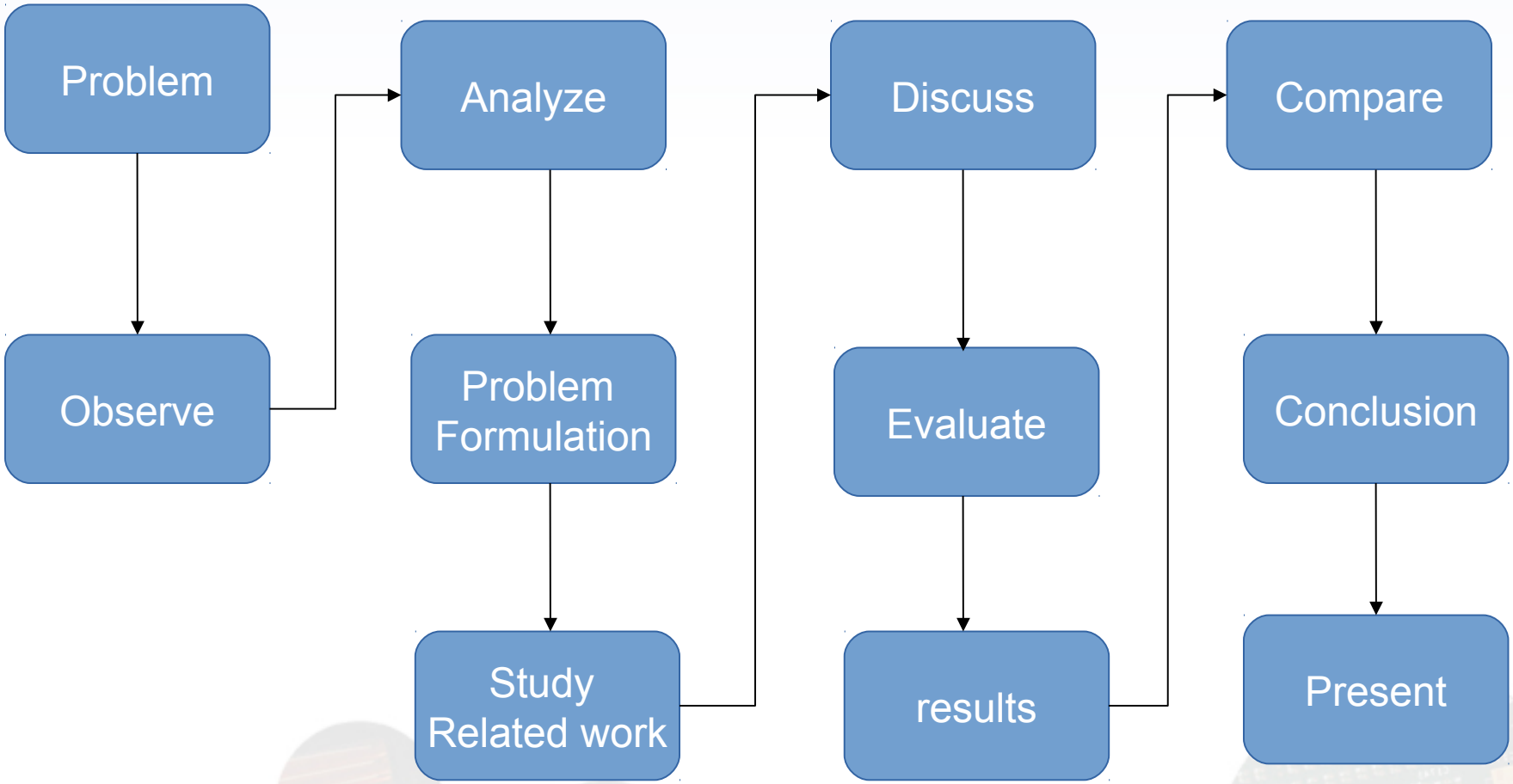
