Amaranth (2, practice)

Hardware System Design

Spring, 2023

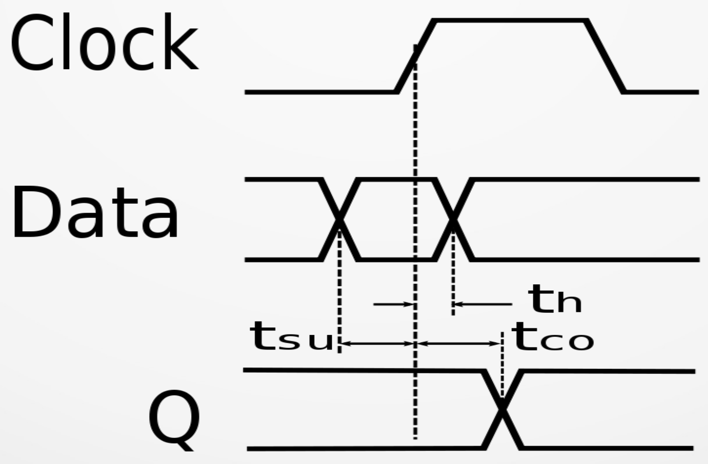
Outline

- **Prevent metastability**

- Practice
  - Implement PE, stacked PE

Flip-Flops – Timing considerations

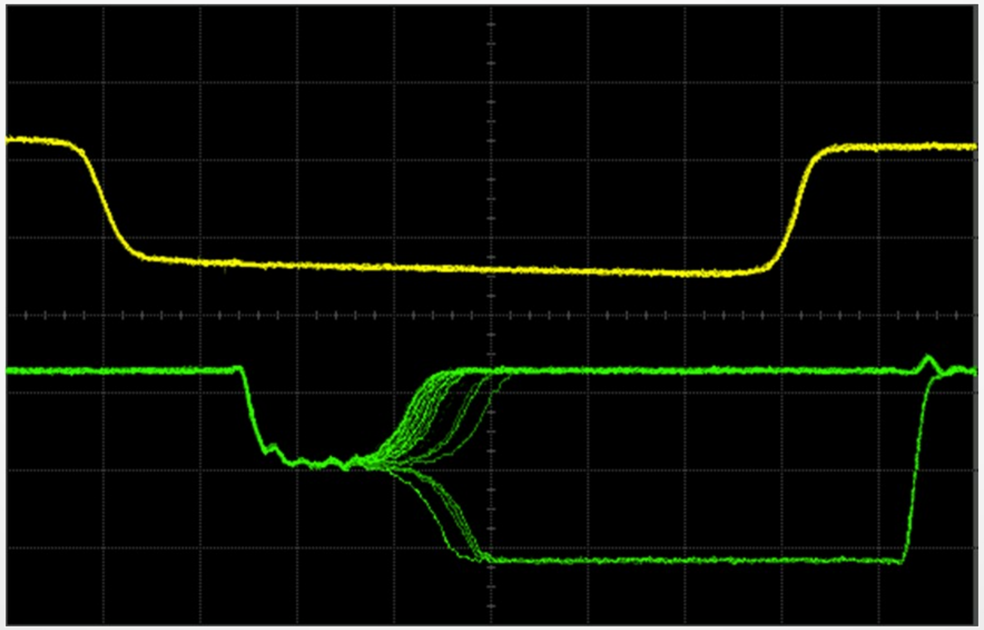
- Because of the construction of a flip-flop [1], the input must be held steady in a period around the rising edge of the clock.
- **Setup time** is the minimum amount of time the data input should be held steady before the clock event, so that the data is reliably sampled by the clock.
- **Hold time** is the minimum amount of time the data input should be held steady after the clock event, so that the data is reliably sampled by the clock.



[1] https://www.edn.com/understanding-the-basics-of-setup-and-hold-time/

Flip-Flops – Metastability

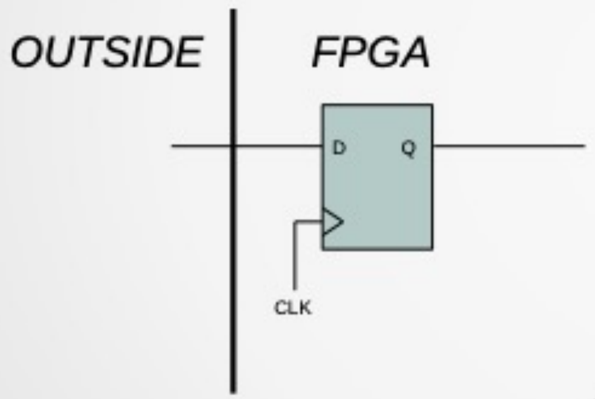
- If setup and hold time are not respected, flip-flops are subject to a problem called **metastability**
- The result is that the output may behave unpredictably, taking many times longer than normal to settle to one state or the other, or even oscillating several times before settling.

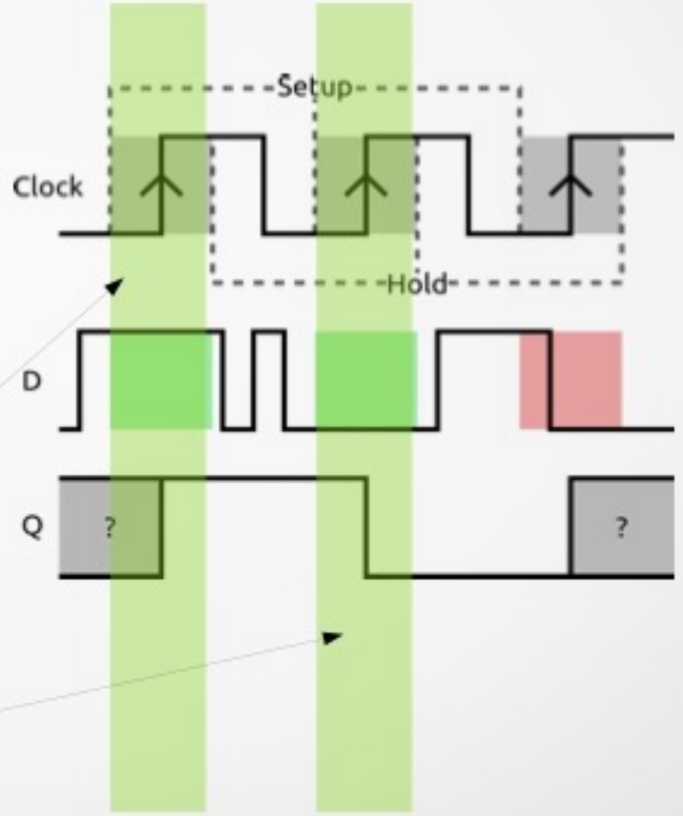


https://youtu.be/5PRuPVIjEcs

Three main reasons for metastability problem (1/3)

- An external signal (user input) is read inside the FPGA:

