Advanced Vlsi Project

Bitcoin

Stage 1 - Synthesis

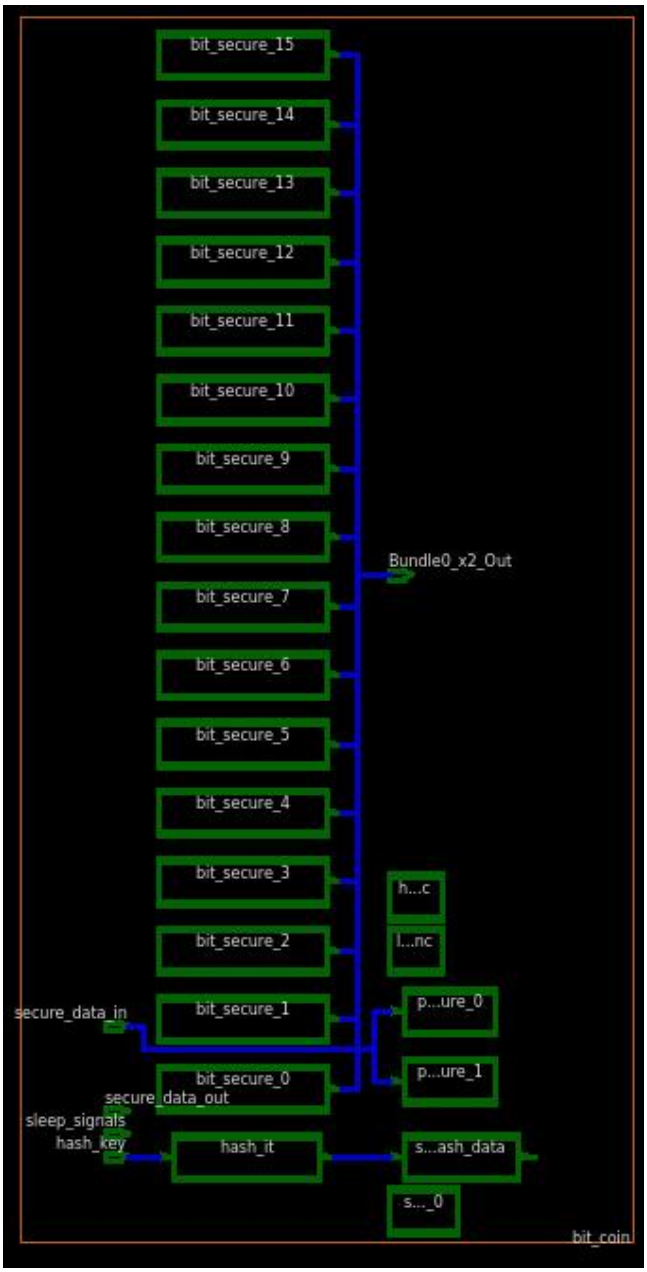
שמות המגישים	אור שאול	עמרי אורן	סטאניסלאב קוגן	נועה פרקש

Workarea path for part 3.3: /project/advvlsi/users/stanislavk2/ws/bitcoin

Workarea path for part 3.1~3.2: /project/advvlsi/users/orshaul/ws/bitcoin

#[P3.1_Q1]

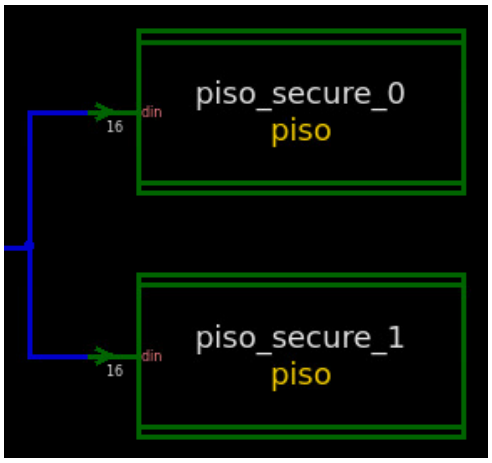
2.4.1 Bit_coin:



Hash:

