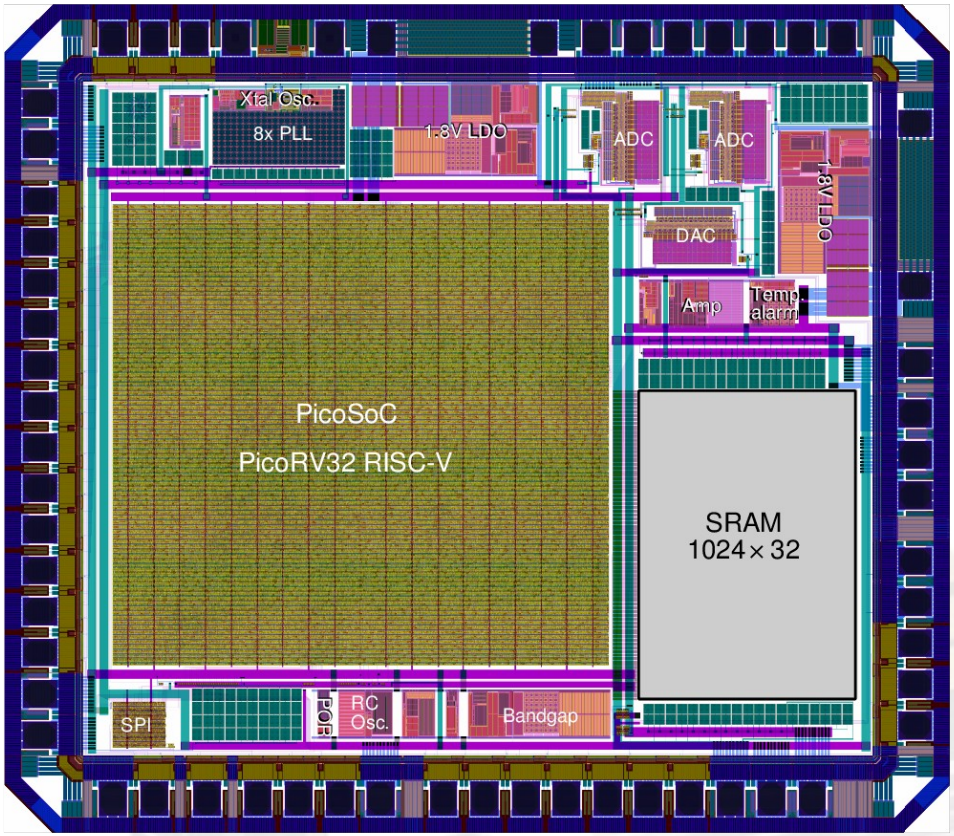
VLSI Design

Prof. Tassadaq Hussain Cheema

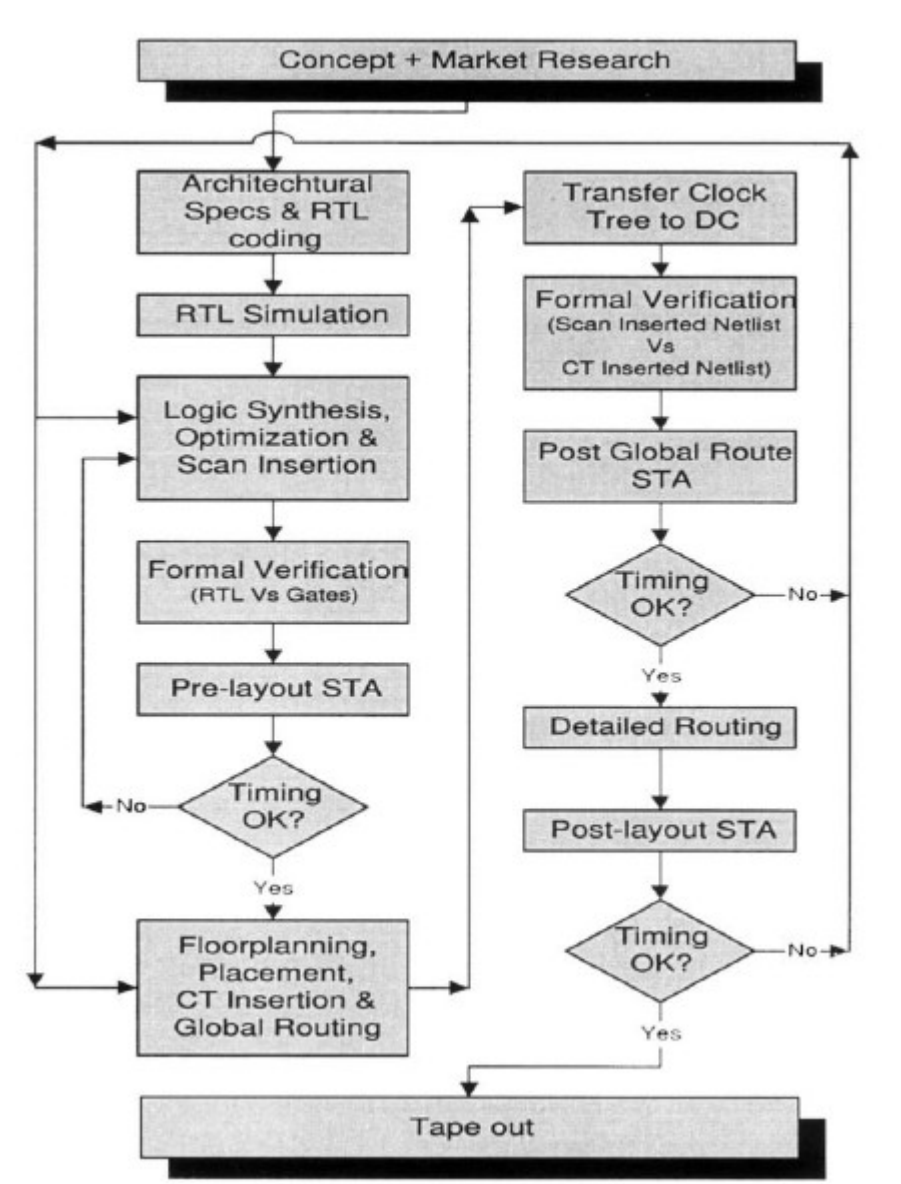
VLSI Design Compiler

Digital Synthesis Flow

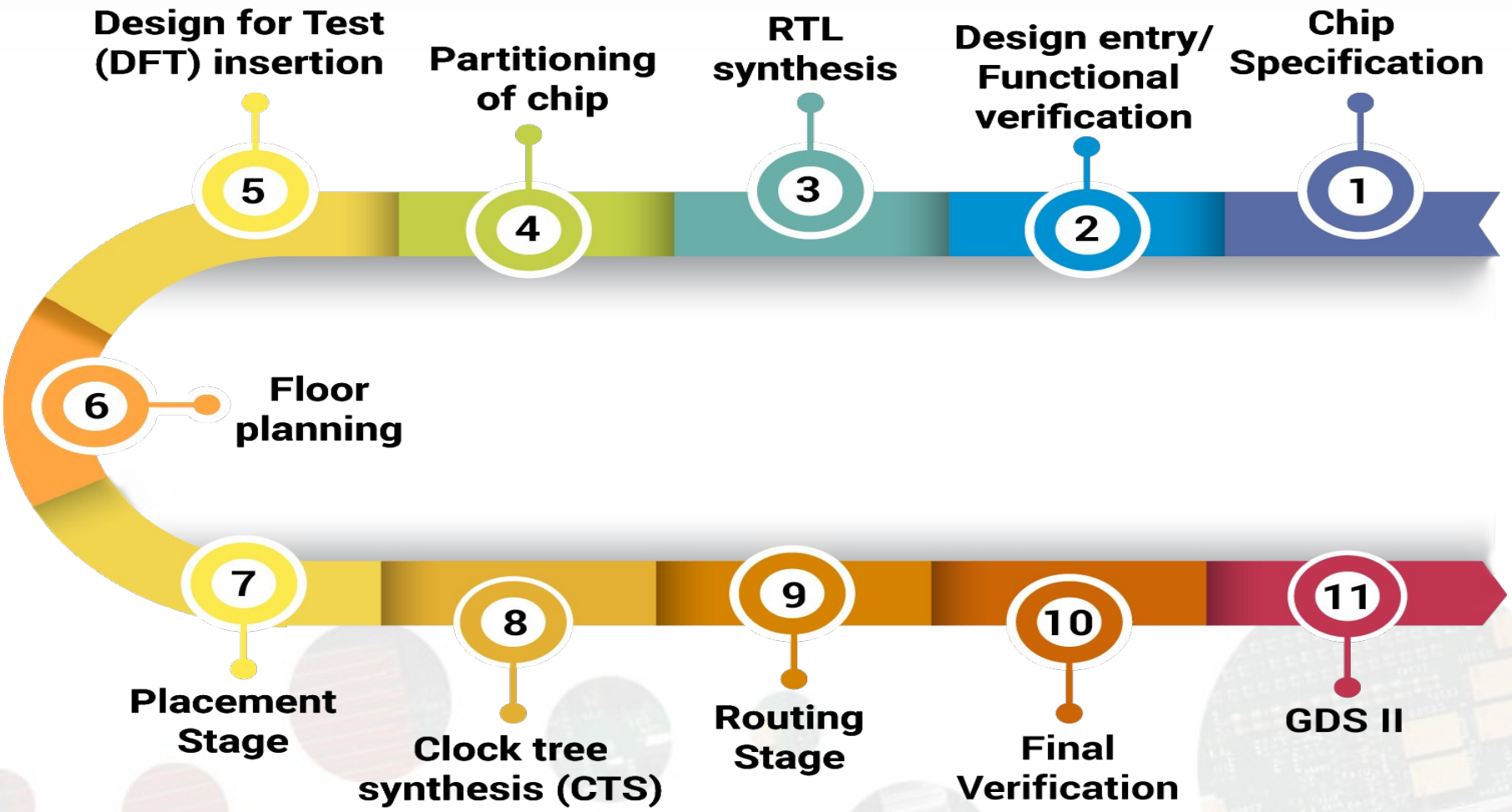
A digital synthesis flow is a set of tools and methods used to turn a circuit design written in a high-level behavioral language like verilog or VHDL into a physical circuit, which can either be configuration code for an FPGA target like a Xilinx or Altera chip, or a layout in a specific fabrication process technology, that would become part of a fabricated circuit chip.



ASIC DESIGN METHODOLOGY



RTL to GDS-II FLOW

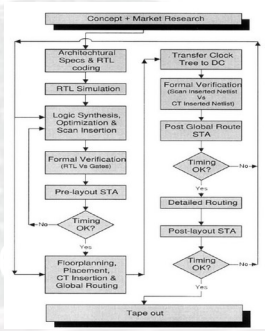


Chip Specification

This is the stage at which the engineer defines features, microarchitecture, functionalities (hardware/software interface), specifications (Time, Area, Power, Speed) with design guidelines of ASIC.