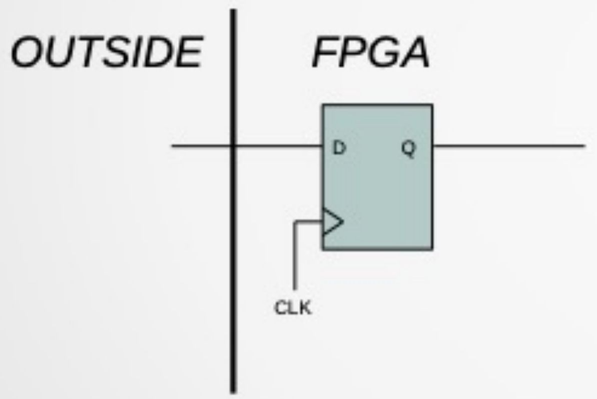


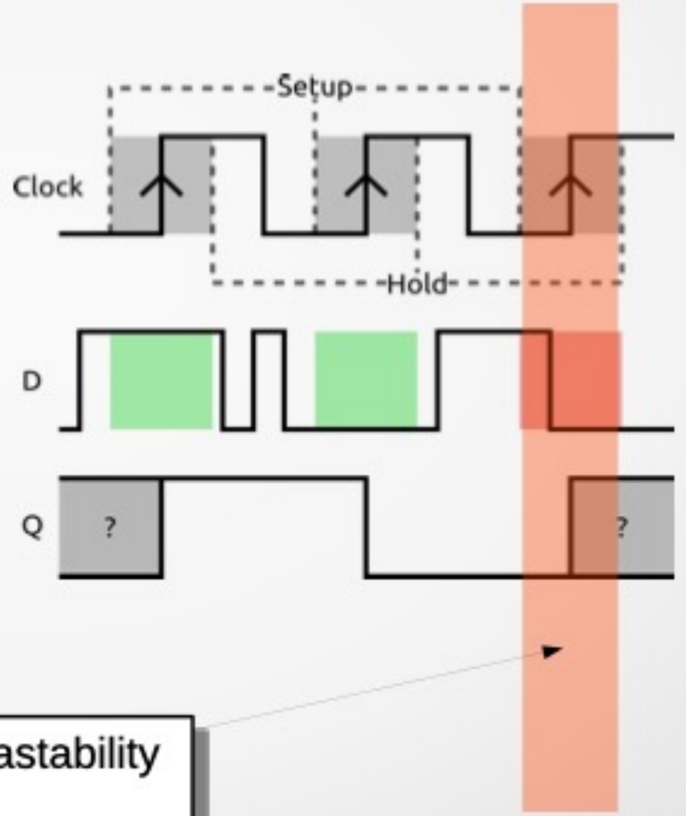


D doesn't change during the aperture.

Three main reasons for metastability problem (1/3)

- An external signal (user input) is read inside the FPGA:

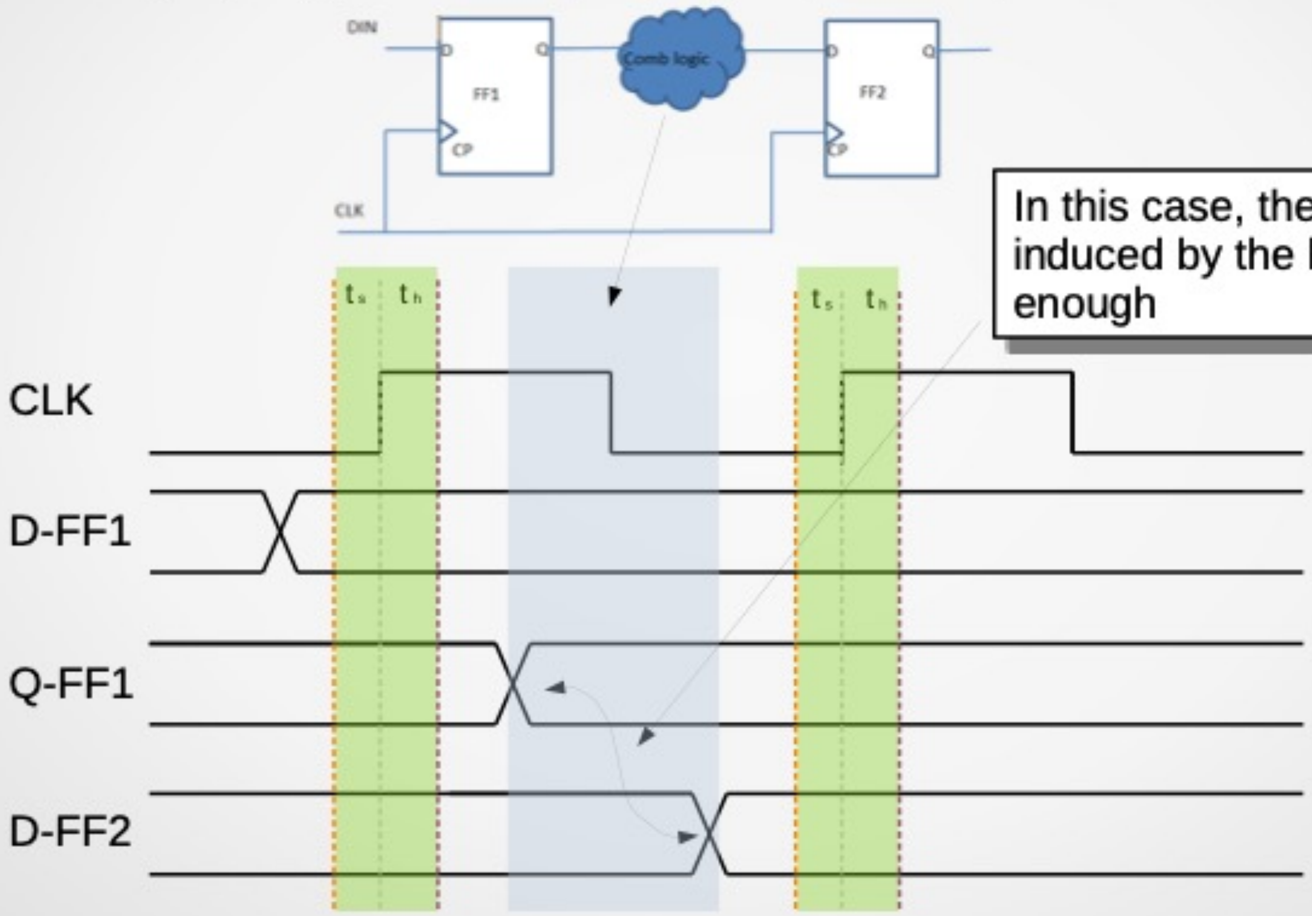




D changes during the aperture→Metastability

Three main reasons for metastability problem (2/3)

