

Design Digitaler Schaltkreise

Synthesis and Timing Analysis Basics

Asic and Detector Lab - IPE

Prof. Ivan Peric ivan.peric@kit.edu

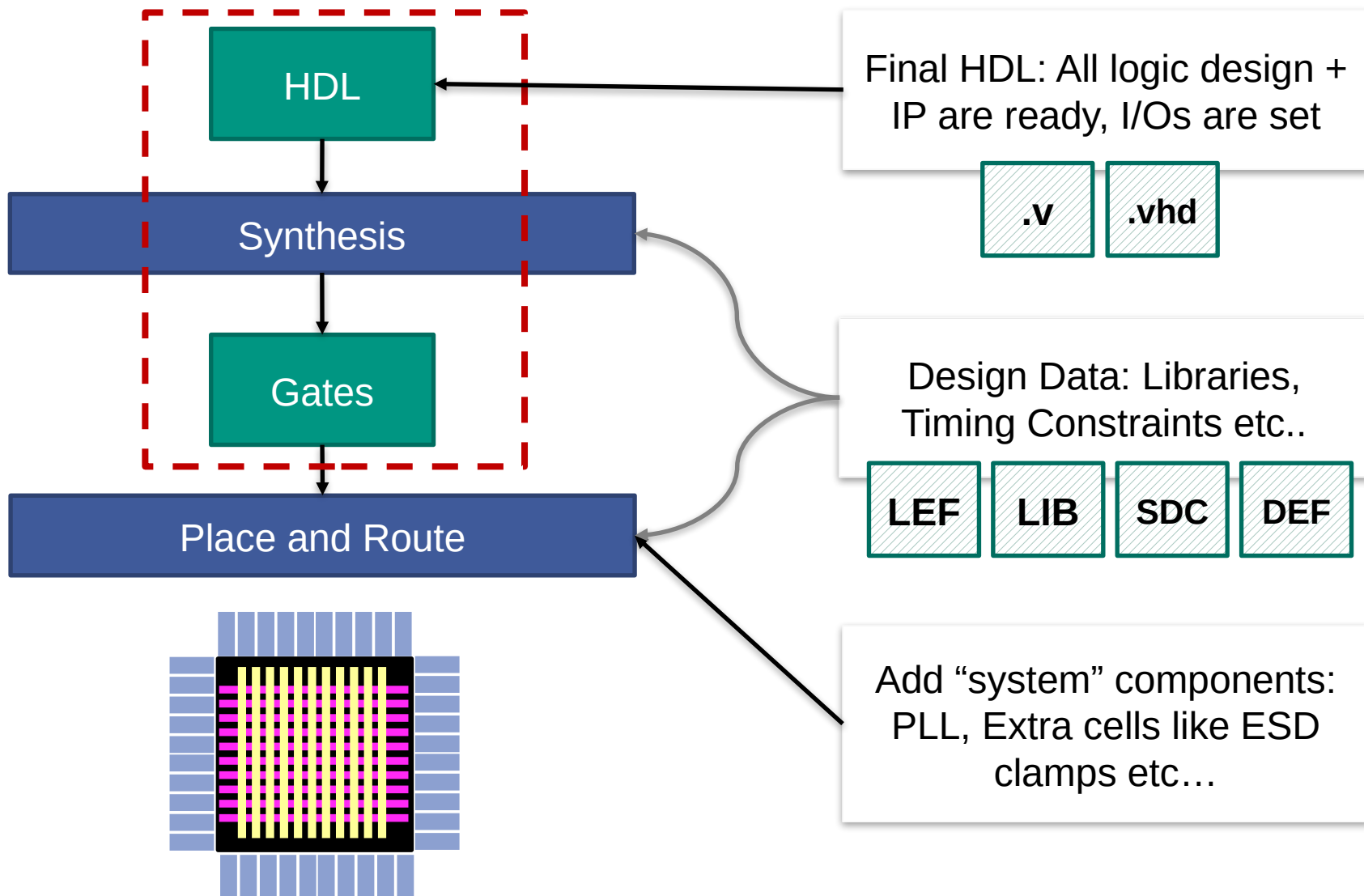
Richard Leys richard.leys@kit.edu

Ilias: https://ilias.studium.kit.edu/goto_produktiv_crs_430424.html

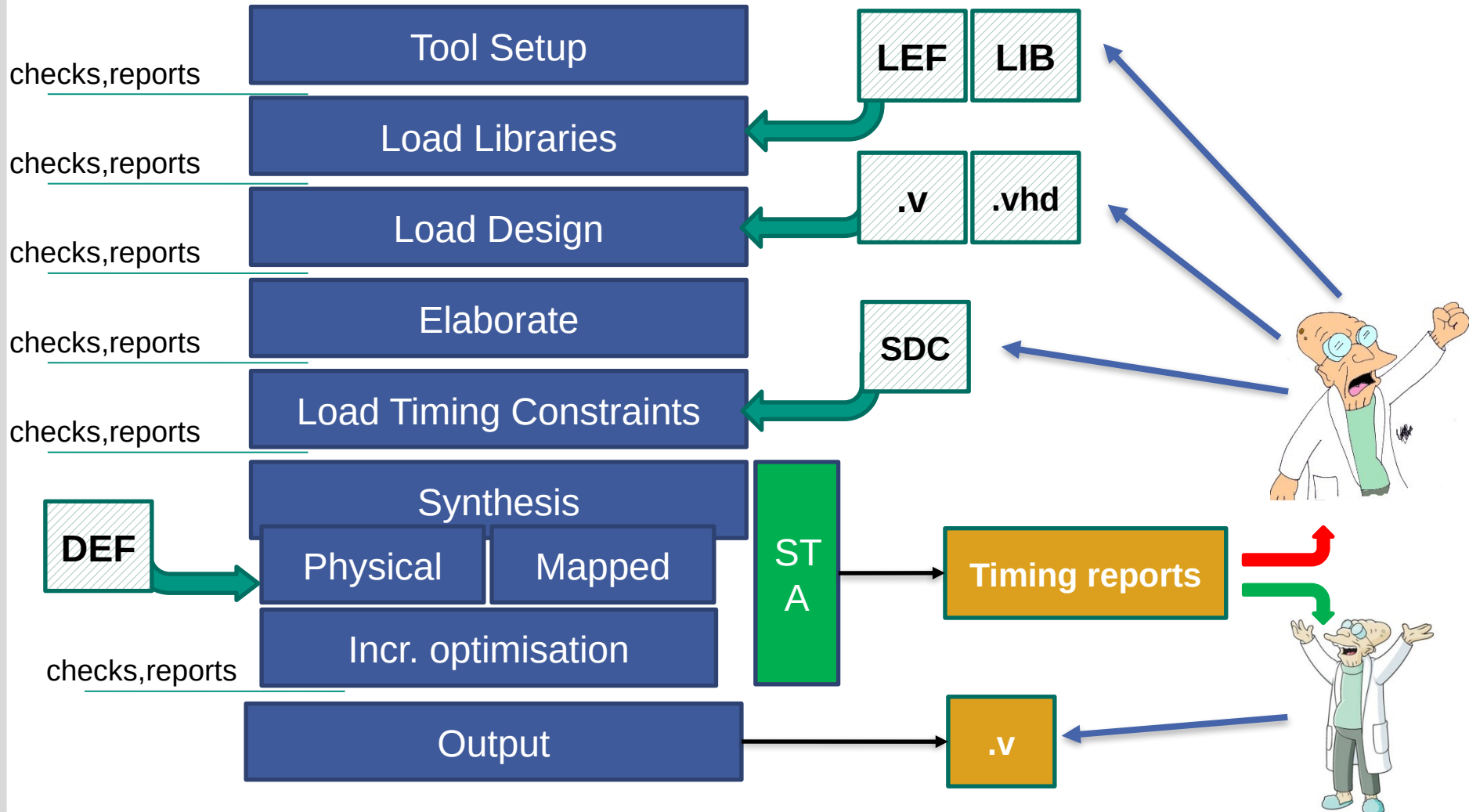
Lecture Goal

- Learn the basics of the Synthesis Steps
- Get used to the required data used by synthesis
 - Sources, Timing Libraries etc..
- Learn the Static Timing Analysis concept:
 - Semantic: Timing delay, slew, slack etc...
 - Understand setup and hold timing violations
 - See how timing reports look like
- Learn the basics of timing constraints writing:
 - Clock domain
 - Input/Output constraining

Flow overview reminder



Generic Synthesis Flow Reference



Timing Library

- Seen low level Flip-Flop and Cells in previous lecture

- Timing Library specified for one corner:

- Process Information: Voltage Temperature etc...

