Synthesis Questions :

1. **What is synthesis?**

Synthesis is a VLSI design process in which the high level design (ie RTL code) is optimized to gate level netlist that targets a specific technology node.

Sythesis= translation + logic optimization + gate mapping

**Q Explain the difference between logic synthesis and physical synthesis**

**Logical synthesis** takes the RTL (register transfer level) description of a design and produces a gate-level netlist.

Logical synthesis operates at a high level of abstraction, where it is concerned with the logical functionality of the design.

Process:

· High-Level Optimization: Optimization of the RTL code to improve performance, reduce area, or minimize power consumption.

· Technology Mapping: Mapping the optimized design onto a specific technology library that contains predefined gate types and their characteristics.

· Constraint Satisfaction: Ensuring that the design meets timing constraints, area constraints, and other specified requirements.

**Physical synthesis** takes the gate-level netlist and produces a physical layout of the design. Physical synthesis operates at a lower level of abstraction, where it is concerned with the physical implementation of the design.

Process:

· Placement: Determining the physical locations of the gates on the chip. This step must take into account timing constraints, power distribution, and area efficiency.

· Clock Tree Synthesis (CTS): Creating a clock distribution network that ensures minimal clock skew and meets timing requirements.

· Routing: Connecting the gates according to the netlist, ensuring signal integrity and meeting electrical constraints.

· Optimization: Iteratively refining the placement and routing to optimize for performance, power, and area.

· Sign-Off: Verifying the physical design through processes such as timing analysis, power analysis, and design rule checking (DRC).

**Q. What are the constraints you use for synthesis?**

Constraints are the designer's specification of timing and environmental restrictions under which Synthesis/Implementation is to be performed

So Constraints provide design goals/specifications that the design must meet through optimization to make the design functional in hardware

Operating conditions:

set\_operating\_conditions

Wire-load models:

set\_wire\_load\_mode

set\_wire\_load\_model

set\_wire\_load\_selection\_group

Environmental:

set\_drive

set\_driving\_cell

set\_load

set\_fanout\_load

set\_input\_transition

set\_port\_fanout\_number

Design rules:

set\_max\_capacitance

set\_max\_fanout

set max transition

Timing:

create\_clock

create\_generated\_clock

set\_clock\_latency

set\_clock\_transition

set\_disable\_timing

set\_propagated\_clock

set\_clock\_uncertainty

set\_input\_delay

set\_output\_delay

Exceptions:

set false\_path

set max delay

set\_multicycle\_path

set\_max\_dynamic\_power

set\_max\_leakage\_power

**Q Explain the role of Synopsys designware libraries and Gtech libraries in synthesis**

**Design ware Library**

➤ A DesignWare library is a collection of reusable circuit-design building blocks (components) that are tightly integrated into the Synopsys synthesis environment.

➤ During synthesis, Design Compiler selects the right component with the best speed and area optimization from the DesignWare Library.

Here’s how they play a role in synthesis:

Pre-verified IP Blocks: DesignWare libraries provide high-quality, silicon-proven IP blocks that can be directly instantiated in your design. This reduces the time and effort needed to design and verify these components from scratch.

Consistency Across Platforms: These libraries ensure that the same IP can be used across different platforms, whether you’re targeting an ASIC or an FPGA. This consistency helps in maintaining a single source of RTL for both prototyping and final implementation.

Accelerated Time to Market: By leveraging pre-designed components, designers can significantly reduce the development time, allowing for faster time to market.

**G tech**

GTECH Libraries are generic technology libraries provided by Synopsys, representing a technology-independent intermediate netlist format. These libraries are used during the initial stages of synthesis before mapping the design to a specific technology library.

Here’s how they play a role in synthesis:

Generic Technology Mapping: GTECH libraries provide a set of generic technology-independent gates. During the initial stages of synthesis, the RTL is mapped to these generic gates, which helps in optimizing the design at a high level before mapping it to specific technology libraries.

Intermediate Representation: GTECH serves as an intermediate representation of the design. This allows for various optimizations and transformations to be applied before the design is finally mapped to the target technology-specific cells.

Simplifies Technology Mapping: By first mapping to GTECH, the synthesis tool can perform technology-independent optimizations. This makes the subsequent mapping to the actual technology library more efficient and effective.

1. **What is clock uncertainty?**

Clock uncertainty is the total amount of possible time variation between any pair of clock edges. The uncertainty consists of the computed clock jitter (system, input, and discrete); the phase error introduced by certain hardware primitives; and any clock uncertainty specified by the user in the design constraints (set\_clock\_uncertainty).

