CPEG 422/622
EMBEDDED SYSTEMS DESIGN

Chengmo Yang

chengmo@udel.edu

Evans 201C

LECTURE 6
VIVADO REPORTS

OUTLINE

- Last lecture:
  - Testbench
  - Sequential circuits in VHDL

- This lecture:
  - Project 1 test cases
  - Vivado reports

PROJECT1 TEST CASES

- Before test, make sure your design is complete and no syntax error.
- Then perform three test cases:

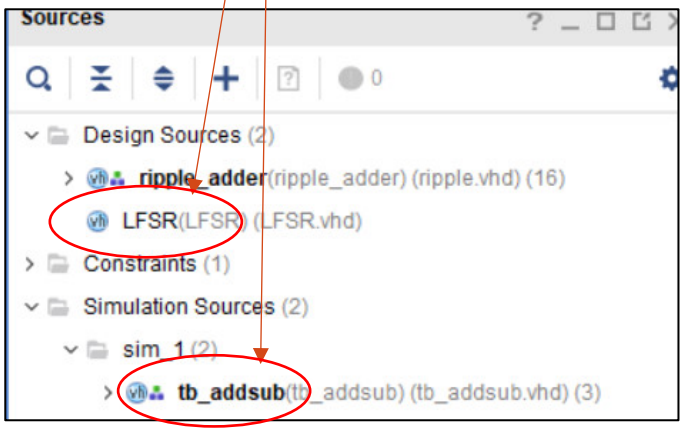
Phase	Goal	Modules	Test bench	Run simulation	Run synthesis & implementation
1	Behavioral correctness	Your design+ LFSR	Yes	Yes	No
2	Report design hardware, power and schematics	Your design	No	No	Yes
3	Report timing	Your design+ FF_wrapper	No	No	Yes

- This lecture will explain these cases one by one.

PHASE 1

- **Goal**: Test your design code correctness.
- **Module**: Design + LFSR

  LFSR is used for generating random numbers to test your design.
- **Four steps:**
  1. Add LFSR file into your project design sources.
  2. Create a testbench file in simulation sources.



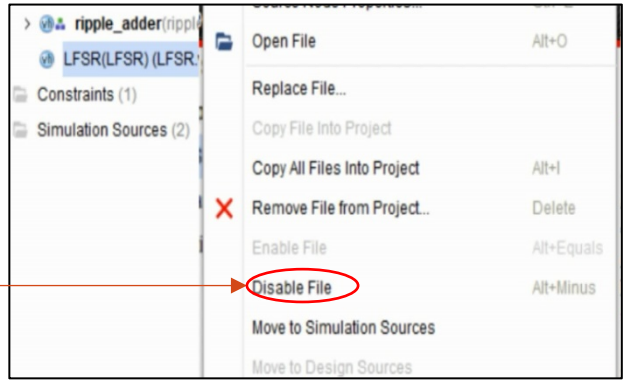
PHASE 1

- **Goal**: Test your design code correctness.
- **Module**: Design + LFSR

  LFSR is used for generating random numbers to test your design.
- **Four steps:**
  1. Add LFSR file into your project design sources.
  2. Create a testbench file in simulation sources.
  3. Make sure your testbench has (a-e):
     a) Add LFSR as a component in testbench architecture.
     b) Instantiate three LFSRs to generate A and B for your design.
     c) Add all connecting signal declarations for LFSR in testbench.
     d) Add setup process for the LFSR in testbench.
     e) Add a clock generation process for the LFSR in testbench.
  4. Run simulation and check waveform to see if it is correct.

PHASE 2

- **Goal**: Report your design hardware, power and schematics.
- **Module**: Design



- **Three Steps:**
  1. Disable LFSR file.
  2. Run *Synthesis* and *Implementation*.
  3. Check *Project summary*.

VIVADO REPORT

- Vivido has two big steps to run:

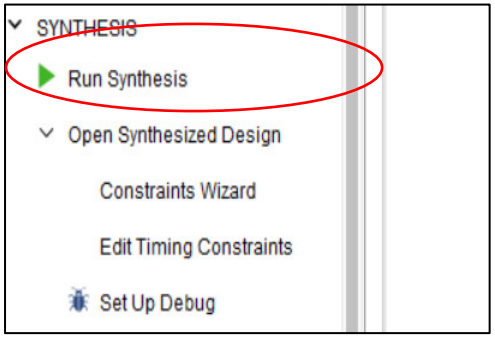
1. **Synthesis –** translate VHDL description to low level circuit design
2. **Implementation** – implement the circuit design on the FPGA

- We ask for reports of:

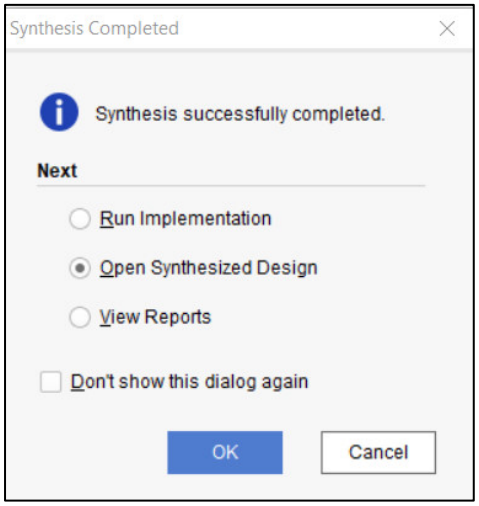
1. Hardware utilization
2. Power
3. Schematic
4. Timing

→ After running synthesis and implementation.

RUN SYNTHESIS



Once your design code is done,
click "run synthesis" on the control panel.



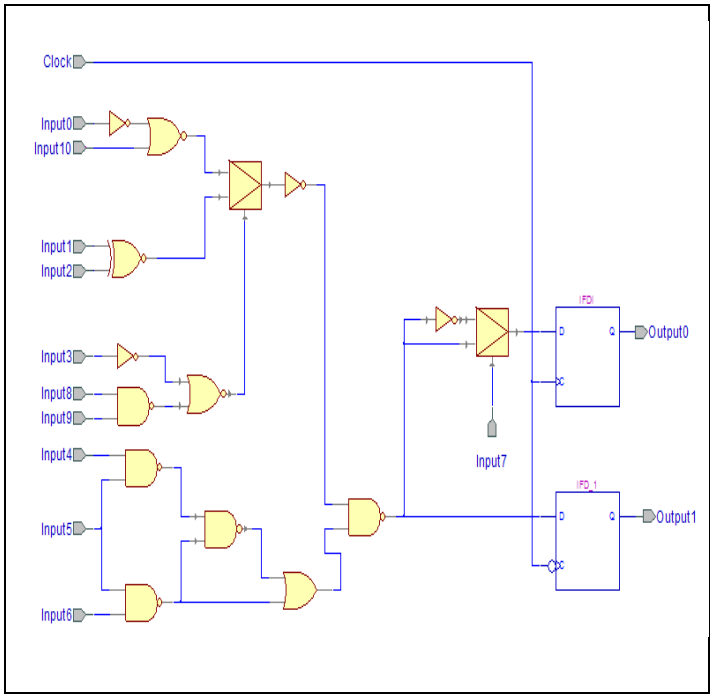
A window is popped up when synthesis is finished successfully.

