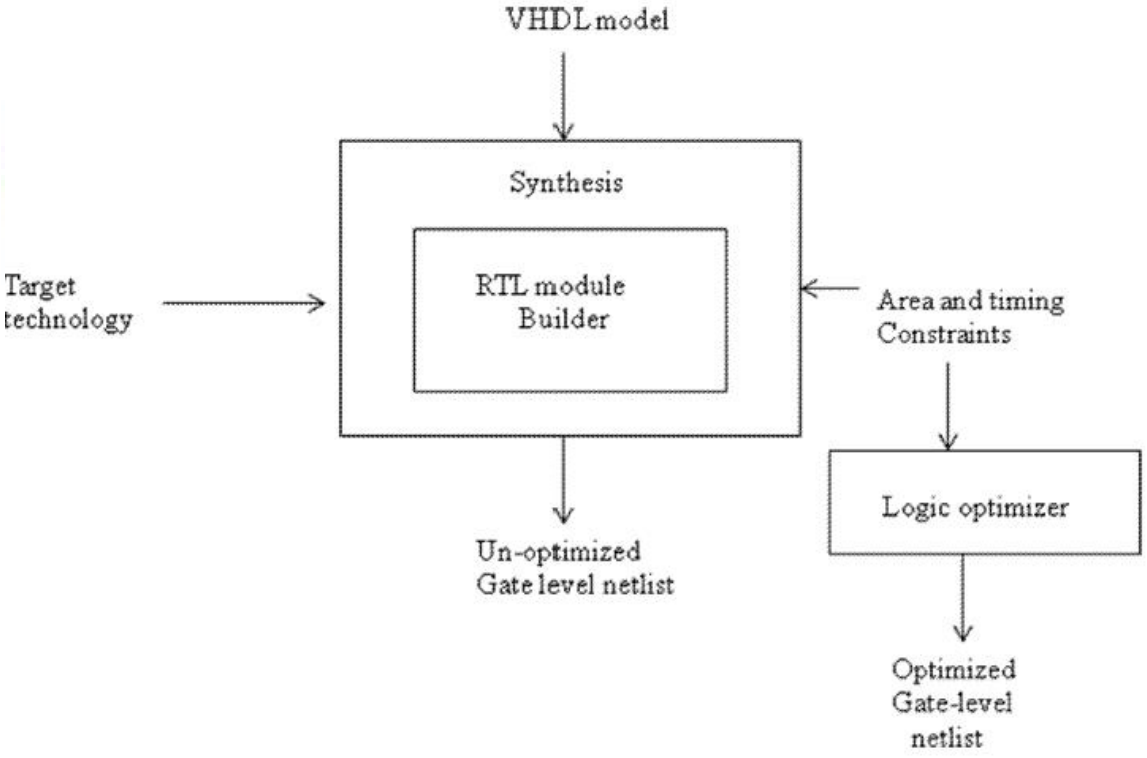
UNIT-5

VHDL SYNTHESIS:

It is the process of constructing a gate level net list from a model of a circuit described in VHDL. The process is explained by the following figure.



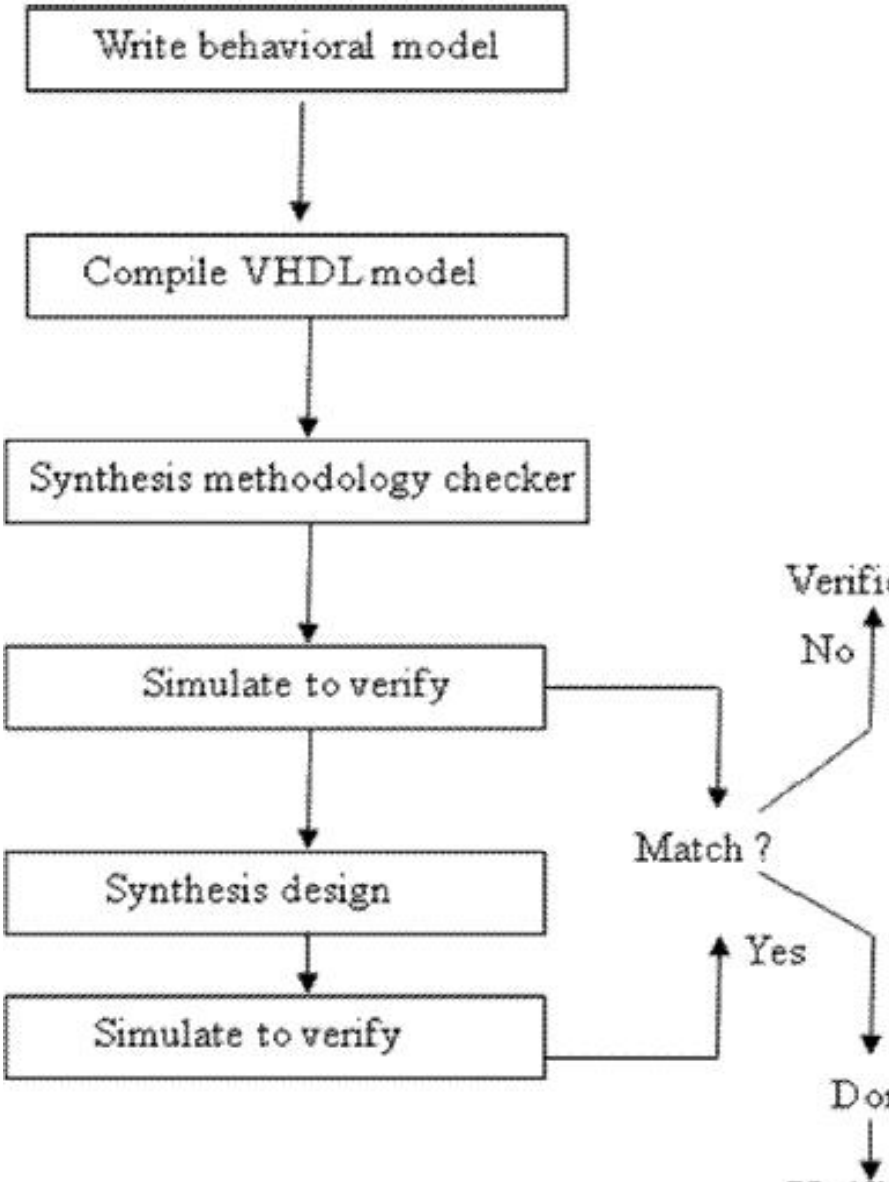
A synthesis process may alternately a RTL net list, which is comprised of generate Register – Transfer level blocks.