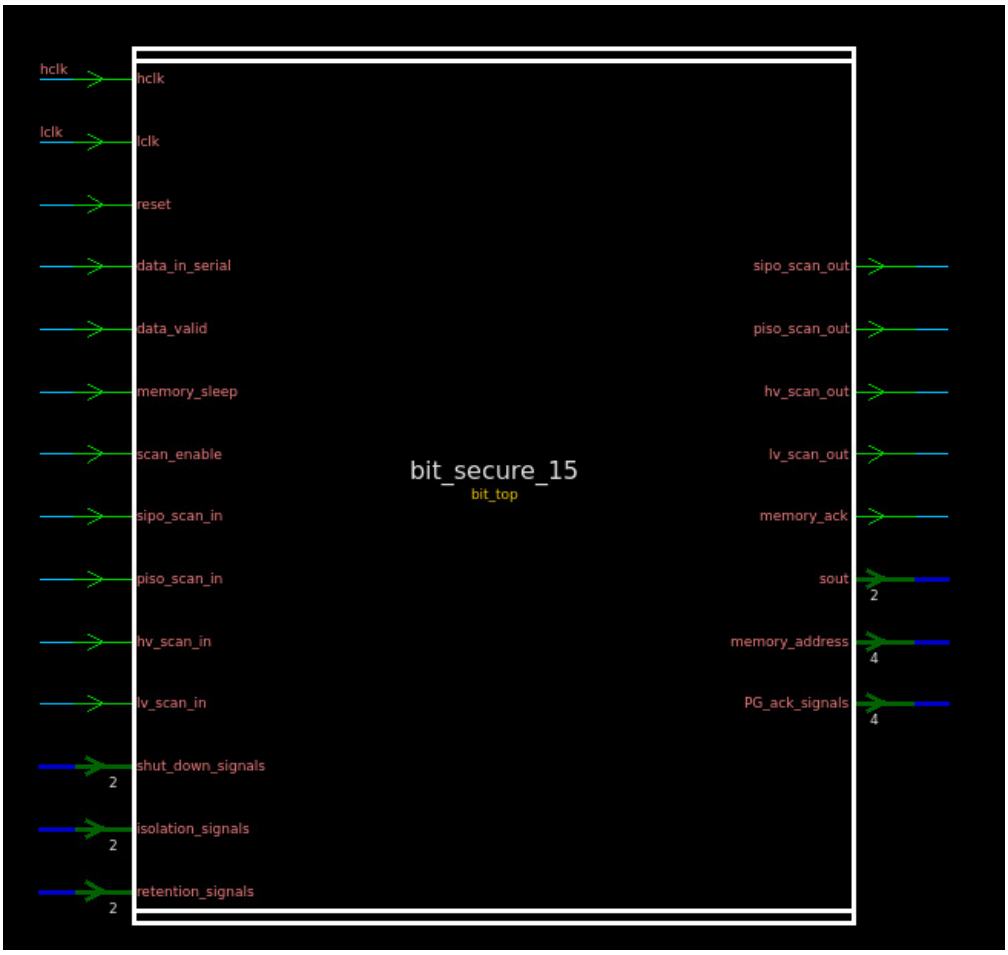

clk
reset
data_in
2
hash_key
32
hash_it
secure_data
secure_data_out
16

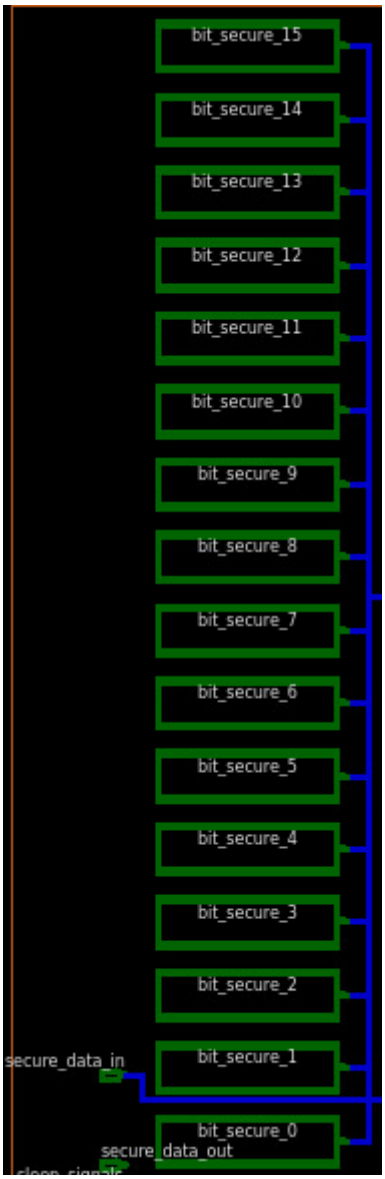
Piso:


16
din
piso_secure_0
piso
16
din
piso_secure_1
piso


clk
reset
load
shut_down_signals
isolation_signals
retention_signals
din
16
piso_secure_0
piso
dout

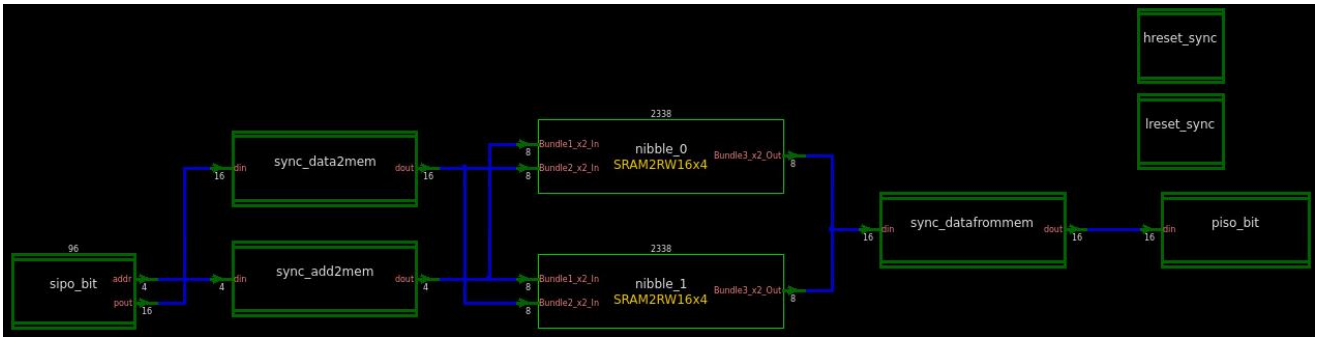
16 bit_top (bit secure):

