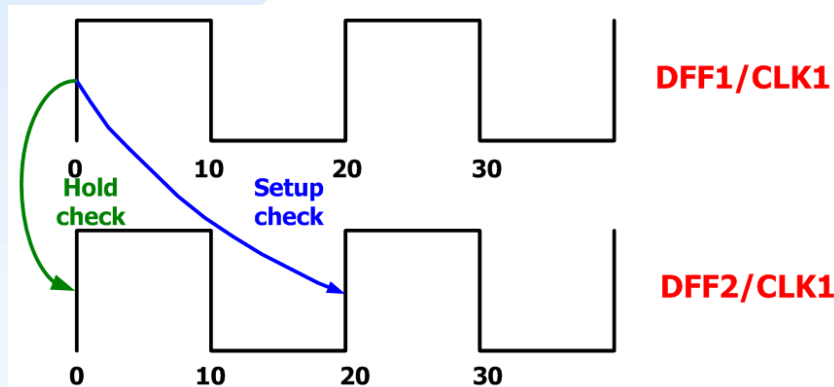
endmodule
```



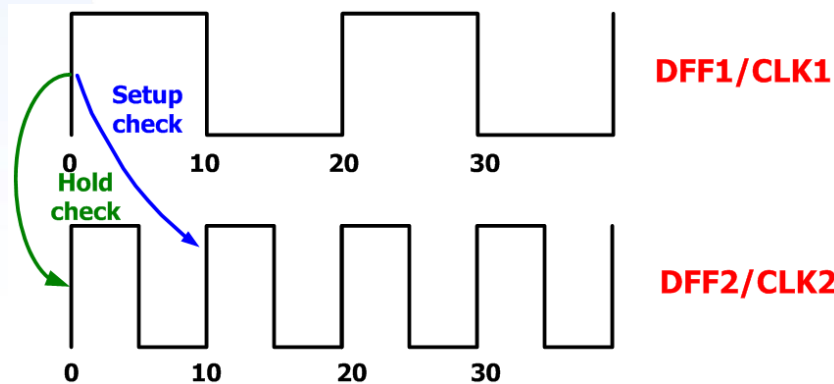
Set Clock Uncertainty_(cont.)

```
dc_shell>create_clock -period 20 CLK1
```

```
dc_shell>create_clock -period 10 CLK2
```



setup and hold check
in clock domain 1



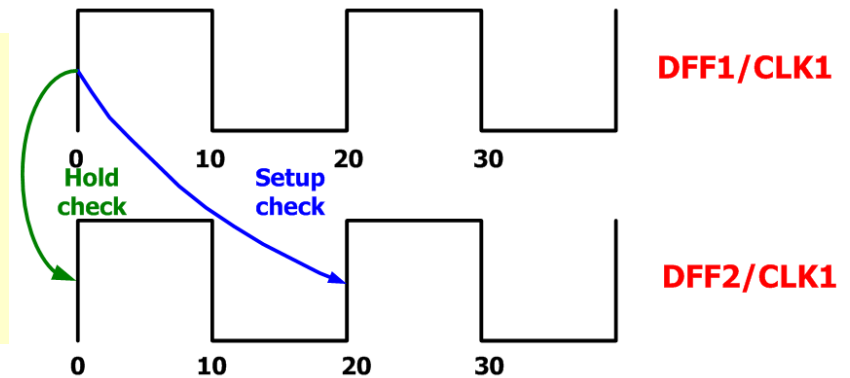
setup and hold check
from clock domain 1
to clock domain 2



Set Clock Uncertainty(cont.)

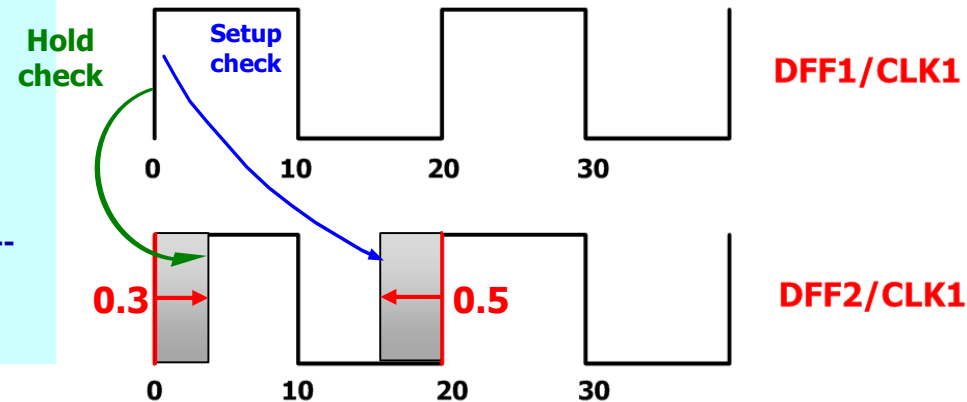
- During timing analysis, clock uncertainty will **tighten up** the timing constraints.
- **Plus** uncertainty is for **hold time** check.
- **Minus** uncertainty is for **setup time** check.

```
set_clock_uncertainty -setup 0.5 CLK1
set_clock_uncertainty -hold 0.3 CLK1
set_clock_uncertainty -setup 0.6 CLK2
set_clock_uncertainty -hold 0.4 CLK2
report_clock -skew
```



```
*****
Report : clock_skew
Design : MULTICLOCK
*****
```

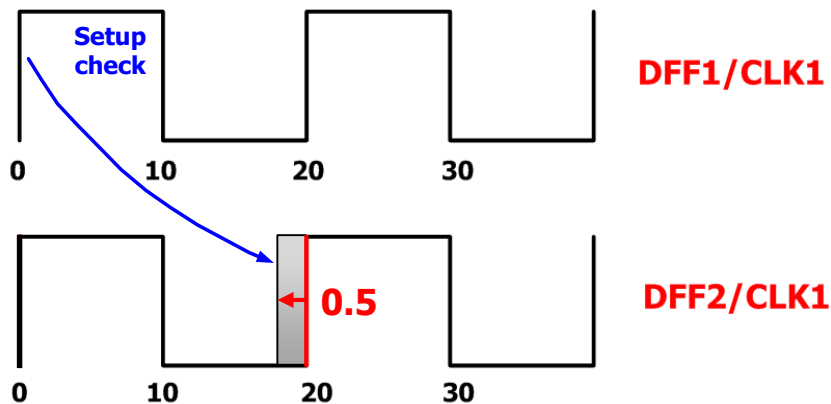
| Object | Rise Delay | Fall Delay | Min Rise Delay | Min fall Delay | Uncertainty Plus | Minus |
|--------|------------|------------|----------------|----------------|------------------|-------|
| CLK1 | - | - | - | - | 0.30 | 0.50 |
| CLK2 | - | - | - | - | 0.40 | 0.60 |



Set Clock Uncertainty_(cont.)

- Timing report (setup time check)

```
dc_shell>compile
dc_shell>report_timing
```



Minus uncertainty 0.5 of CLK1 for setup time check

Report : timing -path full -delay max -max_paths 1

Design : MULTICLOCK

Path Group: CLK1

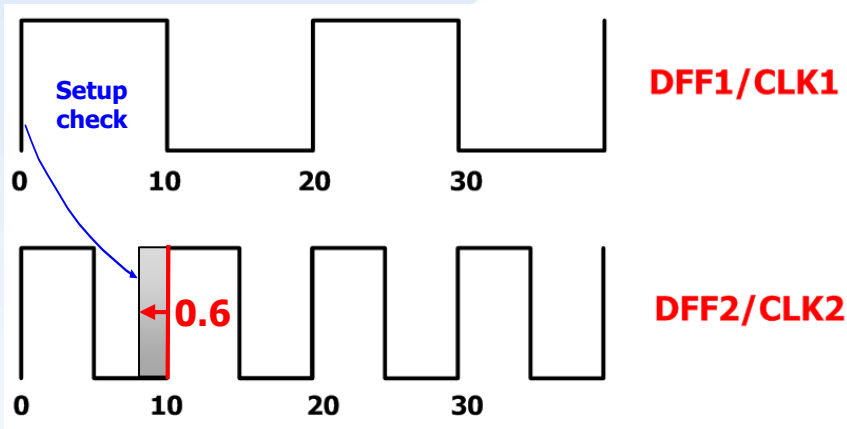
Clock domain : CLK1

Path Type: max

| Point | Incr | Path |
|--------------------------------------|-------|---------|
| ----- | | |
| clock (input port clock) (rise edge) | 0.00 | 0.00 |
| input external delay | 0.00 | 0.00 f |
| IN1 (in) | 0.00 | 0.00 f |
| U8/Y (DLY4X1) | 1.03 | 1.03 f |
| TEMP1_reg/D (DFFX1) | 0.00 | 1.03 f |
| data arrival time | | 1.03 |
| | | |
| clock CLK1 (rise edge) | 20.00 | 20.00 |
| clock network delay (ideal) | 0.00 | 20.00 |
| clock uncertainty | -0.50 | 19.50 |
| TEMP1_reg/CK (DFFX1) | 0.00 | 19.50 r |
| library setup time | -0.33 | 19.17 |
| data required time | | 19.17 |
| ----- | | |
| data required time | | 19.17 |
| data arrival time | | -1.03 |
| ----- | | |
| slack (MET) | | 18.13 |

Set Clock Uncertainty_(cont.)

- Timing report (setup time check)



Because **inter-clock uncertainty is not defined**, here compiler uses **uncertainty of CLK2 automatically**.

Clock domain : from CLK1 to CLK2

(cont.)

Path Group: CLK2

Path Type: max

| Point | Incr | Path |
|-----------------------------|-------|--------|
| clock CLK1 (rise edge) | 0.00 | 0.00 |
| clock network delay (ideal) | 0.00 | 0.00 |
| TEMP2_reg/CK (DFFX1) | 0.00 | 0.00 r |
| TEMP2_reg/Q (DFFX1) | 0.41 | 0.41 r |
| U5/Y (XOR2XL) | 0.40 | 0.81 f |
| U4/Y (DLY4X1) | 1.09 | 1.90 f |
| OUT_reg/D (DFFX1) | 0.00 | 1.90 f |
| data arrival time | | 1.90 |
| clock CLK2 (rise edge) | 10.00 | 10.00 |
| clock network delay (ideal) | 0.00 | 10.00 |
| clock uncertainty | -0.60 | 9.40 |
| OUT_reg/CK (DFFX1) | 0.00 | 9.40 r |
| library setup time | -0.33 | 9.07 |
| data required time | | 9.07 |
| data required time | | 9.07 |

```
set_clock_uncertainty -setup 0.5 CLK1
set_clock_uncertainty -hold 0.3 CLK1
set_clock_uncertainty -setup 0.6 CLK2
set_clock_uncertainty -hold 0.4 CLK2
report_clock -skew
```



Set Clock Uncertainty_(cont.)

- Timing report (setup time check)
- Add inter-clock uncertainty and re-compile

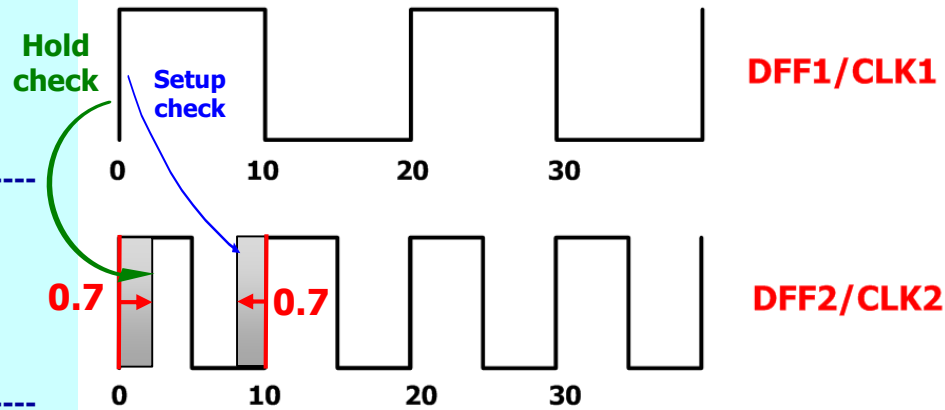
```
dc_shell>set_clock_uncertainty -from CLK1 -to CLK2 0.7
dc_shell>report_clock -skew
```

Report : clock_skew

Design : MULTICLOCK

| Object | Rise Delay | Fall Delay | Min Rise Delay | Min fall Delay | Uncertainty Plus | Uncertainty Minus |
|--------|------------|------------|----------------|----------------|------------------|-------------------|
| CLK1 | - | - | - | - | 0.30 | 0.50 |
| CLK2 | - | - | - | - | 0.40 | 0.60 |

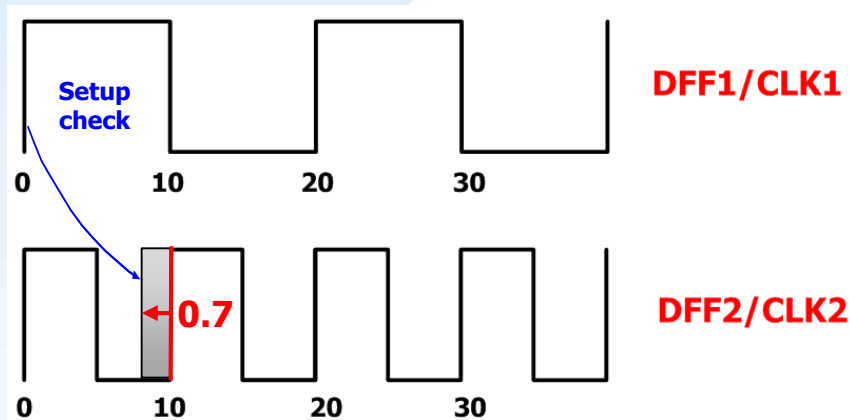
| From Clock | To Clock | Hold Uncertainty Rise | Hold Uncertainty Fall | Setup Uncertainty Rise | Setup Uncertainty Fall |
|------------|----------|-----------------------|-----------------------|------------------------|------------------------|
| CLK1 | CLK2 | 0.70 | 0.70 | 0.70 | 0.70 |



Set Clock Uncertainty_(cont.)

- Timing report (setup time check)