- Speed Case: Slow, Typical, Fast

- Timing Libraries provide characterisation of cells:

- timing “arcs” between input/outputs

- Ex: Q to CK

- Wire loading models

- Power

- Pin Capacitance etc...

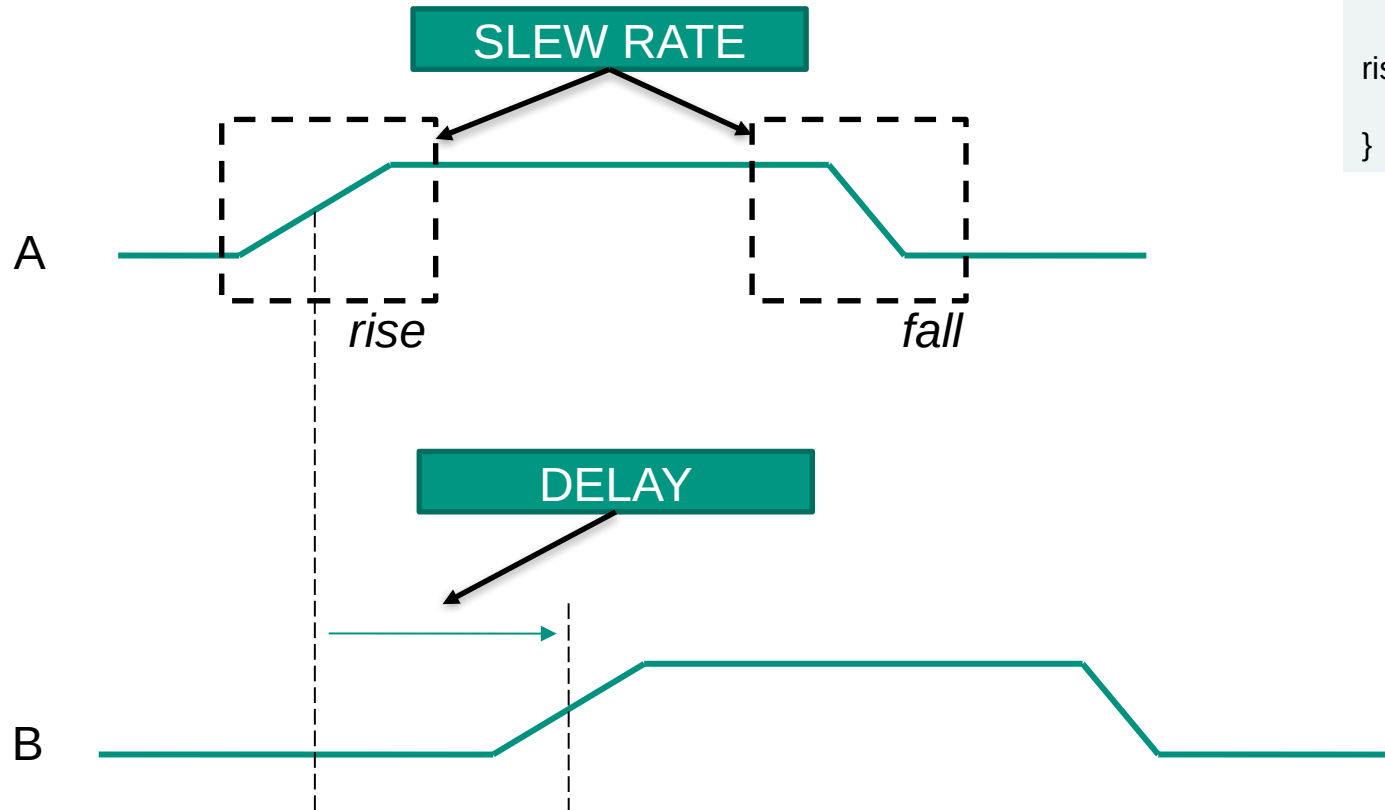
- Format used in RTL compiler: Synopsis liberty

```
library (uk65lsc1mvbbr_090c125_wc) {
    ...
    nom_process : 1.0 ;
    nom_temperature : 125.0 ;
    nom_voltage : 0.9 ;
    operating_conditions(uk65lsc1mvbbr_090c125_wc) {
        process : 1.0;
        temperature : 125.0;
        voltage : 0.9;
        tree_type : balanced_tree
    }
    ...
    cell(DFQBM1RA) {
        area : 7.200000 ;
        cell_footprint : DFQB ;
    }
    ...
    pin(QB) {
        capacitance : 0.000000 ;
        direction : output ;
    }
    ...
    timing() {
        related_pin : "CK" ;
        timing_type : rising_edge ;

        cell_rise(...) {
        }

        rise_transition(...) {
        }
    }
    ...
}
```

Timing Arcs: Delay and Slew



```
rise_transition(...) {  
}
```

```
cell_rise(...) {  
}
```