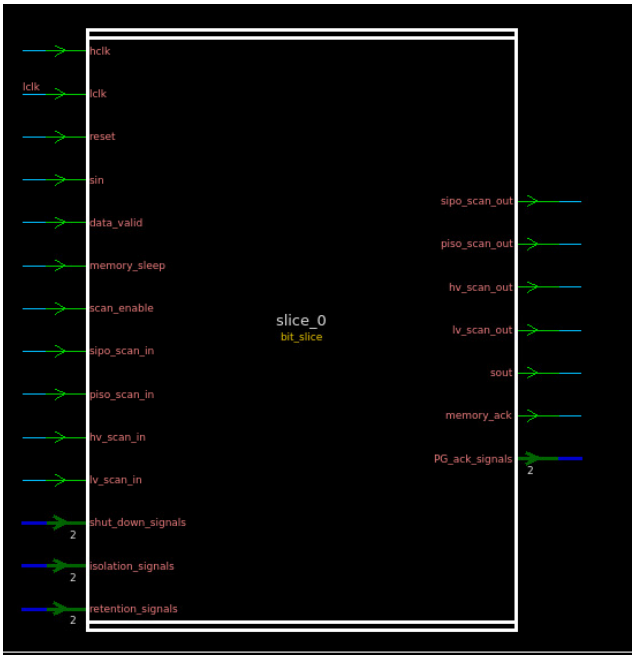




2.4.2

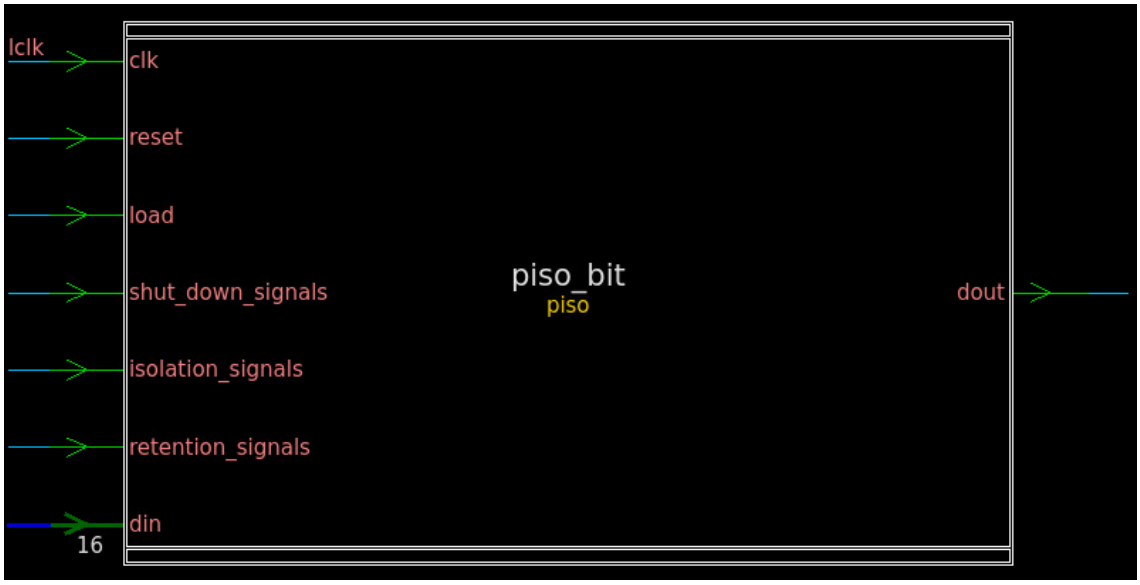
Bit_Slice



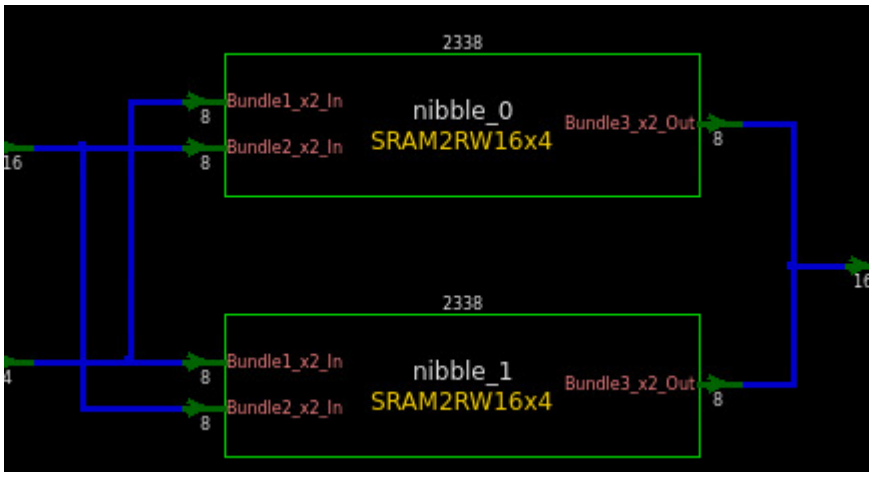


Sipo_bit:

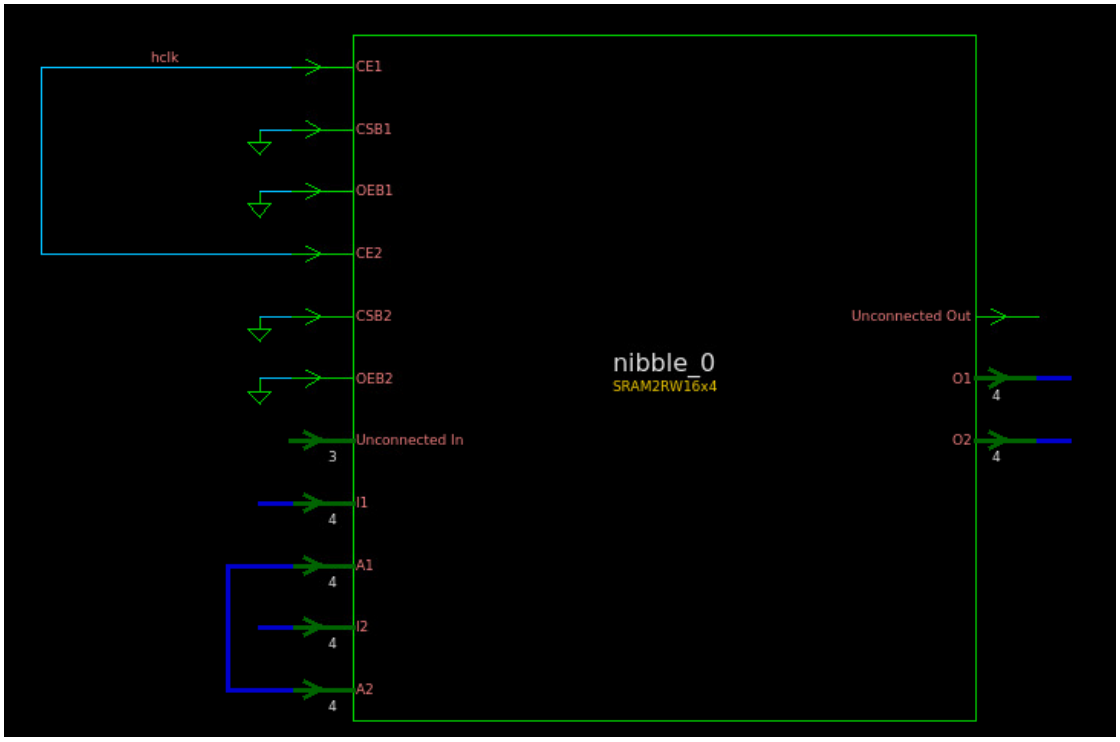
Piso_bit:



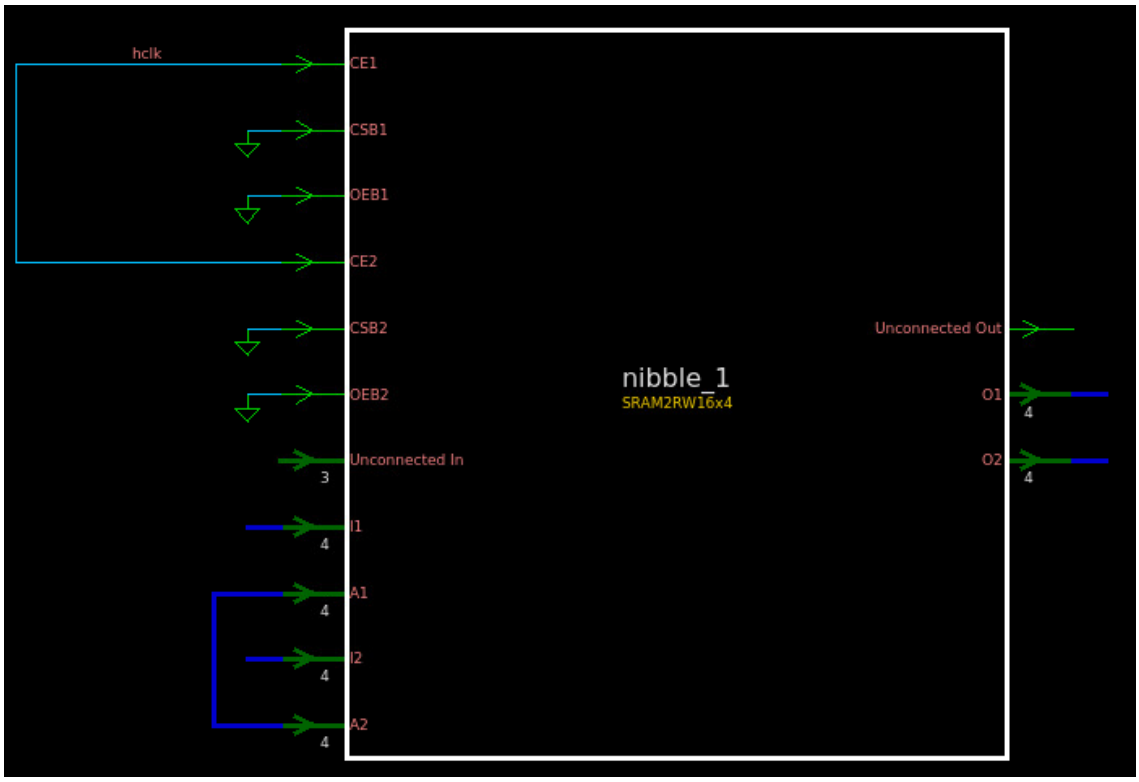
2 nibbles:



nibble_0:



nibble_1:



[#P3.2_Q1]

Constraints:

1.1

5 different constraint examples:

1. **Maximum Junction Temperature constraint** - The highest temperature allowed at the junctions of transistors to prevent thermal runaway and ensure reliable operation.

Supplier: Thermal Engineering Team

Domain: Thermal

2. **Clock Periods constraint** - The maximum allowable time for one complete cycle of a clock signal.

Supplier: Architecture or design team

Domain: Timing

3. **Power Constraint** - Maximum power consumption to ensure efficiency and avoid overheating.