Having produced a gate-level net list, a logic optimizer reads this net list and the circuit for the user specified area is optimized. These constraints are also used by the module builder for appropriate selection or generation of RTL blocks.

With the help of VHDL the designer can model a circuit at different levels of abstraction ranging from the gate level, RTL lever, and behavioral level to the algorithmic level.

There is no standardized subset of VHDL, for synthesis. Each synthesis system may provide a different mechanism to model a flip-flop or a latch. Therefore, an own subset is defined by each system.

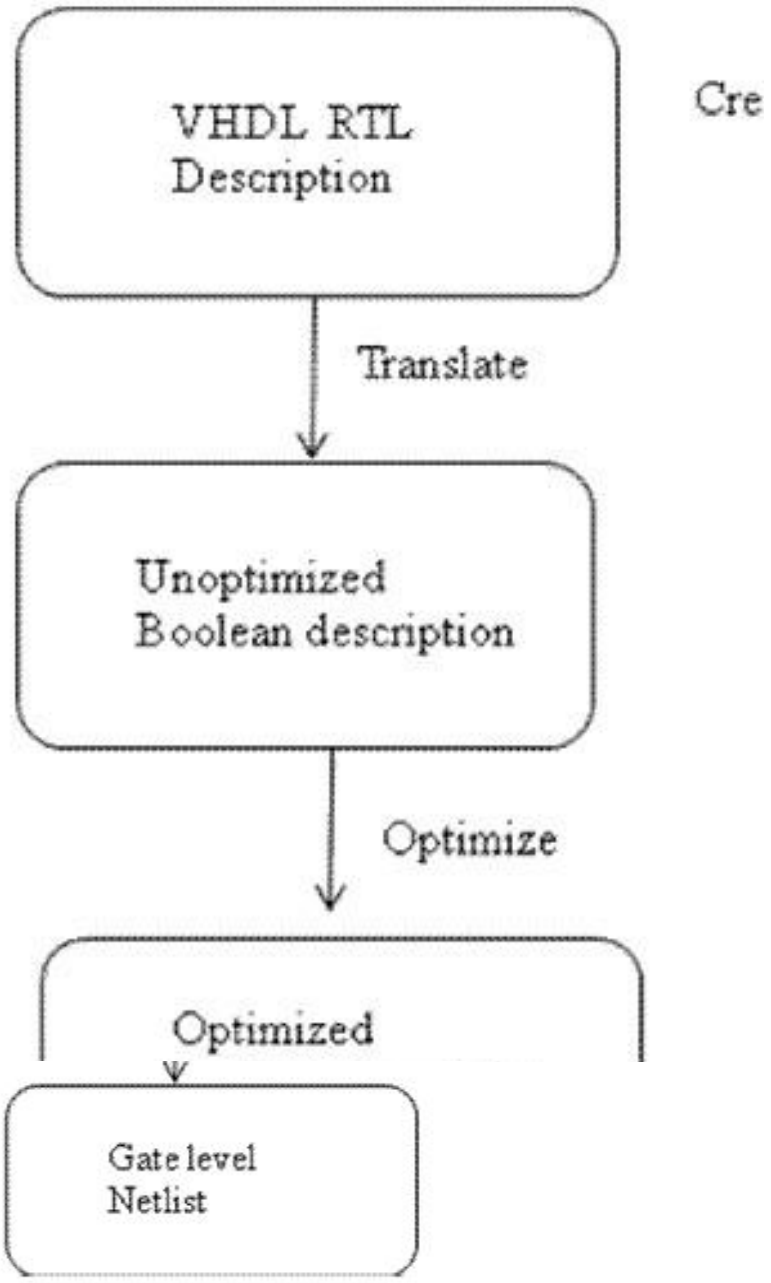
The design process shown in the following figure has to be followed for VHDL synthesis:



In this design process, a synthesis methodology checker is needed to check if the model being written is compatible for synthesis or not. This is done in the first Simulation loop. In this way, after the Simulation results have been verified, a verified synthesizable model exists, which can then be synthesized.

VHDL Synthesizer:

The RTL description is converted to gates by three steps basically.



The RTL description at first is converted to a un optimized Boolean description which consists of basic gates, flip flops and latches. This is functionally correct but completely an unoptimized description. Next, Boolean optimization algorithms are executed on this Boolean equivalent description to produce a optimized Boolean equivalent description. Finally this is mapped to acheal logic gates by using technology library of target process. Each stage is explained below:

Translation: It is done from RTL description to unoptimized Boolean description, not user controllable. All if, case and other statements are converted to their Boolean equivalent in this intermediate form.

Boolean optimixation: If converts the unoptimized Boolean description to optimized Boolean description. Many algorithms and rules are used to do this

Ex: convert to very low-level description (PLA format) and then optimized by reducing the logic generated by sharing common terms.

Flattening: The process of converting unoptimized Boolean description to PLA format is known as flattening, because it creates a flat signal representation of only two levels: an AND level and an OR level.

Ex: x= y and z

Where y = a or (b or c)

Z = c or d

Then x= (a or (b or c)) and (c or d)

Converting into a single statement removing the intermediate variables is called flattening.

Factoring: It is the process of adding intermediate terms to a description to add structure tot it. It is there verse process of flattening.