LOGIC SYNTHESIS

VHDL description

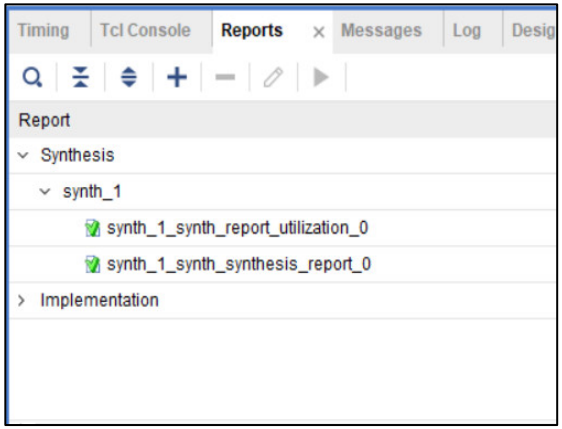
```
architecture MLU_DATAFLOW of MLU is

signal A1:STD_LOGIC;
signal B1:STD_LOGIC;
signal Y1:STD_LOGIC;
signal MUX_0, MUX_1, MUX_2, MUX_3: STD_LOGIC;

begin
                A1<=A when (NEG_A='0') else
                                not A;
                B1<=B when (NEG_B='0') else
                                not B;
                Y<=Y1 when (NEG_Y='0') else
                                not Y1;

                MUX_0<=A1 and B1;
                MUX_1<=A1 or B1;
                MUX_2<=A1 xor B1;
                MUX_3<=A1 xnor B1;

                with (L1 & L0) select
                                Y1<=MUX_0 when "00",
                                                MUX_1 when "01",
                                                MUX_2 when "10",
                                                MUX_3 when others;

end MLU_DATAFLOW;
```

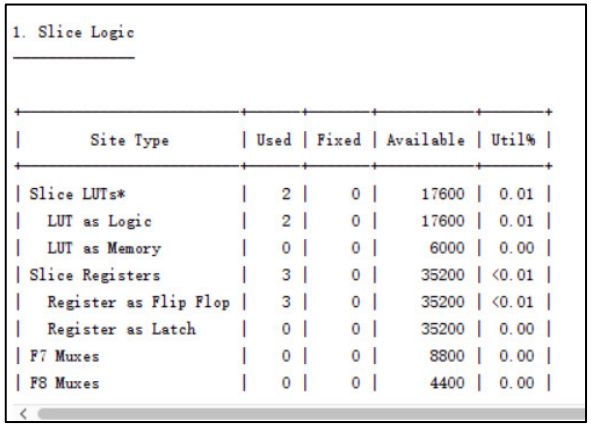
Circuit netlist

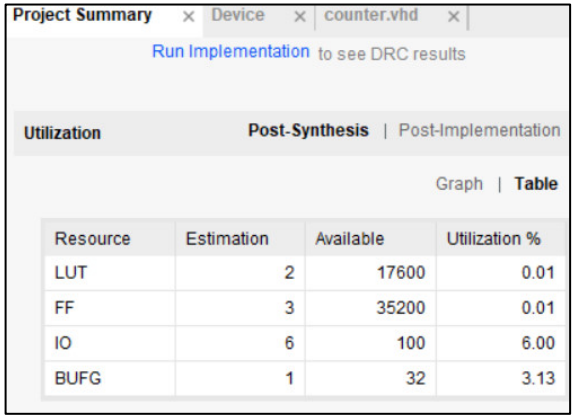


SYNTHESIS REPORTS

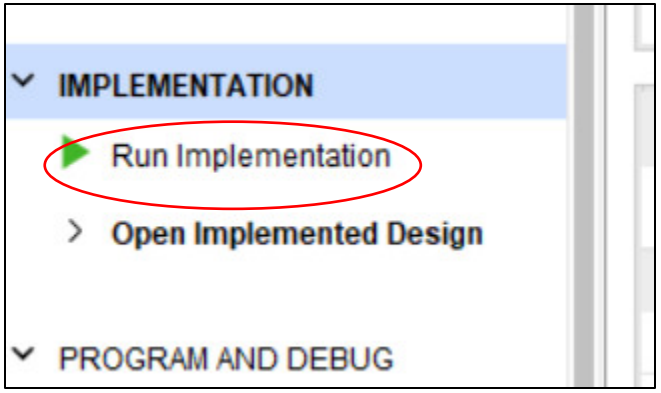
- When synthesis is done successfully, you can click *synthesis reports* to see detailed report or open *project summary* for summary. Note data at this stage are only estimations.



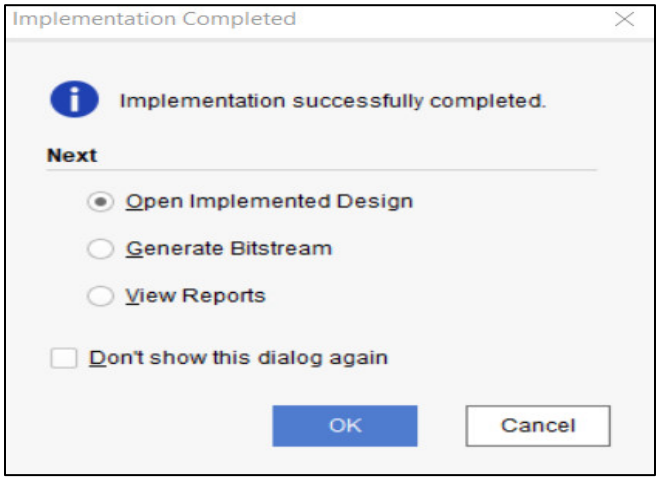




RUN IMPLEMENTATION



After synthesis is done, click "run implementation" on the control panel.

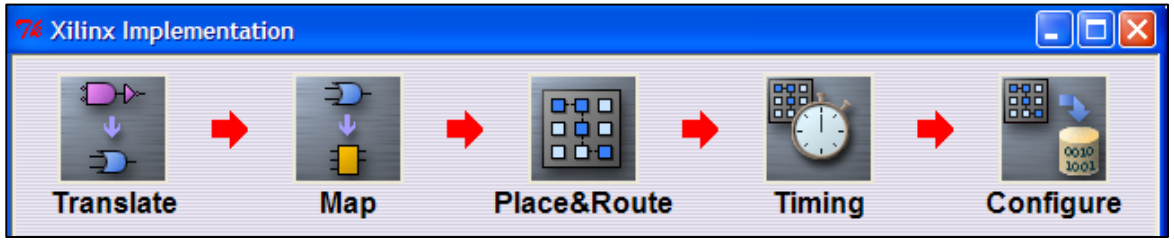


A window is popped up when implementation is finished successfully.