```
dc_shell>compile
dc_shell>report_timing
```



Because inter-clock uncertainty is **specified**, here compiler **uses inter-clock uncertainty**

**Clock domain :
from CLK1 to CLK2**

| Path Group: CLK2 | | |
|-----------------------------|-------|--------|
| Path Type: max | | |
| Point | Incr | Path |
| ----- | | |
| -- | | |
| clock CLK1 (rise edge) | 0.00 | 0.00 |
| clock network delay (ideal) | 0.00 | 0.00 |
| TEMP2_reg/CK (DFFX1) | 0.00 | 0.00 r |
| TEMP2_reg/Q (DFFX1) | 0.41 | 0.41 r |
| U13/Y (XOR2XL) | 0.40 | 0.81 f |
| U12/Y (DLY4X1) | 1.09 | 1.90 f |
| OUT_reg/D (DFFX1) | 0.00 | 1.90 f |
| data arrival time | | 1.90 |
| ----- | | |
| clock CLK2 (rise edge) | 10.00 | 10.00 |
| clock network delay (ideal) | 0.00 | 10.00 |
| inter-clock uncertainty | -0.70 | 9.30 |
| OUT_reg/CK (DFFX1) | 0.00 | 9.30 r |
| library setup time | -0.33 | 8.97 |
| data required time | | 8.97 |
| ----- | | |
| -- | | |
| data required time | | 8.97 |
| data arrival time | | -1.90 |
| ----- | | |
| -- | | |
| slack (MET) | | 7.07 |

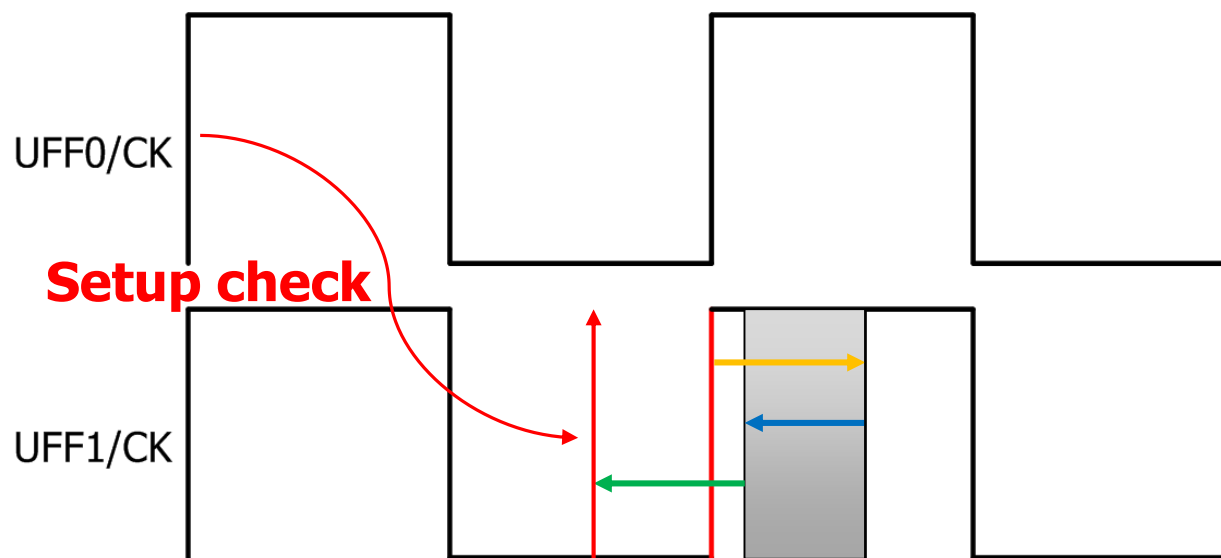


Summary timing check behavior

✓ Positive edge trigger circuit timing check

– Setup check

- check_edge : $\text{ideal_clock_rise_edge}$
+ $\text{rise_delay}(\text{network latency} + \text{source latency})$
– $\text{setup_uncertainty}(\text{minus uncertainty})$
– $T_{\text{setup}}(\text{library setup time})$

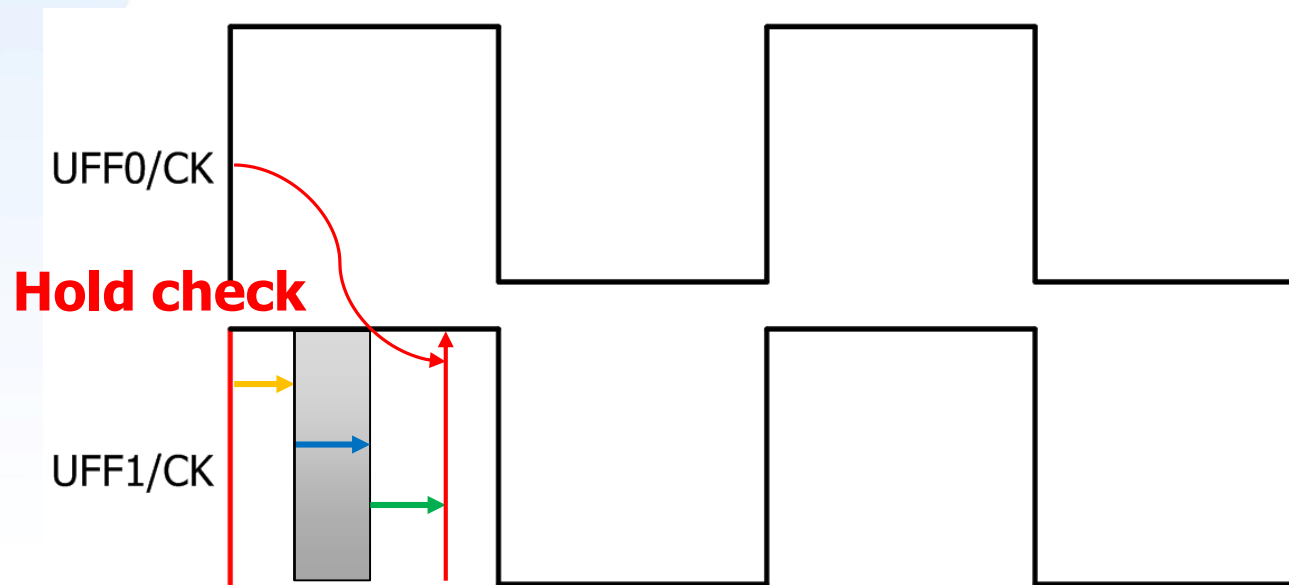


Summary timing check behavior

✓ Positive edge trigger circuit timing check

— Hold check

- check_edge : **ideal_clock_rise_edge**
+ rise_delay(network latency+source latency)
+ hold_uncertainty(plus uncertainty)
+ T_{hold} (library hold time)



✓ **Timing Analysis Considering Clk Latency**

- STA vs DTA
- CLK latency
- CLK uncertainty

✓ **Multicycle Path Specification**

- Specify Multicycle Path
- Single clock system
- Multiple clocks system

✓ **Clock Domain Crossing (CDC)**

✓ **Appendix**



✓ Timing Analysis Considering Clk Latency

- STA vs DTA
- CLK latency
- CLK uncertainty

✓ Multicycle Path Specification

- Specify Multicycle Path
- Single clock system
- Multiple clocks system

✓ Clock Domain Crossing (CDC)

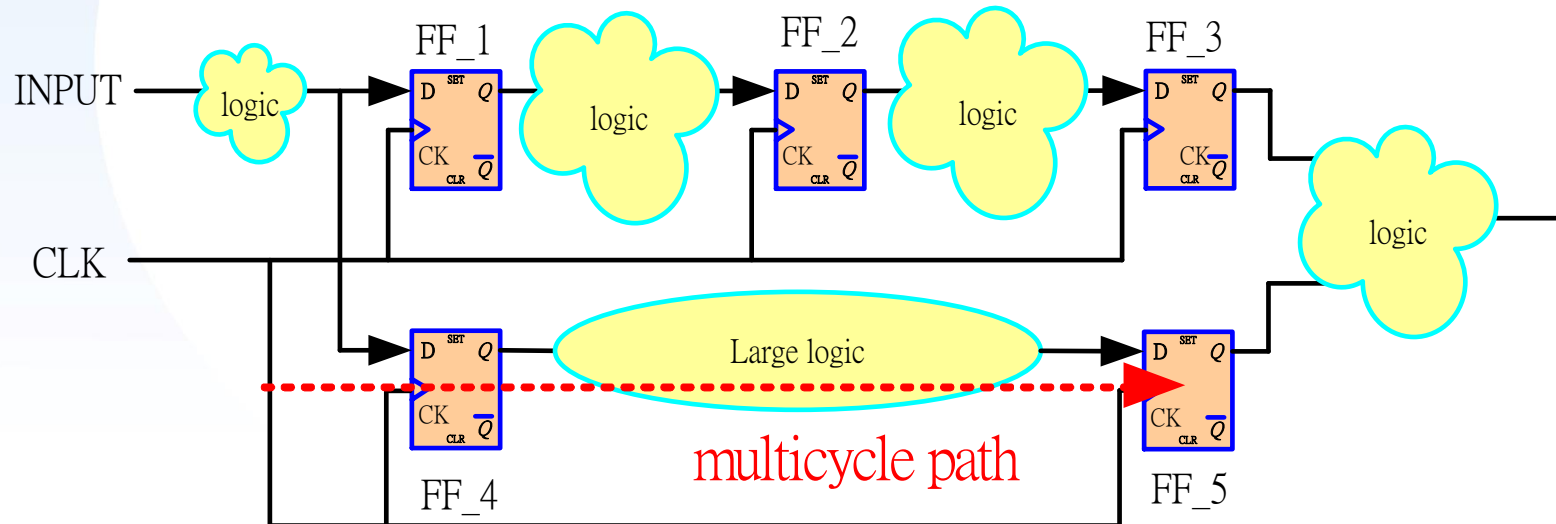
✓ Appendix



Multicycle Path

✓ A multicycle path

- A timing path which is not expected to propagate a signal in one cycle.
- Normally, all paths are constrained for single-cycle timing. Multicycle paths are exceptions to the default single-cycle timing.
- Set multicycle paths to direct compiler to allow multiple clock cycles for data to propagate along a path.



Specify Multicycle Path

✓ Set multicycle path