For primary clocks, the jitter is defined by set\_input\_jitter and set\_system\_jitter. For clock generators such as MMCM and PLL, the tool computes the jitter based on user-specified jitter on its source clock and its configuration. For other generated clocks (such as flop based clock dividers), the jitter is the same as that of its source clock.

The user-specified clock uncertainty is added to the uncertainty computed by the Design Suite timing engine. For generated clocks (such as from MMCM, PLL, and flop-based clock dividers), uncertainty specified by the user on source clock does not propagate through the clock generators.

Uncertainty defines a window within which this clock edge can occur. It’s like saying,

Factors Included in Uncertainty:

Jitter: This represents the deviation of the actual clock edge from its ideal position. Sometimes ticking a little early or late.

Skew: Skew refers to the variation in arrival times of the clock signal at different parts of the chip. Some areas might receive the clock earlier, while others get it later.

Setup Uncertainty:

Pre-Cts = Jitter + Skew + Extra setup margin

Post Cts = Jitter + Extra setup margin

Here for slack analysis setup uncertainity is subtracted from required time

Ie slack =(requiredtime - setup uncertainity) -arrival time

Hold Uncertainty:

Pre-Cts = Skew + Extra hold margin

Post cts = Extra hold margin

Here for slack analysis hold uncertainity is added to required time

Ie slack =arrivaltime -( requiredtime + hold uncertainity)

**Q. What will happen to a design that is synthesized without constraints?**

* If a design is not provided with constraints the synthesis tool will not have any timing constraints to meet
* This will lead to setup and hold violations which will in turn lead the design to not function as expected
* The design will enter metastable state since not setting constraints will lead to setup and hold violation

**Q. List out some synthesizable and non-synthesizable verilog constructs**

Synthesizable constructs are those that can be directly translated into hardware during the synthesis process, while non-synthesizable cannot be translated into constructs they are typically used for simulation.

Synthesizable Constructs

* Basic Logic Gates and Operators

Logical operators (&&, ||, !)

Arithmetic operators (+, -, \*, /, %)

Bitwise operators (&, |, ^, ~, <<, >>)

Relational operators (<, <=, >, >=)

* Continuous assignment(assign)
* Procedural assignment

Inside always(=,<=)

* Conditional Constructs(if-else,case,for)

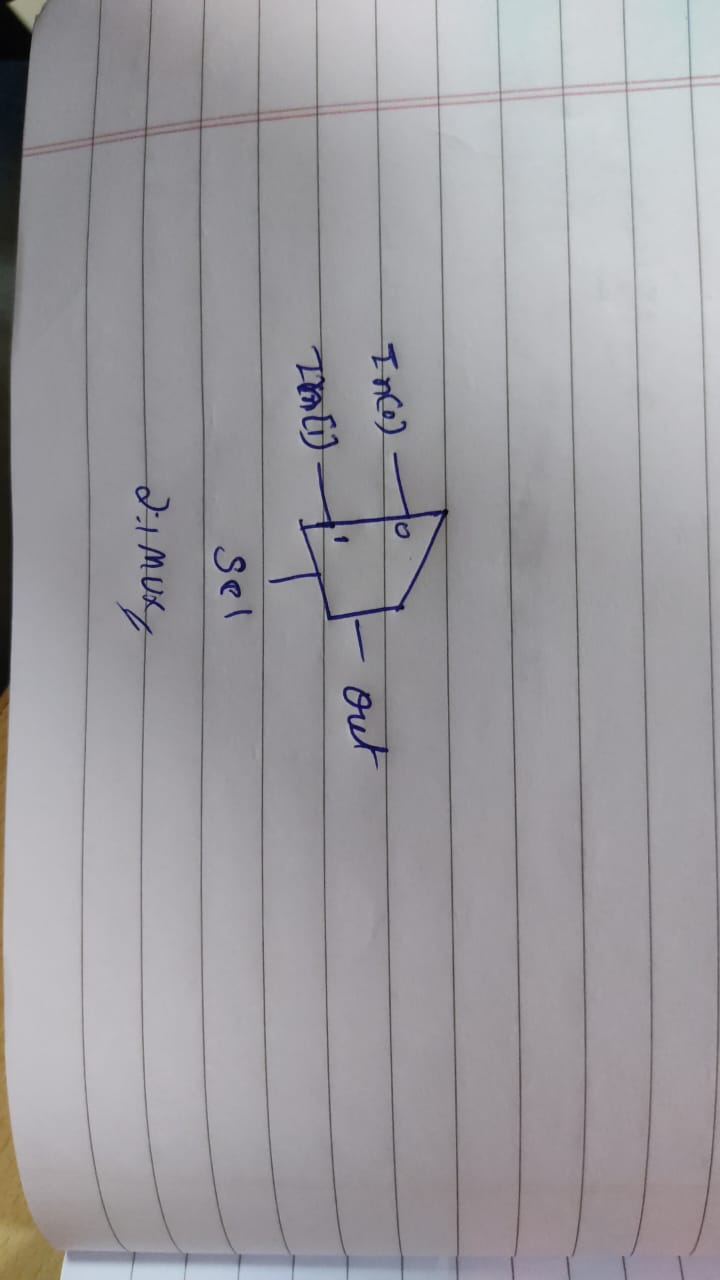
Non-synthesizable Constructs:

* Procedural Block(initial)
* Case equality Operators(===,!==)
* Data types(real, time, event, tri0,tri1)
* System Tasks($random,$display,$monitor)
* Procedural statements(fork, join)

**Q. What is the hardware that is inferred by the conditional operator?**

A conditional operator is infered as a mux after synthesis .

Example : assign out=sel?in[1]:in[0];



**Q. Given two ASICs, one has setup violation and other has hold violation. How do you fix them without modifying the design?**

Setup violations are essentially where the data path is too slow compared to the clock speed at the capture flip-flop. With that in mind there are several things a designer can do to fix the setup violations.

**Method 1: Reduce the amount of buffering in the path.**

• It will reduce the cell delay but increase the wire delay. So if we can reduce more cell delay in comparison to wire delay, the effective stage delay decreases.

**Method 2: Replace buffers with 2 Inverters placed farther apart**

• Adding 2 inverters in place of 1 buffer, reducing the overall stage delay.

**Method 3: Clock skew:**

• By delaying the clock to the end point can relax the timing of the path, but you have to make sure the downstream paths are not critical paths.

**Method 4** : Increase Driver Size or say increase Driver strength (also known as upsize the cell)

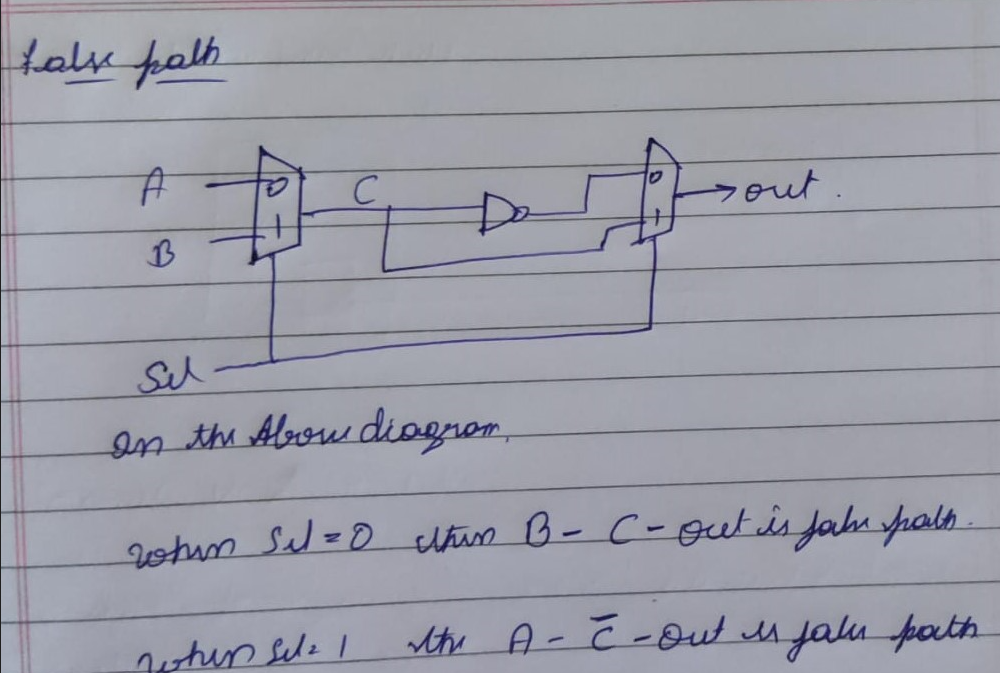
**Method 5:** Inserting repeaters:

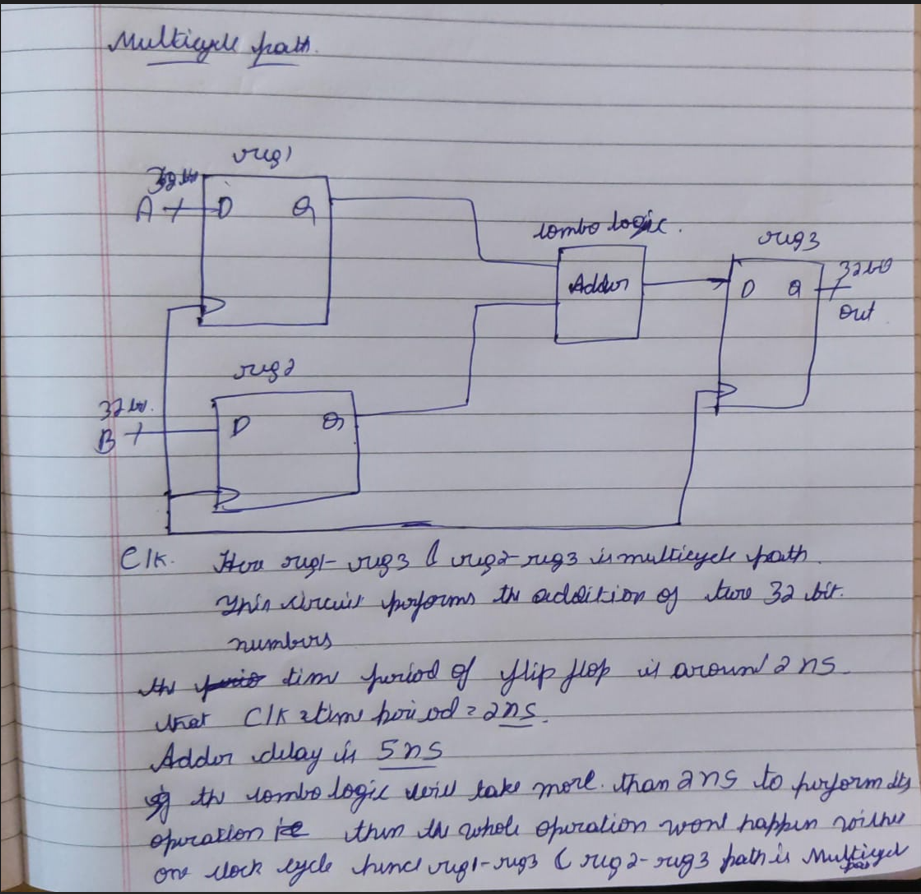
Hold violation is the opposite of setup violation. Hold violation happen when data is too fast compared to the clock speed. For fixing the hold violation, delay should be increases in the data path. Note: Hold violations is critical and on priority basis in comparison are not fixed before the chip is made, more there is nothing that can be done post fabrication to fix hold problems unlike setup violation where the clock speed can be reduced. The designer needs to simply add more delay to the data path. This can be done by

**Method 1** : By Adding delays. Adding buffer / Inverter pairs /delay cells to the data path helps to fix the hold violation.

**Method 2** : Decreasing the size of certain cells in the data path. It is better to reduce the cells closer to the capture flip flop because there is less likely hood of affecting other paths and causing new errors.

**Q. Give one example of false path and multicycle path**





**Q. How do you handle false paths and multicycle paths in timing analysis?**

False Paths

False paths are signal paths that are not critical for the correct operation of the circuit and should be ignored during timing analysis.

We can handle false path during timing analysis using the set\_false\_path Command

set\_false\_path -from <start\_point> -to <end\_point>

above command marks a specific path between the start and end points as a false path, which will be ignored during timing analysis.

Multicycle Paths

Multicycle paths are paths where signals are allowed to take multiple clock cycles to propagate from the start point to the end point.

We can handle multicycle path during timing analysis by Specifying Multicycle Paths

Using the set\_multicycle\_path Command

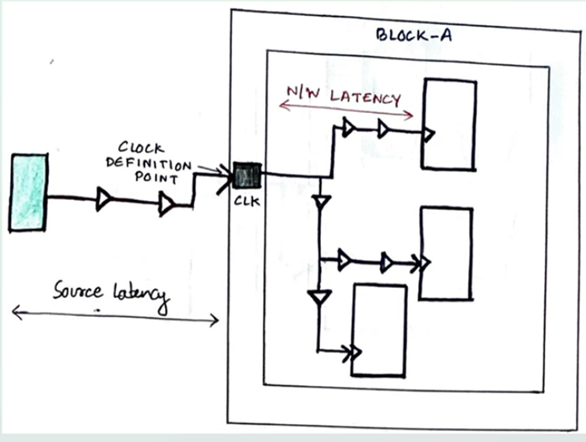
set\_multicycle\_path -from <start\_point> -to <end\_point> <multicycle\_value>

above command specifies that a path between the start and end points can take multiplecycles as indicated by the multicycle\_value.