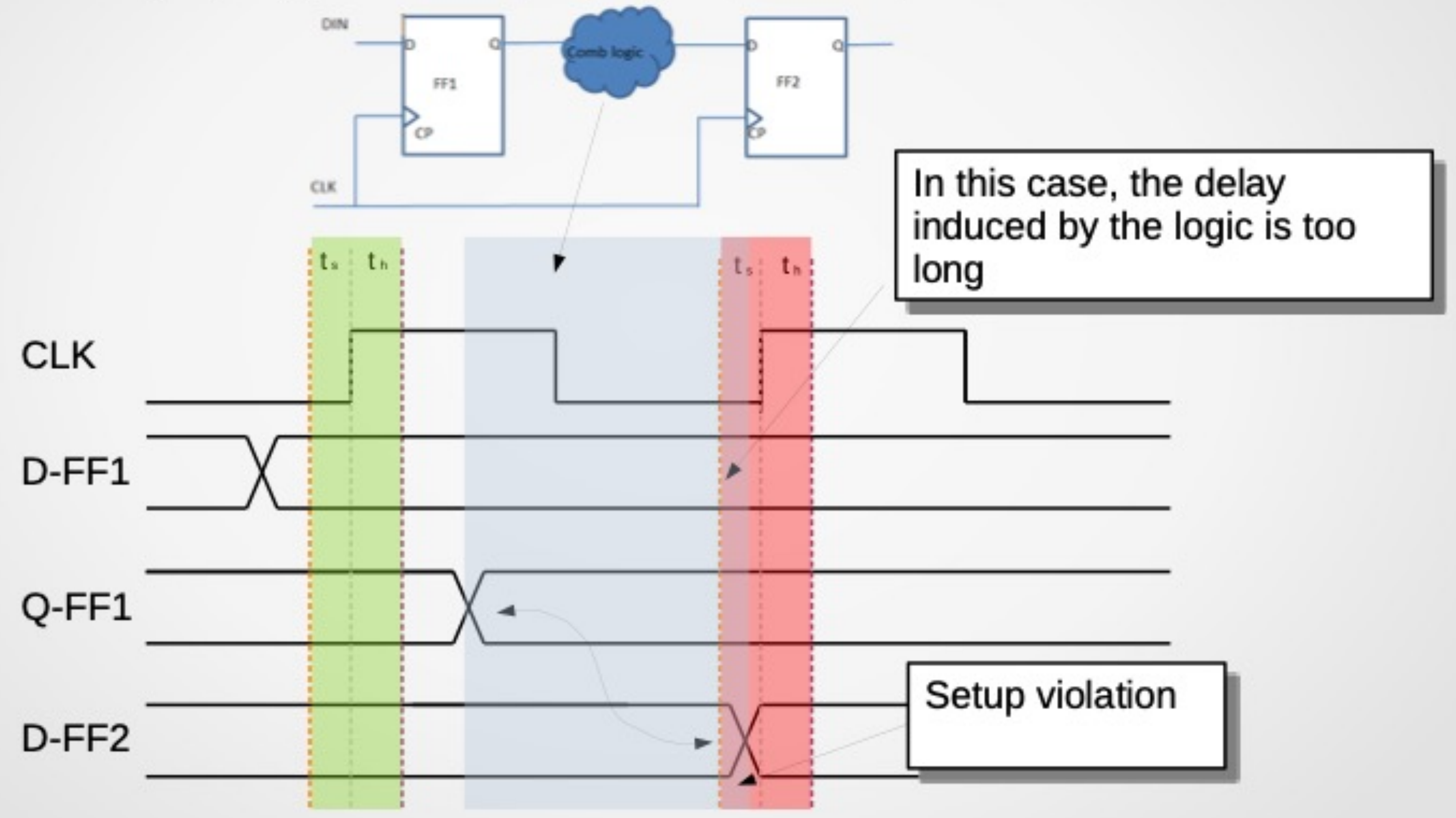
- Too much logic (delay) between flip-flops (setup violation):



In this case, the delay induced by the logic is short enough

Three main reasons for metastability problem (2/3)

- Too much logic (delay) between flip-flops (setup violation):

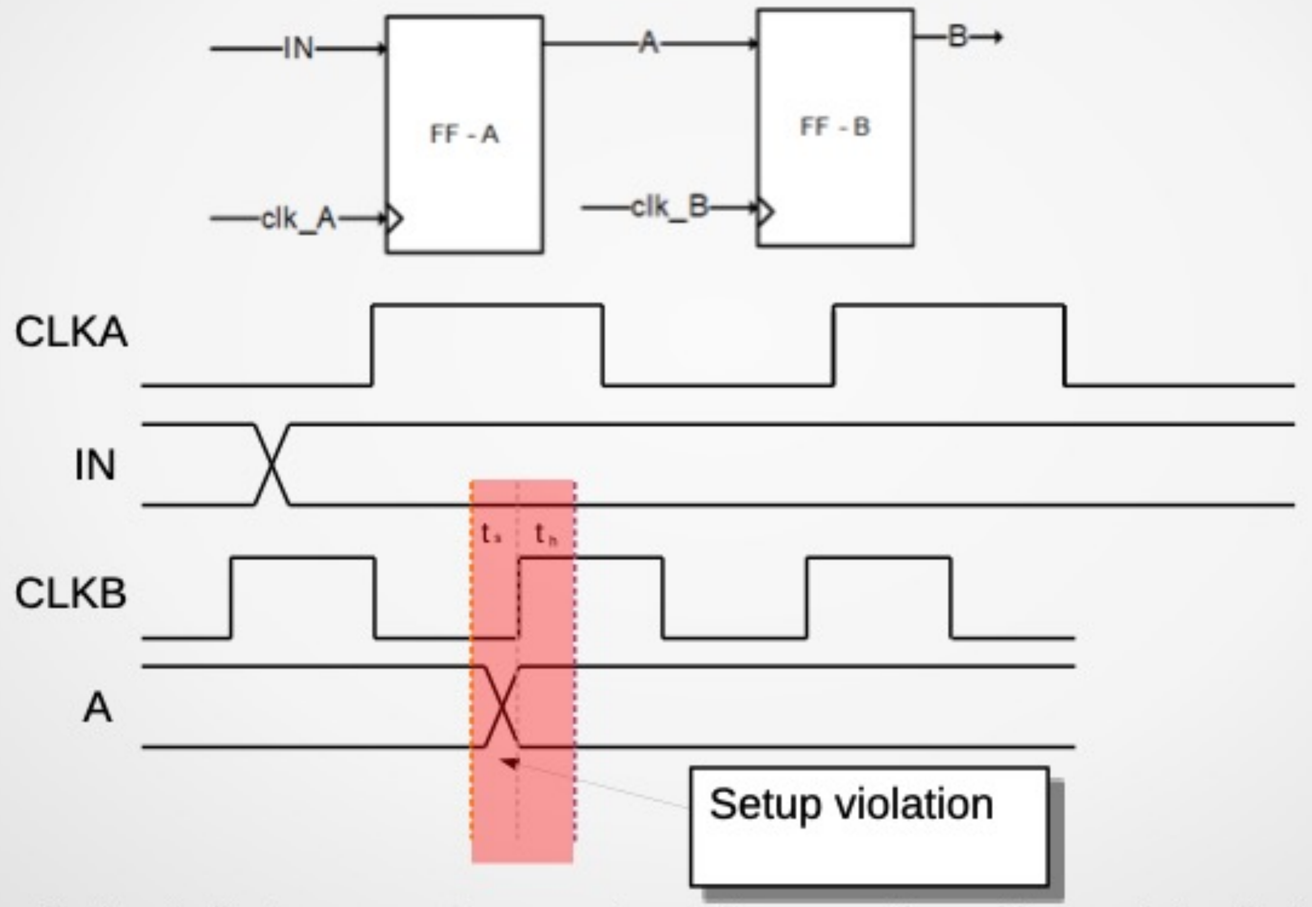


In this case, the delay induced by the logic is too long

Setup violation

Three main reasons for metastability problem (3/3)

- Multiple **clock domains**



Because clk_A and clk_B are asynchronous, A can change anytime with regards to clk_B rising edge.

