ASIC DESIGN FLOW

ASICS

- What are ASICS?

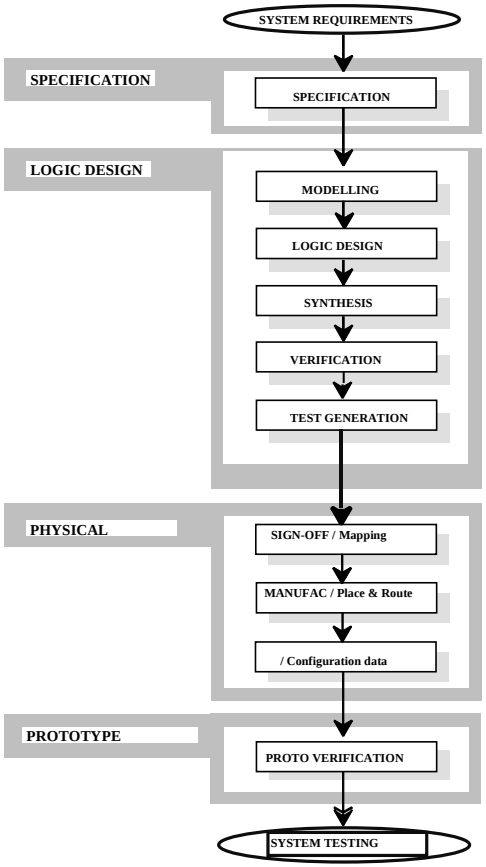
Any IC other than a general purpose IC which contain the functionality of thousands of gates is usually called an ASIC(Application Specific Integrated Circuit). ASICs are designed to fit a certain application.

An ASIC is a digital or mixed-signal circuit designed to meet specifications set by a specific project.

ASIC Project

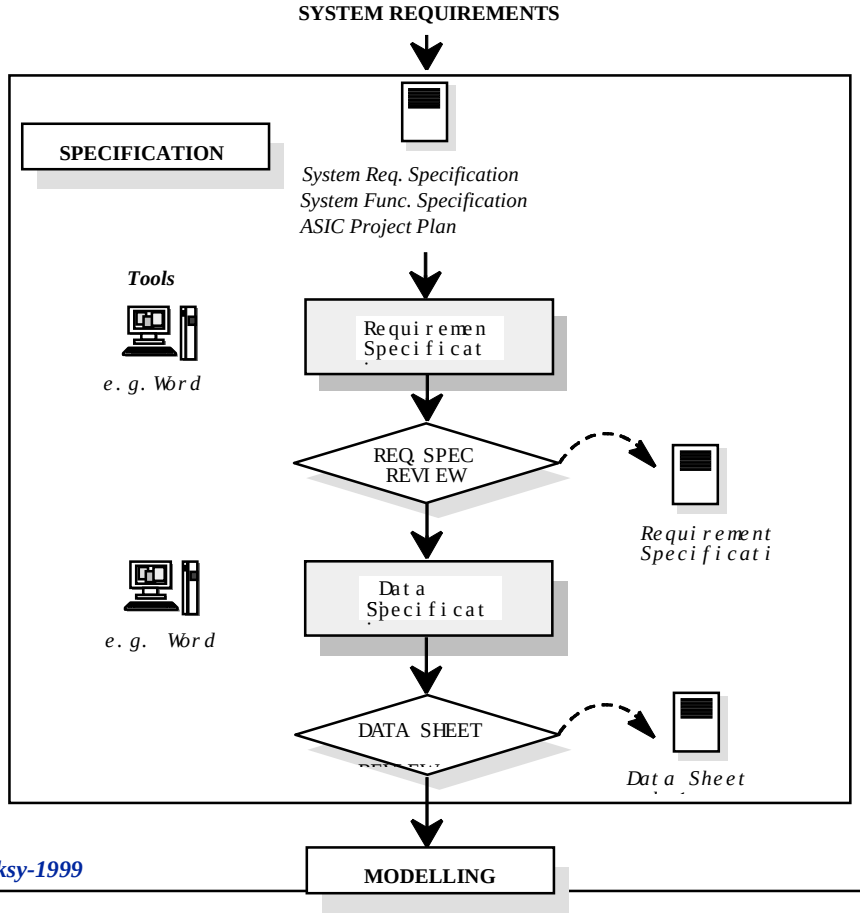
- ASIC design team (Project leader, designers for different tasks)
- Information share with closely related projects/design teams (software, analog HW design, system design) - Documentation!
- ASIC project is a part of bigger project - Scheduling is important!
- Design flow must be defined and approved

ASIC Design Flow



ASIC Specification

- The goal is to specify the functional requirements for the design and define the external interfaces to the related designs.