**Q. What is clock latency? How do you specify it?**

Clock latency is the time taken by the clock edge to reach to the clock pin of flip flop from the clock port.

Clock latency = source latency + network latency



Source Latency(clock insertion delay)

It is defined as the amount of time a Clock Signal takes to travel from a Clock Source to reach to Clock Definition Point of the Chip.

Network Latency(network insertion delay)

It is defined as the amount of time the Clock Signal takes to travel from the Clock Definition Point of the Chip to reach the Clock pin of the Flops. Every clock pin in a Circuit has a different Network Delay based on the location of the flop from the Clock Definition Point.

Using below command we can specify clock latency

set\_clock\_latency -latency <latency\_value> [get\_clocks <clock\_name>]

**Q. What are the commands in your DC synthesis script?**

read\_verilog counter.v : Reads the Verilog file counter.v into the design environment.

current\_design up\_counter : Sets the current design context to up\_counter.

link : Links all the modules and libraries in the current design.

check\_design : Verifies the design for errors or issues.

write -format ddc -hier -output unmapped/counter.ddc : Writes the design data in DDC (Design Data Container) format to unmapped/counter.ddc.

source up\_counter.sdc : Loads and applies the constraints specified in up\_counter.sdc.

write -format ddc -hier -output unmapped/counter\_constraint.ddc : Writes the design constraints in DDC format to unmapped/counter\_constraint.ddc.

check\_timing: Checks the timing analysis results for any violations or issues.

compile : Compiles the design, performing synthesis and optimization.

write -format verilog -hierarchy -output netlist/counter\_netlist.v : Writes the synthesized netlist in Verilog format to netlist/counter\_netlist.v.

write -format ddc -hierarchy -output mapped/upcounter.ddc : Writes the mapped design data in DDC format to mapped/upcounter.ddc.

exit : Exits the Design Compiler environment

**Q. What are the advantages of using set\_app\_var over set when specifying DC variable values?**

Using set\_app\_var over set when specifying DC variable values provides several advantages, including improved consistency, context-sensitive behavior, better scope management, enhanced script readability, and proper handling of complex values. These benefits make set\_app\_var the preferred choice for setting application-specific variables in Design Compiler, ensuring a more reliable and predictable synthesis process

**Q. What is the difference between read\_verilog and analyze and elaborate commands?**

analyze-elaborate and read\_verilog are used for the same purpose

analyze-eloborate:-

analyze command first checks the HDL codes for syntax errors and creates the intermediate files (\*.mr , \*.pvl, \*.syn) which are used latter in elaborate command.

elaborate command use the intermediate files and change the parameter values (if given) and convert the code and arithemetic operaters to GTECH and DW components.here intermediate file creation happens.

read\_verilog :-

performs the same operation what analyze-eloborate does ,but does not create the intermediate files.

**Q. What kind of timing violations are seen in a typical timing analysis report?**

Setup Time Violations: These occur when the data signal does not arrive at the destination flip-flop or latch before the setup time requirement of the clock edge. [This can cause incorrect data to be captured](https://vlsiweb.com/timing-violations/" \t "https://edgeservices.bing.com/edgesvc/_blank)

Hold Time Violations: These happen when the data signal changes before the hold time requirement after the clock edge. [This can lead to data corruption as the previous data might not be held long enough](https://www.synopsys.com/glossary/what-is-static-timing-analysis.html" \t "https://edgeservices.bing.com/edgesvc/_blank)

Slack Violations

Slack Violations occur when the slack is negative, indicating that the signal arrives too late.

Critical path violations

Critical path violations occur when the timing requirements along the longest paths in a digital circuit are not met. These violations can prevent the circuit from operating correctly at its intended clock frequency.

Clock Skew Violations: Clock skew is the difference in timing between clock signals reaching different parts of the circuit. [Excessive skew can cause setup or hold violations](https://www.synopsys.com/glossary/what-is-static-timing-analysis.html" \t "https://edgeservices.bing.com/edgesvc/_blank).

[Clock Gating Violations: These occur when the clock gating logic does not meet the required timing constraints, leading to incorrect clock signals being propagated](https://www.synopsys.com/glossary/what-is-static-timing-analysis.html" \t "https://edgeservices.bing.com/edgesvc/_blank)