```
set_multicycle_path path_multiplier –from $start_point –to  
$end_point [-setup] [-hold][-start][-end]
```

✓ Option definition:

- path_multiplier: Specify the number of cycles
- \$start_point: Specifies start points (clocks, ports, pins, or cells) of multicycle path
- \$end_point: Specifies through points (ports, pins, or leaf cells) of multicycle path
- -setup: Used for setup time calculation
- -hold: Used for hold time calculation
- -start (clock launch edge)(Hold check default): Indicate the multicycle information is relative to the start clock
- -end (clock capture edge)(Setup check default): Indicate the multicycle information is relative to the end clock



✓ Timing Analysis Considering Clk Latency

- STA vs DTA
- CLK latency
- CLK uncertainty

✓ Multicycle Path Specification

- Specify Multicycle Path
- Single clock system
- Multiple clocks system

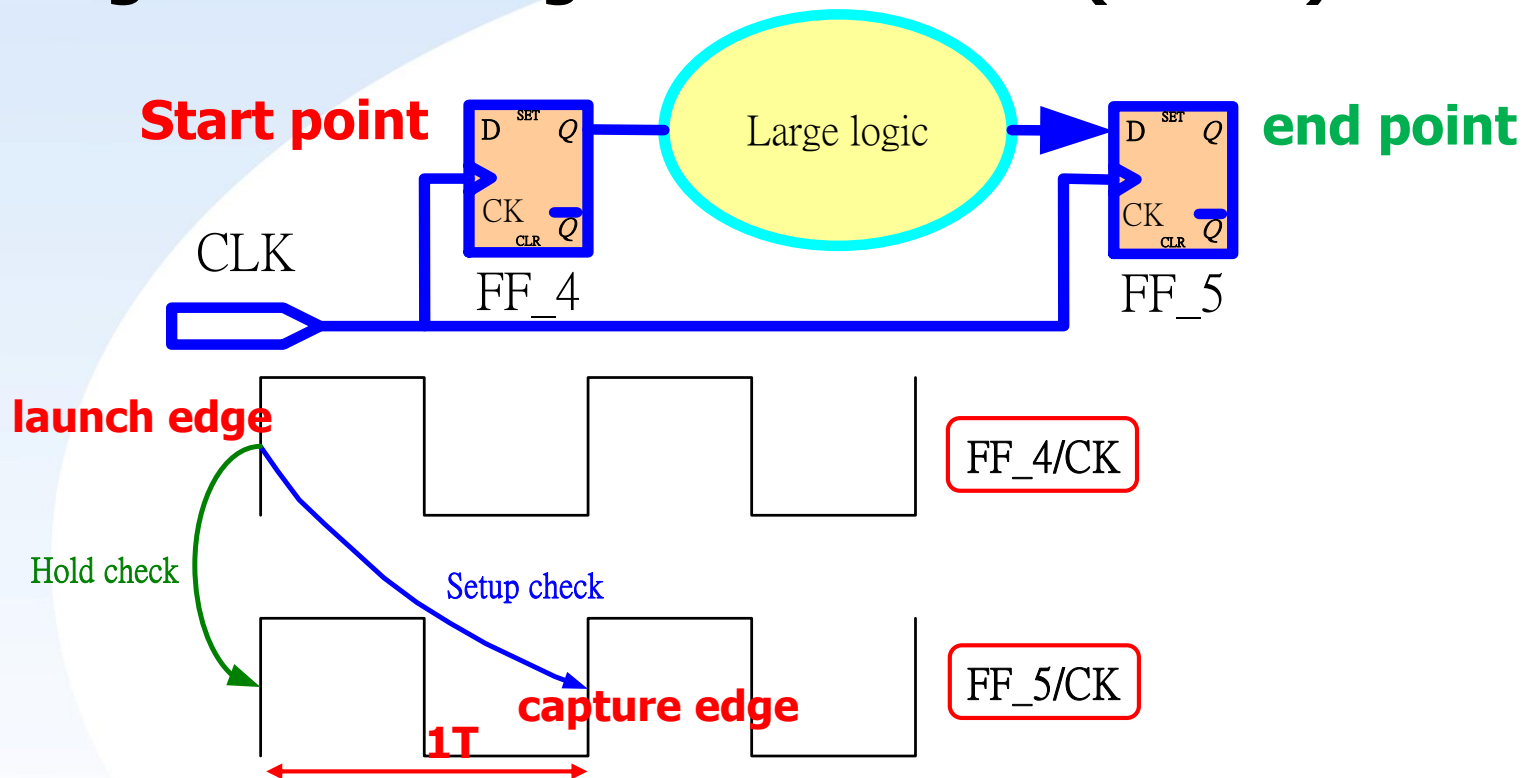
✓ Clock Domain Crossing (CDC)

✓ Appendix



Multicycle Path of Single Clock Domain

✓ Timing check in single clock domain(Initial)



Equivalent(default setting, don't need to write):

`set_multicycle_path` **1** –from **FF_4/CK** –to **FF_5/D**

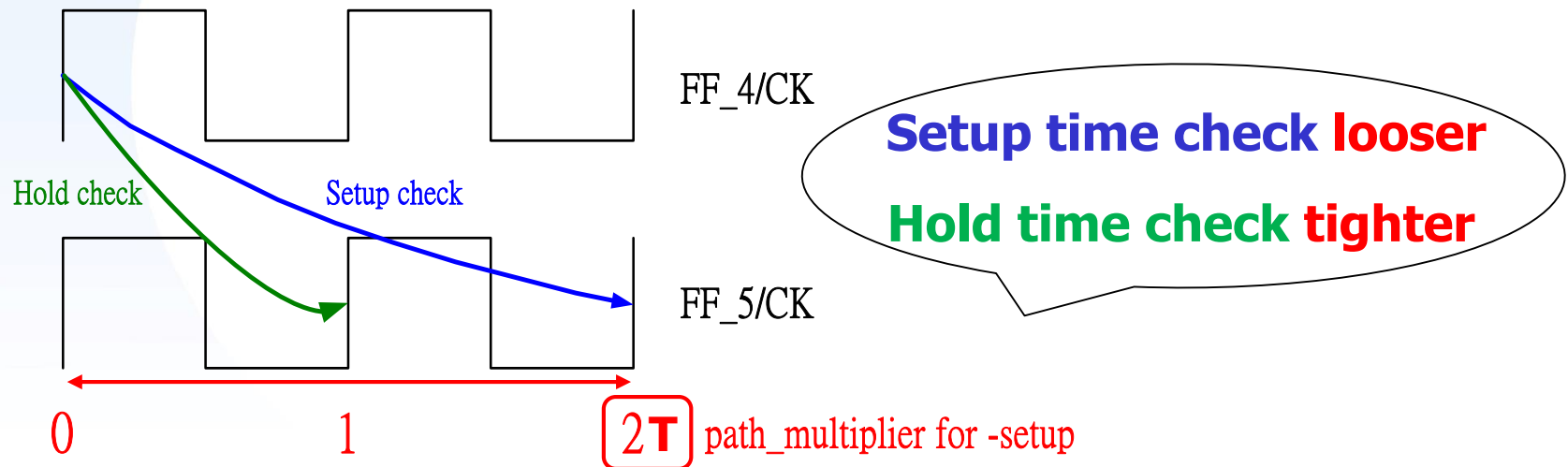
`set_multicycle_path` **0** –hold –from **FF_4/CK** –to **FF_5/D**

Multicycle Path of Single Clock Domain_(cont.)

- ✓ Specify multiple path for **setup** time check
 - Hold time check edge will be changed too!!

`set_multicycle_path 2` –from FF_4/CK –to FF_5/D

`set_multicycle_path 0` –hold -end –from FF_4/CK –to FF_5/D



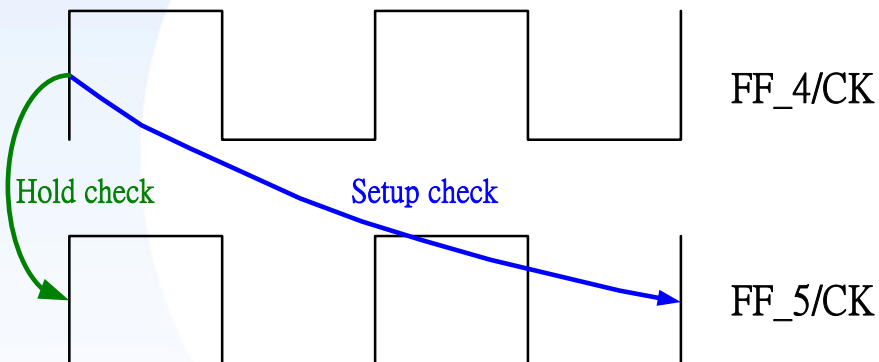
Multicycle Path of Single Clock Domain_(cont.)

- ✓ Specify multiple path for **hold** time check