Supplier: Power or Implementation team

Domain: Power

4. **Area Constraint** - Maximum Die size: The maximum physical size of the silicon dies to ensure the design fits within the intended package and manufacturing limits.

Supplier: Layout team

Domain: Physical design

5. **Routing Constraint** - These constraints include limits on wire length, width, and spacing, as well as the proper assignment of metal layers and via placement to optimize signal integrity and to reduce congestion.

Supplier: Routing or Implementation team.

Domain: Routing

6. **Testability Constraint** - These constraints ensure that the integrated circuit can be effectively tested for faults and defects. These constraints include requirements for design-for-test (DFT) features, such as scan chains and built-in self-test (BIST) circuits, which facilitate easier and more comprehensive testing, ensuring the reliability and functionality of the final product.

Supplier: Testing or Verification team.

Domain: Testing

1.2

3 techniques that we can implement during the synthesis stage to meet low power constraints:

1. **Voltage Scaling**: Voltage scaling involves operating the IC at lower supply voltages to decrease power consumption. Since dynamic power consumption is proportional to the square of the voltage, reducing the supply voltage can significantly lower power usage.
2. **Clock Gating**: Clock gating is a power-saving technique that involves selectively disabling clock signals to inactive or idle parts of a circuit. By turning off the clock to these unused components, unnecessary switching activity and power consumption are minimized. This technique is implemented using clock gating cells that control the clock signals based on specific conditions or enable signals, effectively reducing dynamic power consumption in digital circuits.
3. **Power Gating**: Power gating, also known as power shutdown, is a technique that completely turns off the power supply to inactive or idle blocks within an IC. By isolating and disabling power to these inactive blocks, leakage power (power consumed even when the circuit is idle) is minimized. This is achieved using special power gating cells that act as switches to cut off power to specific circuit blocks, effectively reducing both static and dynamic power consumption, and making it valuable for low-power design.

1.3

While clock gating, voltage scaling, and power gating are effective methods for reducing power consumption, they come with certain tradeoffs that need to be considered:

1. Clock Gating Tradeoffs:

- **Increased Design Complexity:** Implementing clock gating necessitates additional logic in the form of clock gating cells, which can increase design complexity and potentially impact timing closure and overall design productivity.
- **Clock Network Impact:** Using clock gating cells can affect the clock distribution network, leading to increased skew or added routing congestion.
- **Power and Area Overhead:** Although clock gating reduces dynamic power consumption, the extra circuitry for gating introduces some static power consumption and increases the overall chip area.

2. Voltage Scaling Tradeoffs:

- **Performance Impact:** Lowering the supply voltage reduces power consumption but can also degrade performance. This reduction in voltage slows down the circuit, leading to increased propagation delays and potentially reduced operating frequencies, or more stringent design constraints to meet timing requirements.
- **Sensitivity to Process Variations:** Reducing the supply voltage can make circuits more vulnerable to process variations, leading to greater performance inconsistencies among manufactured chips.
- **Design Complexity and Verification Challenges:** Voltage scaling necessitates careful design and validation since it impacts multiple aspects of the circuit, such as timing, noise margins, and voltage levels. This adds extra complexity and verification effort.

3. Power Gating Tradeoffs:

- **Increased Design Complexity:** Power gating involves adding power gating cells and control logic, which complicates the design process. Managing control signals and ensuring proper functionality during power transitions can be challenging.
- **Impact on Signal Integrity:** Power gating can introduce additional power domains and isolation mechanisms, potentially causing signal integrity issues such as increased IR drop, crosstalk, and delay variations.
- **Design Constraints and Trade-offs:** Power gating requires careful consideration of design constraints, such as wake-up latency, power-up sequence, and control signal generation. Achieving desired power savings while meeting these constraints can involve trade-offs.

[#P3.2_Q2]

2.1

A false path in a circuit refers to a pathway that does not need timing verification because it is not essential for the proper operation of the circuit. These paths are deliberately excluded from the timing analysis process. Key aspects of false paths include:

1. **Non-Critical Timing Paths:** False paths are those that do not impact the circuit's functionality or performance. They typically involve non-critical signals, such as control signals, that do not participate in key data transfer or computation processes.

2. **Exclusion from Timing Analysis:** False paths are explicitly designated as "false" to instruct synthesis and timing analysis tools to disregard them during timing verification. This reduces the complexity and computational load of the timing analysis, allowing focus on critical paths that require precise timing characterization.

3. **Timing Constraints:** False paths are not subjected to specific timing constraints, such as setup or hold time requirements. Since they do not influence the circuit's operation, there is no need to analyse or optimize their timing properties.

2.2

False paths are crucial for simplifying the design process and easing the efforts required for timing closure. By omitting non-critical paths from timing analysis, designers can concentrate on optimizing timing constraints and meeting the requirements for critical paths, which are essential for the circuit's proper operation. Accurate identification and marking of false paths in the design are imperative, as mistaking a critical path for a false one can lead to functional failures or timing violations in the final circuit. Therefore, careful identification and verification are necessary to ensure that false paths are correctly handled throughout the design flow.

[#P3.2_Q3]

3.1

The part related to the corners, modes, and scenarios (MCMM) in the "bitcoin_stage_1.tcl" file:

```
# mcmm_setup:
# Remove all MCMM related info
remove_corners   -all
remove_modes     -all
remove_scenarios -all

# Create Corners
create_corner Fast
create_corner Typical
create_corner Slow

# Set parasitics parameters
set_parasitics_parameters -early_spec Cmin -late_spec  Cmin -corners {Fast}
set_parasitics_parameters -early_spec Cnom -late_spec  Cnom -corners {Typical}
set_parasitics_parameters -early_spec Cmax -late_spec  Cmax -corners {Slow}
```

```
# Create Mode
create_mode  FUNC
current_mode FUNC

# Create Scenarios
create_scenario -mode FUNC -corner Fast    -name FUNC_Fast
create_scenario -mode FUNC -corner Typical -name FUNC_Typical
create_scenario -mode FUNC -corner Slow    -name FUNC_Slow

# Read Constraints for each scenario
suppress_message UIC-034

foreach scenario [ list FUNC_Fast FUNC_Typical FUNC_Slow ] {
        current_scenario ${scenario}
        read_sdc ../../libraries/bit_coin.sdc
}

# Set operating conditions for each corner and scenario
set PVT_FF "tt0p85v125c"
current_corner Fast
current_scenario FUNC_Fast
set_operating_conditions ${PVT_FF}

set PVT_TT "tt0p85v125c"
current_corner Typical
current_scenario FUNC_Typical
set_operating_conditions ${PVT_TT}

set PVT_SS "tt0p85v125c"
```

```
current_corner Slow

current_scenario FUNC_Slow

set_operating_conditions ${PVT_SS}


# Scenario configuration example

set_scenario_status FUNC_Fast    -setup false -hold true  -leakage_power false -
dynamic_power true  -max_transition false  -max_capacitance true  -active false

set_scenario_status FUNC_Typical -all -active true

set_scenario_status FUNC_Slow    -setup true  -hold false -leakage_power true  -
dynamic_power true  -max_transition true   -max_capacitance false  -active false.
```

The only scenario that active is FUNC_Typical. The scenarios FUNC_Fast and FUNC_Slow are not active.