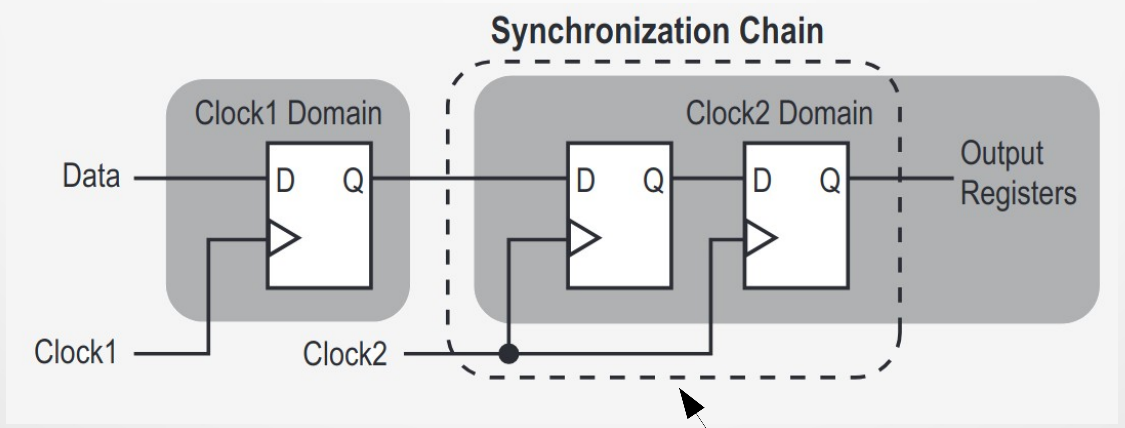
Who is responsible ?

- You are responsible for this. Designers must prevent ***timing*** problems:

  - External asynchronous signals must be handled properly with ***synchronizers***,
  - when using multiple clock domains, use proper ***clock domain crossing*** (CDC) circuits,
  - look at **static timing analysis** report from your synthesis tool and take care (at least evaluate) of every (most) warnings.

Synchronizer

- Use a sequence of registers in the destination clock domain to resynchronize the signal to the new clock domain.
- Allows additional time for a potentially metastable signal to resolve to a known value before the signal is used in the rest of the design.



Must be kept close each other

https://trilobyte.com/pdf/golson_snug14.pdf

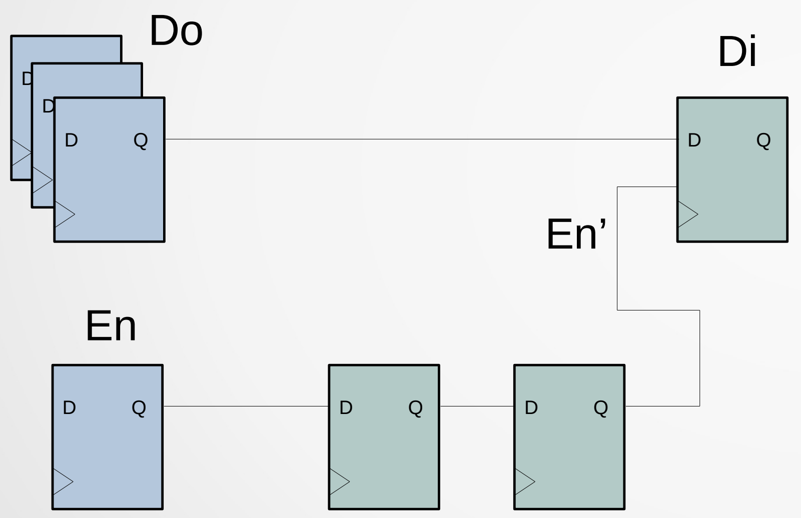
Clock Domain Crossing

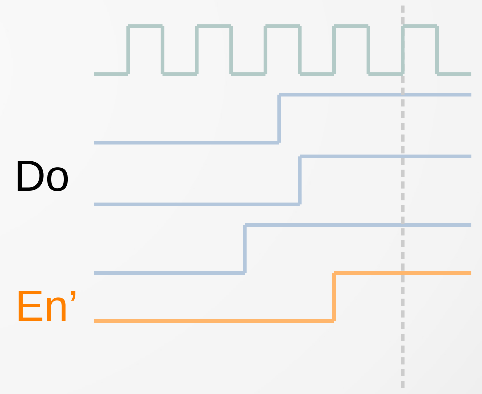
- Used when transferring datas (busses) across clock domain boundaries.

- Two methods:
  - Control based data synchronizers
  - FIFO based data synchronizers

Control based data synchronizers

- The enable signal is responsible to inform the receiving domain that data is stable and ready to be captured.