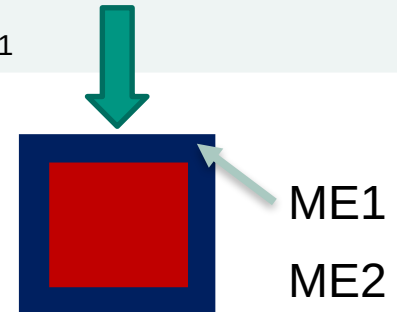
LEF (Library Exchange Format) File

- LEF files contain physical information for Cells
- Technology rules and specs can be defined
 - Ex: Number of layers
 - Ex: VIA Definition
- Cells physical information include:
 - Size
 - I/O (Pins) size and locations
 - Layer obstructions
- Full layout is not included

```

LAYER VI1
TYPE CUT ;
SPACING 0.1 ;
SPACING 0.11
ADJACENTCUTS 3 WITHIN 0.14 ;
...
END VI1

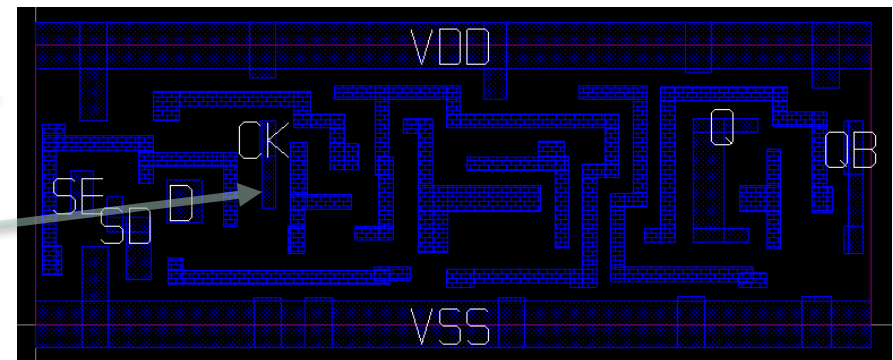
```



```

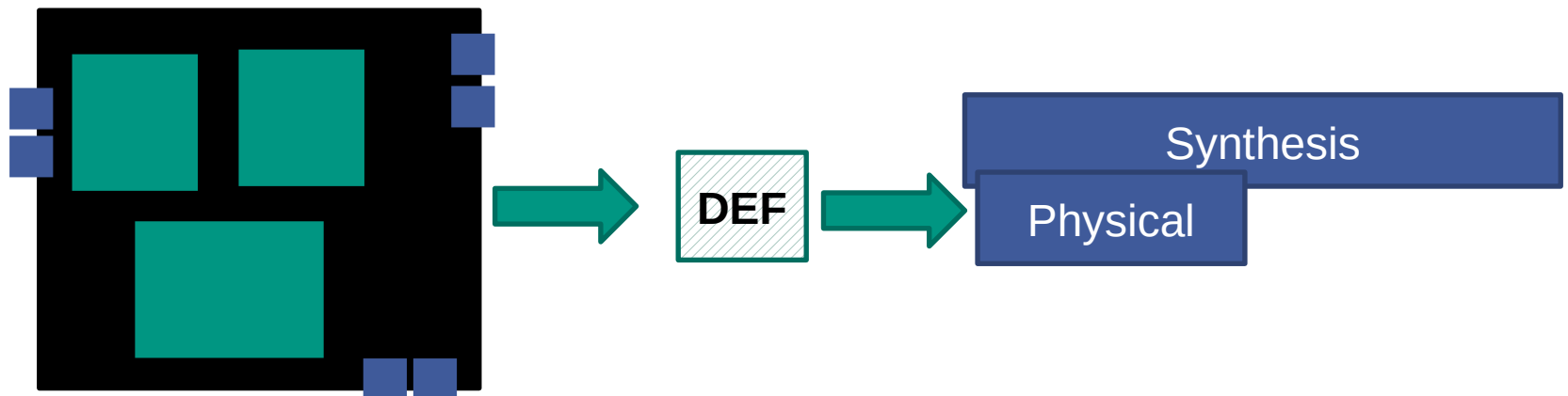
MACRO SDFQBRM1RA
CLASS CORE ;
SIZE 6.0000 BY 1.8000 ;
SYMMETRY X Y ;
SITE CORE ;
PIN CK
DIRECTION INPUT ;
PORT
LAYER ME1 ;
RECT 1.4400 0.4900 1.5500 0.8900 ;
END
END CK
ENDMACRO

```



DEF File

- Design Exchange Format
- Contains Design Data put together:
 - Floorplaning info
 - Netlist etc...
- Used by Synthesis to load floorplaning information



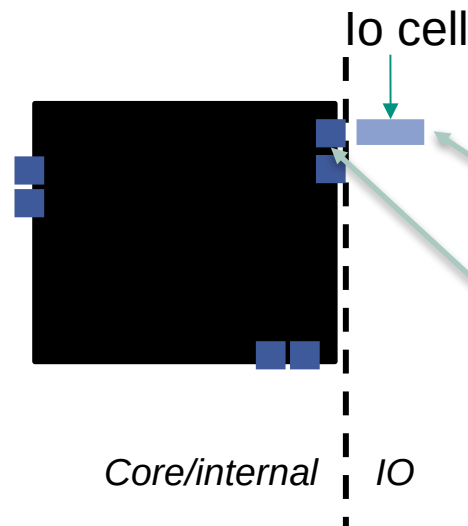
Floorplan: Size and block, io placements

Synthesis Ready HDL: Top Level

- Two types of Top Level HDL
 - Partition Module: Just I/O which are pins
 - Final Top Level: add instances of real I/O Cells

Partition: Just pins

Top: With IO Cells



```

1 module xxx (
2
3   input a,
4
5 );
6
7   wire a_internal;
8
9   IUMA a_io(.PAD(a),.DI(a_internal));
10
11
12   mymodule mymodule_I (
13
14     .anotherinput(a_internal)
15 );
16
17 endmodule

```

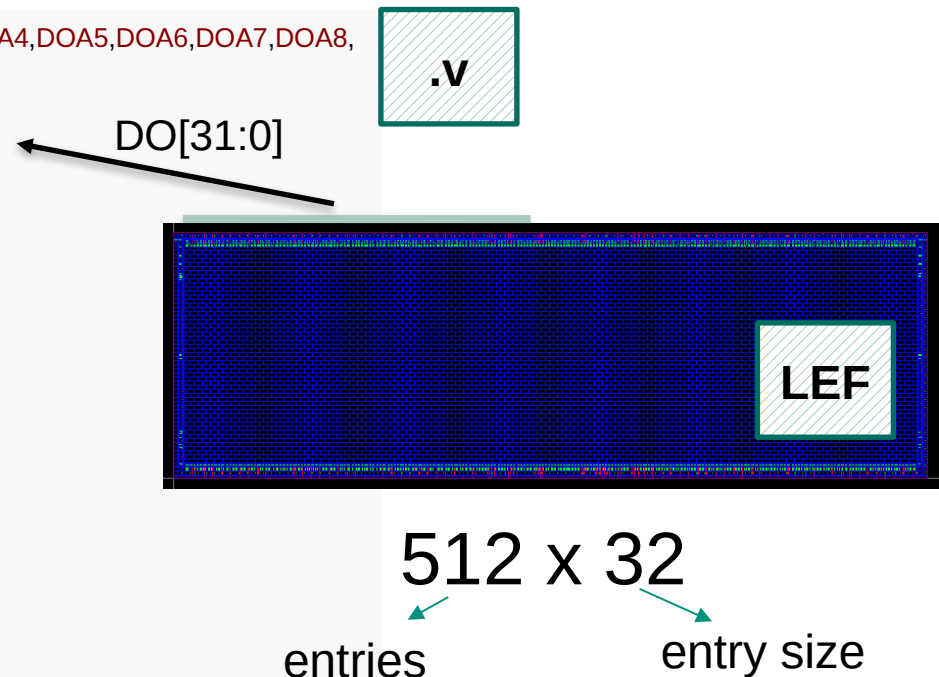
Top Level HDL: IP Blocks (RAM example)