In this part of the script, the corners, modes, and scenarios (MCMM) are defined as:

Corner: Fast, Typical, Slow

Mode: FUNC (Functional mode)

Scenarios:

- FUNC_Fast (corner: Fast)
- FUNC_Typical (corner: Typical)
- FUNC_Slow (corner: Slow)

3.2

In a typical design, the specific corners, modes, and scenarios can be tested depend on the requirements and target application of the design. Here are some factors and their significance:

1. Process Corners:

- **Fast Process Corner:** Represents the best-case scenario where manufacturing parameters result in faster transistor performance. This ensures that the design meets timing requirements under optimal conditions.
- **Slow Process Corner:** Represents the worst-case scenario where manufacturing parameters result in slower transistor performance. This helps identify potential timing violations and ensures robustness across process variations.

2. Voltage Corners:

- **High Voltage Corner:** Represents the scenario where the supply voltage is at its upper limit. This helps validate timing requirements, power consumption, and potential voltage-related issues.
- **Low Voltage Corner:** Represents the scenario where the supply voltage is at its lower limit. This helps identify timing violations, power supply noise issues, and potential functional failures due to reduced voltage margins.

3. Temperature Corners:

- **High Temperature Corner:** Represents the scenario where the operating temperature is at its upper limit. This helps analyse timing violations, power consumption, and potential thermal-related issues.
- **Low Temperature Corner:** Represents the scenario where the operating temperature is at its lower limit. This helps validate timing requirements and

identify potential issues related to reduced performance at low temperatures.

4. Modes and Scenarios:

- **Functional Modes:** Testing different functional modes of the design ensures correct behaviour and timing under various operating conditions.
- **Power-Saving Modes:** If the design includes power-saving features or low-power modes, these should be tested to verify power consumption, wake-up times, and potential timing issues during mode transitions.
- **Realistic Scenarios:** Creating test scenarios that simulate real-world operating conditions and use cases, such as high data traffic, specific input patterns, or environmental conditions, can help identify potential timing violations and functional issues.

[#P3.2_Q4]

Multibit registers are sequential elements in digital circuits, designed to store and synchronize multiple bits of data simultaneously. Unlike single-bit registers that handle individual bits, multibit registers manage several bits as a cohesive unit simultaneously. Here are some important features and characteristics of multibit registers:

Key Features of Multibit Registers:

1. **Data Storage:**
    - Multibit registers are engineered to store multiple bits of data together. They consist of multiple flip-flops or latches, each capable of holding a single bit, integrated within one register element.
2. **Width and Word Size:**
    - The width of a multibit register indicates the number of bits it can store. For instance, a 4-bit register can hold four bits of data. The word size denotes the number of bits in a single data word that the register can store or process as a unit.
3. **Synchronization:**
    - These registers ensure that all the bits are synchronized to a common clock signal. This coordination ensures that data is sampled and

updated simultaneously across all bits, preserving data integrity and preventing timing issues.

2.3.[#P3.3_Q1]

HOLD:

```
Report : timing
        -path_type full
        -delay_type min
        -max_paths 1
        -report_by design
Design : bit_coin
Version: V-2023.12-SP3
Date   : Sun Jul 21 18:10:44 2024
****************************************

  Startpoint: bit_secure_5/sipo_slice_first/count_reg[0] (rising edge-triggered flip-flop clocked by lclk)
  Endpoint: bit_secure_5/sipo_slice_first/count_reg[0] (rising edge-triggered flip-flop clocked by lclk)
  Mode: FUNC
  Corner: Typical
  Scenario: FUNC_Typical
  Path Group: lclk
  Path Type: min

  Point                                          Incr      Path
  -----------------------------------------------------------------------
  clock lclk (rise edge)                         0.00      0.00
  clock network delay (ideal)                    0.00      0.00

  bit_secure_5/sipo_slice_first/count_reg[0]/CLK (SDFFARX1_RVT)
```

```
  bit_secure_5/sipo_slice_first/count_reg[0]/QN (SDFFARX1_RVT)
                                                 0.09      0.09 f
  bit_secure_5/sipo_slice_first/count_reg[0]/D (SDFFARX1_RVT)
                                                 0.00      0.09 f
  data arrival time                                        0.09

  clock lclk (rise edge)                         0.00      0.00
  clock network delay (ideal)                    0.00      0.00
  bit_secure_5/sipo_slice_first/count_reg[0]/CLK (SDFFARX1_RVT)
                                                 0.00      0.00 r
  library hold time                             -0.06     -0.06
  data required time                                     -0.06
  -------------------------------------------------------------------------
  data required time                                     -0.06
  data arrival time                                      -0.09
  -------------------------------------------------------------------------
  slack (MET)                                              0.14
```

Slack for Hold constraint: 0.14

SETUP:

```
Report : timing
        -path_type full
        -delay_type max
        -max_paths 1
        -report_by design
Design : bit_coin
Version: V-2023.12-SP3
Date   : Sun Jul 21 18:10:44 2024
****************************************

  Startpoint: bit_secure_2/lreset_sync/reset_sync_reg (rising edge-triggered flip-flop clocked by lclk)
  Endpoint: bit_secure_2/slice_9/sipo_bit/rd_addr_reg[1] (recovery check against rising-edge clock clocked by lclk)
  Mode: FUNC
  Corner: Typical
  Scenario: FUNC_Typical
  Path Group: lclk
  Path Type: max

  Point                                          Incr      Path
  -----------------------------------------------------------------------
  clock lclk (rise edge)                         0.00      0.00
  clock network delay (ideal)                    0.00      0.00

  bit_secure_2/lreset_sync/reset_sync_reg/CLK (DFFARX2_RVT)
```

```
bit_secure_2/lreset_sync/reset_sync_reg/Q (DFFARX2_RVT)
                                                2824.44   2824.44 r
bit_secure_2/slice_9/sipo_bit/rd_addr_reg[1]/SETB (SDFFASX1_RVT)
                                                   0.00    2824.44 r
data arrival time                                          2824.44

clock lclk (rise edge)                             1.00      1.00
clock network delay (ideal)                        0.00      1.00
bit_secure_2/slice_9/sipo_bit/rd_addr_reg[1]/CLK (SDFFASX1_RVT)
                                                   0.00      1.00 r
library setup time                             -1080.50  -1079.50
data required time                                         -1079.50
-------------------------------------------------------------------------
data required time                                         -1079.50
data arrival time                                          -2824.44
-------------------------------------------------------------------------
slack (VIOLATED)                                           -3903.94
```

Slack for Hold constraint: -3903.94

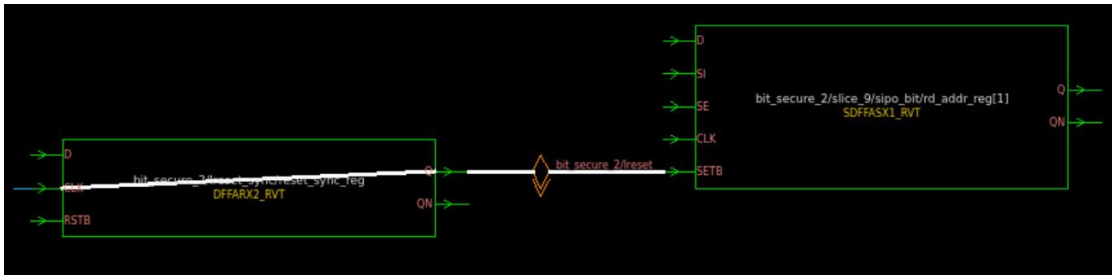
2.4.[#P3.3_Q2]