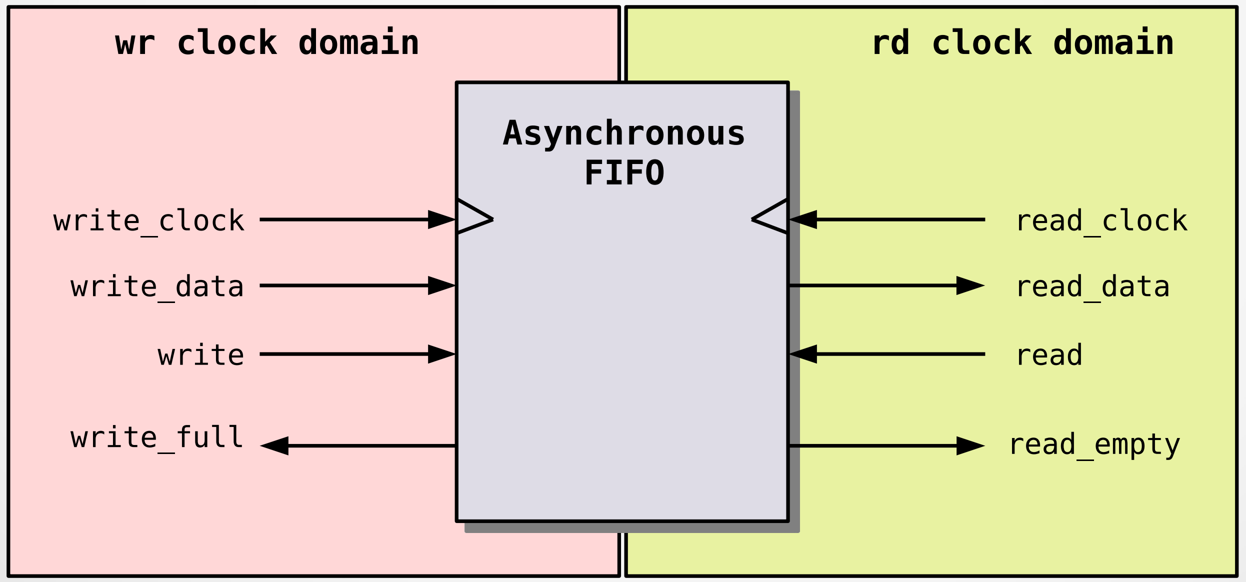


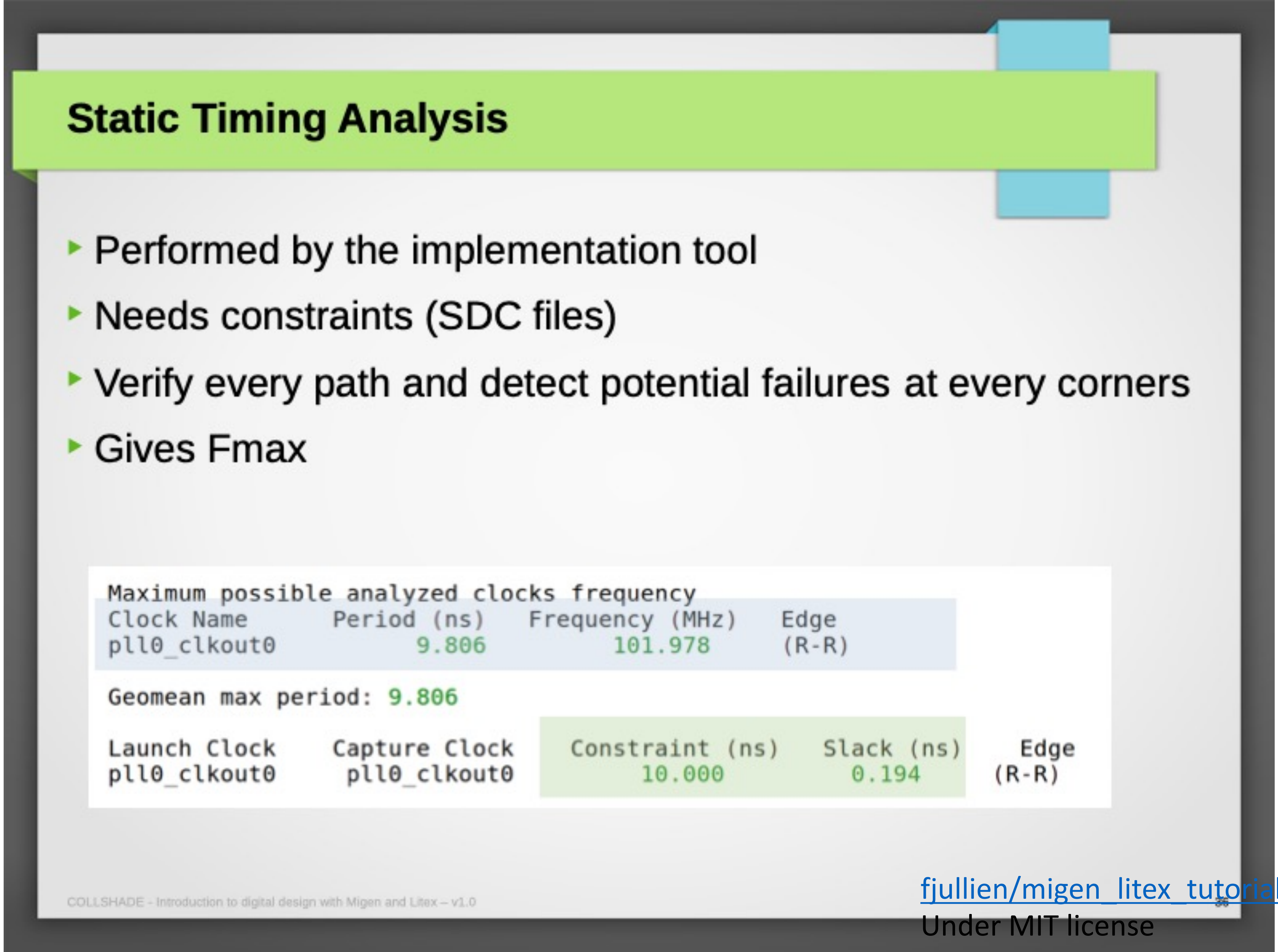


- Control based data synchronizer has limited bandwidth.

FIFO based data synchronizers

- Data is pushed into the FIFO with transmitter clock and pulled out from FIFO with receiver clock.