IMPLEMENTATION

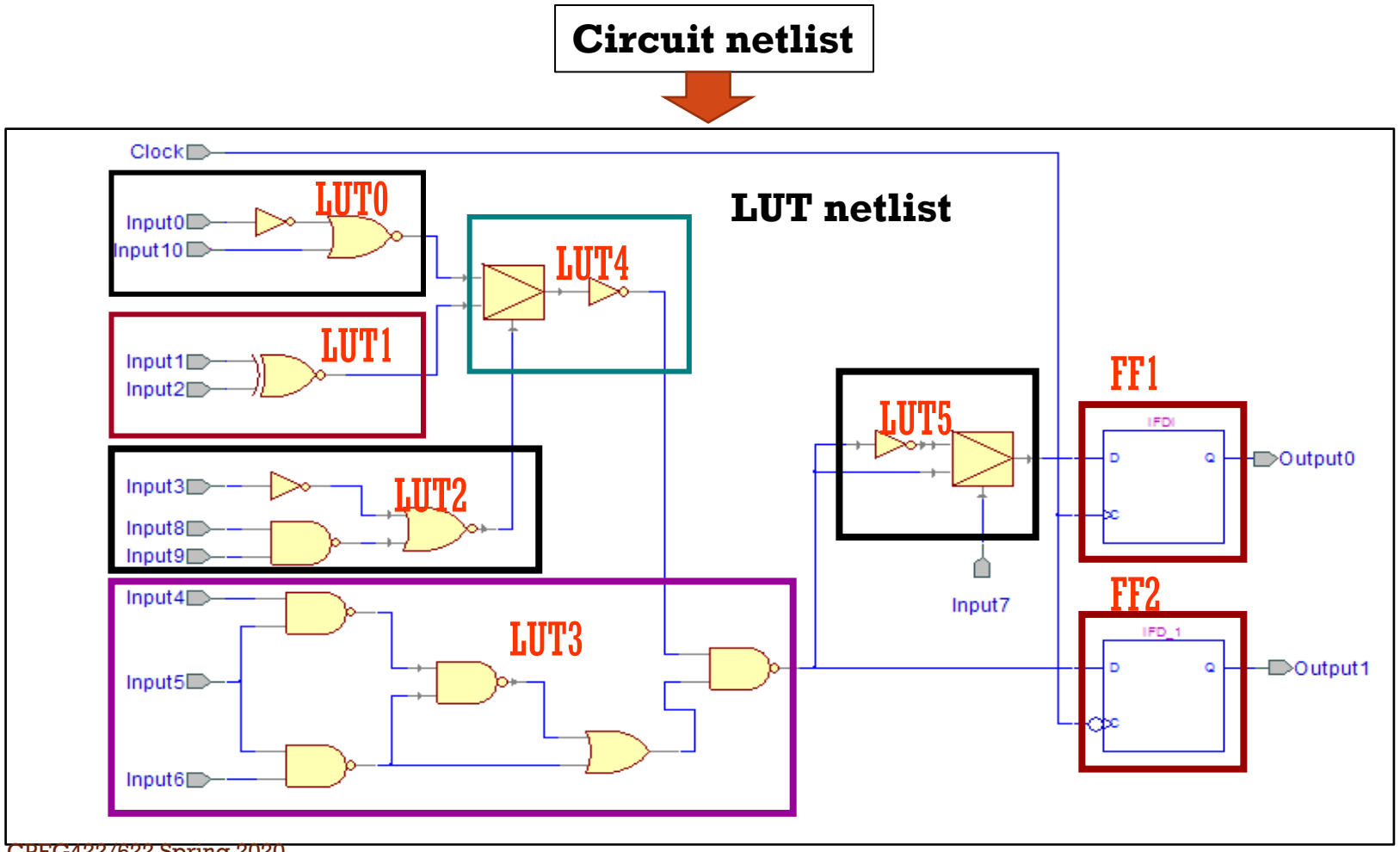
- After logic synthesis, implementation process is performed to generate specific implementation details on FPGA, include:

1. Mapping
2. Placement & Routing
3. Timing analysis
4. Configuration (bit stream generation)



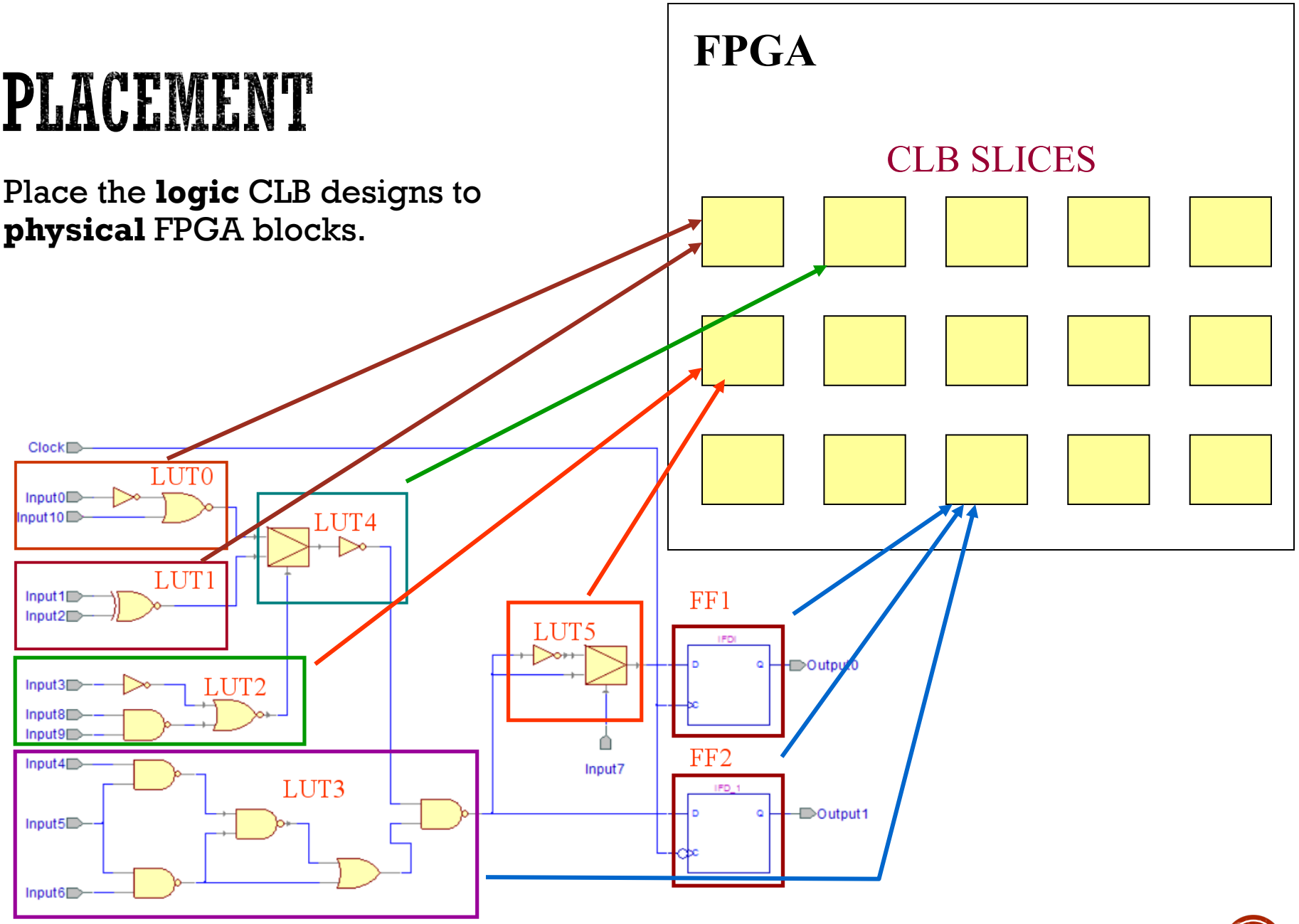
MAPPING

- Cluster circuit netlist to LUT based netlist (further CLB level netlist) for FPGA.