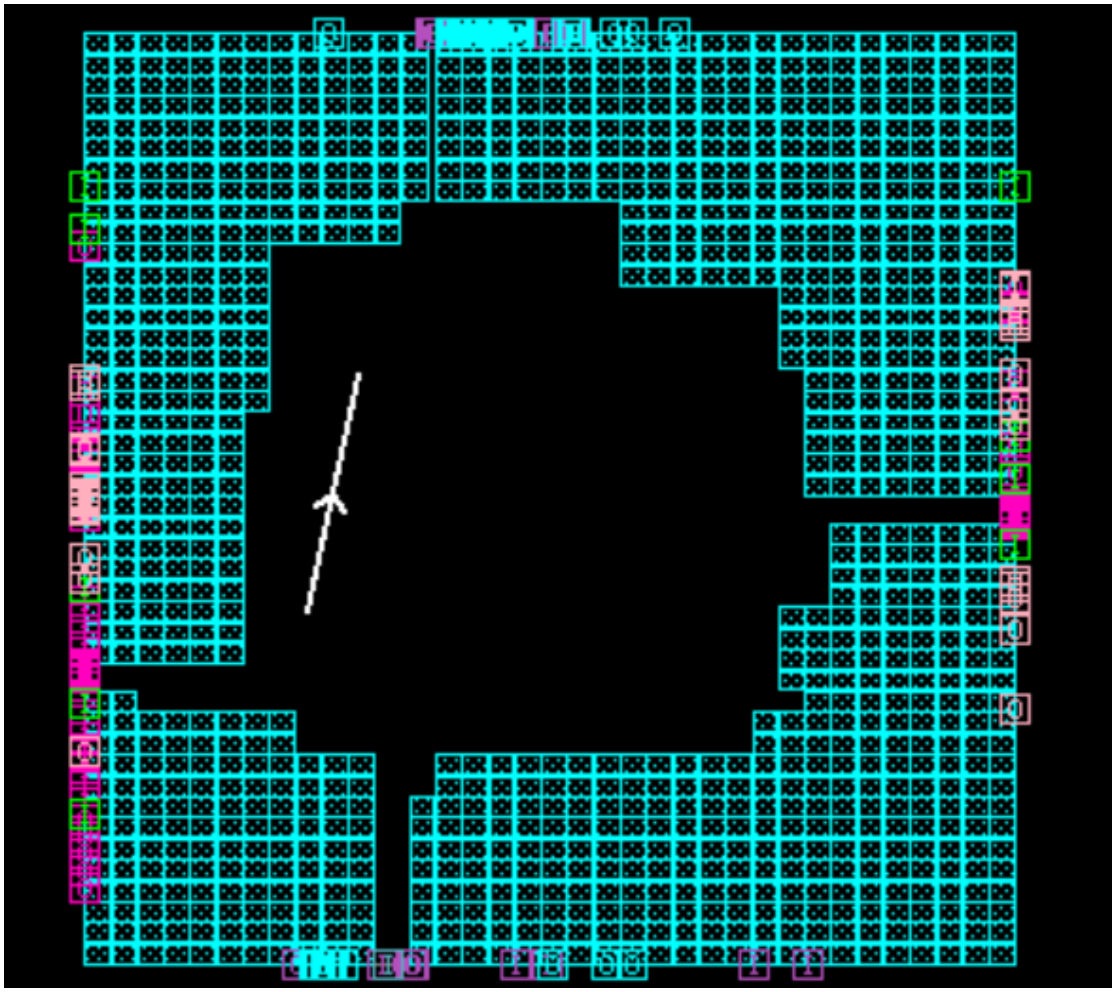
We took the startpoint and Endpoint from the report

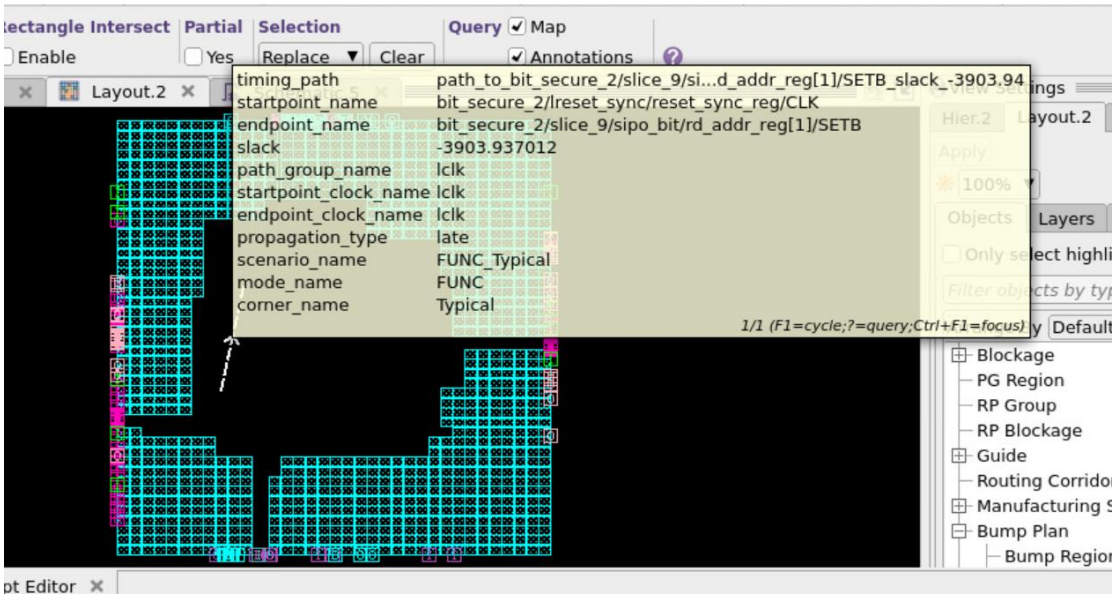
set tp1 [get_timing_paths -from bit_secure_2/lreset_sync/reset_sync_reg -to bit_secure_2/slice_9/sipo_bit/rd_addr_reg[1]]

change_selection $tp1

2.5.[#P3.3_Q3]







Full Disclosure:

During the work on the project, we did not notice that the route should also be taken out for HOLD, but we recorded the route from the report that we took out in the first section and it appears in TCL, after consulting with the practitioner (Elad), it was decided to add it to the report and not show the critical route in the diagram.