```
set_multicycle_path 2 -from FF_4/CK -to FF_5/D
```

```
set_multicycle_path 1 -hold -end -from FF_4/CK -to FF_5/D
```

- Timing check edge is what we desire!



Setup time check **looser**

Hold time check **same**

0

1

2T

path_multiplier for -setup

1T

0

-1

path_multiplier for -hold



✓ Timing Analysis Considering Clk Latency

- STA vs DTA
- CLK latency
- CLK uncertainty

✓ Multicycle Path Specification

- Specify Multicycle Path
- Single clock system
- Multiple clocks system

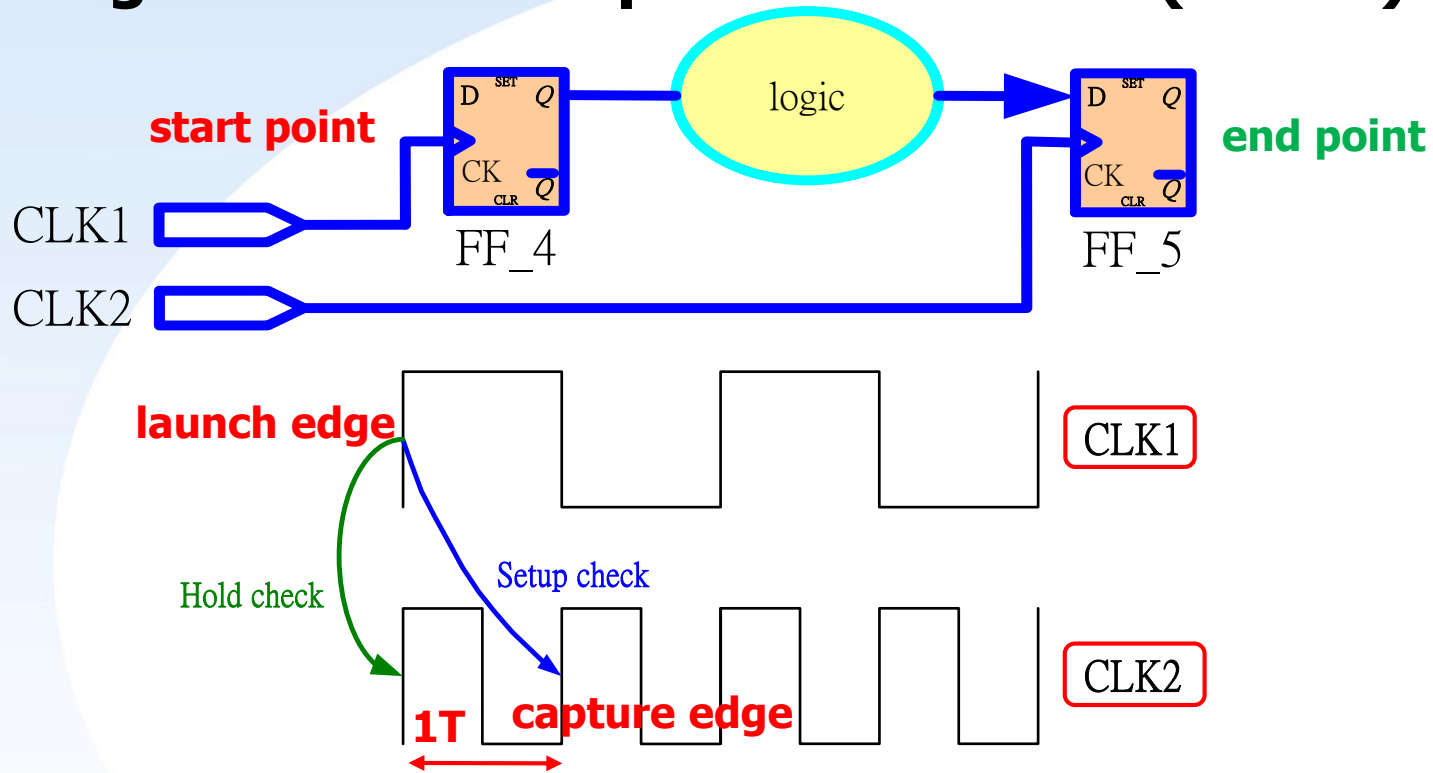
✓ Clock Domain Crossing (CDC)

✓ Appendix



Multicycle Path of Multiple Clock Domain

✓ Timing check in multiple clock domain(Initial)



Equivalent:

set_multicycle_path **1** -end -from CLK1 -to CLK2

set_multicycle_path **0** -hold -start -from CLK1 -to CLK2



Multicycle Path of Multiple Clock Domain_(cont.)

✓ Specify multiple path

`set_multicycle_path` **2** -from CLK1 -to CLK2

– Hold time check edge will be changed too!!

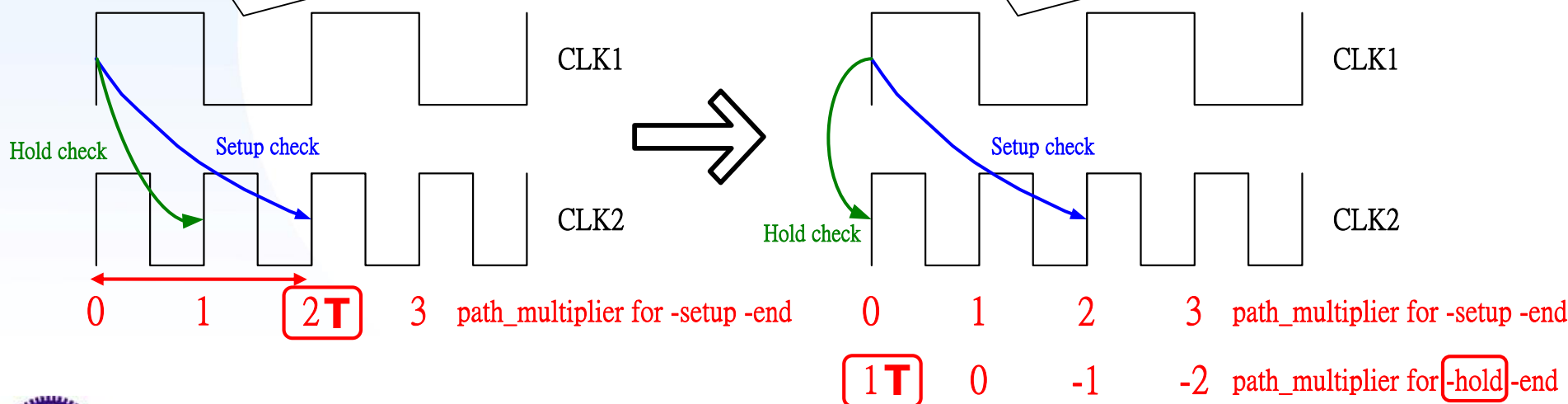
`set_multicycle_path` **1** -hold -end -from CLK1 -to CLK2

Setup time check **looser**

Hold time check **tighter**

Setup time check **looser**

Hold time check **same**



Multicycle Path of Multiple Clock Domain_(cont.)

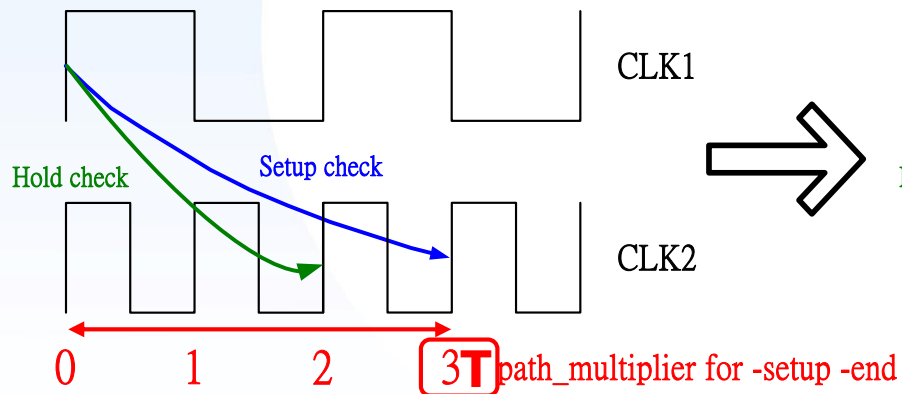
✓ Specify multiple path(Different combinations)

set_multicycle_path **3** –from CLK1 –to CLK2

set_multicycle_path **1** -hold -end –from CLK1 –to CLK2

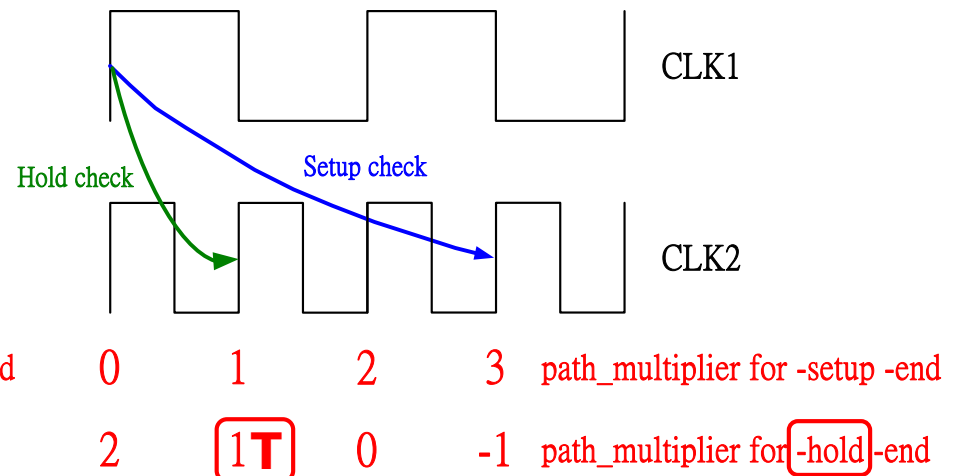
Setup time check **looser**

Hold time check **tighter**



Setup time check **looser**

Hold time check **tighter**



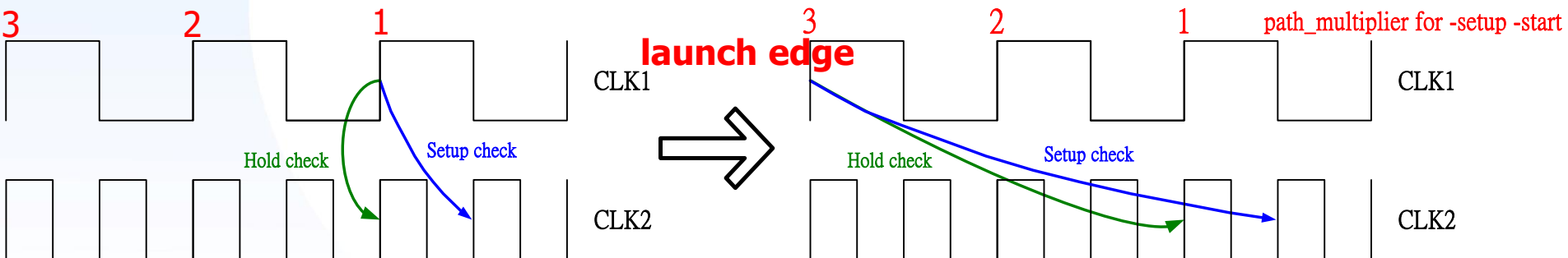
Multicycle Path of Multiple Clocks Domain_(cont.)

- ✓ Specify multiple path for setup time check **relative to start point**

set_multicycle_path **3** -start -from CLK1 -to CLK2

Setup time check **looser**

Hold time check **tighter**

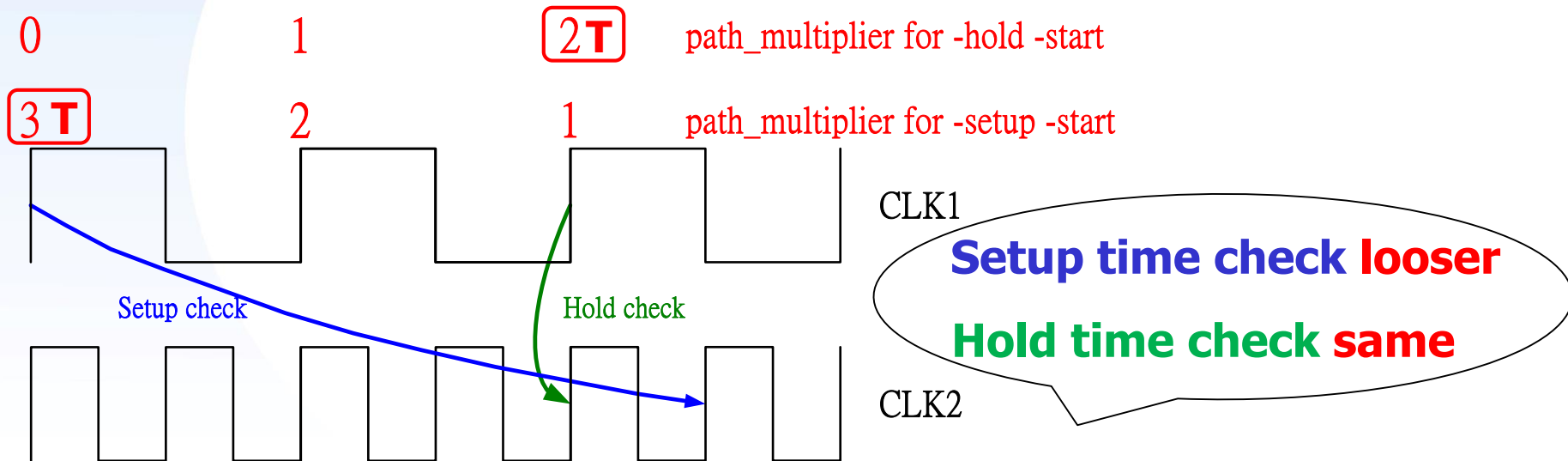


Multicycle Path of Multiple Clock Domain_(cont.)

- ✓ Specify multiple path for hold time check **relative to start point**

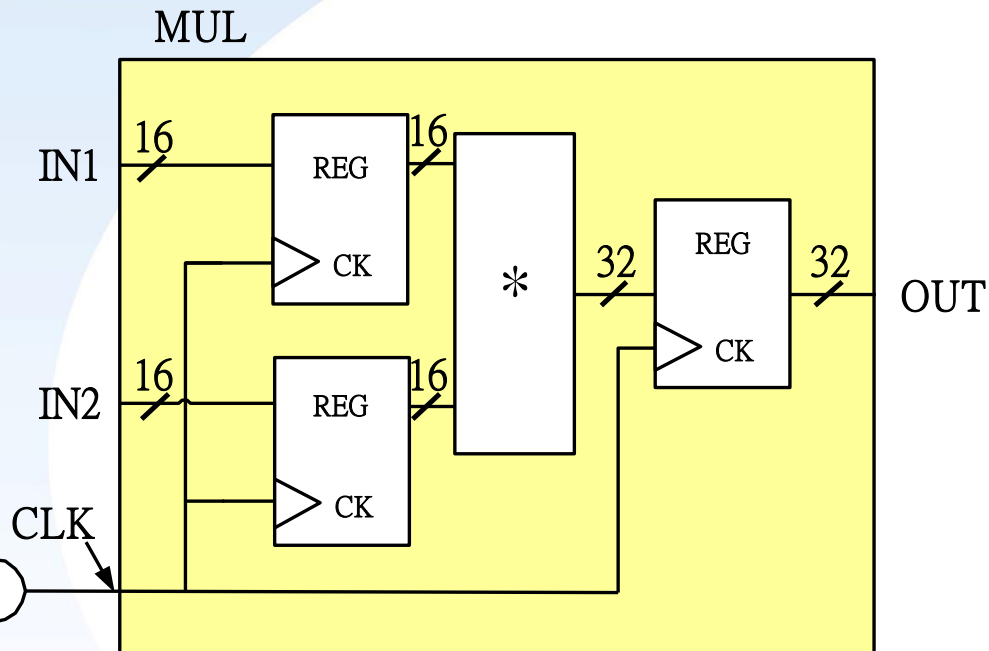
```
set_multicycle_path 3 -start -from CLK1 -to CLK2
```

```
set_multicycle_path 2 -hold -start -from CLK1 -to CLK2
```



Example: Multicycle Path of Single Clock Domain

e.g.1



```
module MUL
(
    // Output Port
    OUT,
    // Input Port
    IN1, IN2, CLK
);

output reg[31:0] OUT;
input  [15:0] IN1, IN2;
input          CLK;

reg [15:0] IN1_R, IN2_R;

always @(posedge CLK)
    IN1_R <= IN1;

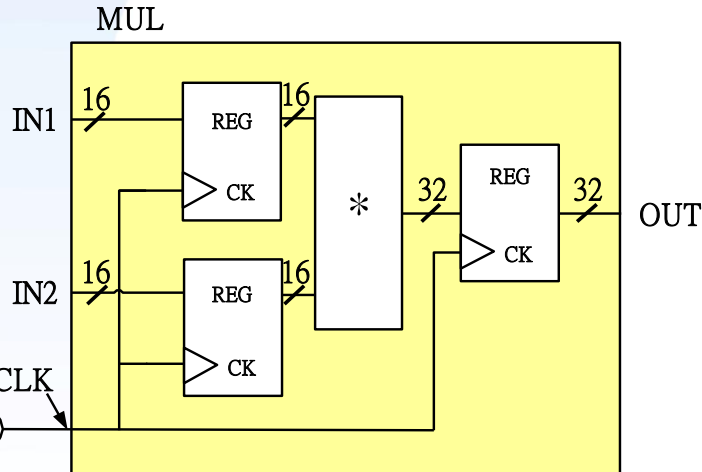
always @(posedge CLK)
    IN2_R <= IN2;

always @(posedge CLK)
    OUT <= IN1_R*IN2_R;

endmodule
```

Example: Multicycle Path of Single Clock Domain

```
dc_shell>create_clock -period 4 CLK
dc_shell>compile
dc_shell>report_timing
```



Report : timing -path full -delay max
-max_paths 1

Design : MUL

| Point | Incr | Path |
|-----------------------------|-------|--------|
| clock CLK (rise edge) | 0.00 | 0.00 |
| clock network delay (ideal) | 0.00 | 0.00 |
| IN1_R_reg[14]/CK (DFFHQX4) | 0.00 | 0.00 r |
| IN1_R_reg[14]/Q (DFFHQX4) | 0.30 | 0.30 r |
| U14/Y (BUFX20) | 0.18 | 0.48 r |
| mult_21/A[14] | | |
| (MUL_DW02_mult_16_16_0) | 0.00 | 0.48 r |
| ... (ignored) | | |
| mult_21/PRODUCT[28] | | |
| (MUL_DW02_mult_16_16_0) | 0.00 | 4.91 r |
| OUT_reg[28]/D (DFFXL) | 0.00 | 4.91 r |
| data arrival time | | 4.91 |
| clock CLK (rise edge) | 4.00 | 4.00 |
| clock network delay (ideal) | 0.00 | 4.00 |
| OUT_reg[28]/CK (DFFXL) | 0.00 | 4.00 r |
| library setup time | -0.10 | 3.90 |
| data required time | | 3.90 |
| data required time | | 3.90 |
| data arrival time | | -4.91 |
| slack (VIOLATED) | | -1.00 |

Timing Violation!!!



Example: Multicycle Path of Single Clock Domain

```
dc_shell>set_multicycle_path 2 -from [get_cells IN1_R_reg[*]] -to [get_cells OUT_reg[*]]
dc_shell>set_multicycle_path 2 -from [get_cells IN2_R_reg[*]] -to [get_cells OUT_reg[*]]
dc_shell>set_multicycle_path 1 -hold -end -from [get_cells IN1_R_reg[*]] -to [get_cells OUT_reg[*]]
dc_shell>set_multicycle_path 1 -hold -end -from [get_cells IN2_R_reg[*]] -to [get_cells OUT_reg[*]]
dc_shell>report_timing_requirements
```

```
*****
Report : timing_requirements -attributes
Design : MUL
*****
```

| Description | Setup | Hold |
|------------------|----------|---------------|
| TIMING EXCEPTION | cycles=2 | cycles=1(end) |