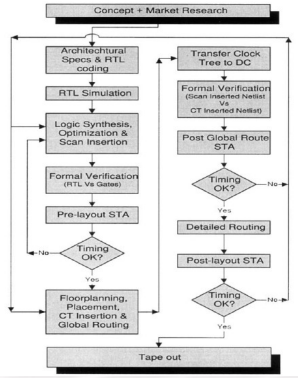
Specification and RTL Coding

- Chip design commences with the conception of an idea dictated by the market.
- These ideas are then translated into architectural and electrical specifications.
- The architectural specifications define the functionality and partitioning of the chip into several manageable blocks, while the electrical specifications define the relationship between the blocks in terms of timing information.



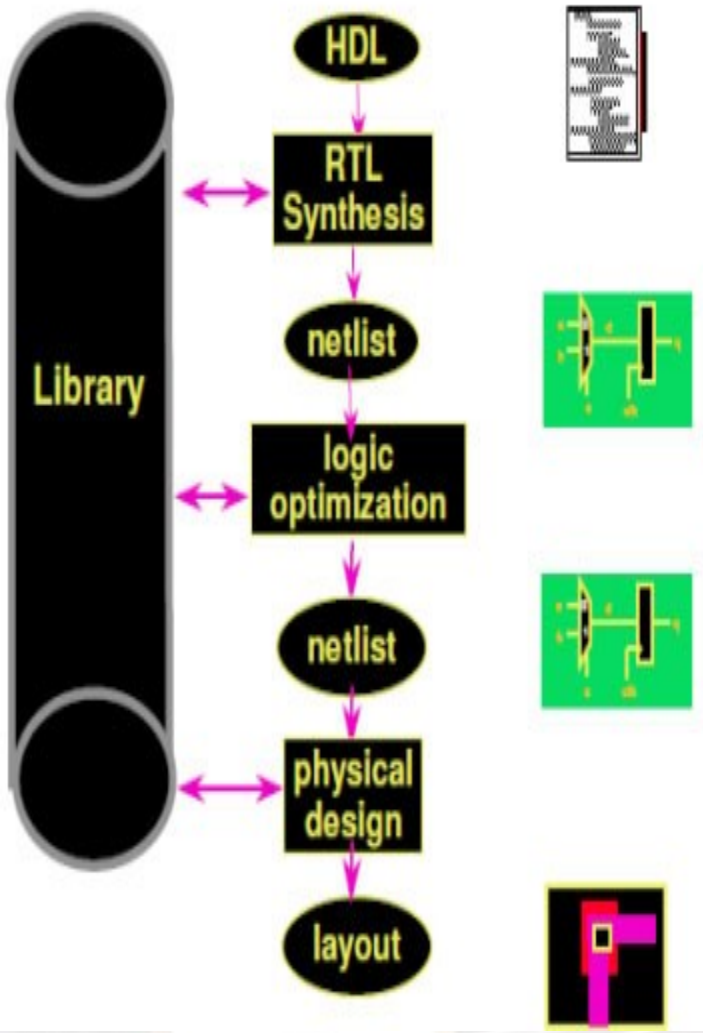
Specification and RTL Coding

- Switch Level
- Gate Level
- RTL Level
- High Level



RTL block synthesis

Translate the RTL code into a gate-level netlis using a logical synthesis tool that meets required timing constraints.



Design Entry / Functional Verification

Functional verification confirms the functionality and logical behavior of the circuit by simulation on a design entry level. This is the stage where the design team and verification team come into the cycle where they generate RTL code using test-benches. This is known as behavioral simulation.

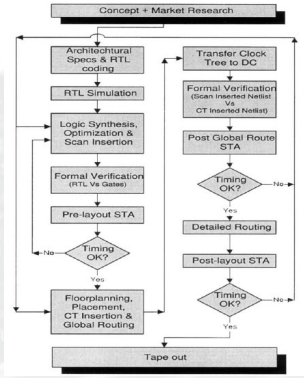
There are two types of simulation tools:

Functional simulation tools: After the testbench and design code, functional simulation verifies logical behavior and its implementation based on design entry.

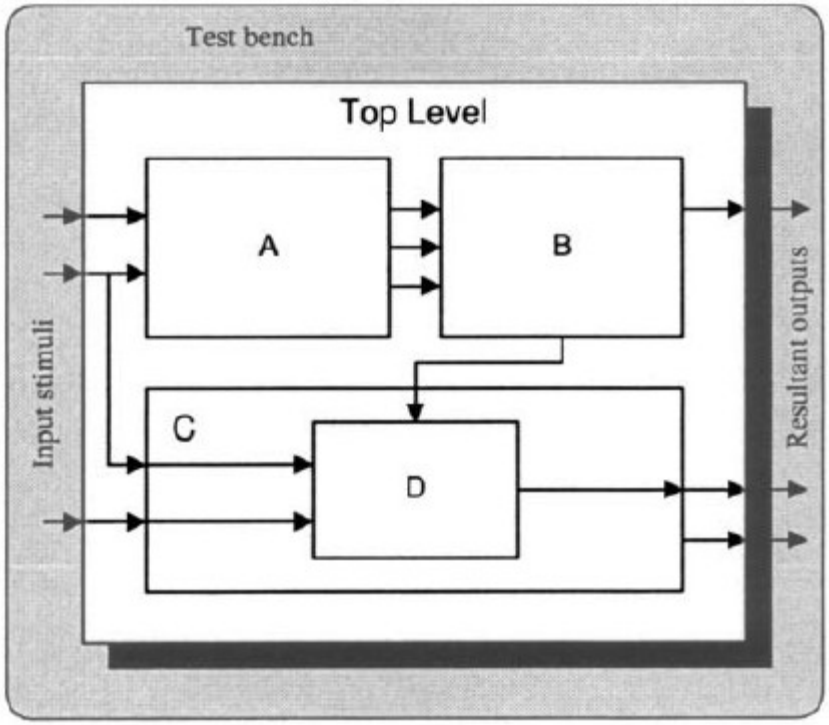
Timing simulation tools: Verifies that circuit design meets the timing requirements and confirms the design is free of circuit signal delays.