- All special components acting in the logic design must be instantiated
 - Use Provided Verilog, Timing and LEF from Provider
- Standard example: RAMS
- In FPGA, a RAM is an array, mapped by the tool
- In ASIC, the RAM is instantiated
- Example in UMC65:

```

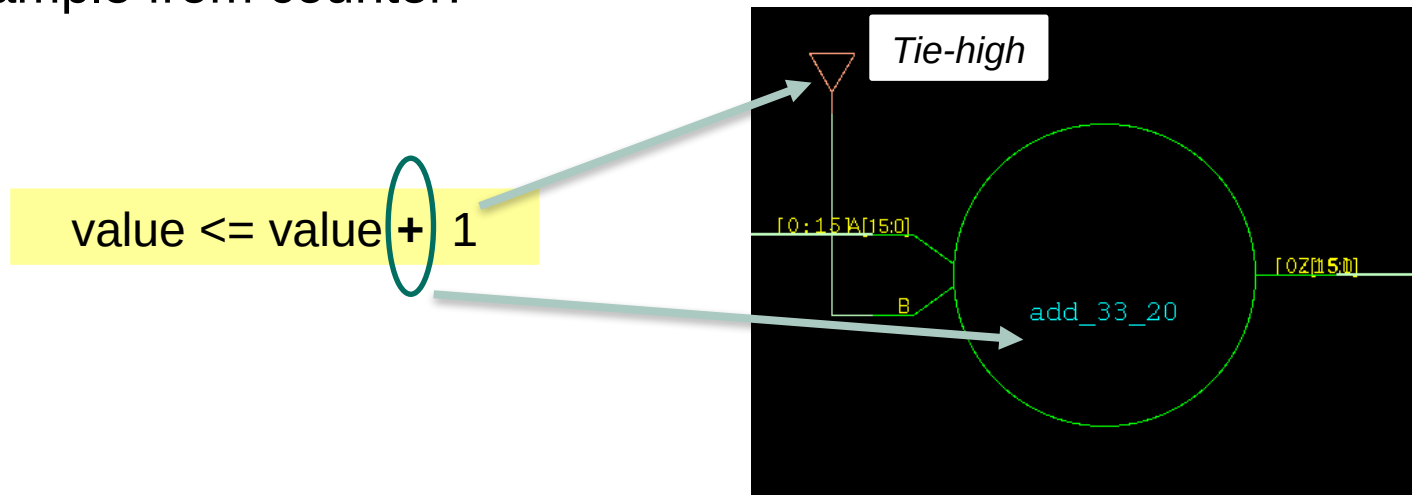
1 module SJKA65_512X32X1CM4 (DOA0,DOA1,DOA2,DOA3,DOA4,DOA5,DOA6,DOA7,DOA8,
2 DOA9,DOA10,DOA11,DOA12,DOA13,DOA14,DOA15,
3 DOA16,DOA17,DOA18,DOA19,DOA20,DOA21,DOA22,
4 DOA23,DOA24,DOA25,DOA26,DOA27,DOA28,DOA29,
5 DOA30,DOA31,DOB0,DOB1,DOB2,DOB3,DOB4,
6 DOB5,DOB6,DOB7,DOB8,DOB9,DOB10,DOB11,DOB12,
7 DOB13,DOB14,DOB15,DOB16,DOB17,DOB18,DOB19,
8 DOB20,DOB21,DOB22,DOB23,DOB24,DOB25,DOB26,
9 DOB27,DOB28,DOB29,DOB30,DOB31,A0,A1,
10 A2,A3,A4,A5,A6,A7,A8,B0,B1,B2,B3,B4,B5,B6,B7,B8,
11 DIA0,DIA1,DIA2,DIA3,DIA4,DIA5,DIA6,DIA7,DIA8,
12 DIA9,DIA10,DIA11,DIA12,DIA13,DIA14,DIA15,
13 DIA16,DIA17,DIA18,DIA19,DIA20,DIA21,DIA22,
14 DIA23,DIA24,DIA25,DIA26,DIA27,DIA28,DIA29,
15 DIA30,DIA31,DIB0,DIB1,DIB2,DIB3,DIB4,
16 DIB5,DIB6,DIB7,DIB8,DIB9,DIB10,DIB11,DIB12,
17 DIB13,DIB14,DIB15,DIB16,DIB17,DIB18,DIB19,
18 DIB20,DIB21,DIB22,DIB23,DIB24,DIB25,DIB26,
19 DIB27,DIB28,DIB29,DIB30,DIB31,WEAN,WEBN,CSAN,
20 CSBN,CKA,DVSE,DVS0,DVS1,DVS2,DVS3,
21 CKB);

```



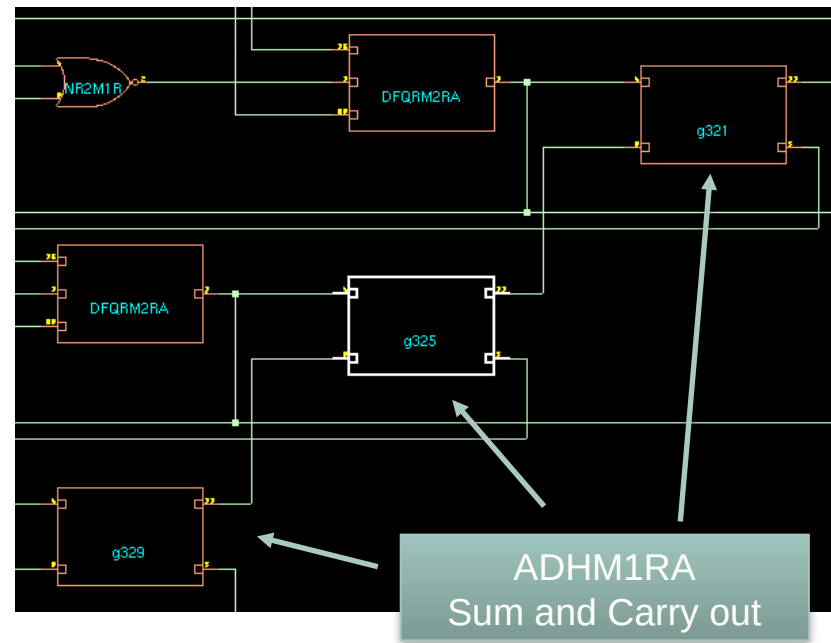
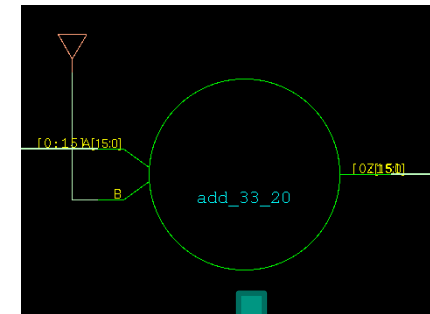
Elaboration

- Parses and compiles the design to an internal logic representation
- Basically the same as in any other tool:
 - Simulation: irun runs “ncelab”
- Check the elaborate output for errors and warnings:
 - Unsupported syntax
 - Connection width mismatches etc..
- Example from counter:



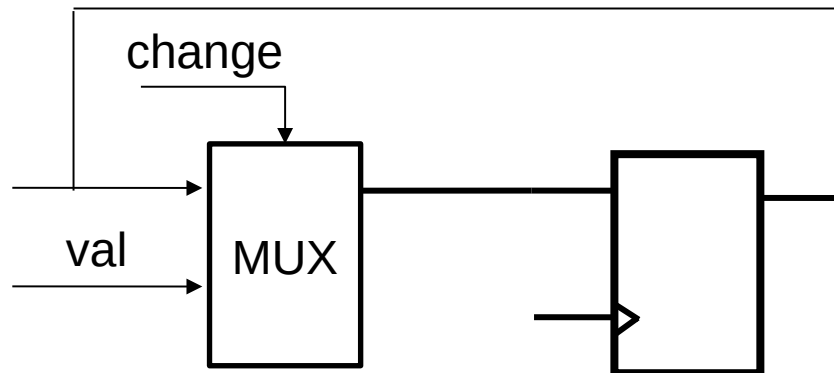
Synthesis: Principle

- Two kinds of operations:
 - Boolean logic (& , ! , ^ ...)
 - Arithmetic operators (+,- etc...)
- Principle:
 - Minimize Boolean logic network
 - Map logic and arithmetic to gates and tries to match costs functions:
 - Area
 - Power
 - Timing
- Main Focus to be chosen:
 - Power ? Area ? Timing?
- Usually: Meet timing for performance

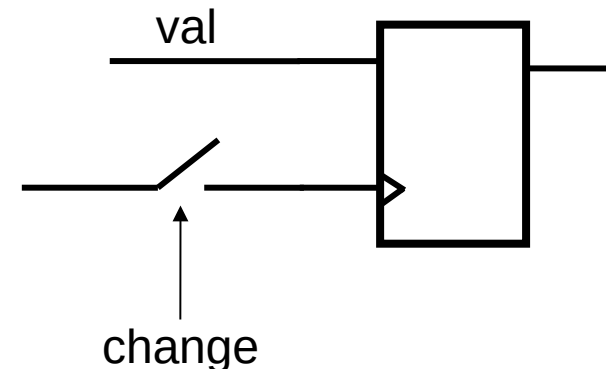


Synthesis: Clock Gating example

- Clock gating is a good example of synthesis optimisation
- It can help save power and modifies timing
- Concept:
 - If a register value changes when an **enable** signal is asserted, then remove the clock when not changing