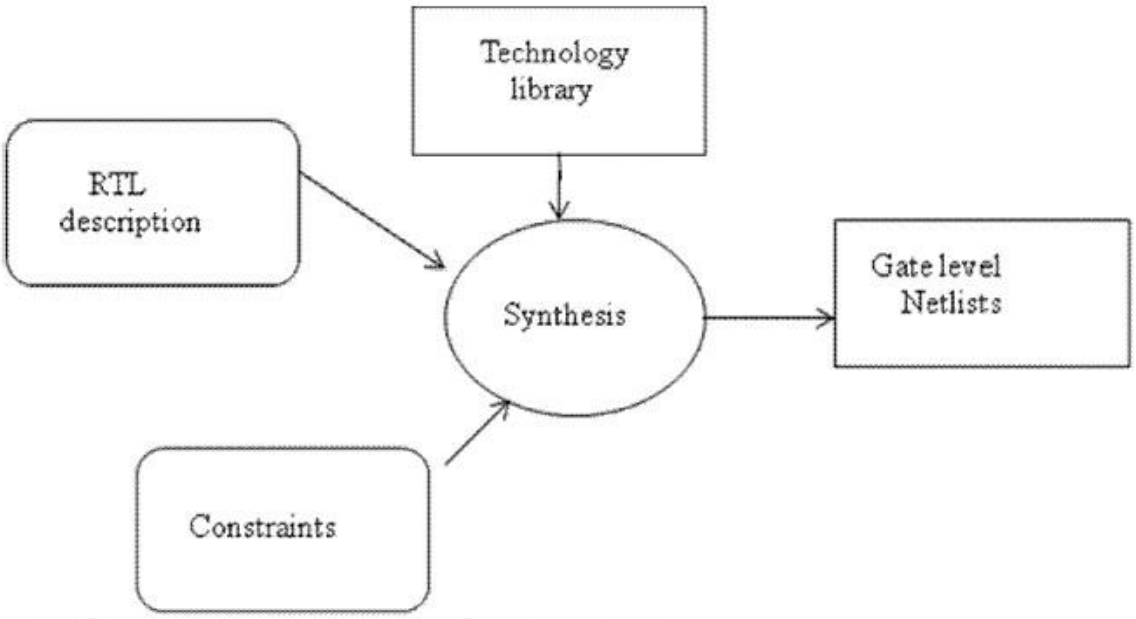
Mapping to gates:

The logical and timing information from a technology library is taken by the mapping process to convert optimized Boolean description to a net list. There are number of net lists which are functionally same but differ in area and speed.

CIRCUIT SYNTHESIS AND DESIGNFLOW:

Circuit synthesis provides a path between VHDL and a netlist, analogous to the c compiler which provider path between C code and machine langeg. The circuit synthesis methodology is needed to check if the model being written is compatible for synthesis or not. The RTL descriptions are converted to the gate lee el netlist by the current synthesis tools available. Interconnected gate level macro cells make up the gate level netlist. The technology libraries consist the models for gate level cells. These nelilsts are optimized for area, speed, testability and so on.

The synthesis process is shown in the following figure:

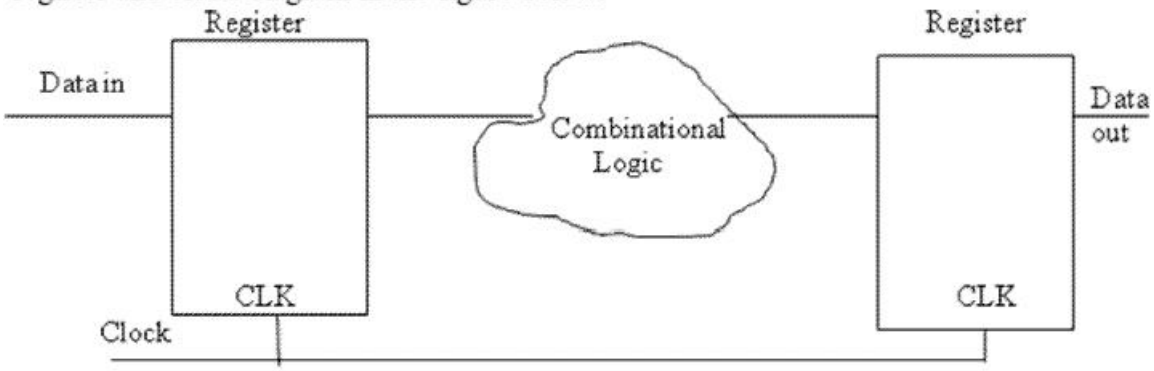


This process has the following as inputs:

1. RTL VHDL description
2. Circuit constraints
3. Attributes for the design and
4. Technology Library

1. RTL VHDL description:

A register transfer level description is characterized by a style that specifies all of the registers in a design and the combinational logic between them. This is shown by the register and cloud diagram in the figure below:

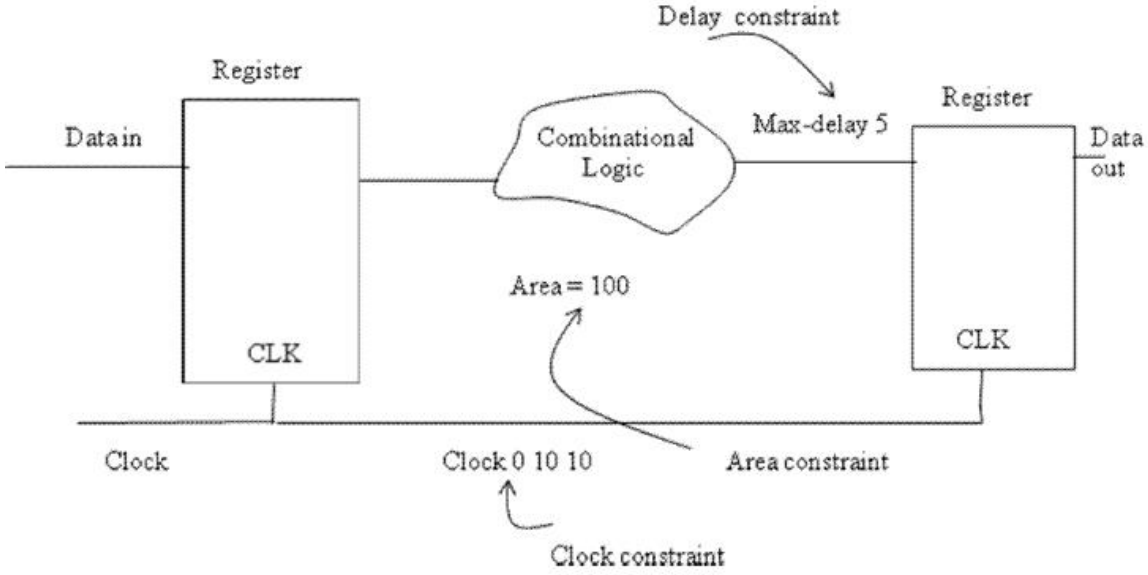


Register and Cloud diagram

The registers are described through component instantiation or inference; RTL descriptions are used for synchronous designs and describe the clock by clock behavior of the design.