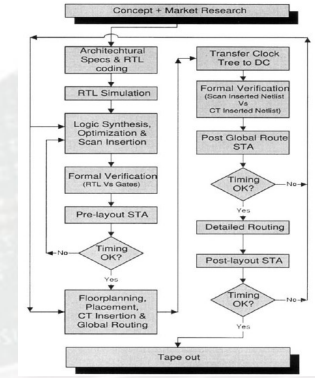
Dynamic Simulation

The next step is to check the functionality of the design by simulating the RTL code. All currently available simulators are capable of simulating the behavior level as well as RTL level coding styles. In addition, they are also used to simulate the mapped gate-level design.



Dynamic Simulation

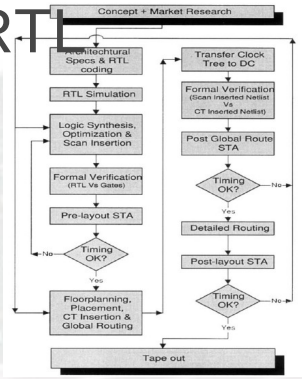


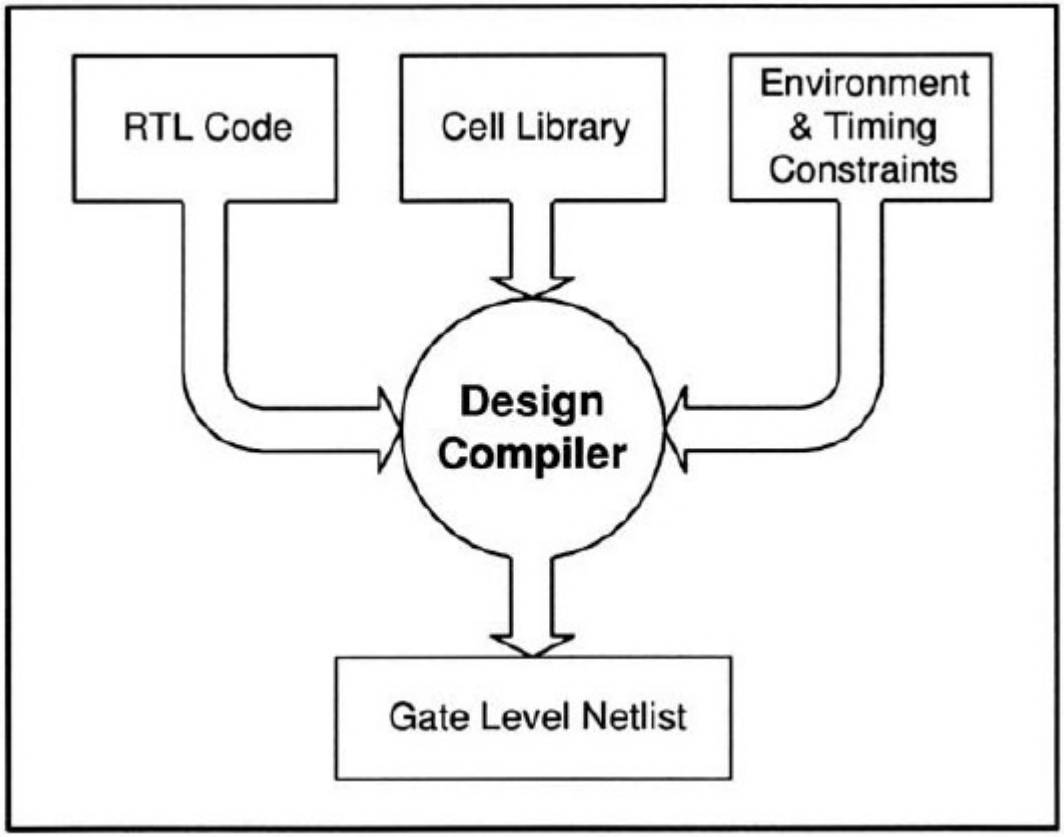


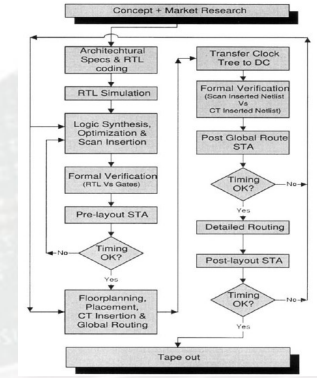
Test Bench

The purpose of the test bench is to provide necessary stimuli to the design. It is important to note that the coverage of the design is totally dependent on the number of tests performed and the quality of the test bench. This is the reason why a sound test bench is extremely critical to the design. During the simulation of the RTL, the component (or gate) timing is not considered.

Therefore, to minimize the difference between the RTL simulation and the synthesized gate-level simulation at a later stage, the delays are usually coded within the RTL source, usually for sequential elements.