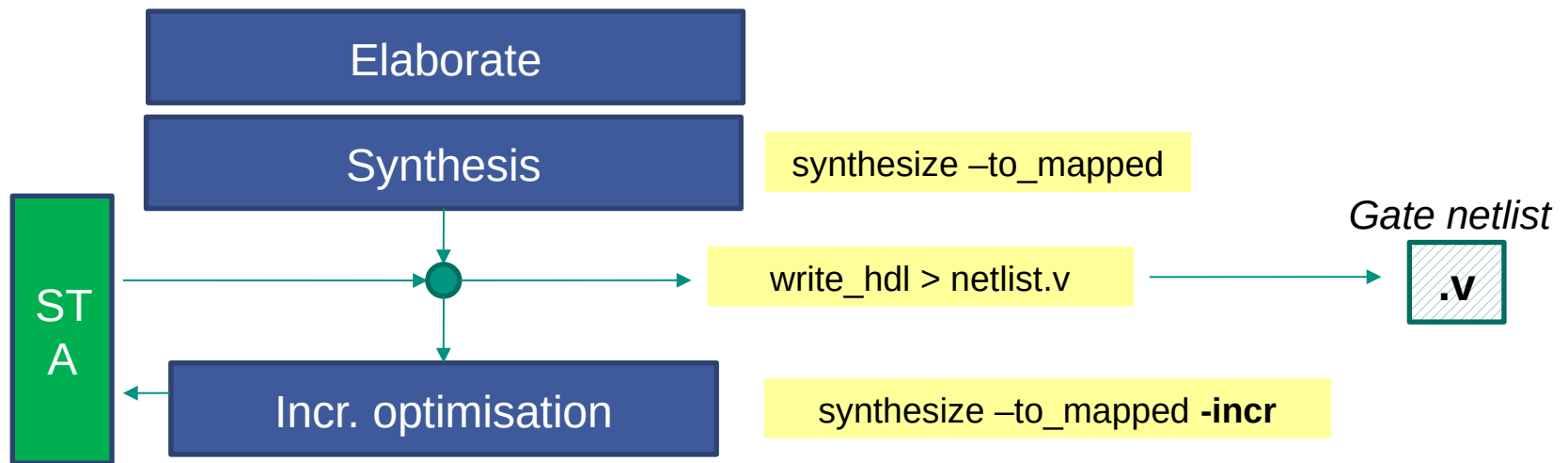
No clock gating



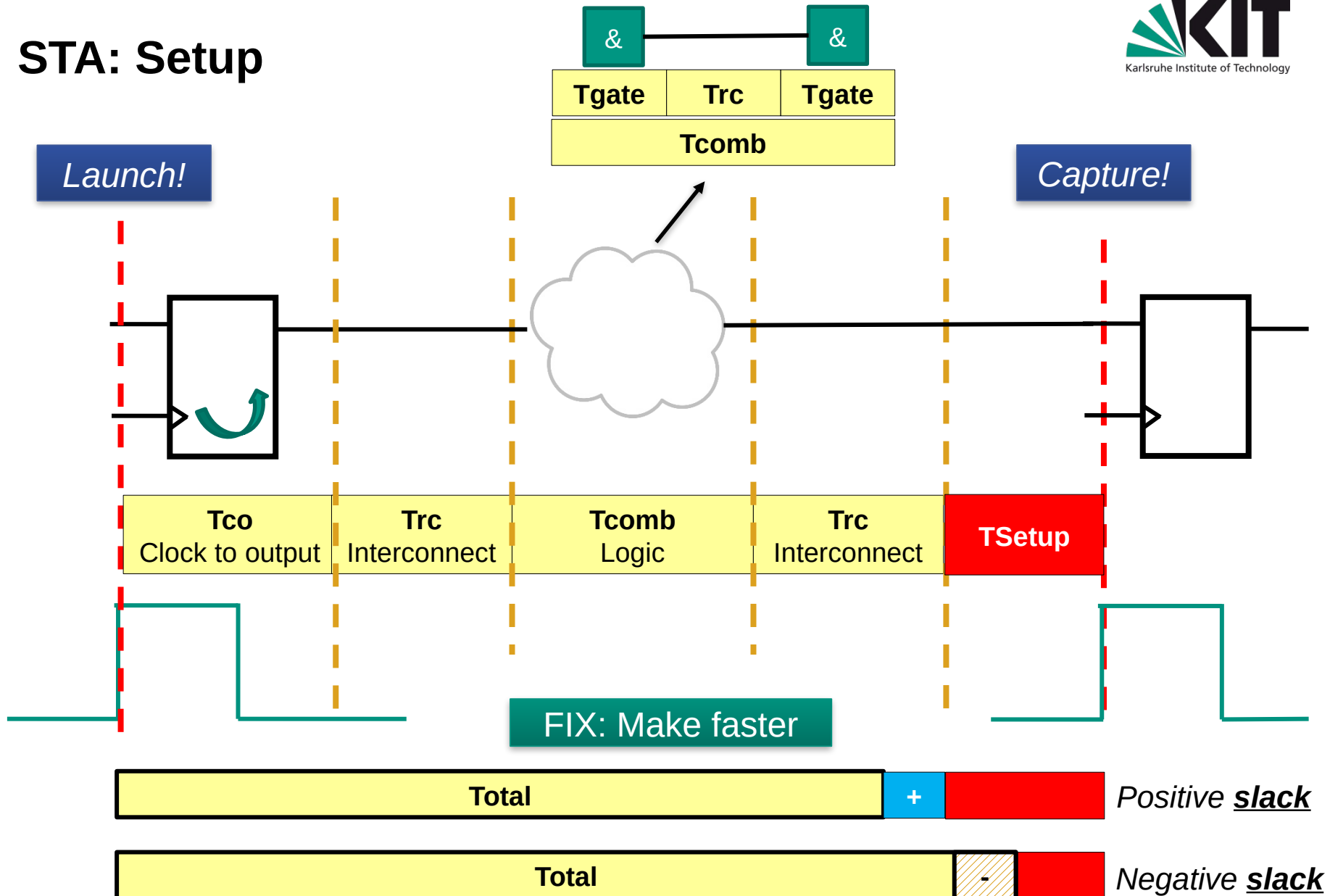
Clock Gating

STA: Overview

- Gate Level + Timing Constraints enable timing the design
- Synthesis tries to fix timing during incremental optimisation
- Important: Synthesis happens during early implementation stage. Its accuracy is not great.
- Two kinds of checks are performed:
 - Setup: Make sure logic is not too slow (slow corner)
 - Hold: Make sure logic is not too fast (fast corner)
- Timing Setup for synthesis: Worst case to ensure it will always work