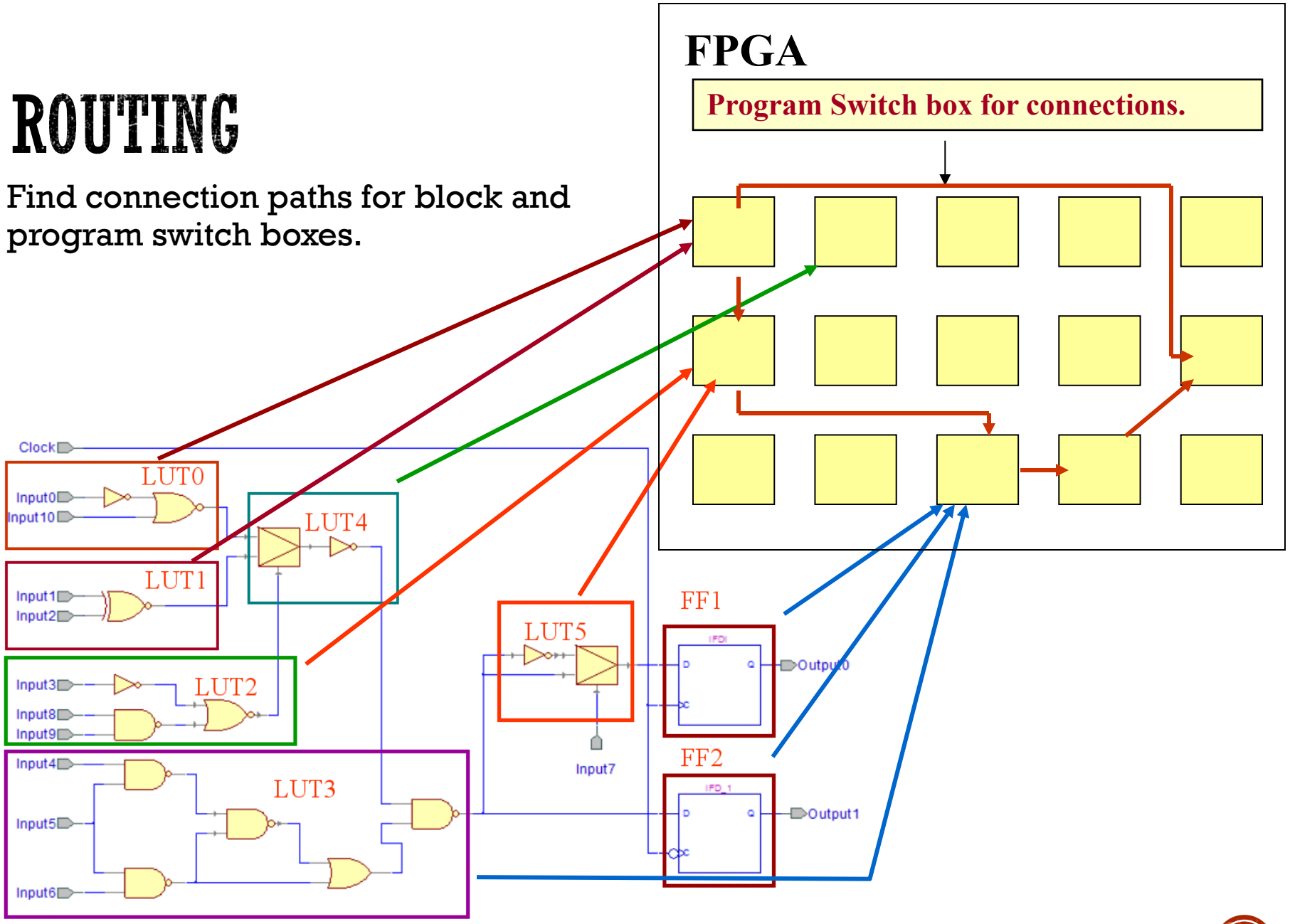
PLACEMENT

- Place the **logic** CLB designs to **physical** FPGA blocks.



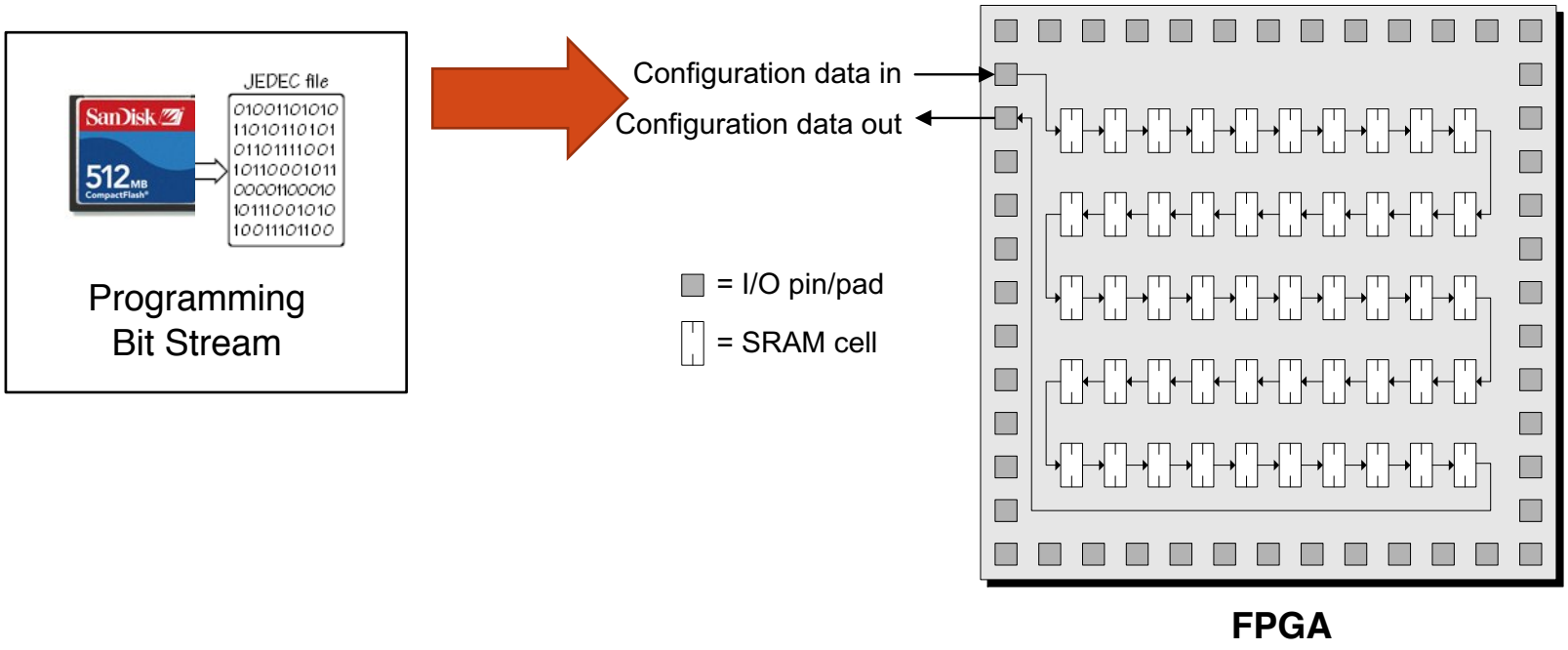
ROUTING

- Find connection paths for block and program switch boxes.