RTL Code
Cell Library
Environment & Timing Constraints
Design Compiler
Gate Level Netlist


Concept + Market Research
Architechtural Specs & RTL coding
RTL Simulation
Logic Synthesis, Optimization & Scan Insertion
Formal Verification (RTL Vs Gatee)
Pre-layout STA
Timing OK?
No
Yes
Floorplanning, Placement, CT Insertion & Global Routing
Transfer Clock Tree to DC
Formal Verification (Scan Inserted Netlist Vs CT Inserted Netlist)
Post Global Route STA
Timing OK?
No
Yes
Detailed Routing
Post-layout STA
Timing OK?
No
Yes
Tape out

Chip Partitioning

Once the design structure is verified with the help of HLL programming languages like C++ or SystemC.

The ASIC is partitioned into multiple functional blocks (hierarchical modules), while keeping in mind ASIC's best performance, technical feasibility, and resource allocation in terms of area, power, cost and time.

Once all the functional blocks are implemented in the architectural document, the engineers need to brainstorm ASIC design partitioning by reusing IPs from previous projects and procuring them from other parties.

Design for Test (DFT) Insertion

ASIC design is complex enough at different stages of the design cycle. Telling the customers that the chips have fault when you are already at the production stage is embarrassing and disruptive. It's a situation that no engineering team wants to be in. In order to overcome this situation, design for test is introduced with a list of techniques:

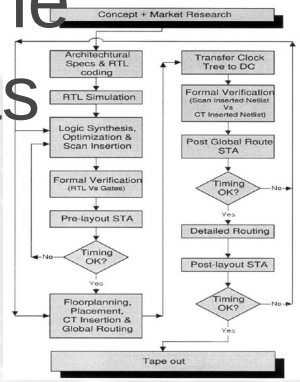
Scan path insertion: A methodology of linking all registers elements into one long shift register (scan path). This can help to check small parts of design instead of the whole design in one go.

Memory BIST (built-in Self-Test): In the lower technology node, chip memory requires lower area and fast access time. MBIST is a device which is used to check RAMs. It is a comprehensive solution to memory testing errors and self-repair proficiencies.

ATPG (automatic test pattern generation): ATPG is a method of creating test vectors / sequential input patterns to check the design for faults generated within various elements of a circuit.

Constraints, Synthesis and Scan Insertion

For a long time, the HDLs were used for logic verification. Designers would manually translate the HDL into schematics and draw the interconnections between the components to produce a gate-level netlist. With the advent of synthesis tools, this manual task has been rendered obsolete. The tool has taken over and performs the task of reducing the RTL to the gate-level netlist. This process is termed as synthesis.



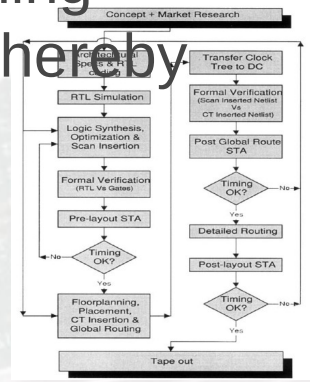
Constraints, Synthesis and Scan Insertion

Synthesizing a design is an iterative process and begins with defining **timing constraints** for each block of the design. **These timing constraints define the relationship of each signal with respect to the clock input for a particular block. In addition to the constraints, a file defining the synthesis environment is also needed.**

The environment file specifies the technology cell libraries

and other relevant information that DC uses during synthesis.

DC reads the RTL code of the design and using the timing constraints, synthesizes the code to structural level, thereby producing a mapped gate-level netlist.



Constraints, Synthesis and Scan Insertion

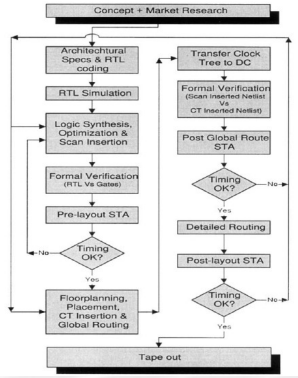
Usually, for small blocks of a design, DC's internal static timing analysis is used for reporting the timing information of the synthesized design. DC tries to optimize the design to meet the specified timing constraints.

Further steps may be necessary if timing requirements are not met.

Most designs today, incorporate design-for-test (DFT) logic to test their functionality, after the chip is fabricated. The DFT consists of logic and memory BIST (built-in-self-test), scan logic and Boundary Scan logic (JTAG) etc.

Formal Verification

Formal verification is the process of checking whether a design satisfies some requirements (properties). We are concerned with the formal verification of designs that may be specified hierarchically this is also consistent with how a human designer operates.



Static Timing Analysis using PrimeTime

This analysis allows the user to exhaustively analyze all critical paths of the design and express it in an orderly report.

Furthermore, the report can also contain other debugging information like the fanout or capacitive loading of each net.

Pre-Layout

The static timing is performed both for the pre and post-layout gate-level netlist.

In the pre-layout mode, PrimeTime uses the wire load models specified in the library to estimate the net delays.

PrimeTime specifies the relationship between the primary I/O signals and the clock.

If the timing for all critical paths is acceptable, then a constraints file may be written out from PrimeTime or DC for the purpose of forward annotation to the layout tool.

This constraint file specifies the timing between each group of logic that the layout tool uses, in order to perform the timing driven placement of cells.

Post Layout

In the post-layout mode, the actual extracted delays are back annotated to PrimeTime to provide realistic delay calculation. These delays consist of the net capacitances and interconnect RC delays.