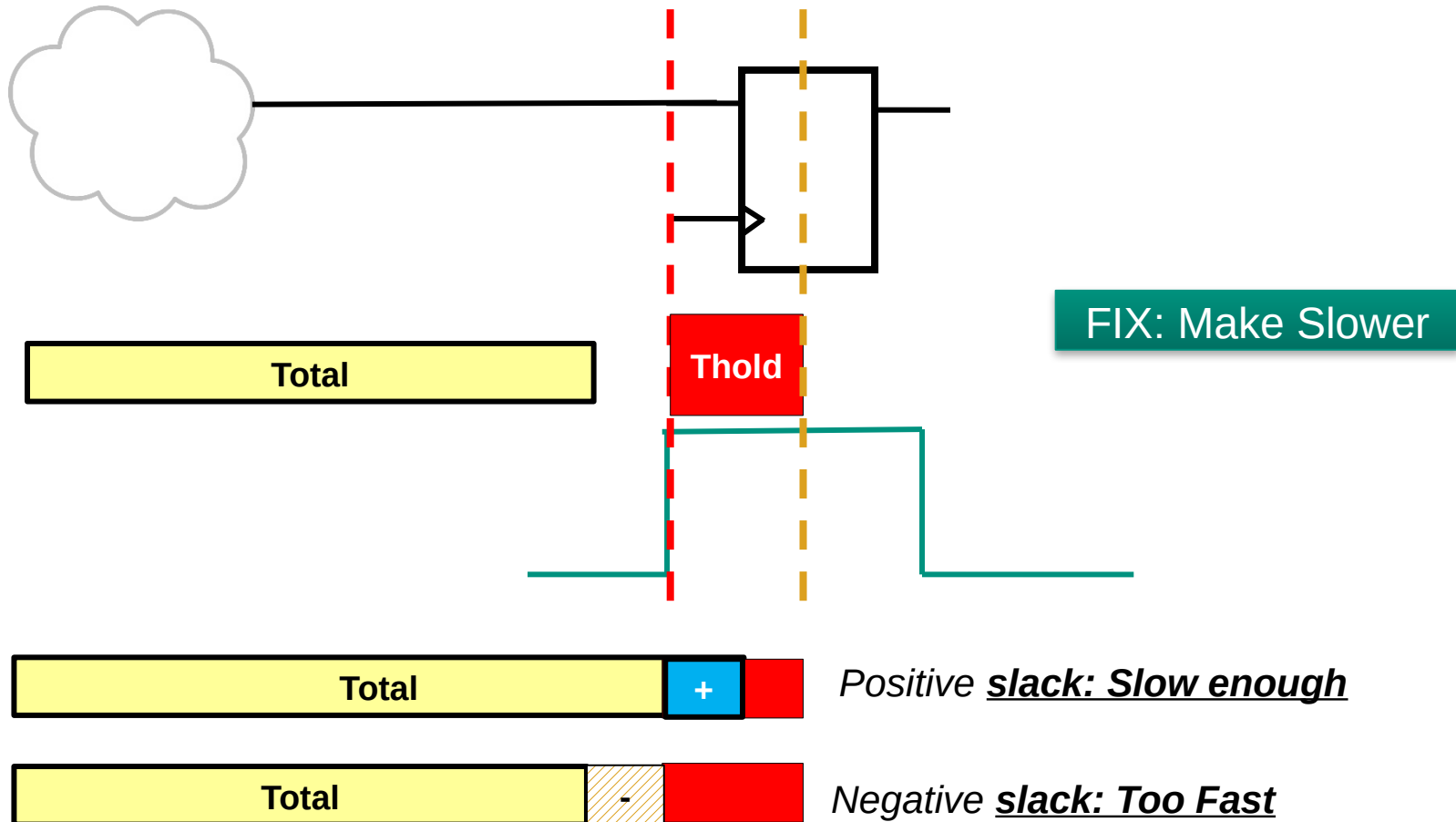


STA: Setup



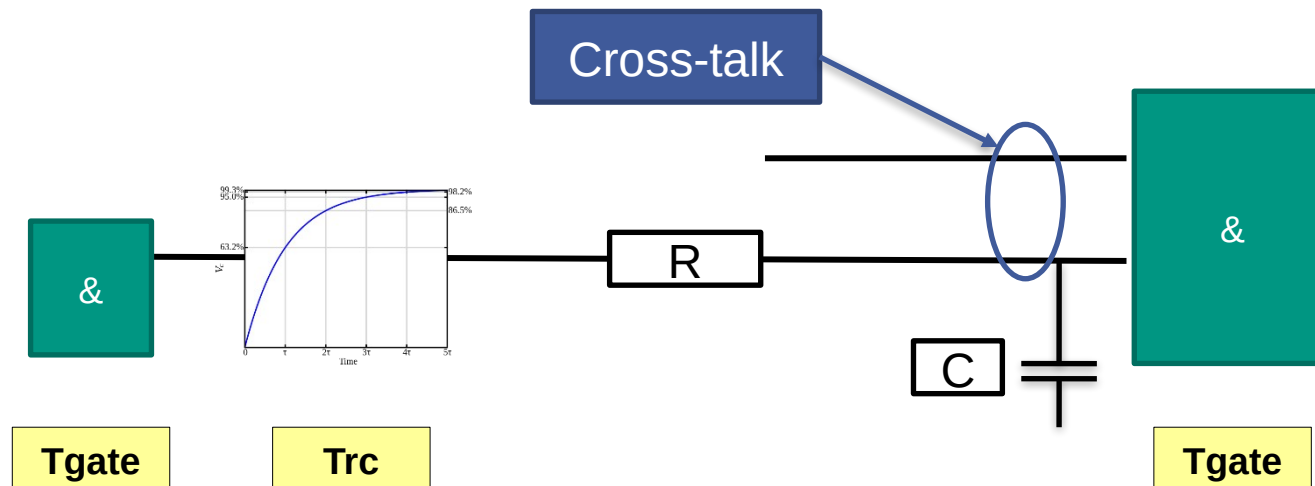
STA: Hold

- Data must remain stable for **Thold** after clock period: Cannot be too fast



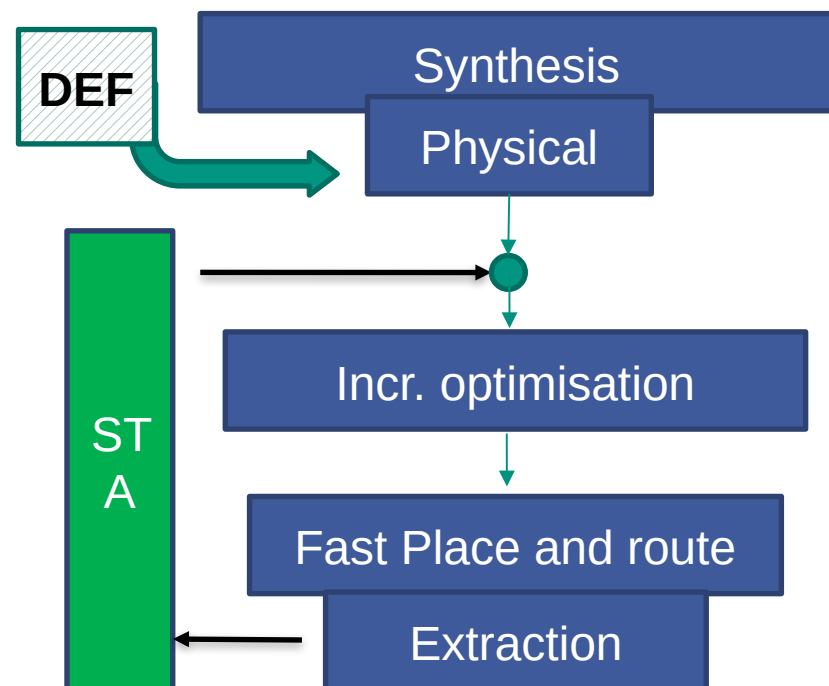
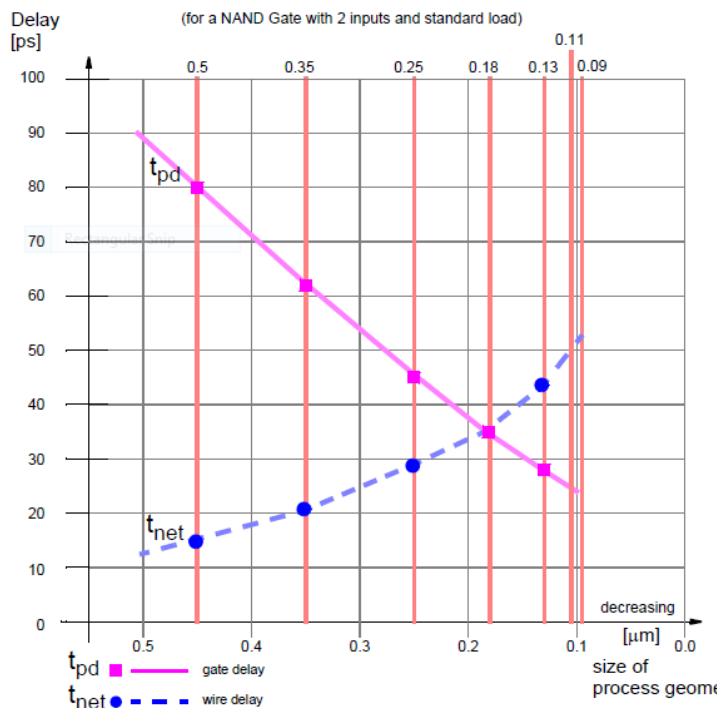
STA: Timing Parameters

- Mostly two types of delays:
 - Gate delays: Input to Output, Clock to output
 - Interconnect Delay (RC extraction)
- RC extraction is called parasitics extraction:
 - R for wire resistance
 - C for interconnect capacitance
 - Cross-Talking can be extracted (can make RC faster)