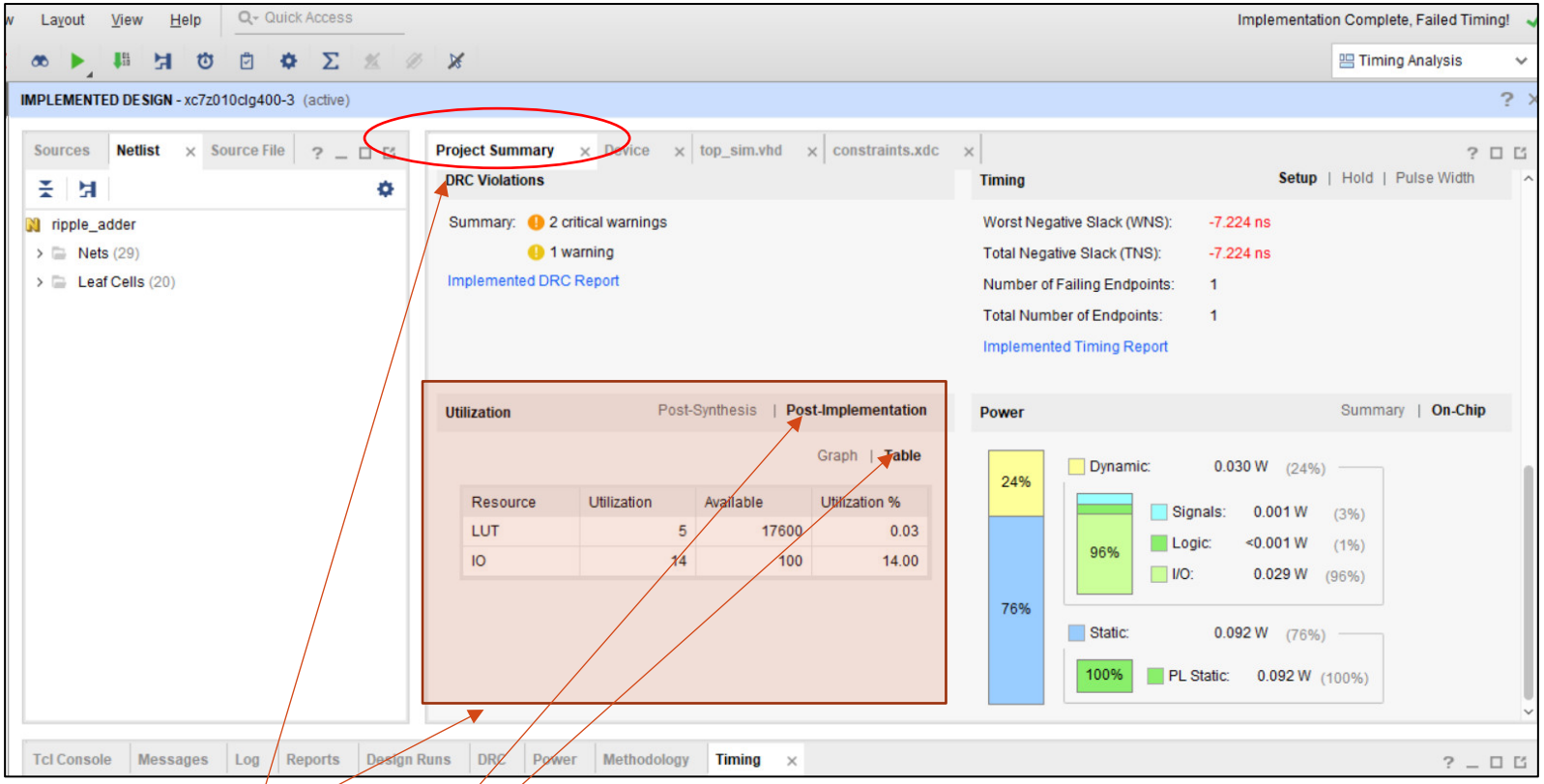


CONFIGURATION

Configuration :Generate bit stream that can be filled in FPGA to make it functional.



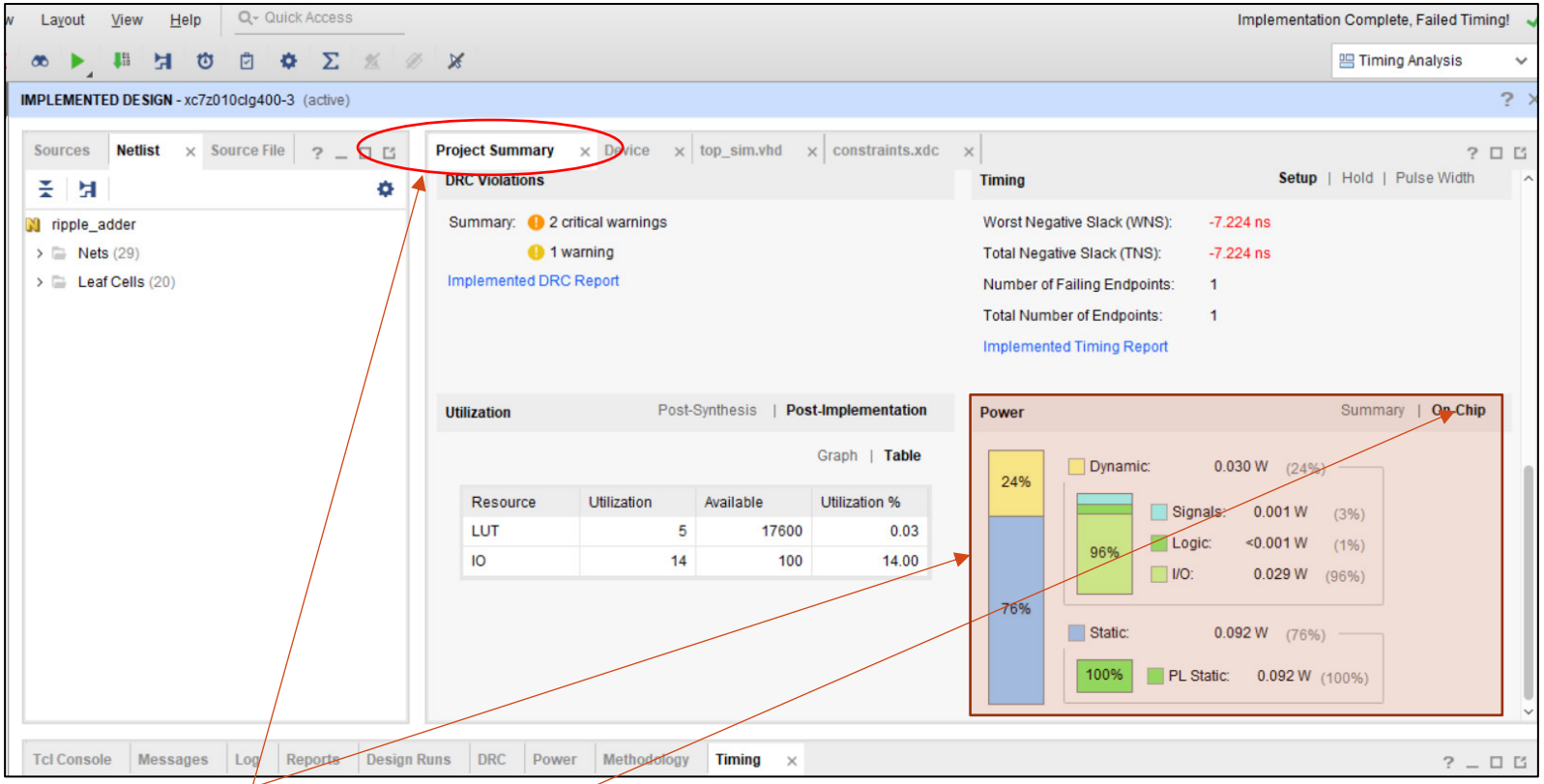
HARDWARE REPORT



Open "project summary" and scroll to "utilization" section for hardware cost.

Choose "post-implementation" and "table" option to view.