```
-from { IN1_R_reg[0]\
```

```
.....
      IN1_R_reg[15]}\
```

```
-to { OUT_reg[0]\
```

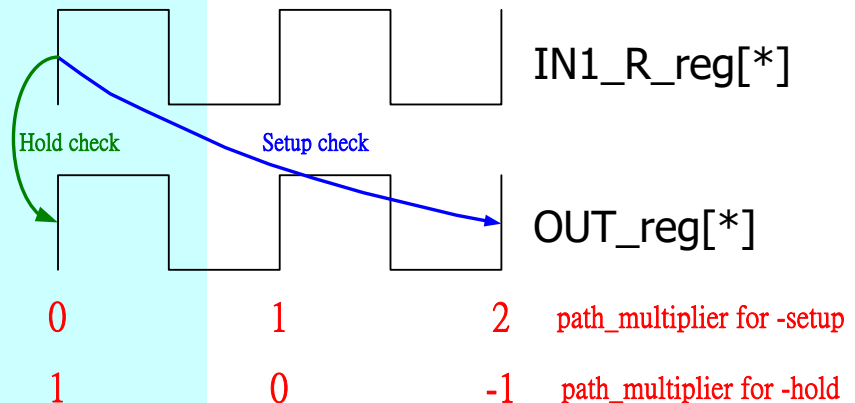
```
.....
      OUT_reg[31] }
```

```
-from { IN2_R_reg[0]\
```

```
.....
      IN2_R_reg[15]}\
```

```
-to { OUT_reg[0]\
```

```
.....
      OUT_reg[31] }
```



Example: Multicycle Path of Single Clock Domain

Report : timing -path full -delay **max**
-max_paths 1

Setup time

Design : MUL

| Point | Incr | Path |
|-----------------------------|-------|----------------|
| ----- | | |
| clock CLK (rise edge) | 0.00 | 0.00 |
| clock network delay (ideal) | 0.00 | 0.00 |
| IN1_R_reg[13]/CK (DFFX1) | 0.00 | 0.00 r |
| IN1_R_reg[13]/Q (DFFX1) | 1.27 | 1.27 r |
| mult_21/A[13] | | |
| (MUL_DW02_mult_16_16_0) | 0.00 | 1.27 r |
|(ignored) | | |
| mult_21/PRODUCT[28] | | |
| (MUL_DW02_mult_16_16_0) | 0.00 | 7.56 f |
| OUT_reg[28]/D (DFFX1) | 0.00 | 7.56 f |
| data arrival time | | 7.56 |
| ----- | | |
| clock CLK (rise edge) | 8.00 | 8.00 2T |
| clock network delay (ideal) | 0.00 | 8.00 |
| OUT_reg[28]/CK (DFFX1) | 0.00 | 8.00 r |
| library setup time | -0.33 | 7.67 |
| data required time | | 7.67 |
| ----- | | |
| data required time | | 7.67 |
| data arrival time | | -7.56 |
| ----- | | |
| slack (MET) | | 0.11 |

dc_shell>compile

dc_shell>report_timing

dc_shell>report_timing -delay min

Report : timing -path full -delay **min**
-max_paths 1

Hold time

Design : MUL

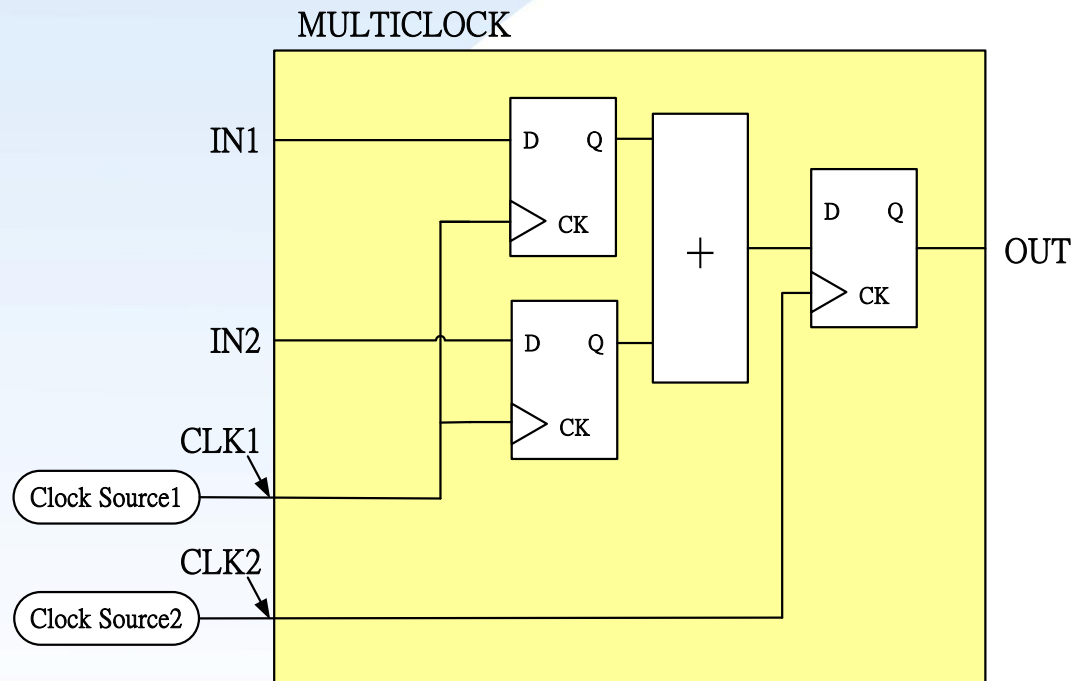
| Point | Incr | Path |
|--------------------------------------|-------|-------------|
| ----- | | |
| clock (input port clock) (rise edge) | 0.00 | 0.00 |
| clock network delay (ideal) | 0.00 | 0.00 |
| input external delay | 0.00 | 0.00 r |
| IN1[0] (in) | 0.00 | 0.00 r |
| IN1_R_reg[0]/D (DFFX1) | 0.00 | 0.00 r |
| data arrival time | | 0.00 |
| ----- | | |
| clock CLK (rise edge) | 0.00 | 0.00 |
| clock network delay (ideal) | 0.00 | 0.00 |
| IN1_R_reg[0]/CK (DFFX1) | 0.00 | 0.00 r |
| library hold time | -0.03 | -0.03 |
| data required time | | -0.03 |
| ----- | | |
| data required time | | -0.03 |
| data arrival time | | 0.00 |
| ----- | | |
| slack (MET) | | 0.03 |

**Timing
Met !!!**



Example: Multicycle Path of Multiple Clock Domain

e.g.2



```
module MULTICLOCK  
(  
    // Output Port  
    OUT,  
    // Input Port  
    CLK1, CLK2, IN1, IN2  
);
```

```
    output reg OUT;  
    input CLK1, CLK2;  
    input IN1, IN2;
```

```
    reg temp1, temp2;
```

```
    assign data = temp1 + temp2;
```

```
    always @(posedge CLK1)  
        temp1 <= IN1;
```

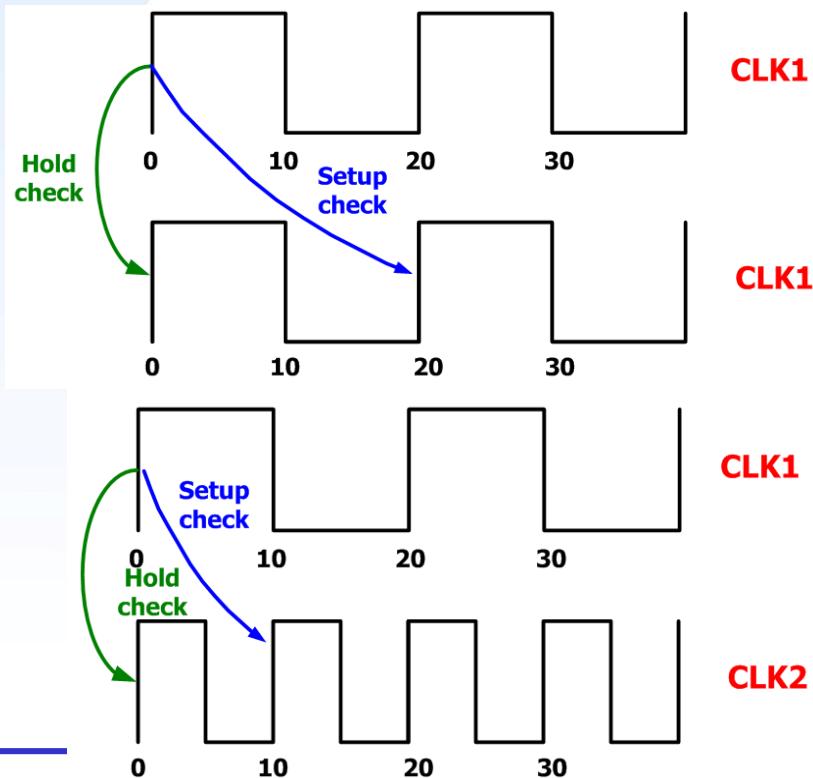
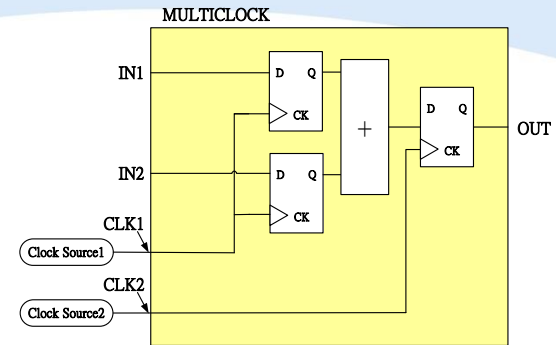
```
    always @(posedge CLK1)  
        temp2 <= IN2;
```

```
    always @(posedge CLK2)  
        OUT <= data;
```

```
endmodule
```

Example: Multicycle Path of Multiple Clock Domain

```
dc_shell>create_clock -period 20 CLK1  
dc_shell>create_clock -period 10 CLK2
```



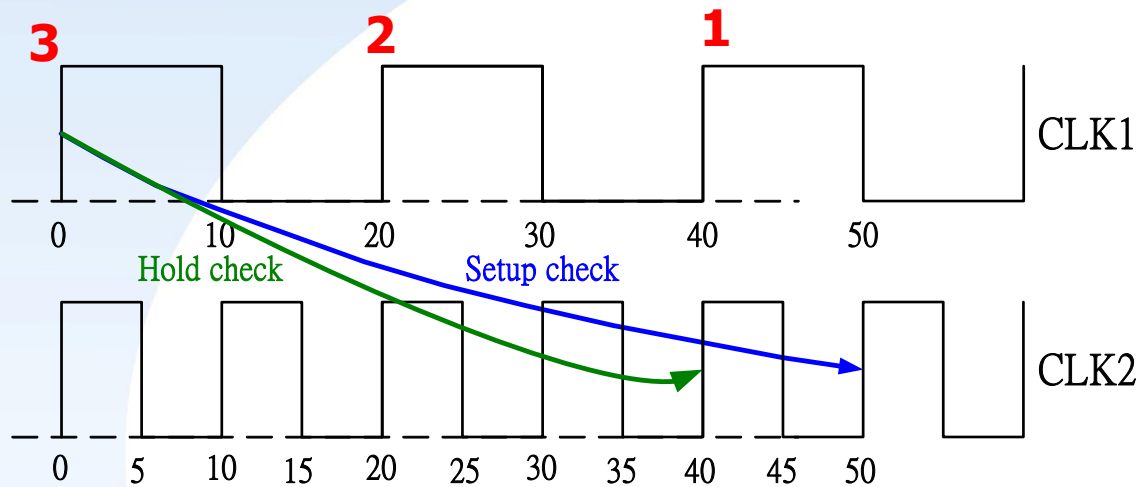
setup and hold check
in clock domain 1

setup and hold check
from clock domain 1
to clock domain 2



Example: Multicycle Path of Multiple Clock Domain (Hold Time violation version)

```
dc_shell>set_multicycle_path 3 -start -from CLK1 -to CLK2
dc_shell>report_timing_requirements
```