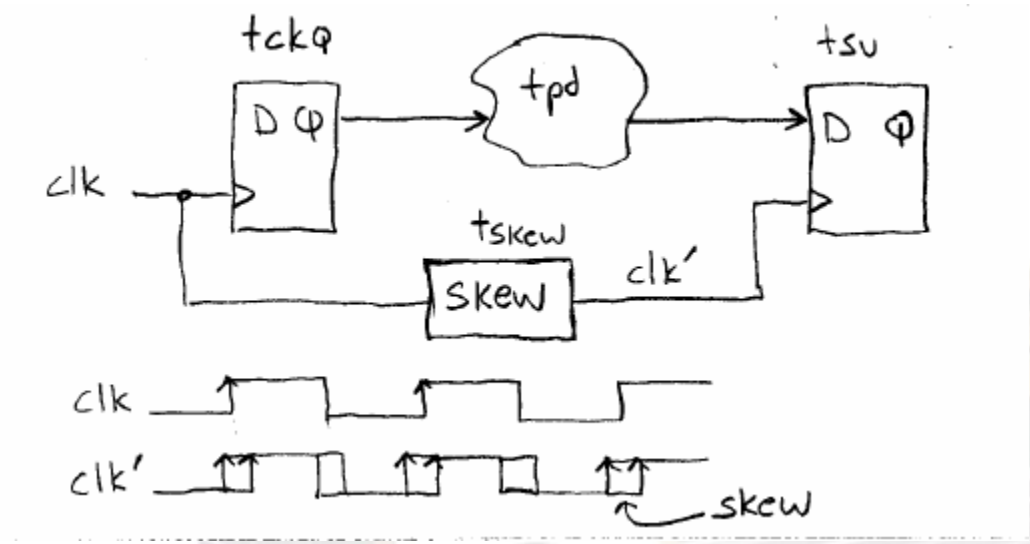
Placement, Routing and Verification

The quality of floorplan and placement is more critical than the actual routing. Optimal cell placement location, not only speeds up the final routing, but also produces superior results in terms of timing and reduced congestion.

The constraint file is used to perform timing driven placement. The timing driven placement method forces the layout tool to place the cells according to the criticality of the timing between the cells.

Clock Tree

After the placement of cells, the clock tree is inserted in the design by the layout tool. The clock tree insertion is optional and depends solely on the design and user's preference.



Clock Tree Insertion

At this stage an additional step is necessary to complete the clock tree insertion.

The layout tool inserted the clock tree in the design after the placement of cells. Therefore, the original netlist that was generated from DC (and fed to the layout tool), lacks the clock tree information (essentially the whole clock tree network, including buffers and nets). Therefore, the clock tree must be re-inserted in the original netlist and formally verified. Some layout tools provide direct interface to DC to perform this step.

Routing (Global Routing)

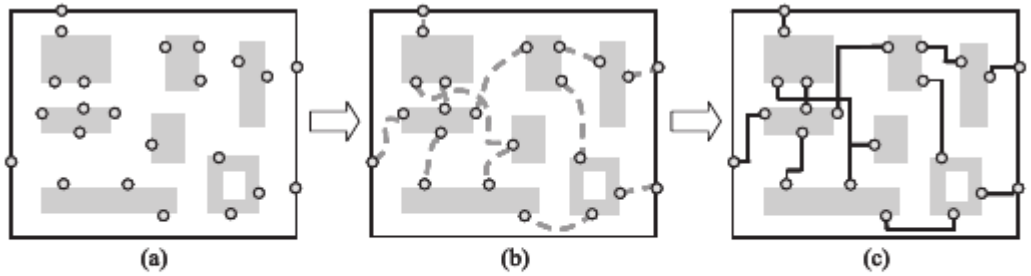
The layout tool generally performs routing in two phases – global routing and detailed routing.

After placement, the design is globally routed to determine the quality of placement, and to provide estimated delays approximating the real delay values of the post-routed (after detailed routing) design.

If the cell placement is not optimal, the global routing will take a longer time to complete, as compared to placing the cells.

Bad placement also affects the overall timing of the design. Therefore, to minimize the

number of synthesis-layout iterations and improve placement quality, the timing information is extracted from the layout, after the global routing phase



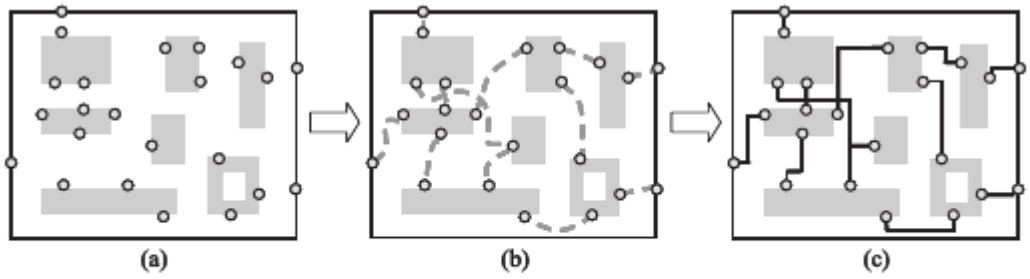
Routing problem: (a) A given placement result with fixed locations of blocks and pins. (b) Global routing. (c) Detailed routing.

Global routing first partitions the routing region into tiles and decides tile-to-tile paths for all nets while attempting to optimize some given objective function (e.g., total wirelength and circuit timing). Then, guided by the paths obtained in global routing, detailed routing assigns actual tracks and vias for nets.

Detailed Routing

Detailed routing is the final step that is performed by the layout tool. After

detailed route is complete, the real timing delays of the chip are extracted, and plugged into PrimeTime for analysis.



Routing problem: (a) A given placement result with fixed locations of blocks and pins. (b) Global routing. (c) Detailed routing.

Engineering Change Order

Many designers regard engineering change order (ECO) as the change required in the netlist at the very last stage of the ASIC design flow. For instance, ECO is performed when there is a hardware bug encountered in the design at the very last stage (say, after tape-out), and it is necessary to perform a metal mask change by re-routing a small portion of the design.