STA: Physical Synthesis for wire delays

- RC Extraction was done in the past using statistical models
 - Estimation of wire length was done based on area and fanout
- Starting at $\sim 0.18\mu$, wire delays become too dominant, so wire delay extraction is required during synthesis $\rightarrow$ physical synthesis



STA: Setup Timing Report example

- List of Gates, interconnect and associated timing delay
- Sum at the end and check against allowed clock period

Tco	Trc	Tcomb			Trc	TSetup
Clock to output	Interconnect	Tgate	Trc	Tgate	Interconnect	

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)
(clock clk)	launch					0 R
counter_value_reg[0]/CK				0		0 R
counter_value_reg[0]/Q	DFRM8RA	3	19.1	126	+376	376 R
g3762/B				+0		376 R
g3762/Z	AN2M12RA	2	16.3	85	+182	558 R
g3761/B				+0		558 R
g3761/Z	ND2M12RA	1	21.3	115	+110	668 F
g10/A				+0		668 R
g10/Z	INVM32R	8	42.8	82	+95	763 R
g37/A				+0		763 R
g37/Z	AN2M12RA	2	16.4	84	+157	920 R
g36/A1				+0		920 R
g36/Z	A0I21M12RA	1	6.6	73	+83	1003 F
g89/NB				+0		1003 R
g89/Z	OAI21B01M12RA	1	5.9	75	+156	1159 F
counter_value_reg[12]/D	DFRM4RA			+0		1159 R
counter_value_reg[12]/CK	setup			0	+62	1221 R
(clock clk)	capture					1300 R
	uncertainty				-150	1150 R

register

Type:
Empty for Trc,
Gate name
otherwise

register

Launch!

Tco

Trc

Tgate

TSetup

Capture!

Cost Group : 'reg2reg' (path_group 'grp_4')
Timing slack : -71ps (TIMING VIOLATION)
Start-point : counter_value_reg[0]/CK
End-point : counter_value_reg[12]/D

$$\text{Slack} = (\text{capture} - \text{uncertainty}) - \text{arrival}$$

$$-71\text{ps} = (1300 - 150) - 1221$$

STA: Timing report types

- Previous Slide: Register to Register
- Other general Timing paths: Register to Output, Input to register, Input to Output

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)
(clock clk)	Launch					0 R
counter_value_reg[12]/CK			0			0 R
counter_value_reg[12]/Q	DFRM4RA	2	15.5	169	+100	405 R
g3405/A					+0	405 R
g3405/Z	BUFM4BRA	3	163.0	156	+214	617 B
value[12]	out port				+0	617 R
(constraints.sdc_line_36_4_1)	ext delay				+780	1397 R
(clock clk)	capture					1300 R
	uncertainty				-150	1150 R

Cost Group : 'reg2out' (path_group 'grp_2')

Timing slack : -247ps (TIMING VIOLATION)

Start-point : counter_value_reg[12]/CK

End-point : value[12]

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)
(clock clk)	Launch					0 R
(constraints.sdc_line_33_1_1)	ext delay				+585	585 R
clear	in port	13	80.5	0	+0	585 R
g3724/A					+0	585 R
g3724/Z	INVNM32R	15	74.1	98	+60	645 R
g3025/B					+0	645 R
g3025/Z	AN2M4R	1	11.6	144	+214	859 R
g39/B					+0	859 R
g39/Z	ND2M12RA	1	8.1	69	+101	960 F
g38/B					+0	960 R
g38/Z	QA121M8B	1	5.9	127	+82	1043 R
counter_value_reg[5]/D	DFRM8RA				+0	1043 R
counter_value_reg[5]/CK	setup			0		1043 R
(clock clk)	capture					1300 R
	uncertainty				-150	1150 R

Cost Group : 'in2reg' (path_group 'grp_1')

Timing slack : 7ps

Start-point : clear

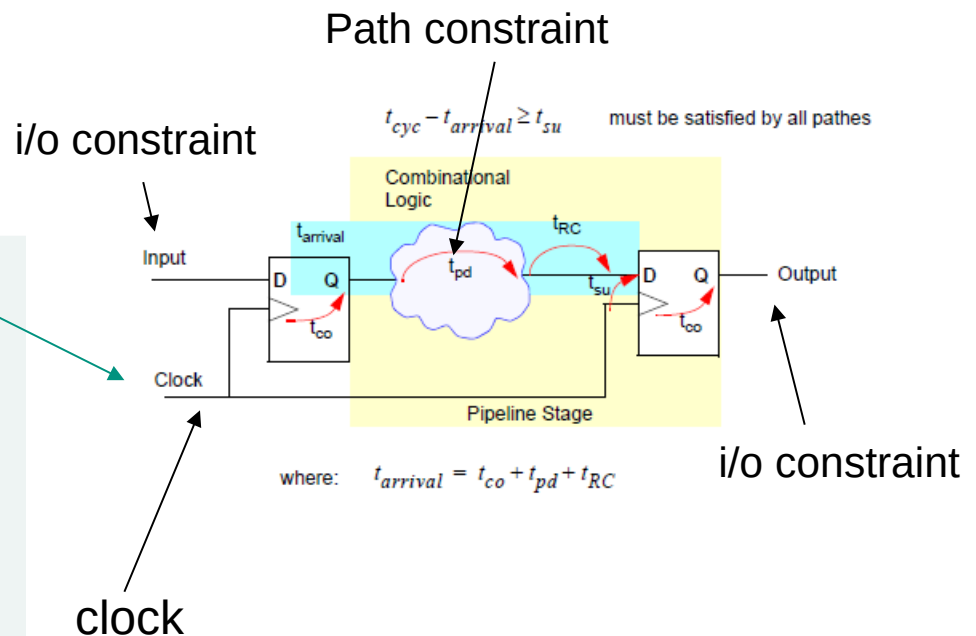
End-point : counter_value_reg[5]/D

Timing Constraints

- Specification for the clocks and paths to be timed
- Format in RTL Compiler: Standard Design Constraints (SDC)
- The timing analysis uses the constraints to perform analysis
- Check Tool documentation

SDC

```
create_clock -name clk -period 1.3 [get_port clk] #750Mhz
set_clock_uncertainty 150 -setup clk # Spare time on setup
set_clock_uncertainty 50 -hold clk # Spare time on hold
set_input_delay -clock clk 50 ...
set_output_delay -clock clk 100 ..
set_load -pin_load 0.15 [all_outputs] #15pf load on outputs
```



TC: Clock Definition

- Specification:
 - Name of the clock
 - Period/Waveform
 - Start point: Typically the top level input (called port)
- Uncertainty: Simply Overconstraining to cover unknown wire delay and physical placement information
 - Will be reduced the closer the design gets to its final stage, extraction takes over then.
- The named created clock is called a **Clock Domain**
- Later in Place and route: A Clock tree will be created

```

1 module top (
2
3 input wire clk
4 );
5
6 ....
7
8 endmodule

```

clk port

Name is arbitrary

In ns per default

```

create_clock -name clk -period 1.3 [get_port clk]
set_clock_uncertainty 150 -setup clk
set_clock_uncertainty 50 -hold clk

```

Just remove time from period

Multi-Mode Multi Corner

- There may be different constraints, like clock period depending on the mode of operation:
 - Test Mode: Clock fed through a different “slow” input, main clock not active
 - Run Mode: Only main clock active
- One cannot create two clock domains driving the same logic
- Create two modes, which are analysed separately
- Corners: Temperature and Voltage combinations
- Synthesis: Only check modes
- Place and route: perform extraction and analysis on all corners and modes

$$\underline{nT \times nV \times nM = \text{many corners}}$$



Limit to the relevants: will my ASIC be in a -40° environment?