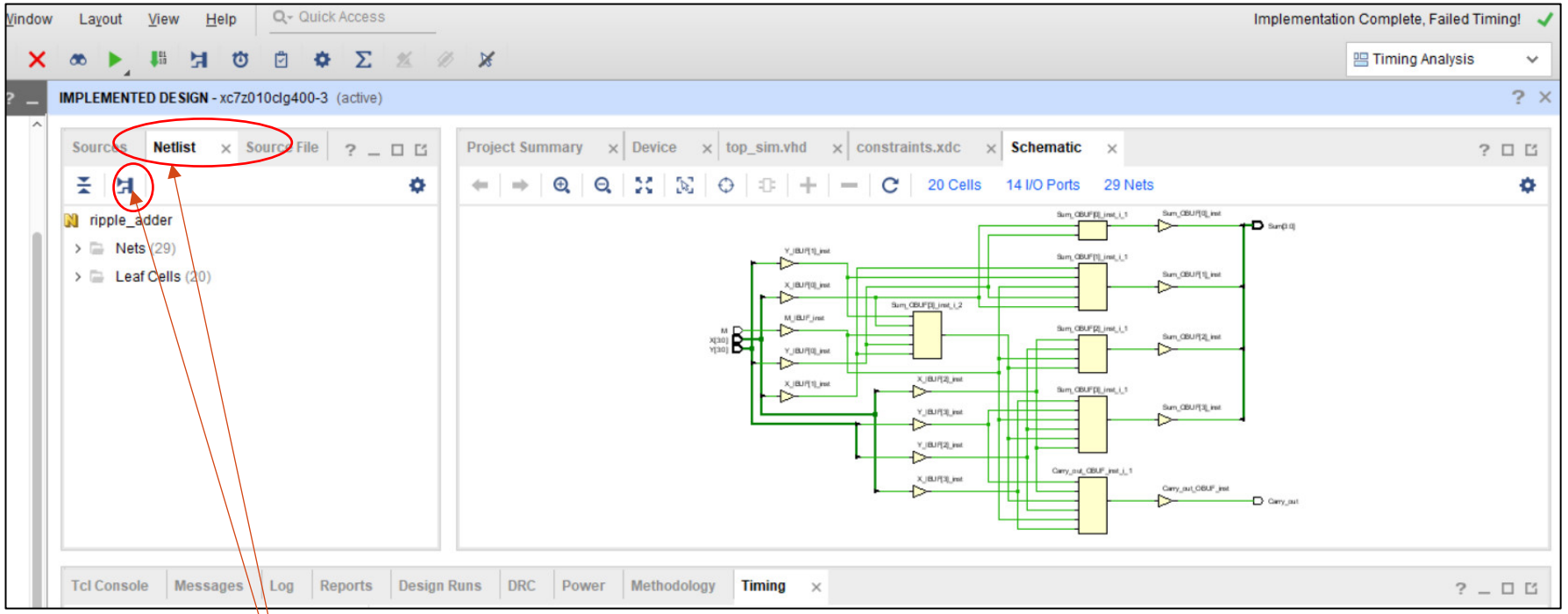
POWER REPORT



Open "project summary" and scroll to "power" section for power cost.

Choose "on-chip" option to view.

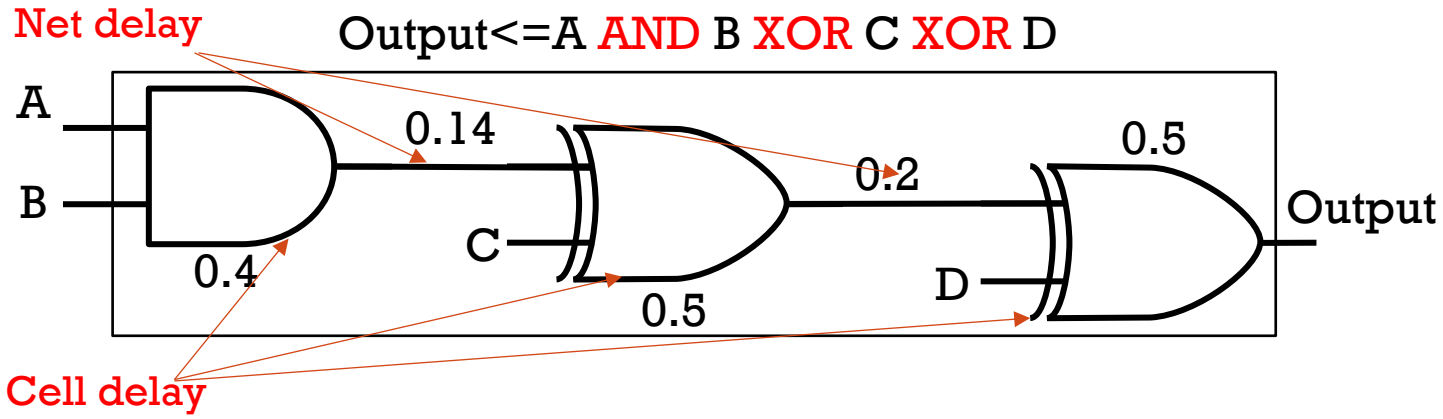
SCHEMATIC REPORT



Click "Netlist" and then click "schematic" button, and you will see design schematic.

PROPAGATION DELAY

- In behavior simulation, the computation has no delay, which is unrealistic.
- In real circuit, there is always **propagation delay**.



Propagation delay can be divided into:
Cell delay: for signal to propagate through a cell.
Net delay: for signal to propagate to the next level.

Total delay from A to Output= 0.4+0.14+0.5+0.2+0.5=1.74 ns

TIMING IN SEQUENTIAL LOGIC

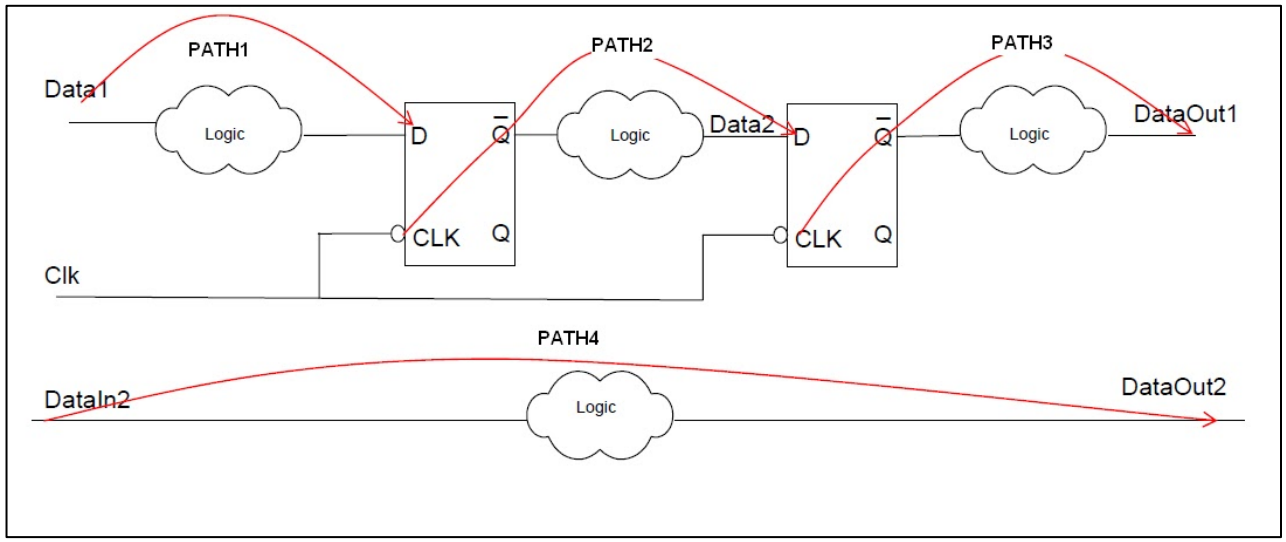
Four types of paths:

Input to register (FF) Path1

register (FF) to register (FF) Path2

register (FF) to output Path3

Input to output Path4



TIMING ANALYSIS

- Timing analysis: Analyze the timing performance of the design.