#Check timing for HOLD violations

report_timing -scenarios [all_scenarios] -delay_type min

#####FILL BY STUDENTS #####

change_selection [get_timing_paths -from **bit_secure_5/sipo_slice_first/count_reg[0]** -to **bit_secure_5/sipo_slice_first/count_reg[0]]**

#####FILL BY STUDENTS #####

2.6.[#P3.3_Q4]

We used the command:

Fc_shell> **report_timing -max_path 10**

We noticed the pattern that in the critical routes the problem comes when the route passes through:

set_false_path -from **bit_secure_2/lreset_sync/reset_sync_reg/Q** -to *

Full disclosure: during the work on the project due to a reading error in the project document we neglected to fill in the line after the "-to" operation due to the fact that we left

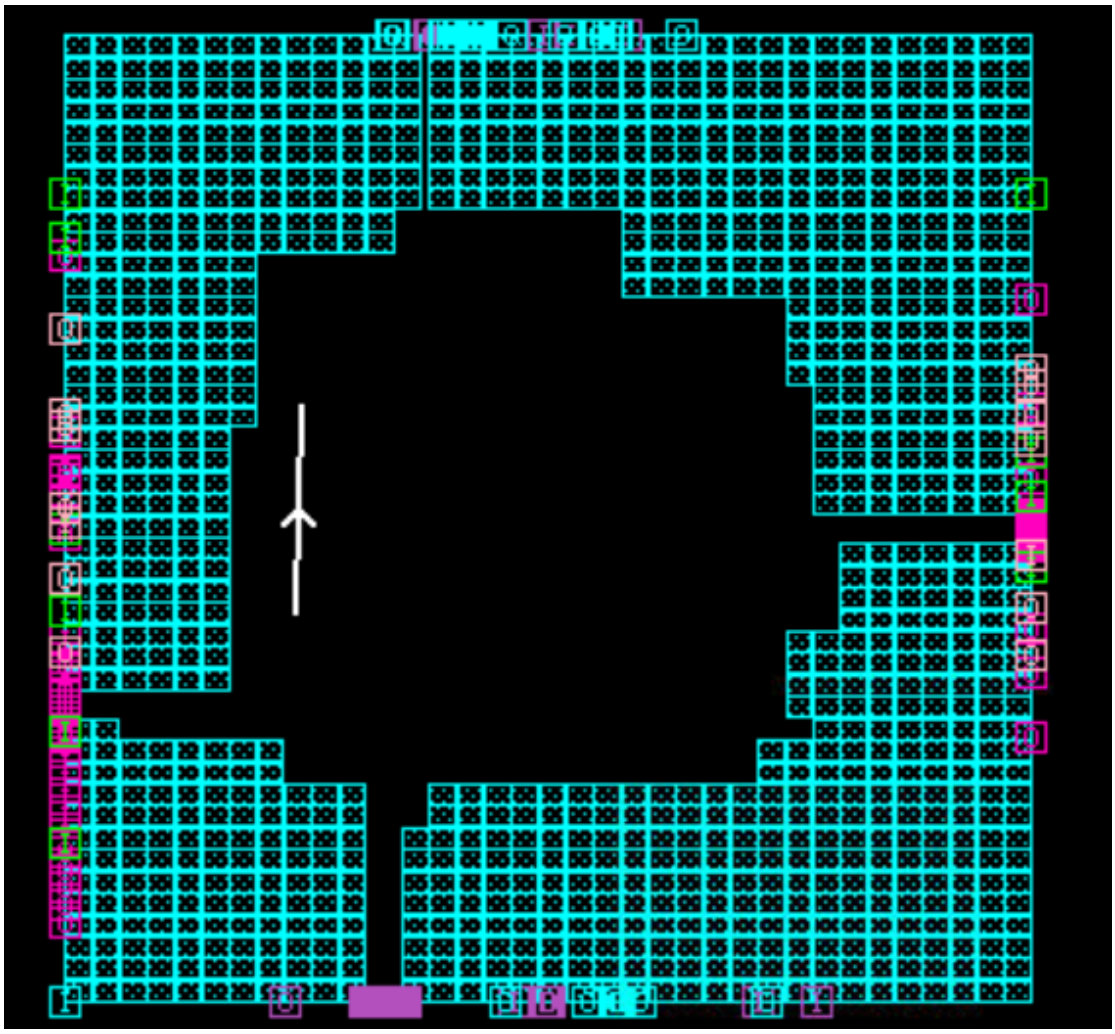
the asterisk sign we actually caused the software to ignore all routes starting with the string bit_secure_2/lreset_sync/reset_sync_reg/Q

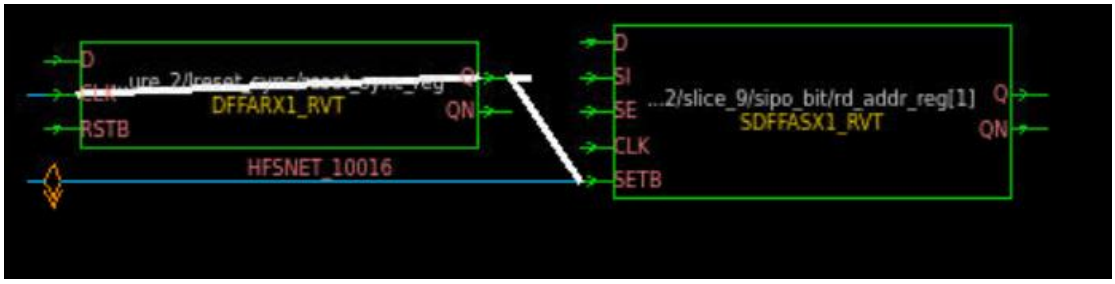
This form of running caused an overly aggressive filtering and caused an error in the running and in the optimization results.

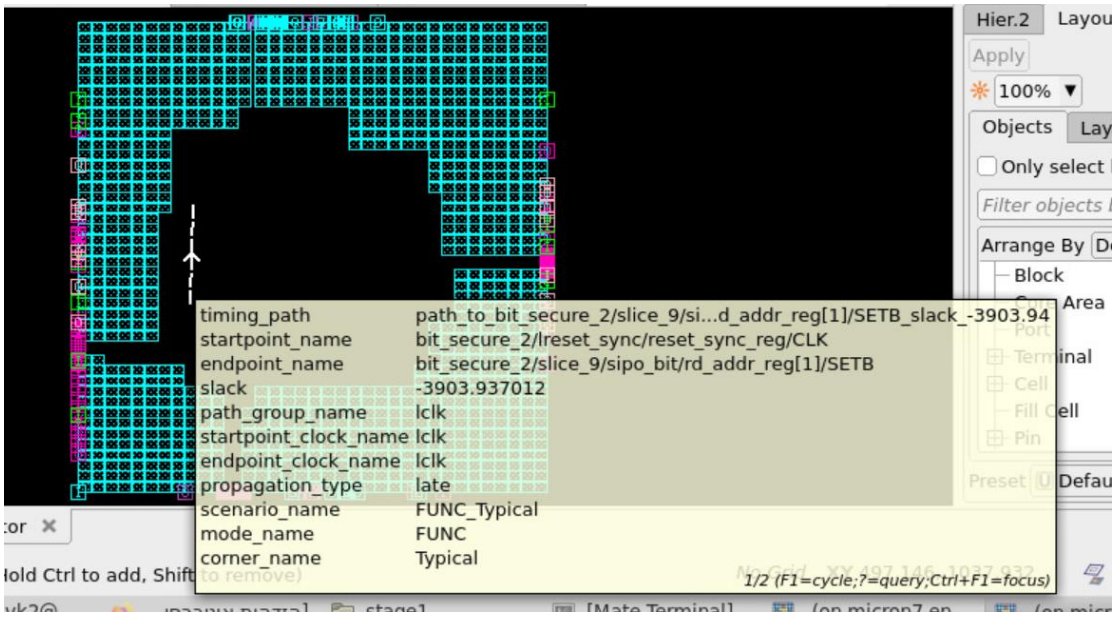
A mistake that we realized during DEBUG that we did after the results seemed abnormal to us and after consulting with the course practitioner (Elad), it was decided not to correct it but to mention it during the report.