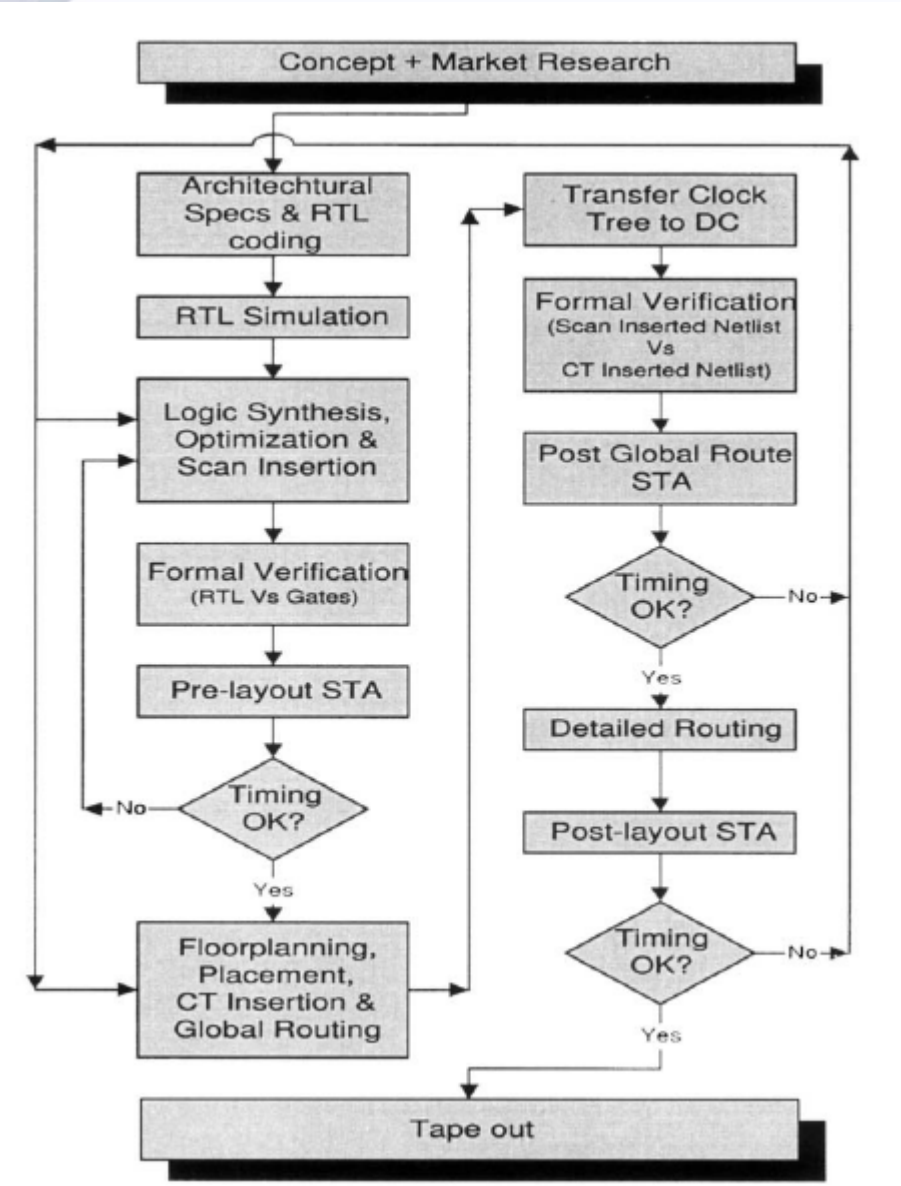
Tassadaq Hussain