Floor Planning

After, DFT, the physical implementation process is to be followed. In physical design, the first step in RTL-to-GDSII design is floorplanning. It is the process of placing blocks in the chip. It includes: block placement, design portioning, pin placement, and power optimization.

Floorplan determines the size of the chip, places the gates and connects them with wires. While connecting, engineers take care of wire length, and functionality which will ensure signals will not interfere with nearby elements. In the end, simulate the final floor plan with post-layout verification process.

A good floorplanning exercise should come across and take care of the below points; otherwise, the life of IC and its cost will blow out:

- Minimize the total chip area
- Make routing phase easy (routable)
- Improve signal delays

Placement

Placement is the process of placing standard cells in row. A poor placement requires larger area and also degrades performance. Various factors, like the timing requirement, the net lengths and hence the connections of cells, power dissipation should be taken care. It removes timing violation.

Clock tree synthesis

Clock tree synthesis is a process of building the clock tree and meeting the defined timing, area and power requirements. It helps in providing the clock connection to the clock pin of a sequential element in the required time and area, with low power consumption.

Routing

Global Routing: Calculates estimated values for each net by the delays of fan-out of wire. Global routing is mainly divided into line routingand maze routing.

Detailed Routing: In detailed routing, the actual delays of wire is calculated by various optimization methods like timing optimization, clock tree synthesis, etc.

Final Verification

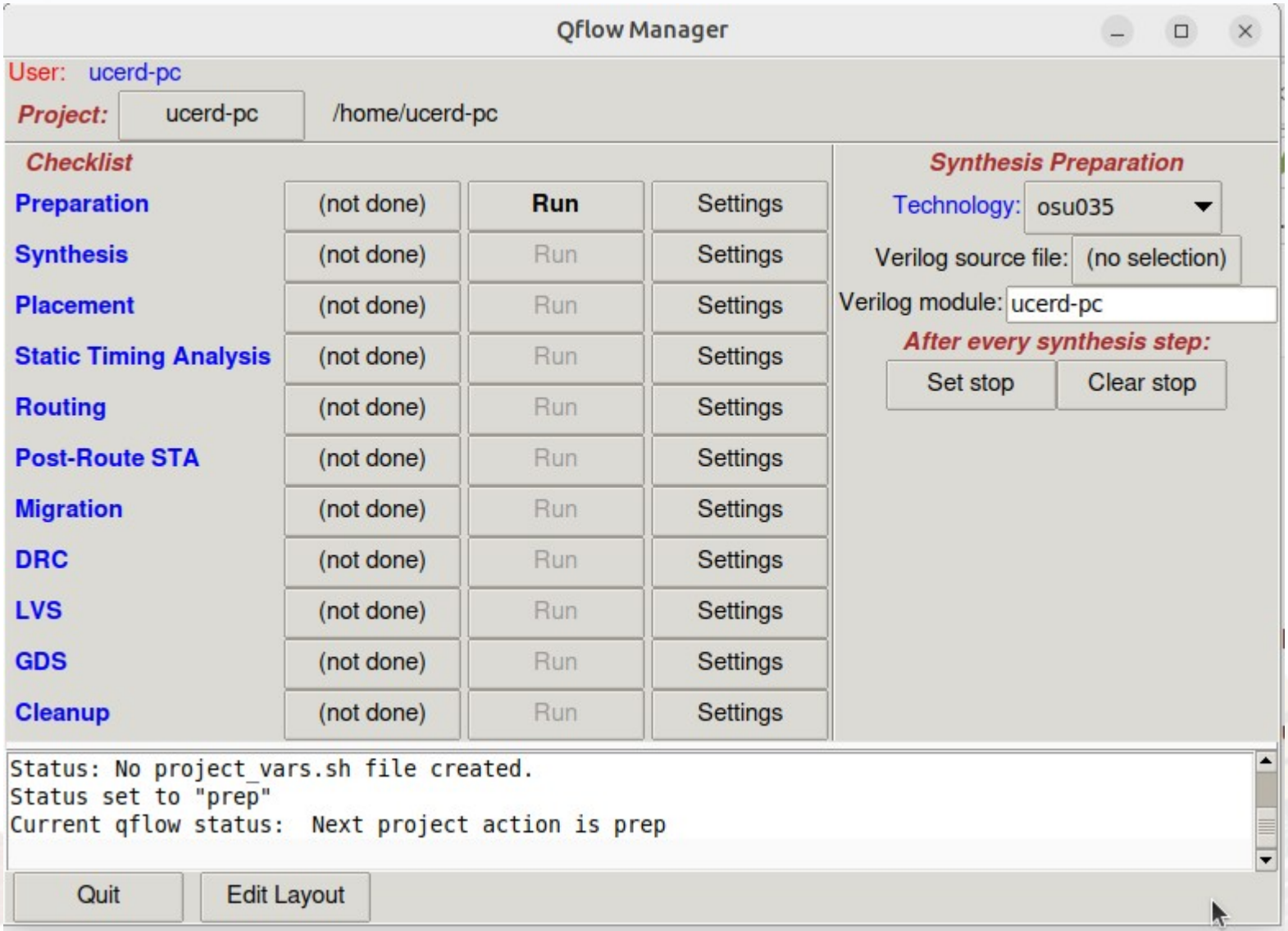
The following checks are followed to avoid any errors just before the tapeout:

Layout versus schematic(LVS) is a process of checking that the geometry/layout matches the schematic/netlist.

Design rule checks(DRC) is the process of checking that the geometry in the GDS file follows the rules given by the foundry.