```
*****
Report : timing_requirements
-attributes
Design : MULTICLOCK
*****
```

| Description | Setup | Hold |
|-------------------------|-----------------|----------|
| TIMING EXCEPTION | cycles=3(start) | cycles=0 |
| -from CLK1\ -to CLK2 | | |



Example: Multicycle Path of Multiple Clock Domain (Hold Time violation version)

dc_shell>compile

dc_shell>report_timing

dc_shell>report_timing -delay min

Try to satisfy hold time check by insert many delay cells

Path Group: CLK2

Path Type: max

| Point | Incr | Path |
|-----------------------------|-------|---------|
| clock CLK1 (rise edge) | 0.00 | 0.00 |
| clock network delay (ideal) | 0.00 | 0.00 |
| TEMP2_reg/CK (DFFX4) | 0.00 | 0.00 r |
| TEMP2_reg/Q (DFFX4) | 0.28 | 0.28 r |
| U48/Y (XOR2XL) | 0.41 | 0.69 r |
|(ignored) | | |
| U5/Y (INVXL) | 0.21 | 49.77 f |
| U4/Y (CLKINX8) | 0.08 | 49.85 r |
| OUT_reg/D (DFFX4) | 0.00 | 49.85 r |
| data arrival time | | 49.85 |
| clock CLK2 (rise edge) | 50.00 | 50.00 |
| clock network delay (ideal) | 0.00 | 50.00 |
| OUT_reg/CK (DFFX4) | 0.00 | 50.00 r |
| library setup time | -0.14 | 49.86 |
| data required time | | 49.86 |
| data required time | | 49.86 |
| data arrival time | | -49.85 |
| slack (MET) | | 0.01 |

Path Group: CLK2

Path Type: min

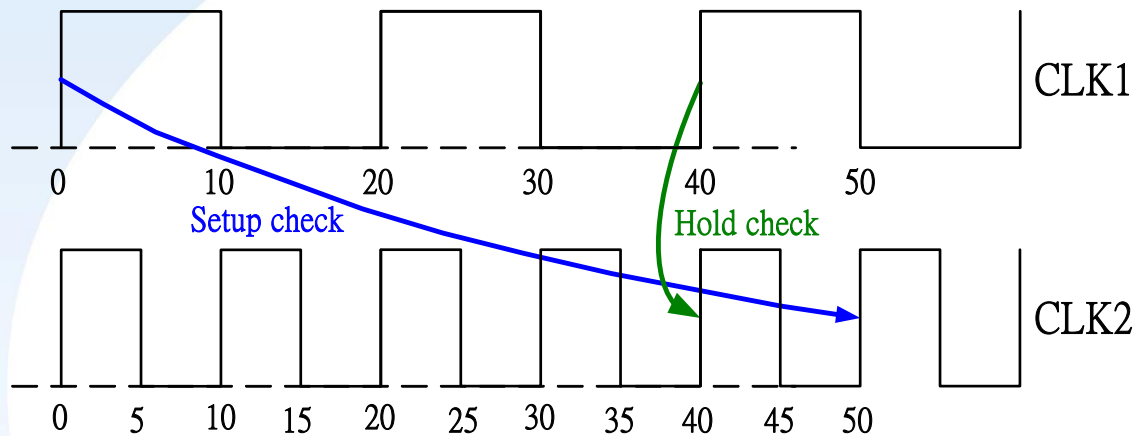
| Point | Incr | Path |
|-----------------------------|-------|---------|
| clock CLK1 (rise edge) | 0.00 | 0.00 |
| clock network delay (ideal) | 0.00 | 0.00 |
| TEMP1_reg/CK (DFFX4) | 0.00 | 0.00 r |
| TEMP1_reg/Q (DFFX4) | 0.10 | 0.10 f |
| U48/Y (XOR2XL) | 0.10 | 0.20 f |
| U47/Y (DLY4X1) | 0.46 | 0.65 f |
| U46/Y (DLY4X1) | 0.47 | 1.12 f |
| U45/Y (DLY4X1) | 0.47 | 1.59 f |
|(ignored) | | |
| U6/Y (DLY4X1) | 0.47 | 19.90 f |
| U5/Y (INVXL) | 0.15 | 20.05 r |
| U4/Y (CLKINX8) | 0.03 | 20.08 f |
| OUT_reg/D (DFFX4) | 0.00 | 20.08 f |
| data arrival time | | 20.08 |
| clock CLK2 (rise edge) | 40.00 | 40.00 |
| clock network delay (ideal) | 0.00 | 40.00 |
| OUT_reg/CK (DFFX4) | 0.00 | 40.00 r |
| library hold time | -0.04 | 39.96 |
| data required time | | 39.96 |
| data required time | | 39.96 |
| data arrival time | | -20.08 |
| slack (VIOLATED) | | -19.88 |

Timing Violation!!!



Example: Multicycle Path of Multiple Clock Domain (Correct version)

```
dc_shell>set_multicycle_path 3 --start --from CLK1 --to CLK2
dc_shell>set_multicycle_path 2 --hold --start --from CLK1 --to CLK2
dc_shell>report_timing_requirements
```



```
*****
Report : timing_requirements
-attributes
Design : MULTICLOCK
*****
```

Description

Setup

Hold

TIMING EXCEPTION

-from CLK1\
-to CLK2

cycles=3(start)

cycles=2



Example: Multicycle Path of Multiple Clock Domain (Correct version)