I/O constraints

■ Input and Output Delay for the outside world

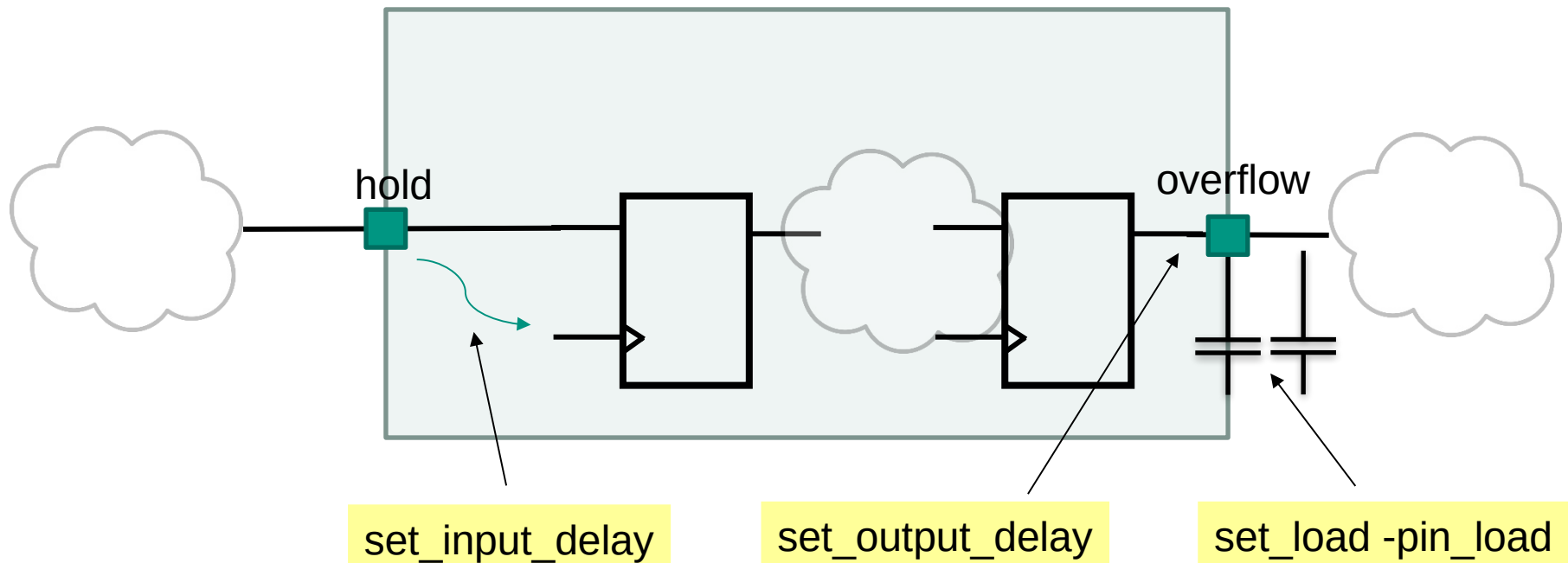
■ Input: When does the input arrive after the clock (setup time from unknown)

■ Output:

■ How Much time is to be spared for the outside

■ How much capacitance do we have to drive

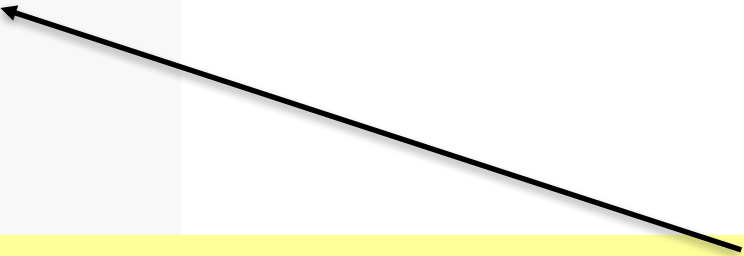
■ Example from counter:



False Paths: Non timed wires

- Some Special Paths can be ignored
 - Configuration Lines if asserted during total inactivity (like reset)
 - Asynchronous Reset: Reset usually asserted for a long time, so don't need to be timed
 - Wires not relevant to the mode of operation

```
1 always @(posedge clk or posedge res_n)
2 begin
3     if (rst) begin
4         // reset
5     end
6     else begin
7
8     end
9 end
```

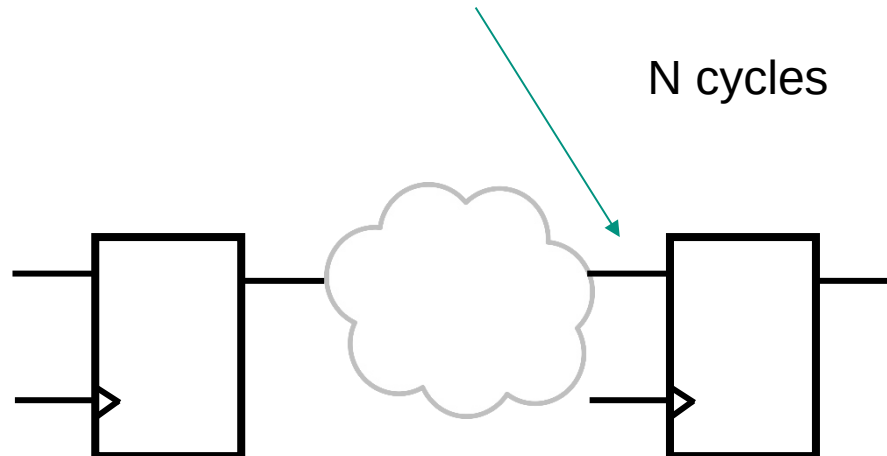


set_false_path -through [get_port res_n]

Multi Cycle: timing relaxing

- Some paths can take more time, like configuration Lines barely changing, or changing when design is inactive anyway
- They have to be timed anyways to respect setup and hold time

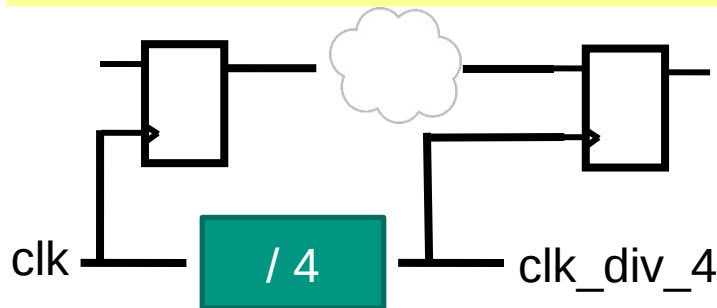
set_multicycle_path time -setup -through net



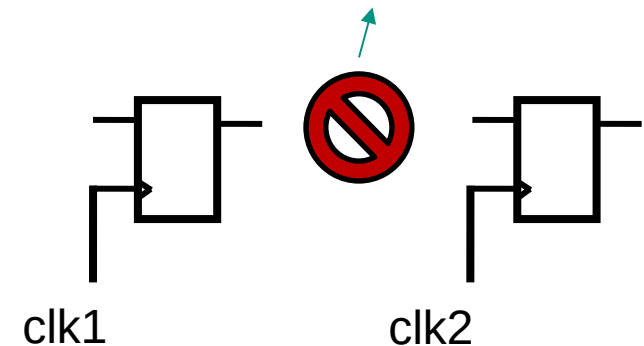
Multiple Clock domains

- Multiple Clocks can be defined in a design, there are then multiple clock domains
- Some clocks are derived from each other, like divided clock
 - -> Generated Clocks
 - Timing is respected between the clock domains
- Separated clocks are uncorrelated, thus false paths are set between flip flops of different clock domains

```
create_generated_clock -divide_by 4 \
                        -name clk_div_4 -from
clk
```



Design should prevent metastability



Results

- Reports: Final timing and other kind of useful information
- Gate Level Netlist
 - Use to re-simulate the design
 - Go to place and route
- LEC tools check netlists for equivalence after each transformation:
 - After each synthesis step and incremental optimisation
- Place and route setup
 - Some scripts to start with encounter
 - In real designs, not used in the end but still works for us
- Reports

End Slide