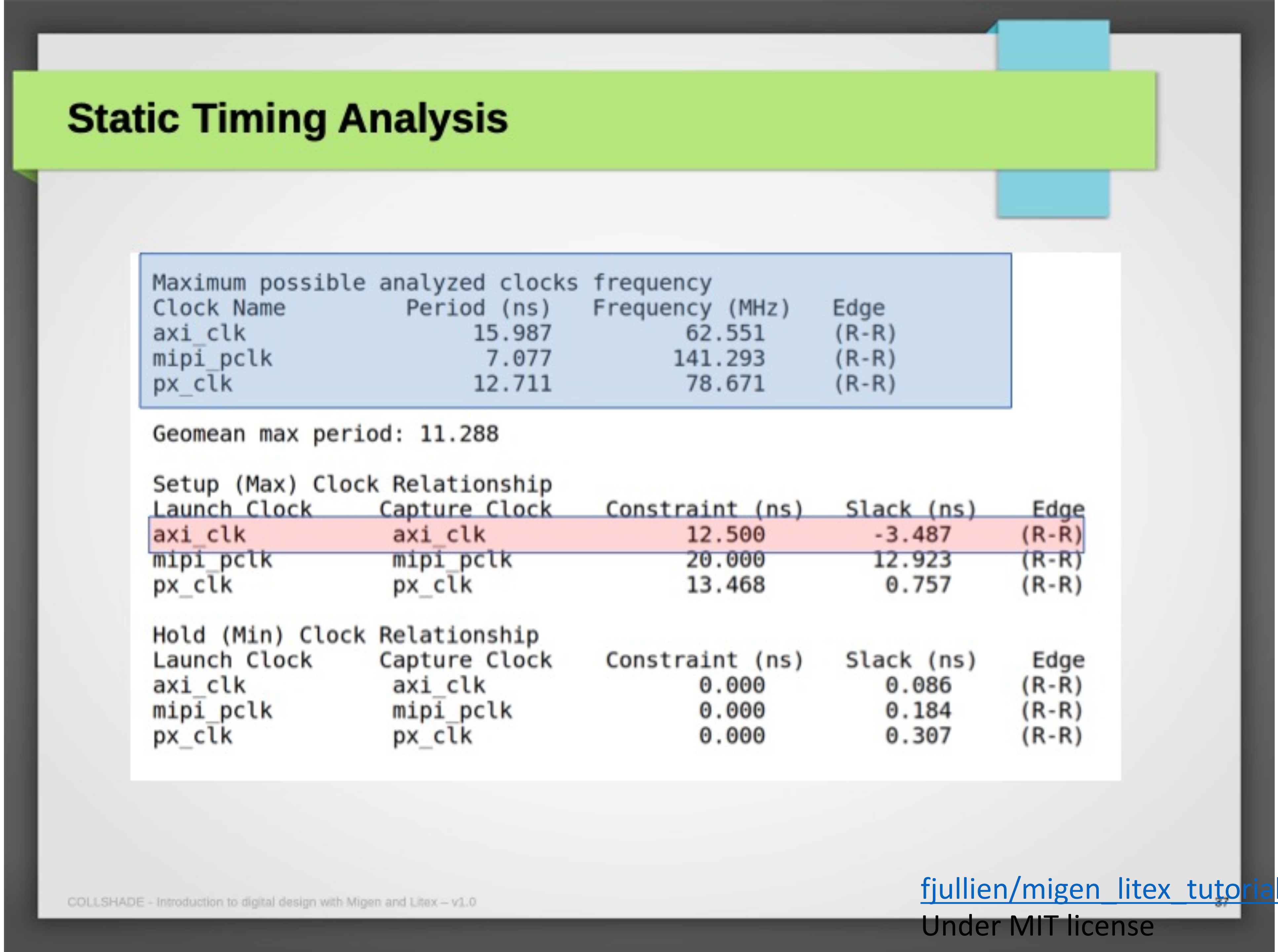




Static Timing Analysis

- Performed by the implementation tool
- Needs constraints (SDC files)
- Verify every path and detect potential failures at every corners
- Gives Fmax

```
Maximum possible analyzed clocks frequency
Clock Name       Period (ns)    Frequency (MHz)    Edge
pll0_clkout0           9.806            101.978    (R-R)

Geomean max period: 9.806

Launch Clock     Capture Clock      Constraint (ns)     Slack (ns)      Edge
pll0_clkout0      pll0_clkout0          10.000             0.194       (R-R)
```



Static Timing Analysis

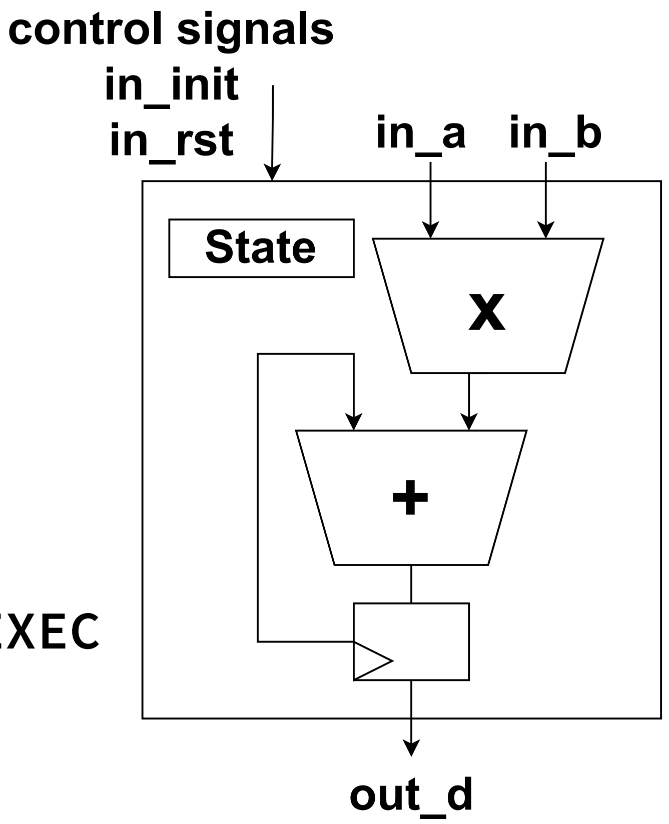
```
Maximum possible analyzed clocks frequency
Clock Name          Period (ns)   Frequency (MHz)   Edge
axi_clk                  15.987            62.551   (R-R)
mipi_pclk                 7.077           141.293   (R-R)
px_clk                   12.711            78.671   (R-R)

Geomean max period: 11.288

Setup (Max) Clock Relationship
Launch Clock     Capture Clock    Constraint (ns)   Slack (ns)    Edge
axi_clk           axi_clk                  12.500       -3.487   (R-R)
mipi_pclk         mipi_pclk                20.000       12.923   (R-R)
px_clk            px_clk                   13.468        0.757   (R-R)

Hold (Min) Clock Relationship
Launch Clock     Capture Clock    Constraint (ns)   Slack (ns)    Edge
axi_clk           axi_clk                   0.000        0.086   (R-R)
mipi_pclk         mipi_pclk                 0.000        0.184   (R-R)
px_clk            px_clk                    0.000        0.307   (R-R)
```

Outline

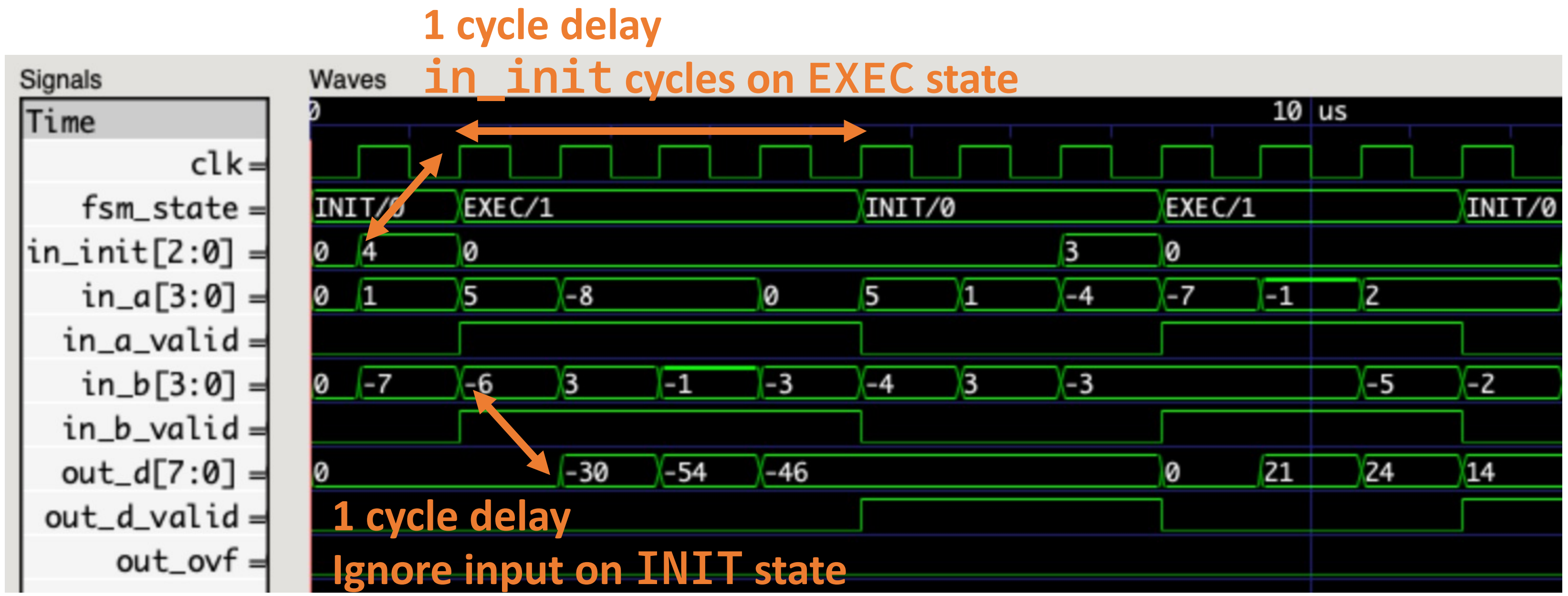
- Prevent metastability
- **Practice**
  - Implement PE, stacked PE