VLSI Design

Basic Steps to Solve a Problem

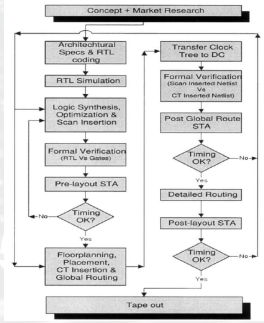


ASIC DESIGN METHODOLOGY



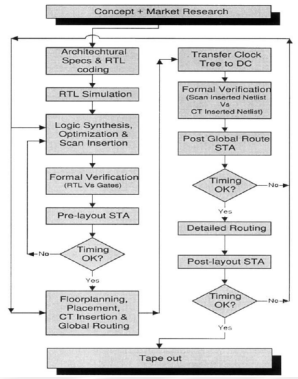
Specification and RTL Coding

- 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.



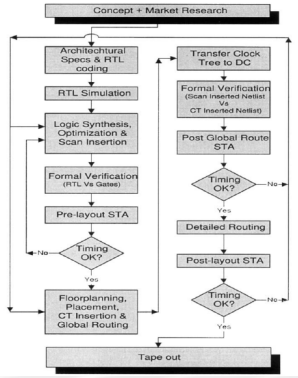
Specification and RTL Coding

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

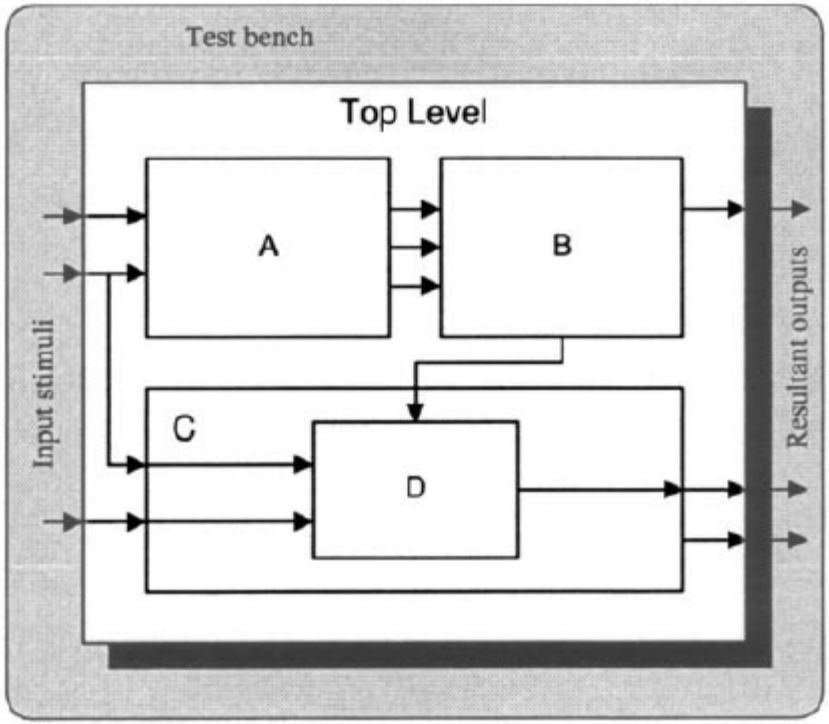


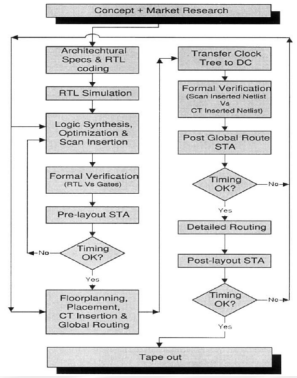
Dynamic Simulation

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



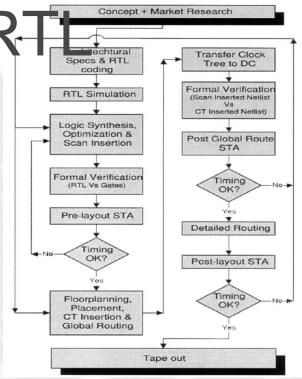


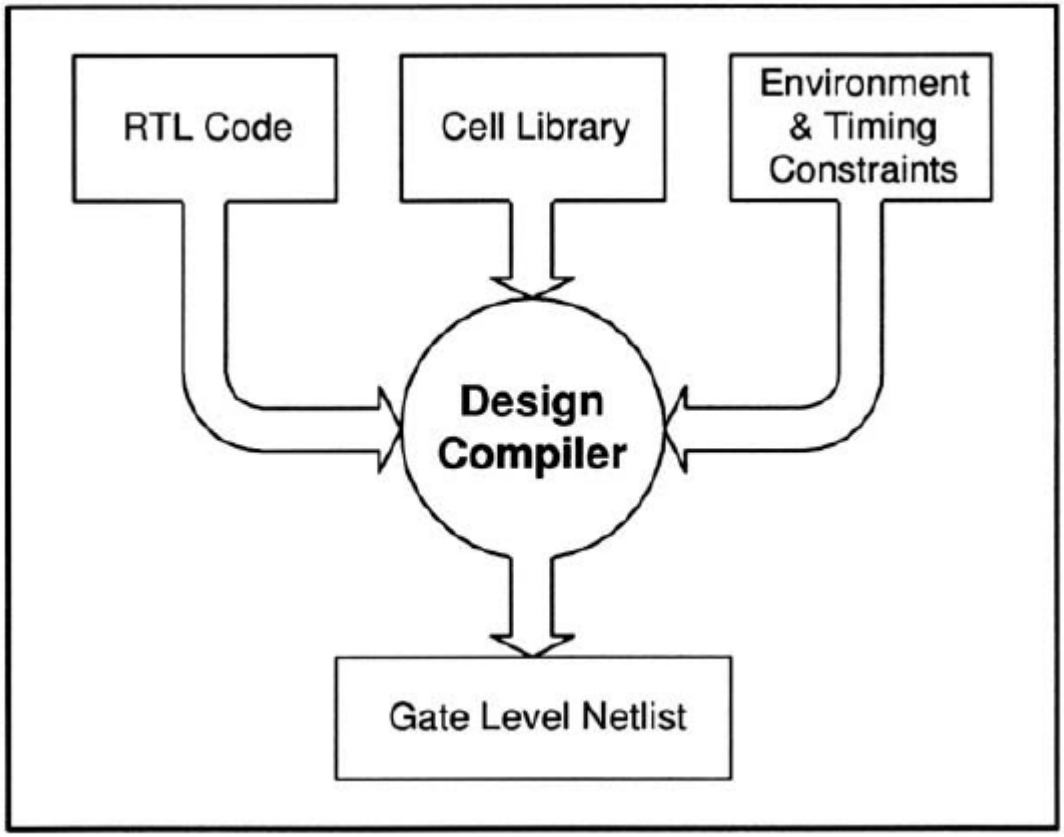
RIPHAH INTERNATIONAL UNIVERSITY

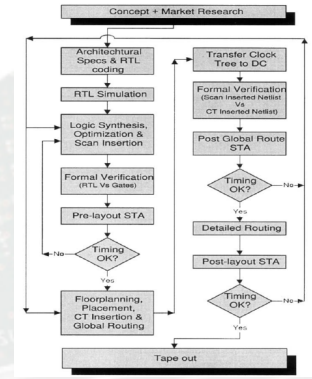