2. Constraints:

Constraints are used to control the output of the optimization and mapping process. They provide goals that the optimization and mapping processes try to meet and control the structural implementation of the design. Constraints include area, timing, power and testability constraints. The most common are timing constraints. The block diagram is shown below:



There are different types of constraints as explained below:

(i) Timing constraints:

- These are used to specify maximum delays for particular pathes in a design.
- A typical delay constraint is shown here:

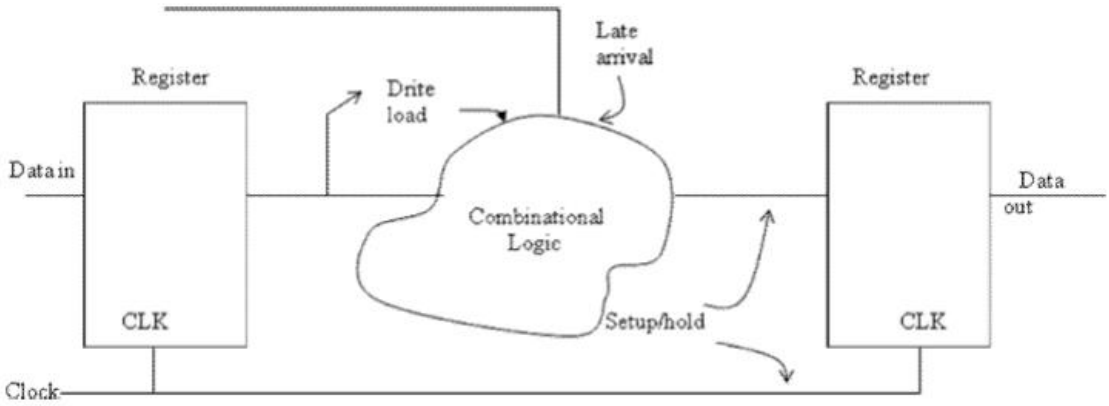
  Set-attribute-port data-out-name, required-time-value 25
- This constraint specifies that the maximum delay for signal data-out should be less than or equal to 25 library units.

(ii) Clock Constraints:

- We add a required time-constraint to every flip-flop input with the value of clock cycle.
- Another method is to add a clock constraint to the design.
- An example clock constraint is set – attribute – port clk- name clock- cycle – value 25

3. Attributes:

These are used to specify the design environment. For example, loading the outpu devices hoe to drive, the drive capability of devices driving the design and timing of inpu signals are specified by the attributes. A cloud diagram is shown in figure below:



Load:

The name of loads that can be driven within a particular time is determined by the drive capability specified by each output. Each input can have a load value specified, that determines how much it will show a particular driver.

The load attribute specifies how much capacitive load exists on a particular output signals.

Ex: Set- attribute-port x bus-name input-load- value 5

Drive:

The drive attribute specifies the resistance of the driver, which controls how much current it can source. An example of a drive specification is:

Set-attribute-port y bus- name output- drive- value 2.7

The attribute specifies that signal y bus has 2.7 library units of drive capability.

Arrival Time:

Setting the arrival time on a particular node specifies to the static timing analyzer when a particular signal will occur at a node.

Late arriving signals drive inputs current block at a later time, but the results of the current block still must meet its own timing constraints on its outputs.

4. **Technology Library:**

   Technology Libraries hold all the information necessary for a synthesis tool to create a netlist for a design based on the desired logical behavior, and constraints on the design.

These libraries contain all the information that allows the synthesis process to make the correct choice and build a design. They contain not only the logical functions of an ASIC cell, but the area o the cell, the input to output timing of the cell, any constraints on fanout of the cell, and the timing checks that are required for the cell.

SIMULATION:

A Simulation is used to predict and verify the performance of a given circuit. They can be divided into following categories:

1. Transistor-level or circuit-level Simulation
2. Static timing analysis or timing Simulation
3. Gate level simulating
4. Switch level Simulation
5. Behavioural Simulation
6. Functional Simulation.

1. Circuit-level Simulation:

It is the most accurate technique. These Simulations operate at the circuit level. It is complex and time consuming. Here the electrical behavior the various parts of circuit is to be implemented inn silicon. Circuit analysis programs are typified by SPICE program. Simulation time is proportional to N^m.

Simulation time α N^m

N. number of non linear devices in the circuit

m= varies between 1 to 2

Timing Simulation:

Once a behavioral or functional Simulation predicts that a system works correctly, the next step is to check the timing performance. At this point a system is partitioned into ASICS and a timing Simulation is performed for each ASIC separately. Timing analysis in a static manner computing delay times for each path is called static timing analysis. The path with the longest delay is called critical path.

The structure of the timing tools ensures that the run times are strictly linearly related to the number of devices and nodes being simulated.

Logic level Simulation (or) gate-level Simulation:

Here the performance is assessed in terms of logic levels with no or little timing information. They can simulate large section of layout at one time.

In a gate level Simulation a logic gate or logic cell is located as a black ox modeled by a function hose variables are the input signal.

These Simulations use primitive models such as NOT, AND, OR, NAND and NOR gates. They can be operated in unit delay mode or timing parameters may be assigned based on prior circuit Simulation, and known circuit parasitic.

Timing is normally specified in terms of inertial delay and load dependent delay for the appropriate edge transition, as follows:

$$T_{gate} = T_{inteinsic} + C_{load} \times T_{load}$$

(delay of gate)	(Intrinsic Delay of Gate)	(actual Load in Some units) i.E PF	(delay per Load)

Logic Simulations with such timing information are accurate for Cmos logic configurations.

Switch level Simulation:

This type of Simulation models transistors as switches – on or off. It can provide more accurate timing predictions than gate-level Simulation, but without the ability to use logic cell delays.

They merge logic-Simulation technique with some circuit Simulation techniques by modeling transistors as switches.

Behavioral Simulation:

Large pieces of system modeled as block boxes with inputs and outputs crating an imaginary Simulation model of the system are called behavioral Simulation.

Functional Simulation:

Functional Simulation ignores timing and includes unit-delay Simulation, which sets delays to a fixed value 7.31.

In the processes in ASIC design flow there are two kinds of Simulation

a. Post layout timing Simulation
b. Post synthesis Simulation.

a. Post layout timing Simulation:

After place and route process has completed the designer uses post route gate level Simulation to verify its results.