Implement PE

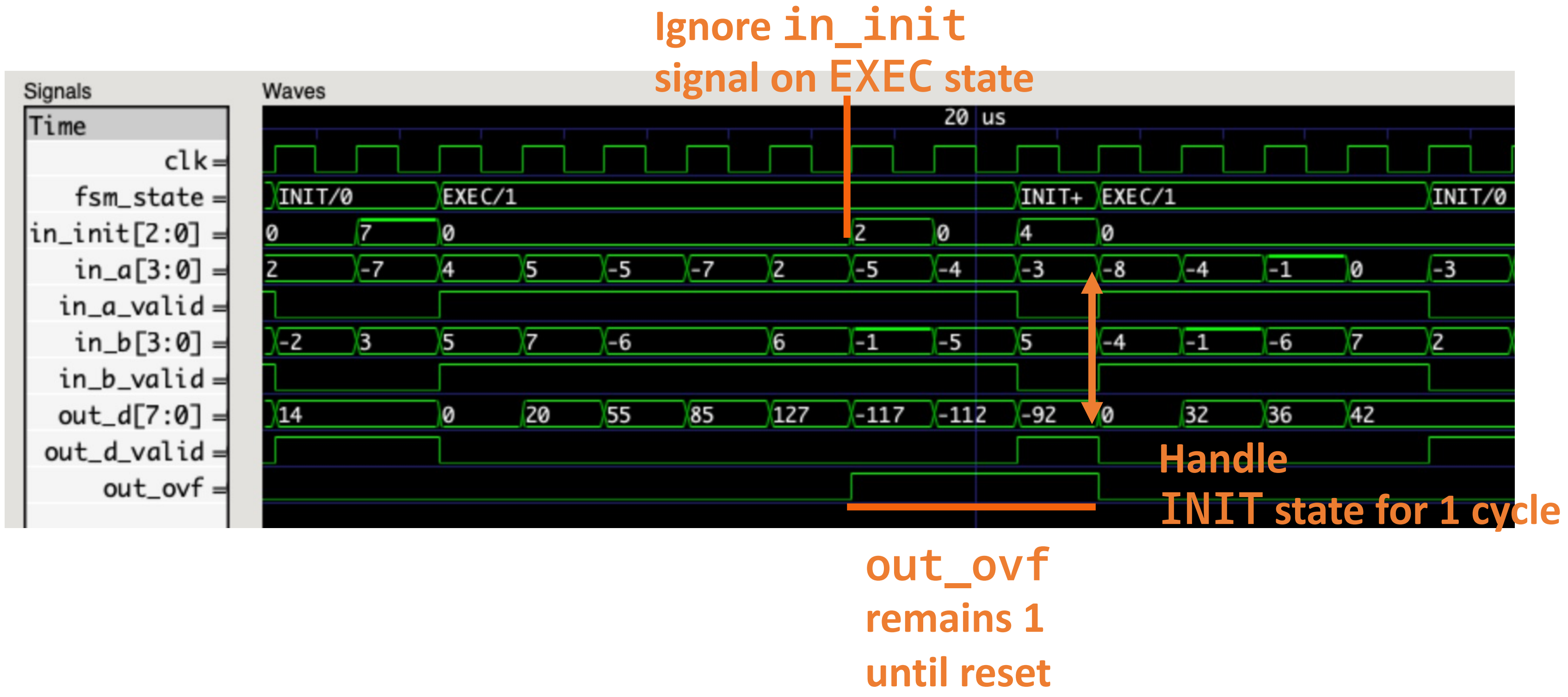
- PE := Processing Element
  - Sequential FMA for V * V inner product
  - Reuse `MAC` of last week
- States
  - **`INIT`**
    - Wait for **`in_init`** signal
    - If **`in_init`** signal is non-0, reset `MAC` and next state is **`EXEC`**
    - NOTE **`in_init`** is of **`cnt_bits`**-bit
  - **`EXEC`**
    - Compute partial sum over **`in_init`** cycles
    - Then return to **`INIT`**



Implement PE



Implement PE

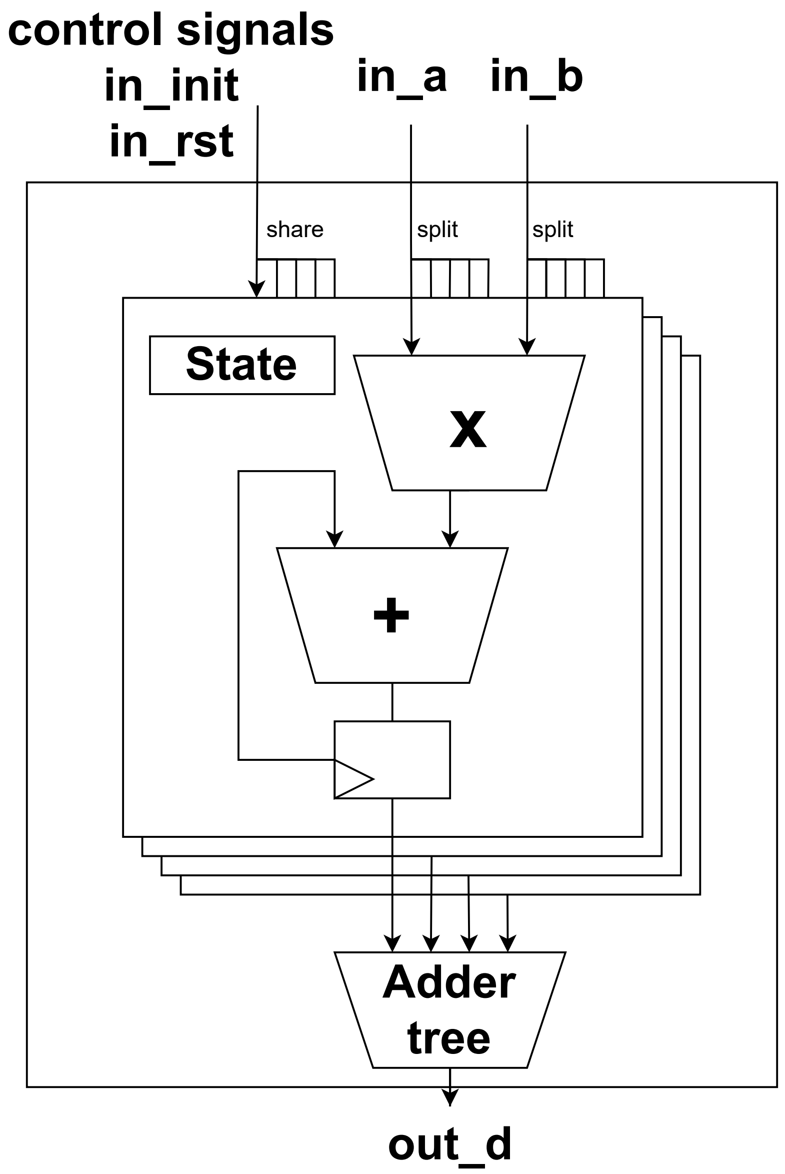


```
in_init[2:0]  0   7   0                     2    0    4   0
in_a[3:0]     2   -7  4   5   -5  -7  2     -5   -4   -3  -8  -4  -1  0   -3
in_b[3:0]     -2  3   5   7   -6      6     -1   -5   5   -4  -1  -6  7   2
out_d[7:0]    14      0   20  55  85  127   -117 -112 -92 0   32  36  42
```