2.8.[#P3.3_Q5]

NEW SETUP PATH







At this stage of the project the goal is to see how the optimizations we made during the tcl result in the improvement of the critical times and routes, we were alerted because we experienced an error at the beginning of the report, we don't see it in this screenshot but the principle was understood to us and as stated before, after consulting with the practitioner of the course (Elad), it was decided to continue with the exercise anyway.

4. **[#P3.3_Q6]** Analyze those reports and answer the following questions:

4.1.<u>Setup timing</u>**:** What is the first stage in the compile_fusion flow that fixes the SETUP violations?

The "Fusion Analysis" stage is the first stage in the “Compile fusion flow” that addresses setup violations. During this stage, the tool performs a detailed analysis of the design to identify and fix any violations related to setup time requirements. The primary goal of this stage is to ensure that the design meets the necessary timing constraints before moving on to subsequent stages of the compilation process.

4.2. <u>Hold timing</u>: From which stage in the general chip implementation (Synthesis, STA, Floorplan, Placement...) should we check the hold violations and why?

Hold violations should be checked during the **Static Timing Analysis (STA)** stage.

Checking hold violations during the STA stage is crucial for several reasons:

a) Early Identification**:** Identifying hold violations early in the design flow allows designers to address timing issues before they propagate further into the design process.

b) Avoiding Complex Routing**:** Fixing hold violations before the routing stage helps to avoid complex and time-consuming routing optimizations and iterations. This can significantly streamline the overall design process.

c) **:** Ensuring that the design meets the required hold time constraints Ensuring Reliability during the STA stage is essential for the reliable and correct operation of the chip. Hold violations can lead to data corruption and operational failures if not properly addressed.

By checking and fixing hold violations during the STA stage, designers can ensure a more robust and efficient design process, leading to a higher quality final product.

4.3. Area: What can we derive out of the following information from the area report?

4.3.1. Number of ports

4.3.2. The ratio between number of the combinational cells and the sequential cells

4.3.3. Number of macros

4.3.4. Number of buffers and inverters

4.3.5. Area of cells + area of macros

4.3.1. The number of ports, referring to the total input and output ports in the design, indicates the design's connectivity and I/O requirements, impacting design complexity, I/O management, resource utilization, and performance, particularly in high-speed or high-data designs.

4.3.2. The ratio between the number of combinational cells and sequential cells provides insight into the design's overall structure and functionality, indicating the mix of combinational and sequential logic, affecting speed and efficiency, influencing power requirements, and impacting the overall area, which is crucial for optimizing performance, power, and area. This balance is crucial for optimizing performance, power, and area.

4.3.3. The number of macros, referring to the total count of predefined, pre-verified, and reusable modules used in the design, is valuable because it incorporates specialized functionalities, speeds up the design process through reuse, enhances performance with optimized blocks, and aids in managing design resources effectively.

4.3.4. The number of buffers and inverters, referring to the count of these cells within the design, is important because they maintain signal integrity by buffering signals, shift signal levels to ensure proper logic, change signal polarity as required, and help understand how well the design manages signal delays and transitions, which are fundamental for maintaining signal quality and ensuring reliable operation.

4.3.5. The area of cells and macros, referring to the sum of the areas of all individual cells and macros in the design, is crucial because it contributes to the overall chip area, affects performance by influencing signal propagation and timing, impacts power consumption, and affects the manufacturing cost of the chip, making it essential to optimize for balancing performance, power, and cost.

4.4. Design: What can we derive out of the following information from the design report?

4.4.1. ICG - integrated clock gates

4.4.2. Latches

4.4.3. Number of Power Domains

4.4.1. Integrated Clock Gates (ICGs) control clock signals to inactive parts of the design, reducing power consumption and enhancing overall efficiency.

4.4.2. Latches are used to store and retain a binary state until a specific triggering

condition occurs. Unlike flip-flops, latches have a transparent or level-sensitive behavior, meaning their outputs can change as long as the enabling condition is met. This characteristic makes latches useful for certain timing and control applications within the design.

4.4.3. The number of power domains refers to the count of distinct power regions within the design, each with independent power supply and control. This information indicates how power management is implemented, affecting power efficiency and isolation of different parts of the design.

4.5. Power: What can we derive out of the following information from the power report?

4.5.1. What is the static, dynamic and total power after all compile_fusion

command?

4.5.2. What does each of the power types mean?

4.5.1.

```
  Cell Internal Power    = 4.13e+11 pW ( 51.8%)
  Net Switching Power    = 3.84e+11 pW ( 48.2%)
Total Dynamic Power      = 7.97e+11 pW (100.0%)

Cell Leakage Power       = 1.44e+11 pW


  Attributes
  ----------
      u  -  User defined power group
      i  -  Includes clock pin internal power

Power Group         Internal Power        Switching Power          Leakage Power            Total Power    (   %  )    Attrs
--------------------------------------------------------------------------------------------------------------------------------
io_pad                   0.00e+00              0.00e+00               0.00e+00               0.00e+00    (  0.0%)
memory                   2.22e+11              1.27e+10               0.00e+00               2.34e+11    ( 24.9%)
black_box                0.00e+00              0.00e+00               0.00e+00               0.00e+00    (  0.0%)
clock_network            1.90e+11              3.63e+11               8.03e+08               5.54e+11    ( 58.8%)        i
register                -1.33e+09              1.17e+09               1.02e+11               1.02e+11    ( 10.8%)
sequential               0.00e+00              0.00e+00               0.00e+00               0.00e+00    (  0.0%)
combinational            2.24e+09              7.85e+09               4.20e+10               5.21e+10    (  5.5%)
--------------------------------------------------------------------------------------------------------------------------------
Total                    4.13e+11 pW           3.84e+11 pW            1.44e+11 pW            9.41e+11 pW
1
```

From the report we can see that:

Total Dynamic Power = $7.97 \cdot 10^{11} pW = 0.797W$

Total Power = $9.41 \cdot 10^{11} pW = 0.941W$

Total Static Power = Total Power- Total Dynamic Power = $0.941W - 0.797W = 0.144W$

As we mention in previous section we had a problem while running the compiler which is why we received suboptimal results as expected.

4.5.2.

<u>Static Power:</u> Also known as leakage power, is the power consumed by a circuit even when it is not switching or performing any active operations. It is primarily caused by current leakage through transistors and other components when they are in the off state.

<u>Dynamic Power:</u> This is the power consumed by a circuit during switching activities, such as charging and discharging capacitive loads and driving signal transistors. It is directly related to the frequency, activity, and switching capacitance in the design.

<u>Total Power:</u> This represents the overall power consumption in the circuit, taking into account both the power consumed during active switching activities (dynamic power) and power consumed during idle or standby periods (static power).