```
dc_shell>compile
```

```
dc_shell>report_timing
```

```
dc_shell>report_timing -delay min
```

Path Group: CLK2

Path Type: **max**

| Point | Incr | Path |
|-----------------------------|-------|---------|
| clock CLK1 (rise edge) | 0.00 | 0.00 |
| clock network delay (ideal) | 0.00 | 0.00 |
| TEMP1_reg/CK (DFFX1) | 0.00 | 0.00 r |
| TEMP1_reg/Q (DFFX1) | 0.44 | 0.44 r |
| U49/Y (XOR2X1) | 0.29 | 0.73 f |
| OUT_reg/D (DFFX1) | 0.00 | 0.73 f |
| data arrival time | | 0.73 |
| clock CLK2 (rise edge) | 50.00 | 50.00 |
| clock network delay (ideal) | 0.00 | 50.00 |
| OUT_reg/CK (DFFX1) | 0.00 | 50.00 r |
| library setup time | -0.31 | 49.69 |
| data required time | | 49.69 |
| data required time | | 49.69 |
| data arrival time | | -0.73 |
| slack (MET) | | 48.95 |

Path Group: CLK2

Path Type: **min**

| Point | Incr | Path |
|-----------------------------|-------|---------|
| clock CLK1 (rise edge) | 40.00 | 40.00 |
| clock network delay (ideal) | 0.00 | 40.00 |
| TEMP2_reg/CK (DFFX1) | 0.00 | 40.00 r |
| TEMP2_reg/Q (DFFX1) | 0.14 | 40.14 f |
| U49/Y (XOR2X1) | 0.08 | 40.22 r |
| OUT_reg/D (DFFX1) | 0.00 | 40.22 r |
| data arrival time | | 40.22 |
| clock CLK2 (rise edge) | 40.00 | 40.00 |
| clock network delay (ideal) | 0.00 | 40.00 |
| OUT_reg/CK (DFFX1) | 0.00 | 40.00 r |
| library hold time | -0.03 | 39.97 |
| data required time | | 39.97 |
| data required time | | 39.97 |
| data arrival time | | -40.22 |
| slack (MET) | | 0.25 |

Timing
Met !!!



✓ Timing Analysis Considering Clk Latency

- STA vs DTA
- CLK latency
- CLK uncertainty

✓ Multicycle Path Specification

- Specify Multicycle Path
- Single clock system
- Multiple clocks system

✓ Clock Domain Crossing (CDC)

✓ Appendix



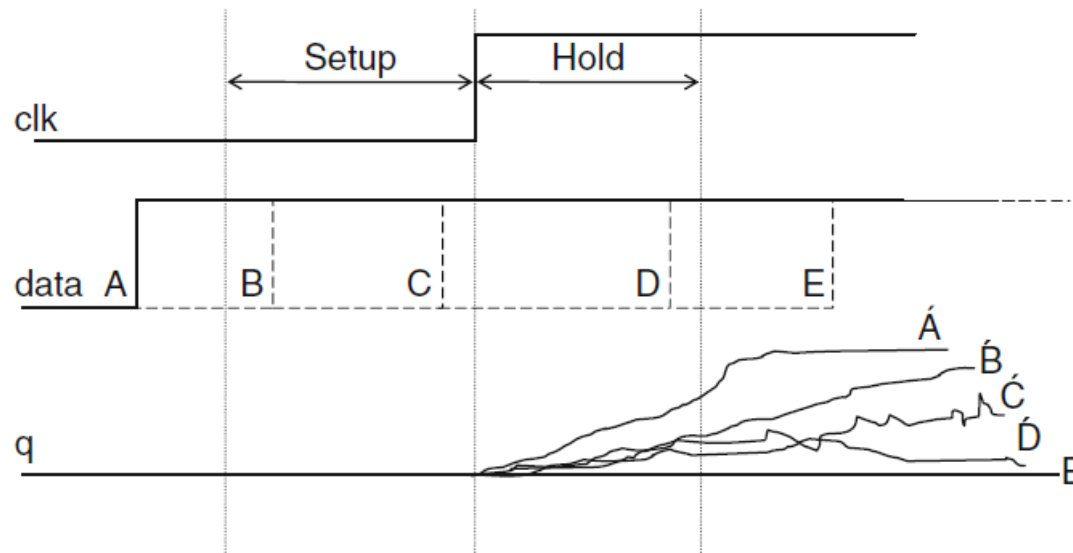
Clock Domain Crossing (CDC)

- ✓ Because different blocks require different clock period, some blocks need long clock period to operate, but others only do simple work. If we all use long clock period, it will cause the performance to be poor. So that's why we need to use clock domain crossing .



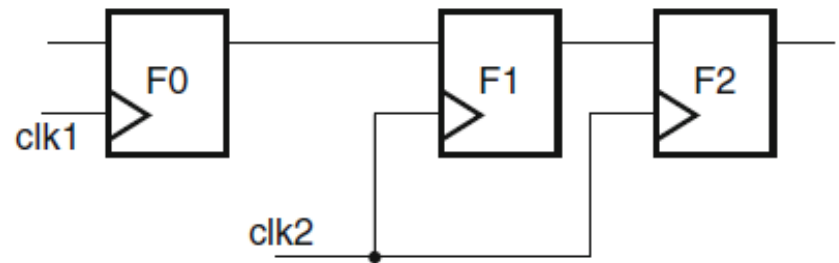
Metastability

- ✓ The unstable status due to non-ideal data transition is called metastability.
- ✓ To avoid this phenomenon, we have to ensure no data transition during setup/hold timing check.
- ✓ However, CDC designs will inevitably face this problem.

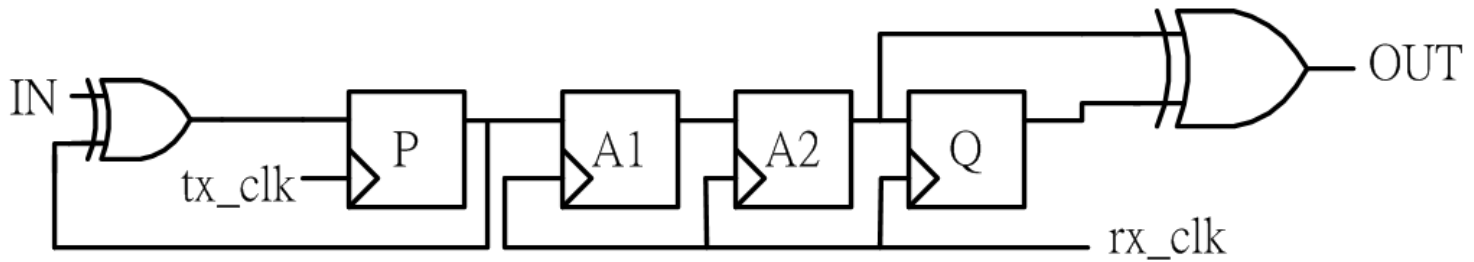


CDC solution