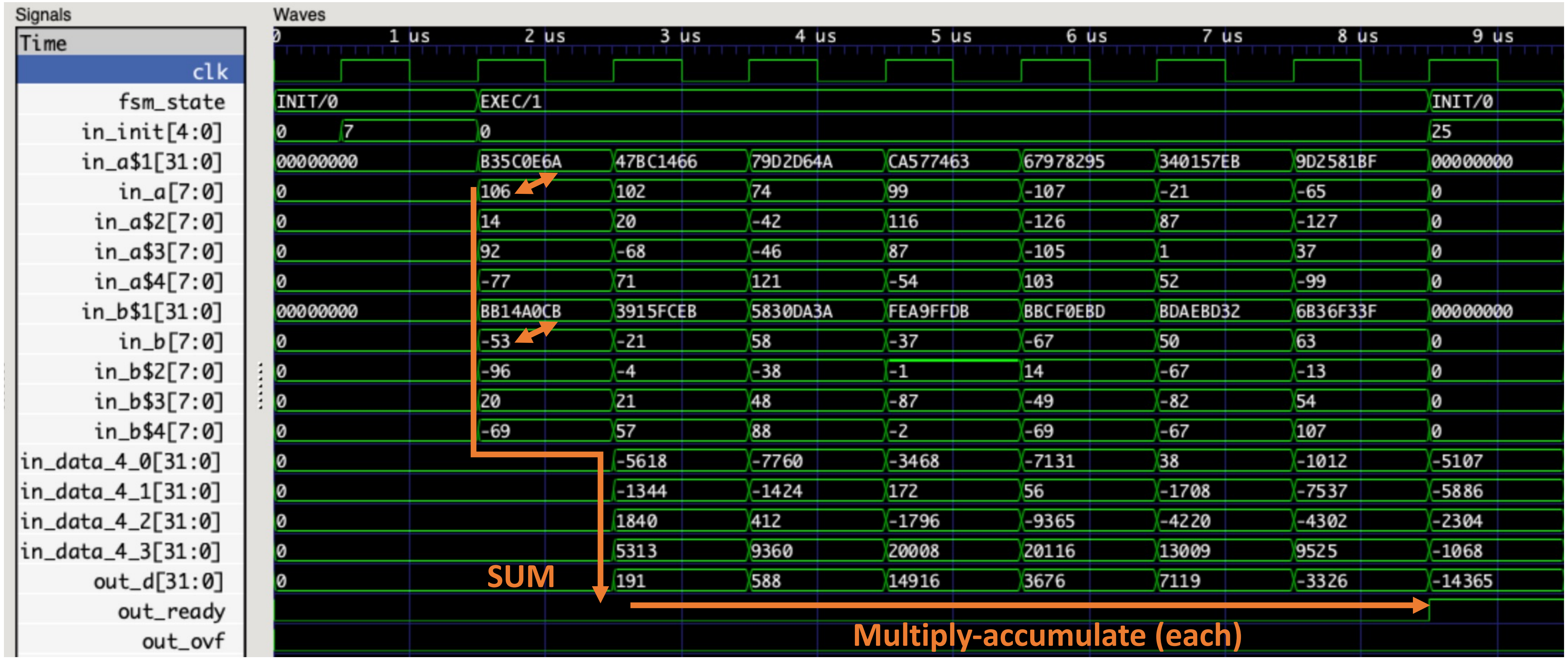
Implement Stacked PE

- `width`-bit `in_a, in_b`
  - Split into multiple `num_bits` signals

- Reuse `PE` and `AdderTree`
  - Don’t forget to add on `m.submodules`



Implement Stacked PE



```
Signals              Waves
Time                 0        1 us      2 us      3 us      4 us      5 us      6 us      7 us      8 us      9 us
clk
fsm_state            INIT/0             EXEC/1                                                              INIT/0
in_init[4:0]         0   7              0                                                                   25
in_a$1[31:0]         00000000           B35C0E6A  47BC1466  79D2D64A  CA577463  67978295  340157EB  9D2581BF  00000000
in_a[7:0]            0                  106       102       74        99        -107      -21       -65       0
in_a$2[7:0]          0                  14        20        -42       116       -126      87        -127      0
in_a$3[7:0]          0                  92        -68       -46       87        -105      1         37        0
in_a$4[7:0]          0                  -77       71        121       -54       103       52        -99       0
in_b$1[31:0]         00000000           BB14A0CB  3915FCEB  5830DA3A  FEA9FFDB  BBCF0EBD  BDAEBD32  6B36F33F  00000000
in_b[7:0]            0                  -53       -21       58        -37       -67       50        63        0
in_b$2[7:0]          0                  -96       -4        -38       -1        14        -67       -13       0
in_b$3[7:0]          0                  20        21        48        -87       -49       -82       54        0
in_b$4[7:0]          0                  -69       57        88        -2        -69       -67       107       0
in_data_4_0[31:0]    0                            -5618     -7760     -3468     -7131     38        -1012     -5107
in_data_4_1[31:0]    0                            -1344     -1424     172       56        -1708     -7537     -5886
in_data_4_2[31:0]    0                            1840      412       -1796     -9365     -4220     -4302     -2304
in_data_4_3[31:0]    0                            5313      9360      20008     20116     13009     9525      -1068
out_d[31:0]          0                            191       588       14916     3676      7119      -3326     -14365
out_ready
out_ovf
```

Practice

- Skeleton code
  https://www.notion.so/skeleton-code-91e6f0b4d8b44ab1bccb71ee4f497cf2?pvs=4
- Implement and show
  - Waveform
  - Code