Logical equivalence checks(LVC) is the process of equivalence check between pre and post design layout.

Q-Flow


Qflow Manager
User: ucerd-pc
Project: ucerd-pc
/home/ucerd-pc
Checklist
Preparation (not done) Run Settings
Synthesis (not done) Run Settings
Placement (not done) Run Settings
Static Timing Analysis (not done) Run Settings
Routing (not done) Run Settings
Post-Route STA (not done) Run Settings
Migration (not done) Run Settings
DRC (not done) Run Settings
LVS (not done) Run Settings
GDS (not done) Run Settings
Cleanup (not done) Run Settings
Synthesis Preparation
Technology: osu035
Verilog source file: (no selection)
Verilog module: ucerd-pc
After every synthesis step:
Set stop
Clear stop
Status: No project_vars.sh file created.
Status set to "prep"
Current qflow status: Next project action is prep
Quit
Edit Layout

Qflow is a toolchain that takes input in the form of a Verilog file and converts it into the physical layout which is ready for fabrication.

The Logic Synthesis (by the yosys tool) converts the design into a gate-level netlist based on the osu18 library (open-source 180nm technology node.osu libraries developed by Oklahoma State University)

The OSU18 library contains LEF formats file which contains information about macros, standard cells and timing information, and technology requirements for the Magic tool.

The synthesis performed optimization like mux tree optimization or D-Flip flop optimization on the design and mapped the design with standard cell. output is in the form of blif format.

Placement uses Graywolf tool: The blif netlist and .cel2 file are input for the Graywolf tool. .cel2 file contains the information about the location and order of pins. Graywolf tool uses the min-cut algorithm for the placement of standard cells.

The placement stage generates a DEF file which contains the information about how all the standard cells are placed in the layout.

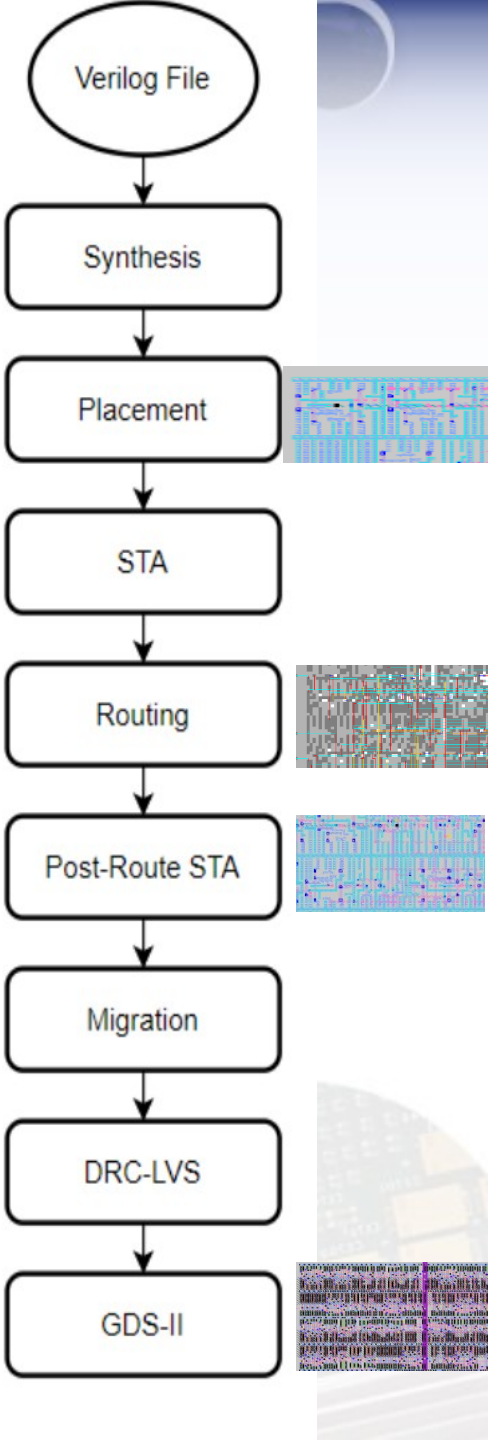
STA on the design is done with the help of the Vesta tool. It checks for timing requirements for all possible paths and checks for the setup and holds violations.

Routing is done with the help of qrouter. Qrouter converts the DEF file into annotated DEF file which is input for the Magic tool.

Post-STA checks that after performing routing is there any violation occurred or not. it checks the condition of Setup and Hold violation of all paths in the design.

Migration creates the layout and abstract view and extracts a new netlist from the layout. And check for DRC and LVS violations.

DRC checks that the final layout follows all the rules made by libraries like spaces between components, minimum feature size, minimum gate width, the minimum separation between two devices, etc. **LVS** checks that the physical design is the same as the gate-level netlist that is no change in functionality of the design is assured after completion of all steps.LVS is performed with the help of the netgen tool.



QFlow Result

Number of Wires:	289
Number of Wire bits:	257
Number of Public Wires:	289
Number of Public Wires bits:	0
Number of Memories:	0
Number of Memory bits:	0
Number of Processes:	0
Number of Cells:	249
AND2X2	23
AOI21X1	10
DFFPOSX1	46
BUFX2	6
INVX1	13
MUX2X1	28
NAND2X1	13
NAND3X1	19
NOR3X1	5
NOR2X1	16
OAI21X1	66
OR2X2	2
XNOR2X1	2

Parameter	Qflow
Cells	249
Area(Micron sq.)	11900
Max. operating Freq.(MHz)	748.765
No. of Metal Layers used	6

https://www.youtube.com/watch?v=5fm_YmcoTiU&ab_channel=AbhishekPandya