This Simulation combines the netlist used for place and route process into a Simulation that checks both functionality and timing of the design. The designer can run the Simulation and generate accurate output waveforms that show whether the device is working properly and if the timing is being met.

Same test vectors and the same Simulation as for the RTL Simulation can be used for post route gate level Simulation if properly structured.

b. Post synthesis Simulation:

This Simulation is carried out after the synthesis has been done.

This is as effective as the post layout timing Simulation.

Applications of Simulation:

Simulation is applied in two major application areas:

- System validation
- Fault Simulation

1. System validation:

In system validation we to deduce that the system conforms to a specification. The system is represented by a model at different points in the design process. Based on the design cycle, a system can have models at the chip, register and circuit level. One uses Simulation to validate that these models conform to that specification.

2. Fault Simulation:

The other application of Simulation is fault Simulation. Here, faults are injected into the system model, and the system is simulated to observe the response. To establish that a test detects a fault this can be done.

This type of Simulation can also be used to create fault dictionaries, which relate output signal states to faults, therefore allowing one to diagnose which fault occurred.

Efficiency of a Simulation:

When simulating models for large systems Simulation efficiency is very important. For logic Simulation, Simulation efficiency (E) is defined as

$$E = \frac{\text{Real Logic time}}{\text{Host CPUtime}}$$

'Real logic time' is the actual time required to complete an activity sequence in real logic circuit. 'Host CPU time' is the time required to simulate the same activity sequence using a logic Simulation running on a host CPU.

Advantages of Simulation:

1. It permits evaluation of a part before it is available.
2. It permits substitution of a range of components into signal paths to study the effects of different timing.
3. The ability to see inside ICs, so that the designer can examine the contents of the registers and flip flops during simulation is one of the benefits.
4. The Simulation can also be used to detect glitches caused by hazards.
5. The Simulation can be designed to detect instances where two or more tristate devices are simultaneously active.

LAYOUT

Graphical Entry Layout:

Special textual entry editors are used for small subsystem layout. These tools have been replaced by highly interactive method of producing layouts for which monochrome or color graphics terminals are used. The layout is built up and displayed during the design process.

These systems are menu driven and possible actions at various stages of the design are displayed on the screen along with details of the current layout.

Two of the earliest available graphical entry packages were KTC and PLAN. PLAN makes use of low- cost monochrome as well as colour graphical terminals.

RC Calculation from layout:

It can be described in terms of sheet resistance and wiring capacitance.

The following table gives the sheet resistance values for different layers:

	Layer	Sheet resistance
1	Metal	0.03
2	Diffusion	10 → 50
3	Silicide	2 → 4
4	Polysilicon	15 → 100

For any layer, Area capacitance is

$$C = \frac{\epsilon_0 \epsilon_i \; ns.A}{D} \; farads$$

D= Thickness of Silicon dioxide

A= area of plates

ε_{ins}= Relative permittivity of SiO_2 =4

$\varepsilon_0 = 8.85 \times 10^{-14}$ p/cm.

DESIGN CAPTURE AND DESIGN VERIFICATION TOOLS:

Place and Route tool:

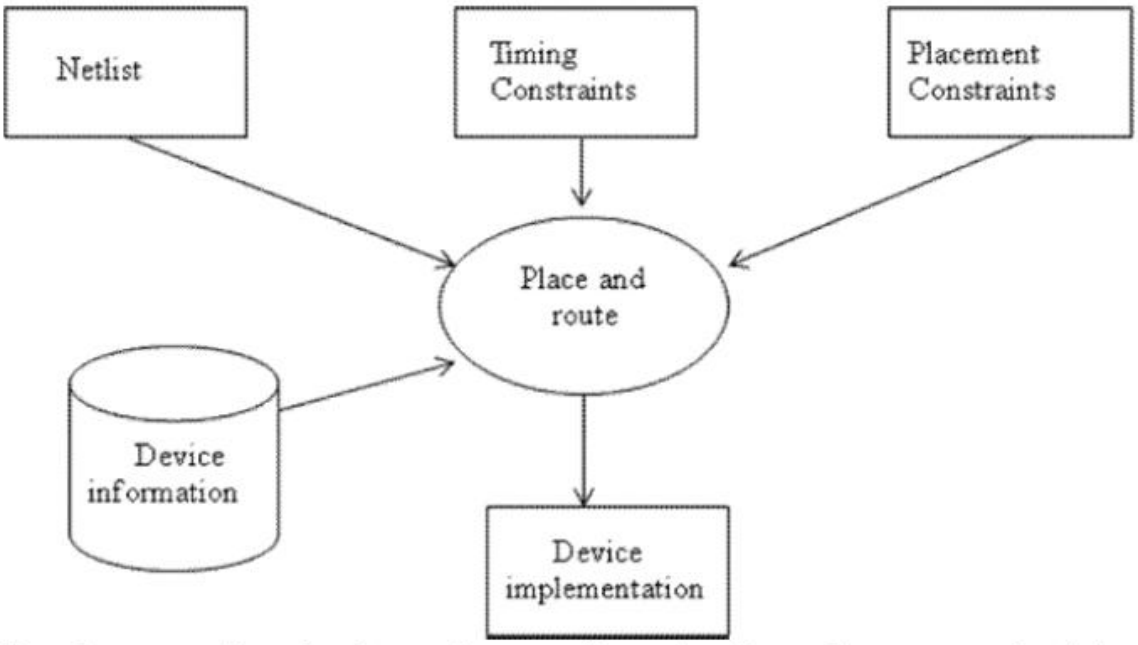
These tools are used to take the design netlist and implement the design in the target technology device. They place each primitive from netlist into an appropriate location on the target device and then route signals between the primitives to connect the devices according to the netlist. These tools are very architecture and device dependent. Place and route tools for FPGA and ASIC devices can be obtained from the respective vendors. The data flow diagram of the place and route tools is shown below:

Inputs to the place and route tools are the netlist in EDIF or another netlist format, and possibly timing constraints.

Another input is the timing constraints which informs the tool about the signals which have critical timing associated with them, and to route these nets, into most timing efficient manner. These constraints tell the place and route tool to place the primitive in los proximity to one another and to sue the fastest routing.

Placement of large parts of the design is known as floor planning. It allows the user to pick location on the chip for large blocks of design so that routing wires are as shot as possible.

After all the cells are placed and routed, the output of the place and route tools consists of dataflow that can be used to implement the chip.