Test Bench

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.



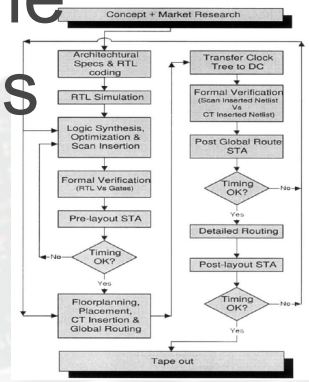




RIPHAH INTERNATIONAL UNIVERSITY

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.

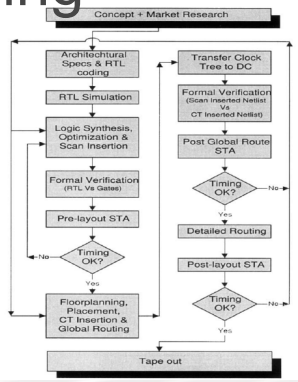


Constraints, Synthesis and Scan Insertion

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.



RIPHAH INTERNATIONAL UNIVERSITY

Constraints, Synthesis and Scan Insertion

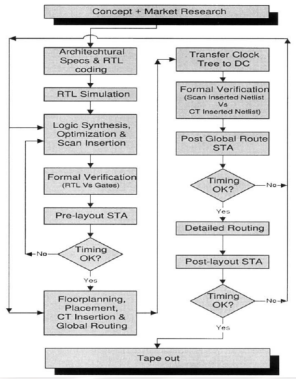
Usually, for small blocks of a design, DC's internal static timing analysis is used for reporting the timing information of the synthesized design. DC tries to optimize the design to meet the specified timing constraints.