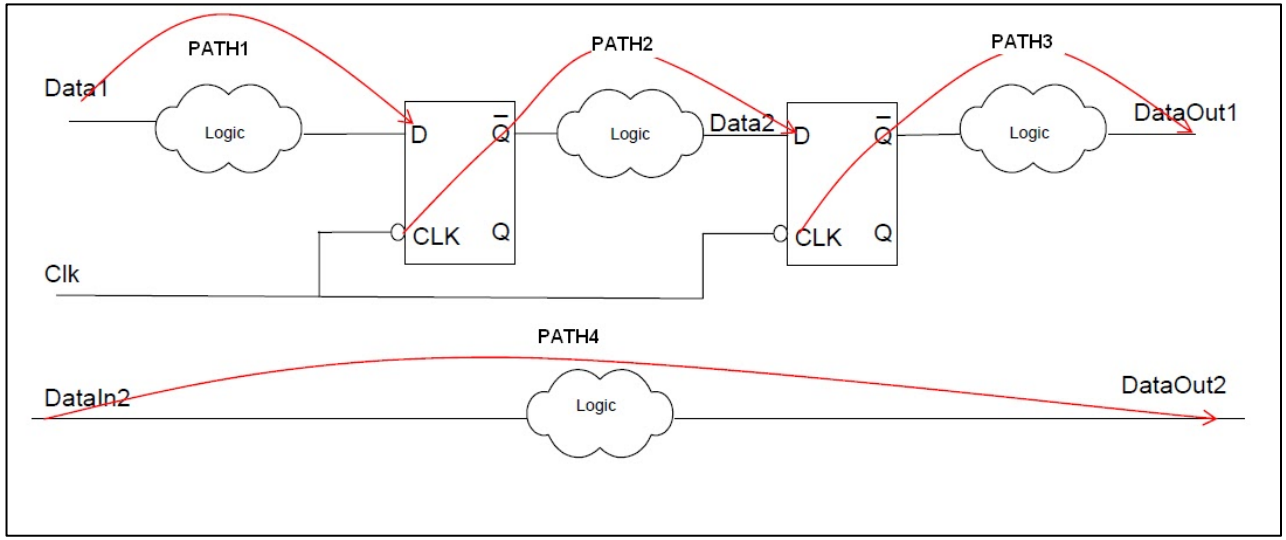
  Path 1 delay: 4ns

  Path 2 delay: 4.5 ns

  Path 3 delay: 5 ns

  Path 4 delay: 4.2 ns

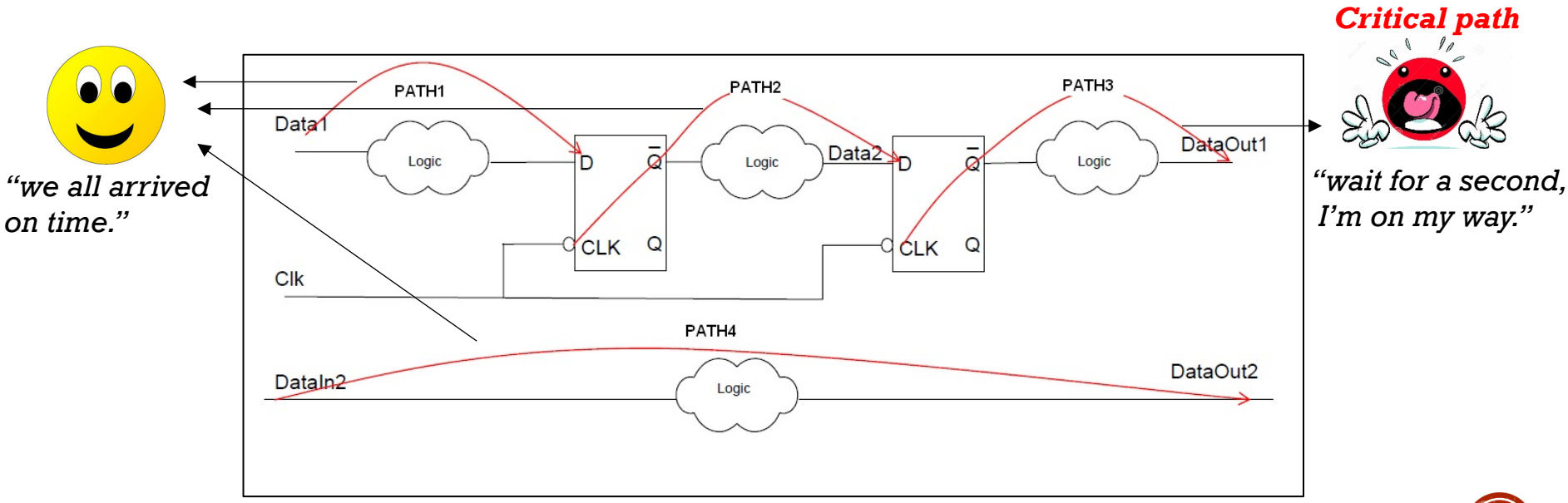
  …



CRITICAL PATH DELAY

- **Critical path delay:** the **maximum** path delay among all timing paths in the design.
- Why important?

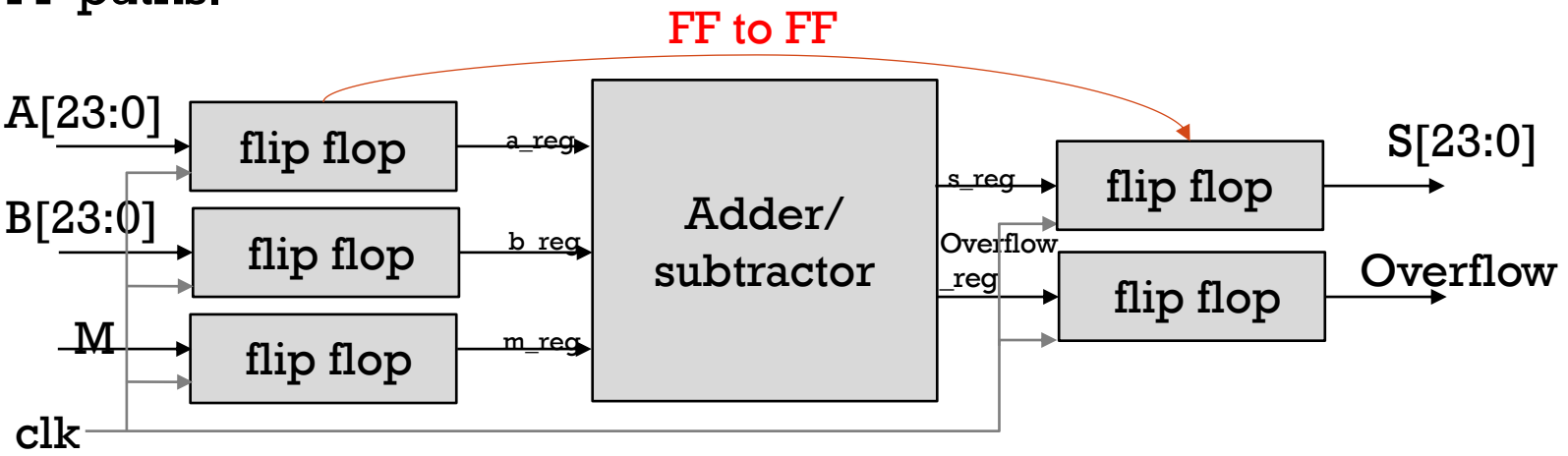
1. Determines the design performance. (How fast)
2. If **critical path delay >clock period**, it causes **timing violation.**



TIMING FOR ADDER/SUBTRACTOR

- Vivado only reports timing for type 2 path (**FF to FF**), and only if there is a **timing violation (clock period<critical path delay)**.

- To evaluate your adder/subtractor timing, we need to create FF to FF paths.