The other output from the place and route software is a file used to generate the timing file. This file describes the actual timing of the programmed FPGA deice or the final ASIC deice.

CAD Tools:

Building a chip requires a variety of CAD Tolls. CAD algorithms are careful chosen because of the complexity.

Most CAD design data consist of structural description graphs, geometric objects etc. Most programming tools appear to the user to be manipulating test files. Layout representation requires geometric information,. Such as rectangles and triangles. Circuit information is stored as enlists or component lists. The CAD includes.

1. Physical design layout and editing capabilities which may be through textual or graphical entry of information.
2. Structure generation/system composition consist of capabilities of design layout software of point 1.
3. Physical verification: These tools include design rule checking (DRC), circuit extractor ratio run and other static checks and to plot out and/or display for visual checking.
4. Behavior verification: Simulation at various levels will be required to check out the design before one embarks on the expense of turning out the design in silicon.

Design Rule Checkers:

All possible errors must be eliminated before mask making proceeds.

- Check for errors at all stages of design
- The different stages are:

- Pencil and paper stage of the design of leaf cells.
- At the leaf cell level/once the layout is complete.
- At the subsystem level check that butting together and wiring up of leaf cells is correctly done.
- Once the system layout has been completed.

Circuit Extractors:

If the design information exists in the form of physical layout then a circuit extractor program is required that wile interpret the physical layout in circuit term. It is fed directly into simulator that computer may be used to interpret the finding of the extractor.

Design verification Prior to Fabrication:

- It is not sufficient to have good design tools, there must be complemented by equally effective verification software capable of handling large systems and with reasonable computing power requirements.
- The nature of tools required will depends on the way in which an integrated circuit design is represented in computer.
- There are two approaches:

(I) Mask level layout languages such as CIF, which are well suited to physical layout description but not for capturing the design intent.

(II) Circuit description languages where the primitives are circuit elements such as transistors, wires and nodes.

TEST PRINCIPLES:

The responsibilities for the various levels of testing and testing methodology lie on the designer. Small imperfections in starting material, processing steps or in photo masking may result in bridged connection or missing features. The aim of a test procedure is to determine the good die to be used in the end system.

Testing a die can occur at

- ✓ Wafer level
- ✓ Packaged chip level
- ✓ Board level
- ✓ System level
- ✓ Field

If we detect a malfunctioning at a earlier stage the cost of manufacturing is minimum.

Some of the test principles are:

(i) Scan – Based Techniques

(ii) Boundary scan test

(iii)Built-in-self test techniques.

(i) Scan – Based Techniques

If more accessible logic nodes with use of additional primary input lines and multiplexers, controllability and observability can be enhanced. But this can be costly, so an alternative s to use scans registers with both shift and parallel load capabilities. The scan design technique is a structured approach to design sequential circuits for testability. The storage cells in registers are used s observation points, control points, or both. The testing of a sequential circuit is reduced to the problem of testing a combinational circuit.

In general a sequential circuit consists of a combinational circuit and some storage elements. The storage elements are connected to from a long serial shift register, so called scan path, by using multiplexers.

In test mode, the scan-in-signal is clocked into the scan path and the output of the last stage of latch is scanned out. The testing sequence as follows:

Step 1: Set the mode to test and let latches accept data from scan-in-input

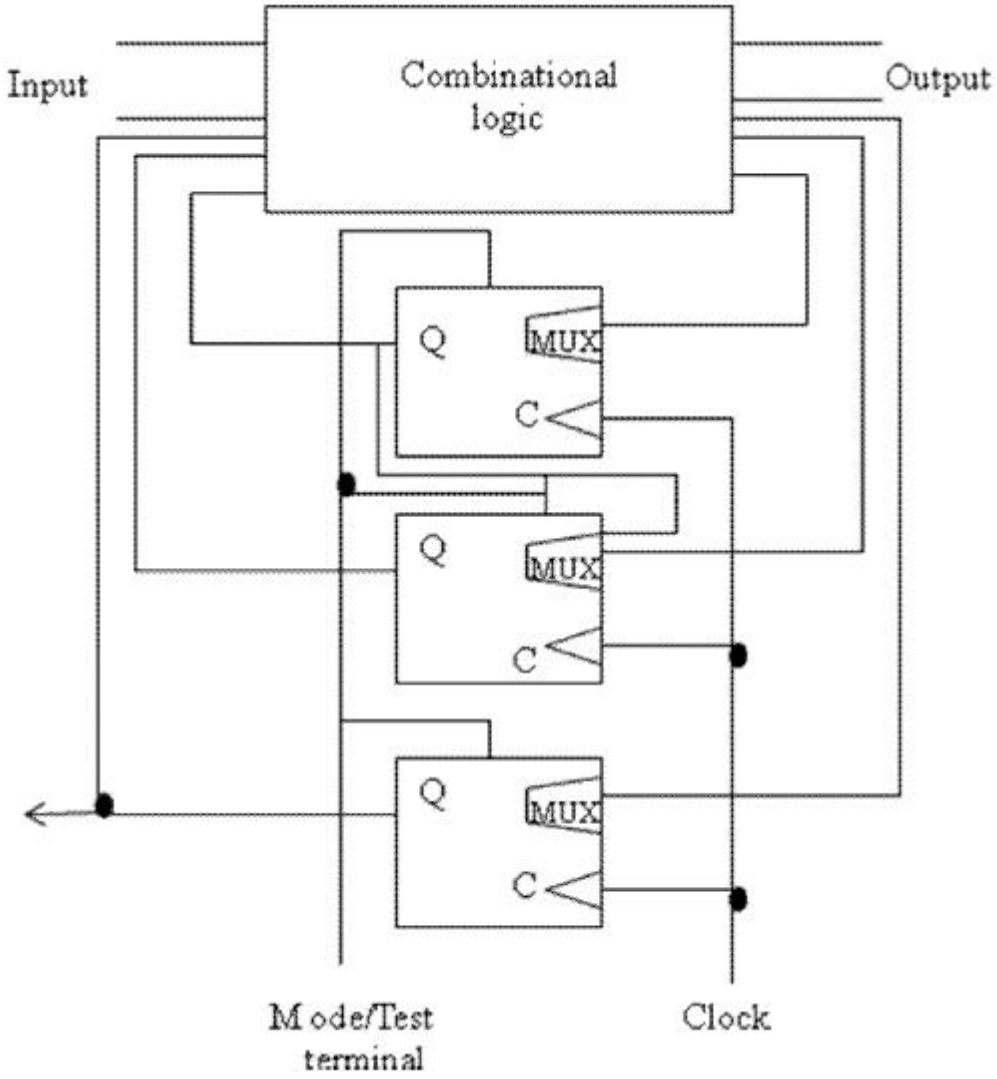
Step 2: Verify the scan path by shifting in and out the test data.