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.



RIPHAH INTERNATIONAL UNIVERSITY

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.

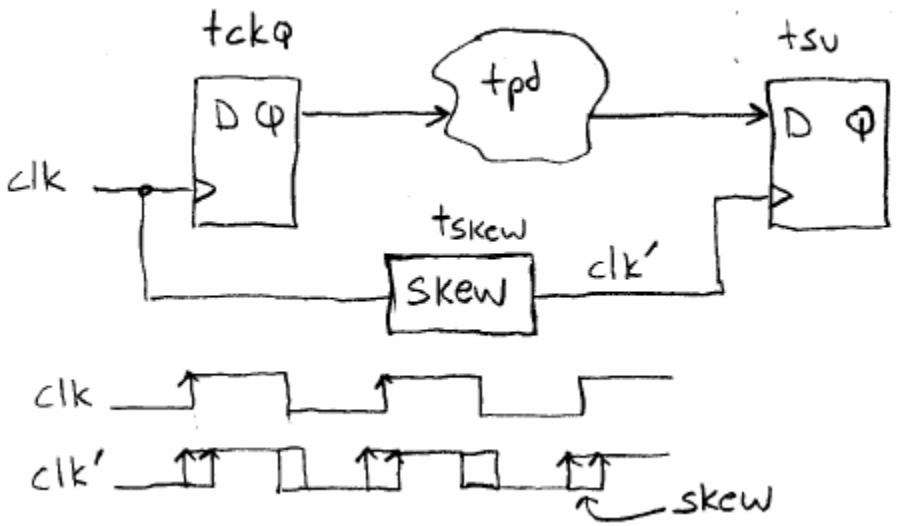
Placement, Routing and Verification

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)

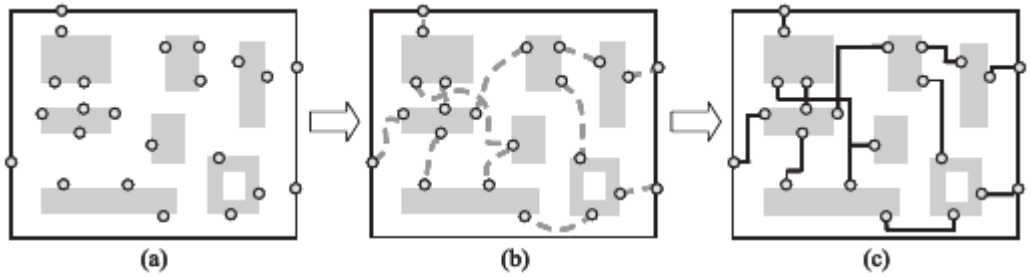
The layout tool generally performs routing in two phases – global routing and detailed routing.

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



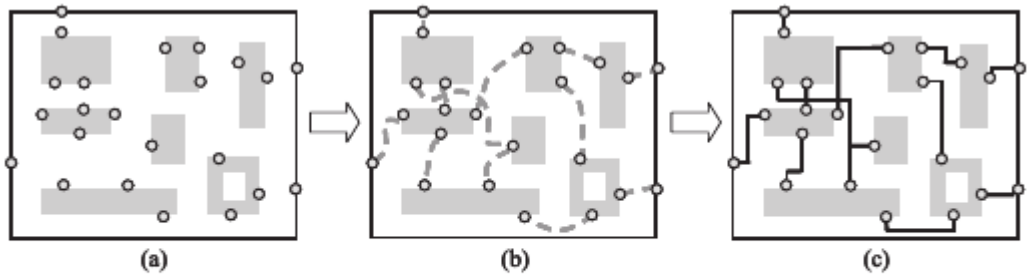
Routing problem: (a) A given placement result with fixed locations of blocks and pins. (b) Global routing. (c) Detailed routing.

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.

Calculate Performance of Graphical System

A graphical system processes 30 images per second. Each image has resolution of 640x CMS.

The system operates at 100 MHz of clock frequency. In a clock cycle the system performs 30 arithmetic operations and generates a single output.