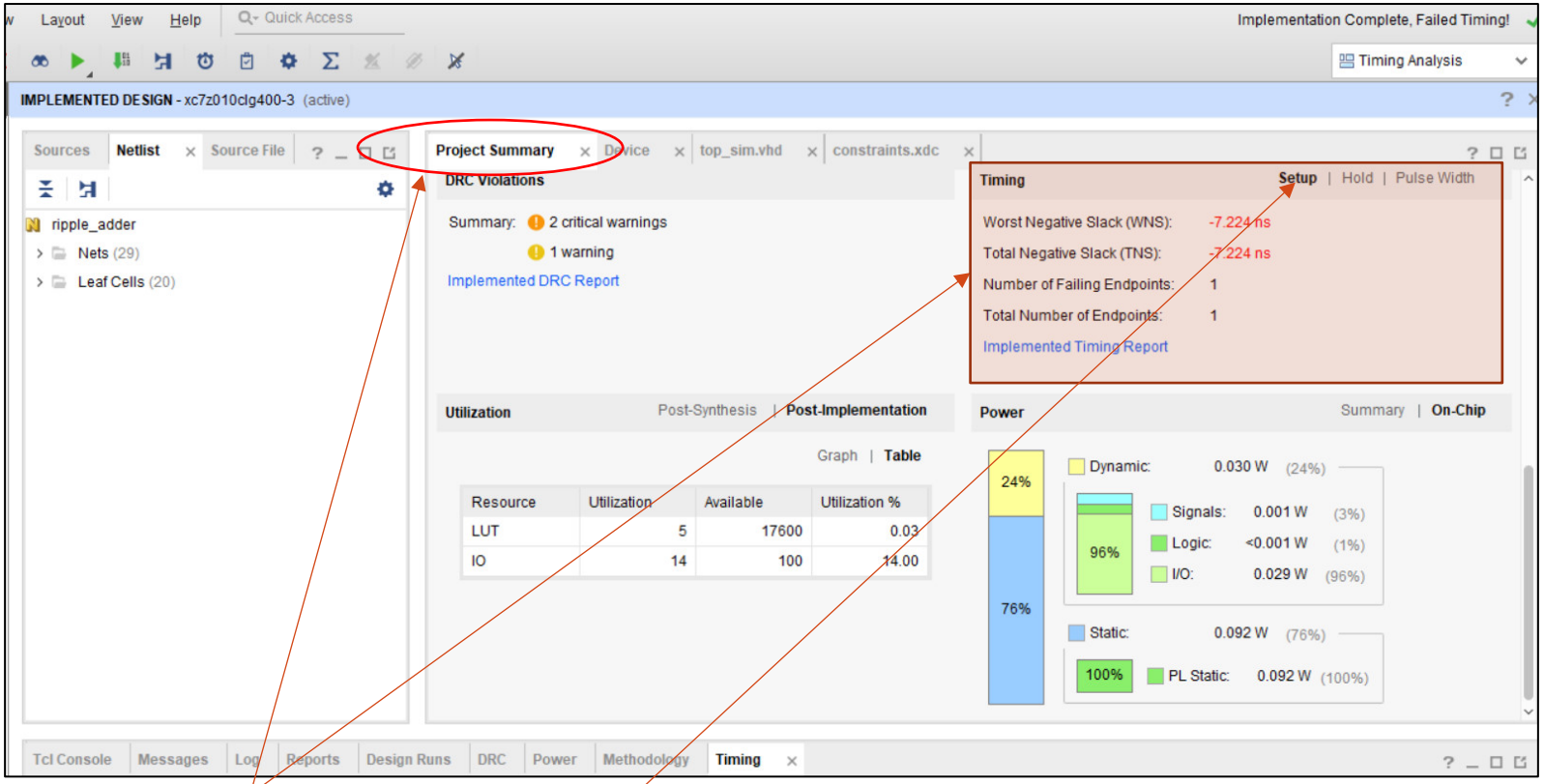


PHASE 3

- **Goal**: Report your design timing.
- **Module**: Design+ FF_wrapper

- **Five Steps:**
  1. ADD FF_wrapper file to your project design sources.
  2. Make your design a component in design wrapper.
  3. Instantiate your design in design wrapper.
  4. Run *Synthesis* and *Implementation*.
  5. Check *Project summary* for timing.

TIMING REPORT



Open "project summary" and scroll to "timing" section.

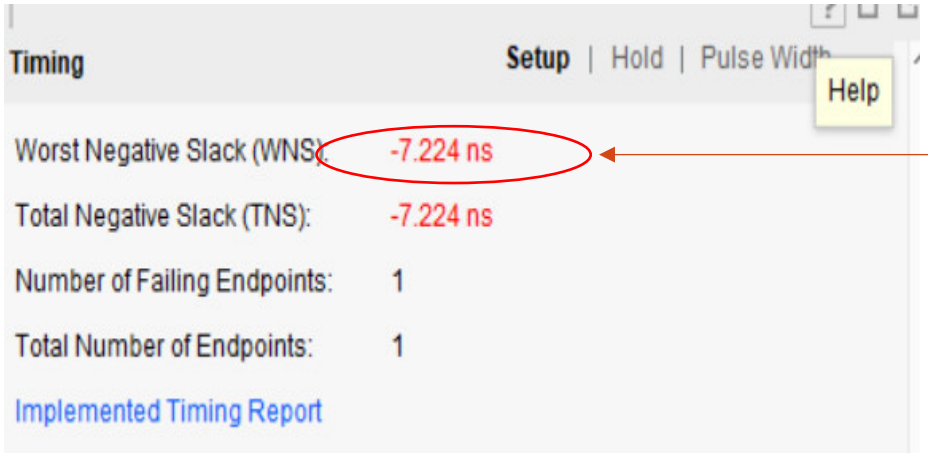
Choose "setup" option to view, and we will see "negative slack".

What is negative slack?

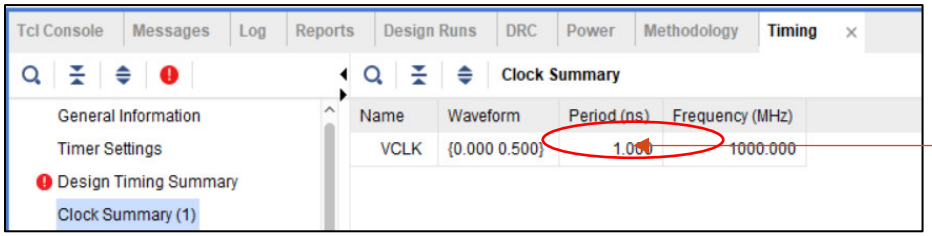
NEGATIVE SLACK

- It means that the timing constraints that you have given for your design are not met.
- **Example:**
  - the clock period is 2ns
  - one path propagation delay is 2.1 ns
  - Then this path has a negative slack of -0.1ns.

TIMING REPORT



Check the WNS of setup.



Check the timing constraints (period) in tcl console "timing->clock summary".

Here period=1ns, WNS=-7.224 ns:
Critical path delay=period –WNS=8.224ns

SET UP FPGA CLOCK

- The FPGA clock setup is in the first two lines of the constraint file. For example:

```
set_property -dict { PACKAGE_PIN K17   IOSTANDARD LVCMOS33 }
[get_ports { sysclk }];      #IO_L12P_T1_MRCC_35 Sch=sysclk

create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4}
[get_ports { sysclk }];
```

- This means the clock signal name is ”sysclk", period is 8ns (125MHZ), and 50% duty cycle.
- If one wants to change it to 4ns, the second line should become:

```
create_clock -add -name sys_clk_pin -period 4.00 -waveform {0 2}
[get_ports { sysclk }];
```

NEXT LECTURE

- Sequential circuits
- Registers
- Finite State Machine (FSM)