Step 3: Scan in the desired state vector into the shift register.

Step 4: Apply the test pattern to the primary input pins.

Step 5: Set the mode to normal and observe the primary outputs of the circuit after sufficient time for propagation.

Step 6: Assert the circuit clock for one machine cycle to capture the outputs of the combinational logic into the registers.

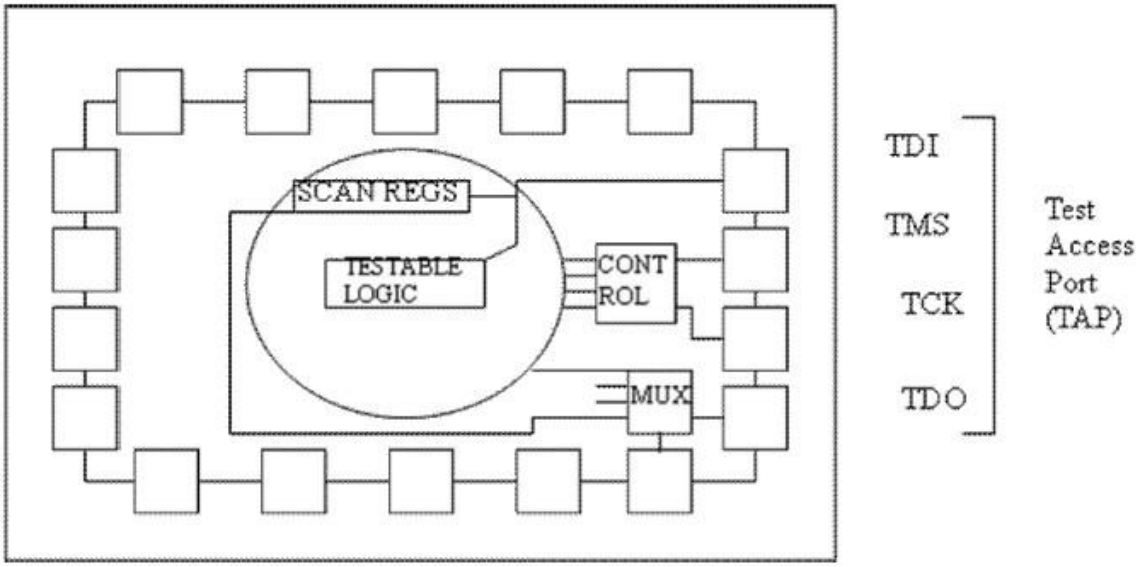


Step 7: Return to test made, scan-out the contents of the registers, and at the same time scan-in next pattern.

Step 8: Repeat steps 3-7 until all test patterns are applied.

(ii) Boundary Scan Test:

The problems associated with the testing of boards carrying VLSI circuits and /or surface-mounted devices are resolved by technique involving scan path and self-testing. It consists of placing a scan path (shift register) cell adjacent to each component pin and to inter connect the cells so as to form a chain around the border of circuit. The technique is explained in diagram below:



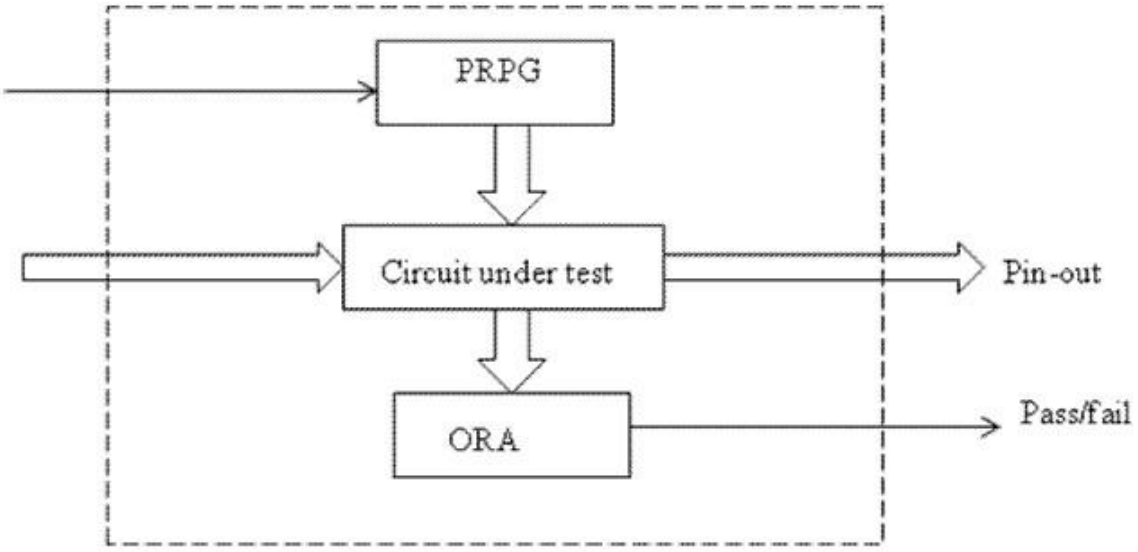
The boundary scan path is provided serial input and output pads and appropriate clock pole which make it possible to

- Test the inter connections between various chips
- Deliver test data to the chips on the board of self testing
- Test the chips themselves with internal self-test facilities.