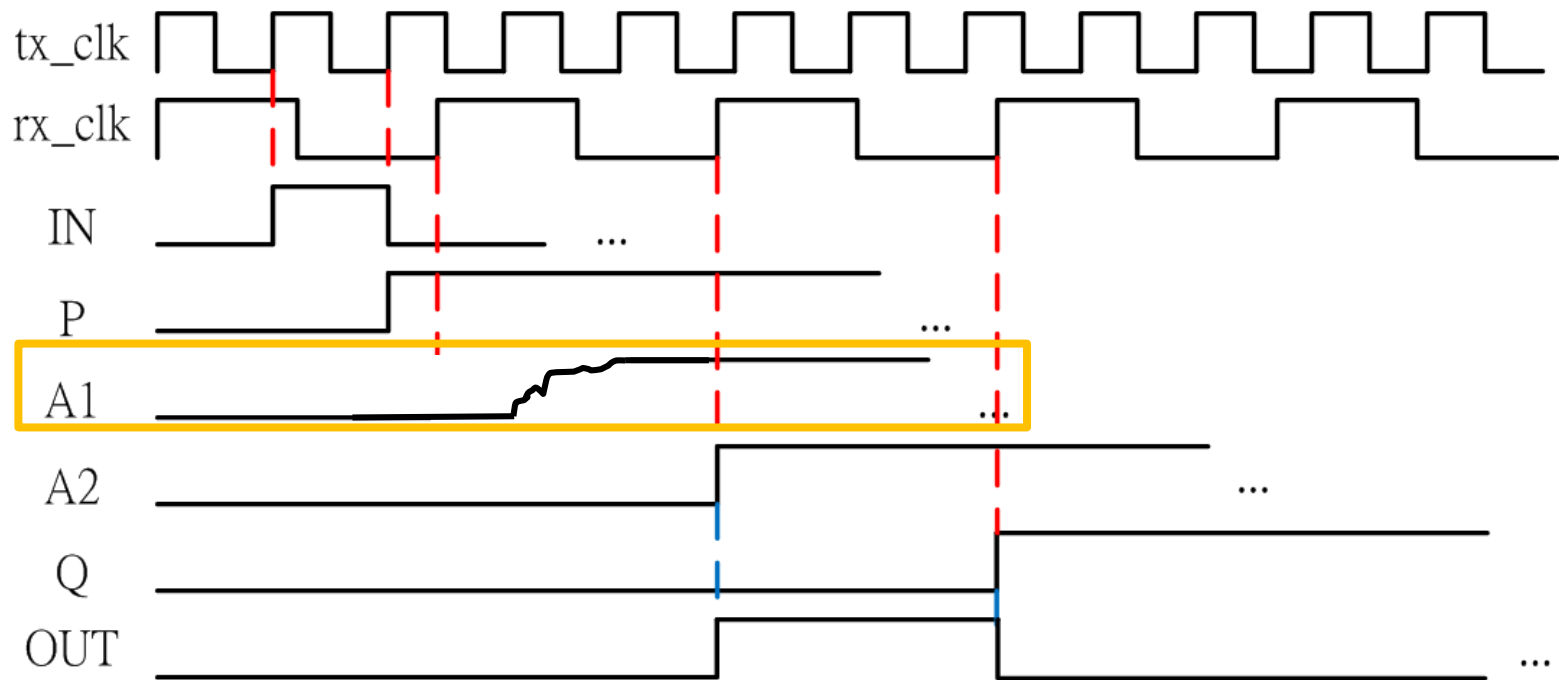
- ✓ There are lots of solutions for CDC designs, here we would only introduce an easy one.
- ✓ Double Flop(2-FF) Synchronizer
 - F1 samples the asynchronous input signal into the destination clock domain, probably a meta-stable value.
 - With a full clk2 cycle to permit any meta-stability on the F1 output signal to decay.
 - Then the signal is sampled by the same clock into F2, which we expect to be a stable and valid signal.
 - Often used in 1-bit flag transmission.



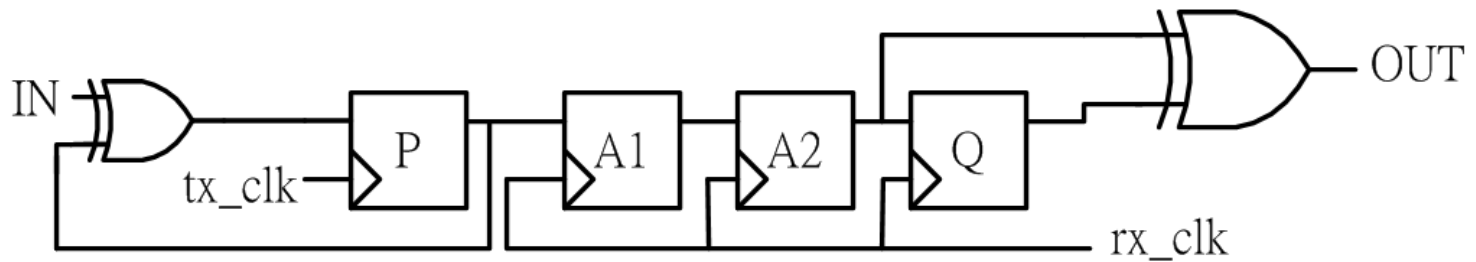
Example



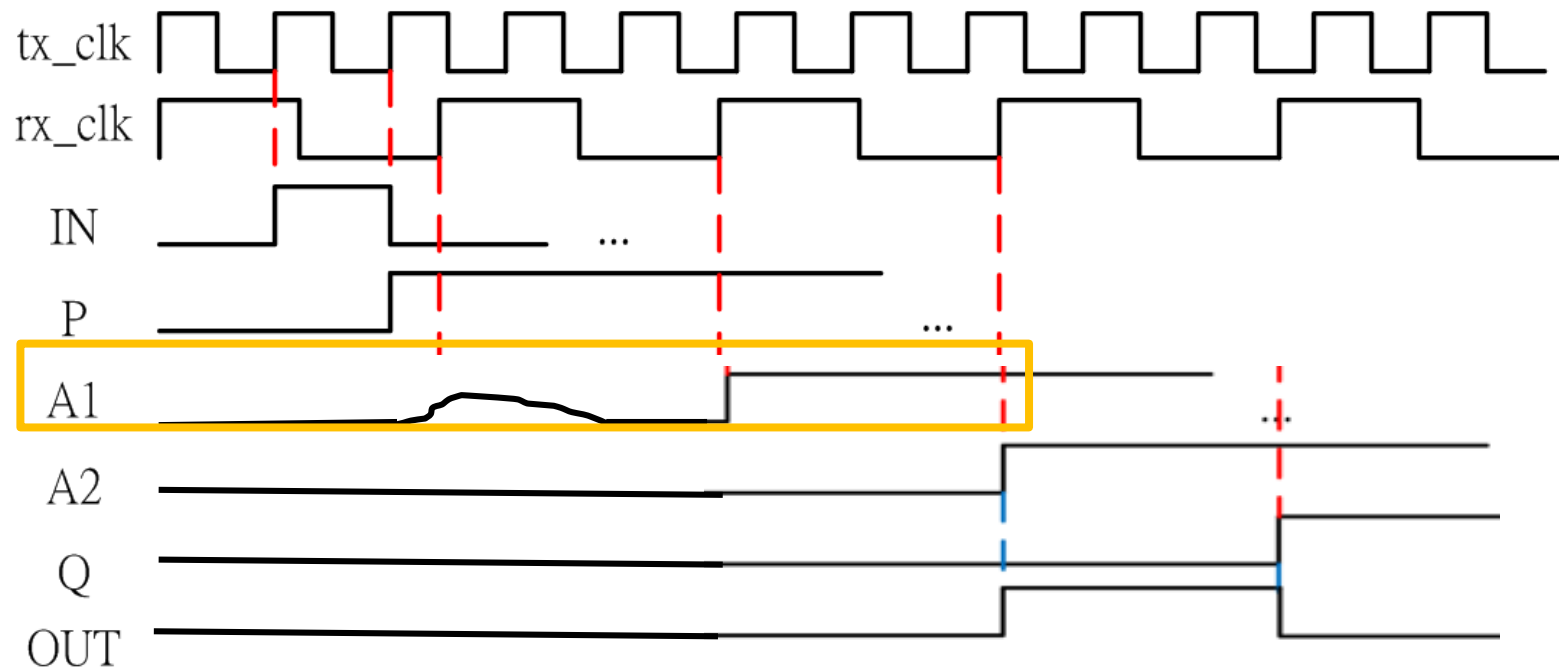
fast to slow



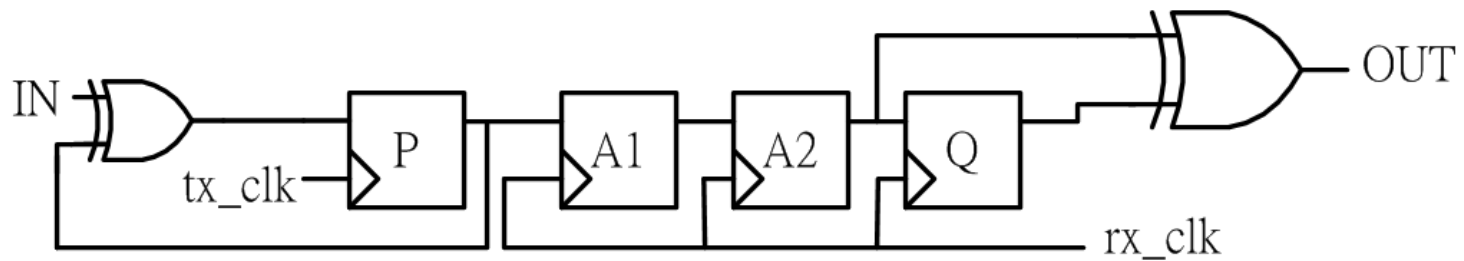
Example



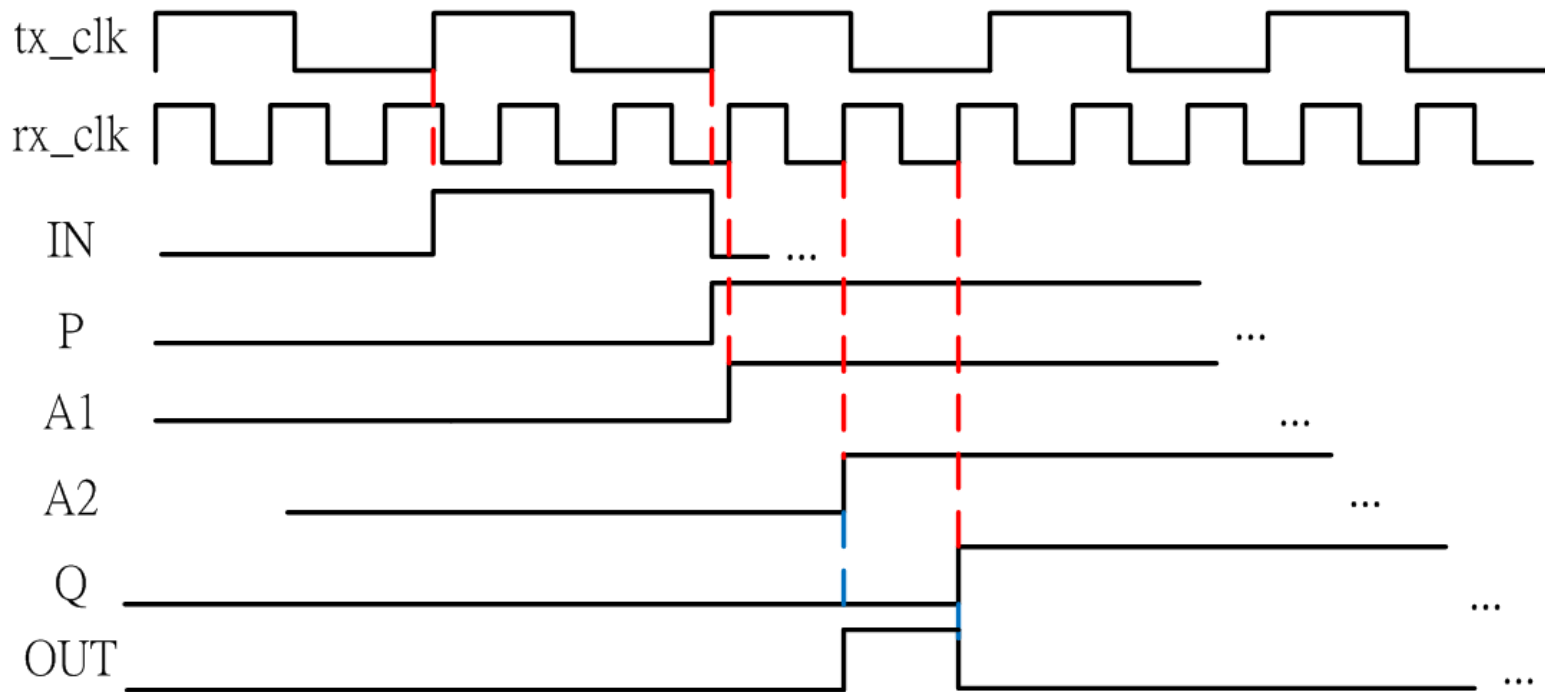
fast to slow



Example

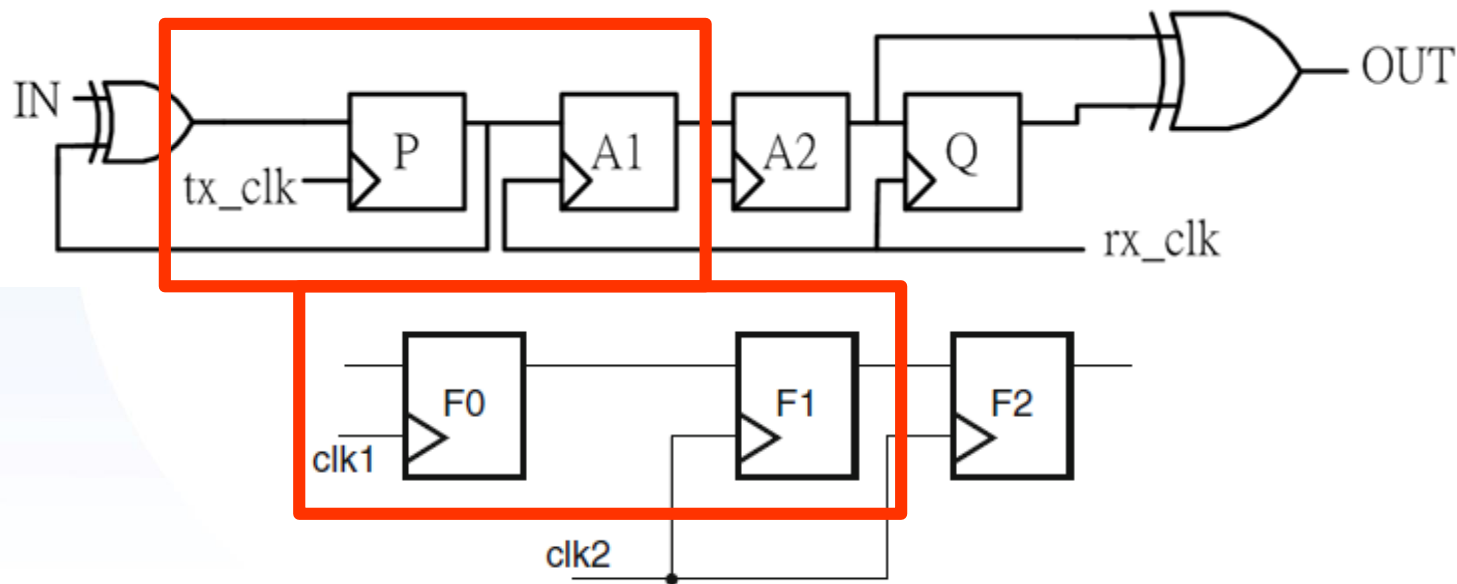


slow to fast



Specify False Path

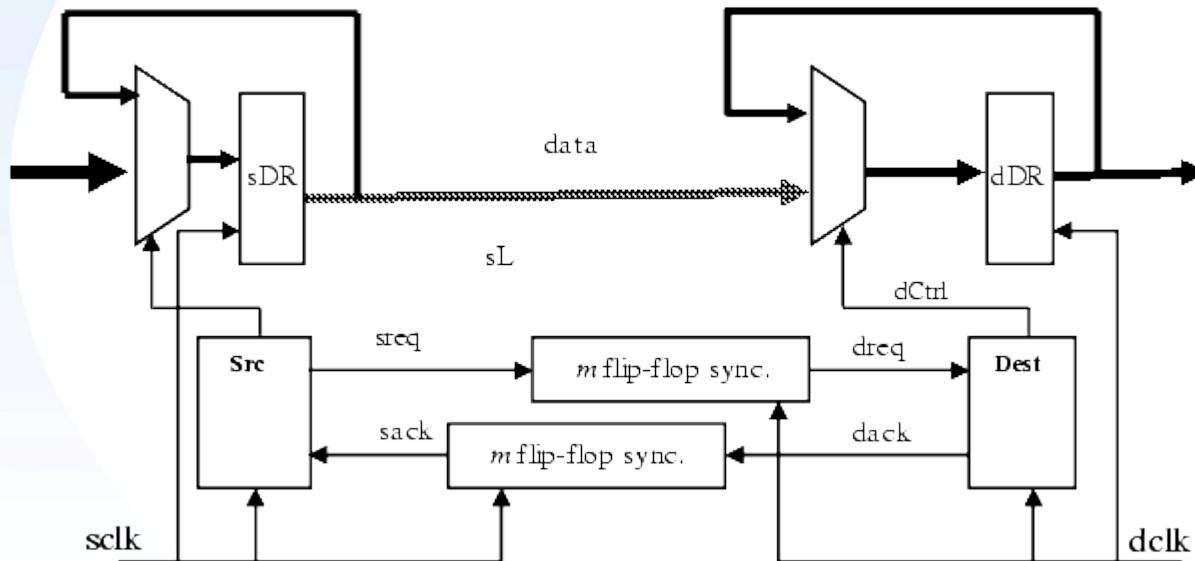
- ✓ **Remove timing constraints from particular path**
 - Timing checks of false path will be disabled. Therefore, use this command carefully
- ✓ **dc_shell>set_false_path –from FF_A/CK –to FF_D/D**



CDC solution

✓ Handshake Synchronizers.

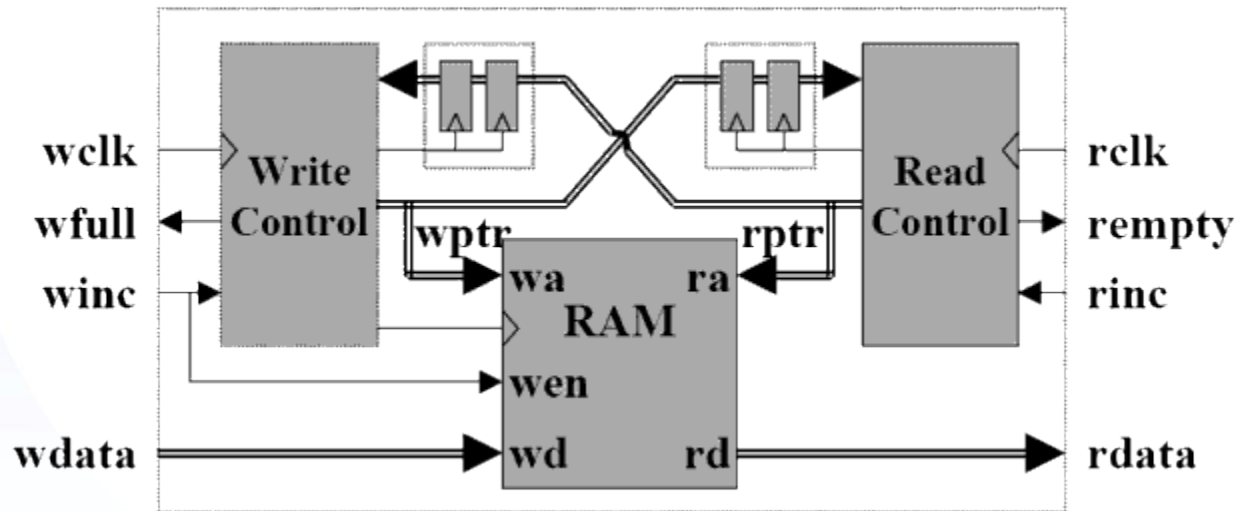
- Synchronize a single-bit request into the destination clock domain and wait for a synchronized version of the acknowledge to come back.
- Time uncertainty and high latency.
- Data stability and protocol check



CDC solution

✓ Dual Clock FIFO synchronizers

- Data is written into a dual-port RAM block from the source clock domain and the RAM is read in the destination clock domain.
- Gray-coded read and write pointers are passed into the alternate clock domain to generate full and empty status flags.
- High throughput but difficult hardware.

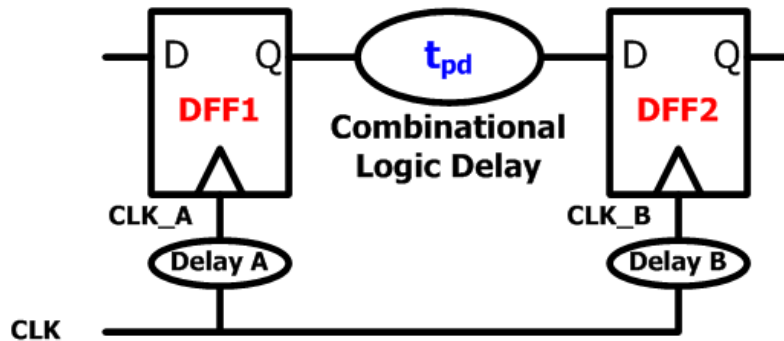
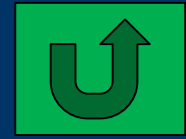


Reference

- ✓ More detail please refer “Principles of VLSI RTL Design A Practical Guide”
- ✓ More detail please refer “Static Timing Analysis for Nanometer Design”

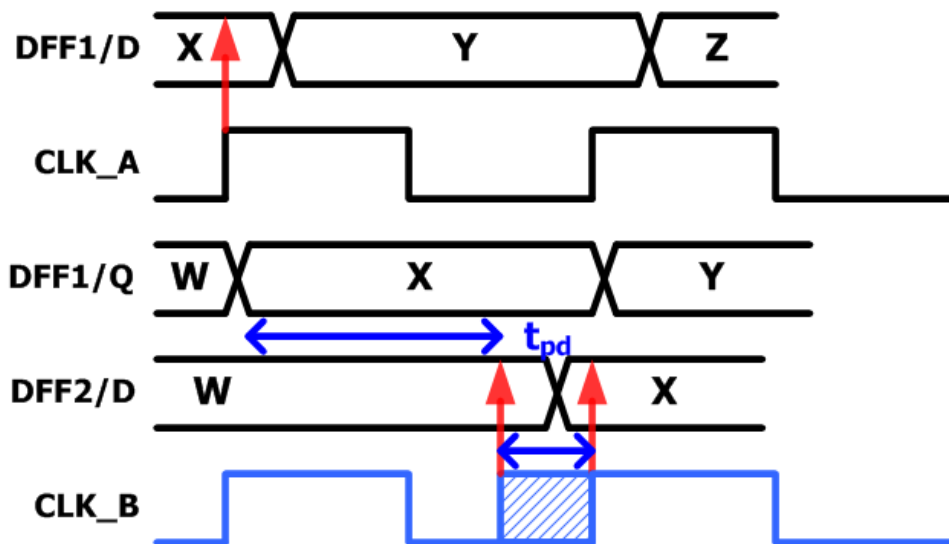


Appendix-Timing Check

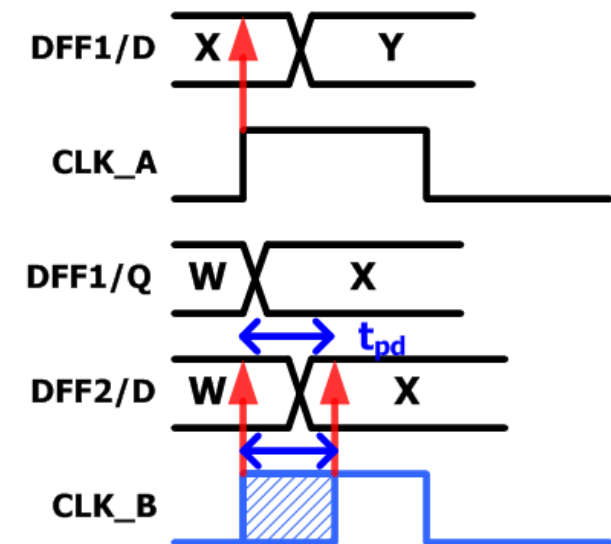


Setup time determines **max** value of t_{pd}

Hold time determines **min** value of t_{pd}

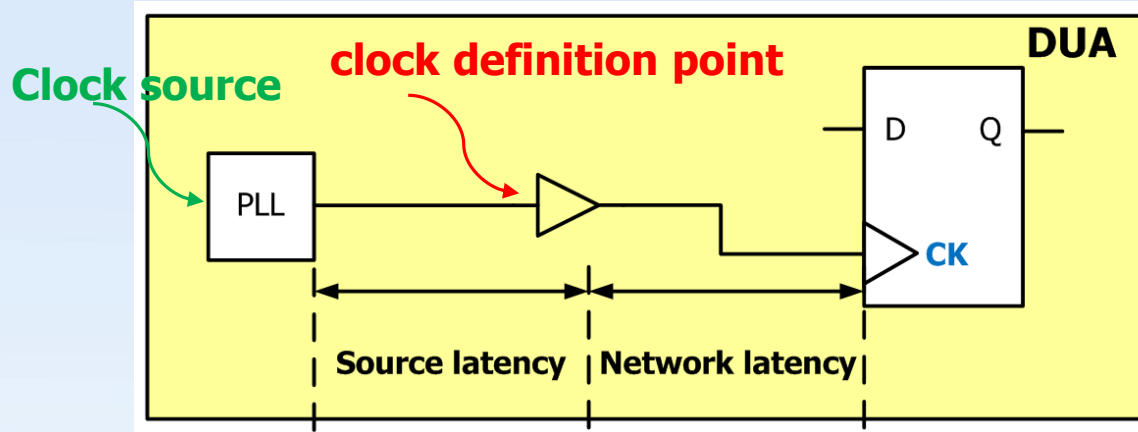
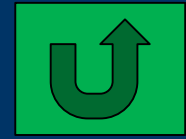


Setup Time Check



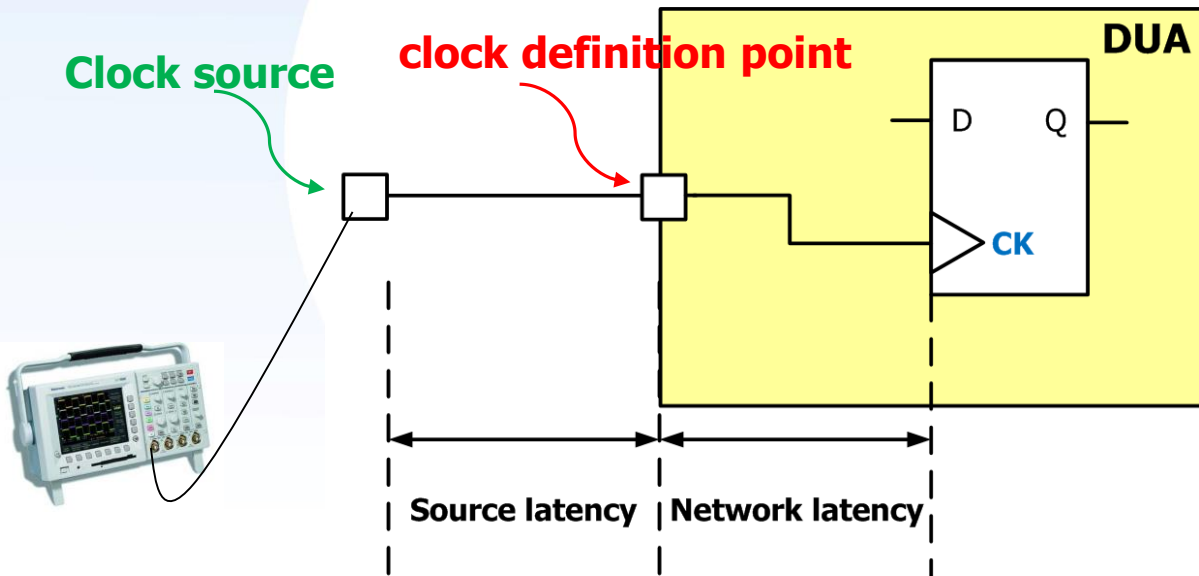
Hold Time Check

Appendix-Clock Latency



p.s. DUA(Design under analysis)

- On-chip clock source



- Off-chip clock source