(iii)Built in self test techniques:

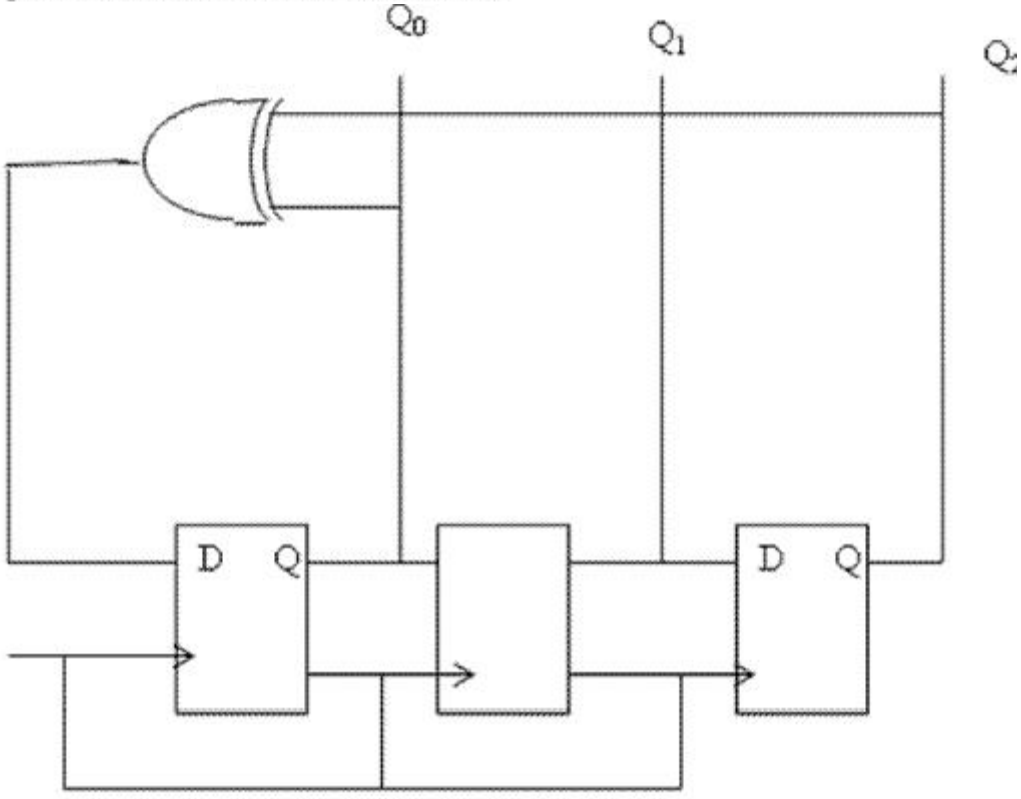
Here parts of the circuit are used to test the circuit itself online BIST is used to perform the test offline. The essential modules include.

- Pseudo random pattern generator (PRPG).
- Output Response analyzer (ORA).



The roles of there two modules are illustrated above. The implementation of both PRPG and DRA can be done with linear feedback shift registers (LFs Rs)

The patterns are to generated to test the circuit. This is done by pseudo random pattern generator. Its circuit is shown below:



A Pseudo-random sequence generator using LFSR

Design for Test (DFT):

There techniques are increasingly gaining momentum as they provide measures to comprehensively test the manufactured device for quality and coverage.

Merging testability features early in the cycle was the final solution, creating thename Design for Test.

Types of DFT:

The main DFT techniques available today are;

(a) Scan insertion – involves various design issues to be resolved

- Involves replacing all flip flops
- Most prevalent is the multiplexed flip flop.

(b) Memory BIST insertion:

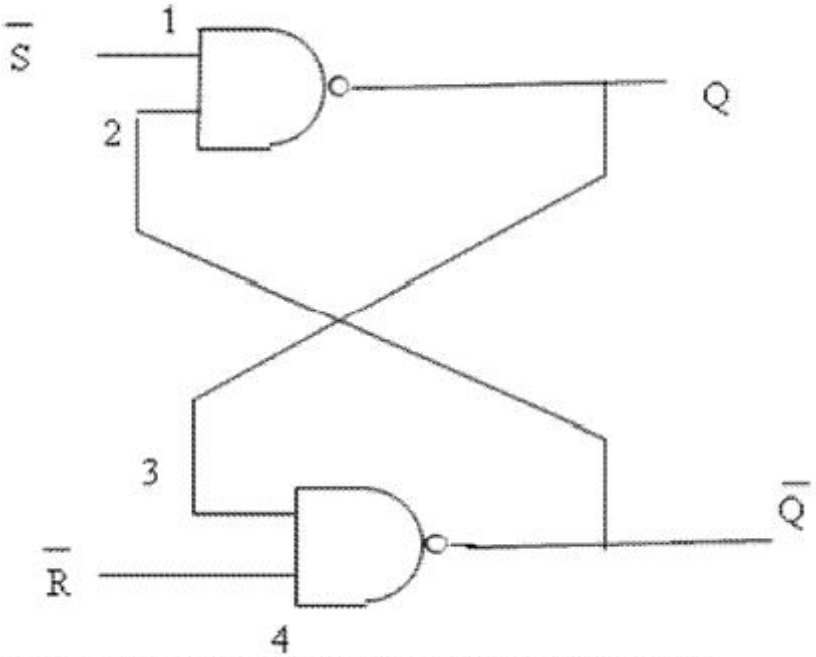
- It is comprised of a controller logic that uses various algorithms to generate input patterns that are used to exercise the memory elements of a design.
- It is automatically generated, based on the size and configuration of memory element.

(c) Boundary Scan DFT:

- Used for testing without unplugging the clip from the board.
- The TAG controller an surrounding logic also may be generated.

Procedure to test sequential logic:

All the sequential circuits exhibit a memory, property of remembering or taking into account the previous conditions. Consider the basic sequential circuit as shown below:



The sequential logic testing effects of memory.

The delay in the fed back path is non-existent, which is a case when the circuit propagation delays contribute to the necessary delay elements. The inert is applied to $\overline{S}, \overline{R}$.

The state of 'Q' for a "good machine (GM)" is tabulated first by the remaining columns and then for the' faulty machine' (FM) for a stud at 1 fault on each of the four inputs (1, 2, 3, 4).

The machine matches the good response in the first table with the SA 1 fault on the line 2 and so this particular test sequence will not detect a SA 1 fault on line 2.

The vectors are applied exactly in the reverse order in the second table. Each of the "?" indicated that the Q will retain the value it had earlier to the application of the test vector for that row. Again if the latches are reset i.e, Q=0 prior to application of the test sequence, then the SA1 fault on line 2 may not be detected.

Inputs		Output Q	
		GM	Input SAS 1
S	R		1 2 3 4
0	0	1	0 1 1 1
0	1	1	0 1 1 1
1	0	0	0 0 0 1
1	1	0	0 0 1 1

Inputs	Output Q