ASIC Modeling

- The goal is to build a simulatable (behavioral) VHDL model corresponding to the specification.
- The function of the model is verified by using a VHDL test bench
- Architecture design
- Model validation ok

  -> Logic design

Logic design

- The goal is to write a synthezisable VHDL description of the design
- Design rules (Naming, vendor independence, Use only IEEE standard types, comments, …).
- Reusing
- Synchronous design !!
- Design partitioning (into RTL blocks)

Naming rules

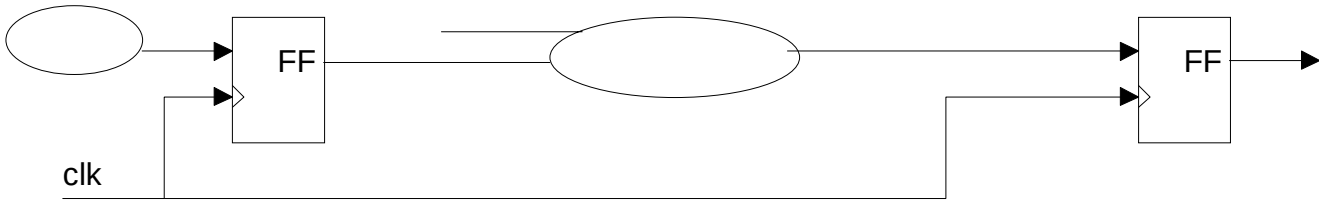
- Use meaningful names for signals, ports, functions and parameters. For example, do not use ra for a RAM address bus. Instead, use ram_addr or RamAddr (capital letters can also be used).
- If your design uses several parameters, use short but descriptive names.
- Use the name clk for the clock signal. If there is more than one clock in the design, use clk as the prefix for all clock signals (for example clk4m, clk8m).
- Use the same name throughout the hierarchy for all clock signals that are driven from the same source.
- For active low signals, end the signal name with an underscore followed by a lowercase character x.
- Use the name reset for reset signals. For active low resets use the name reset_x.
- For multibit buses, use (y downto x) ordering of bits.

Naming rules

- Use the same name or similar names for ports and signals, throughout the hierarchy, that are connected. (for example, a => a; or a => a_int;)
- Use the name enab for an enable signal. If there is more than one enable in the design, use enab as the prefix for all enable signals.
- Testbench is named 'name_of_the_block'_TB.

Guidelines for clocks and resets

The preferred clocking structure is a single global clock and positive edge-triggered flip-flops as the only sequential devices.



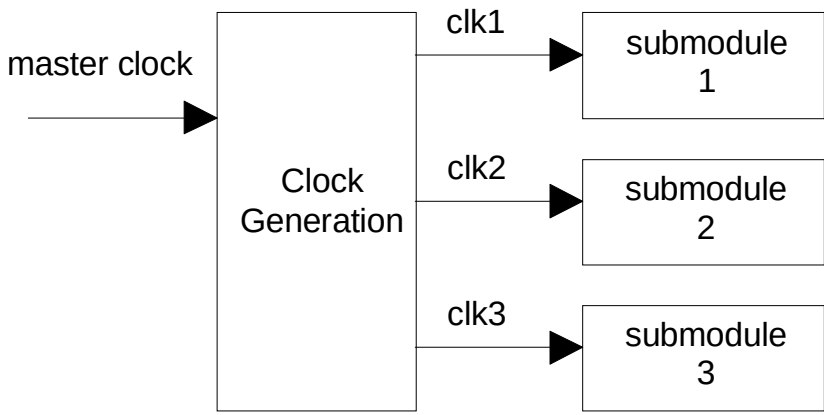
Guidelines for clocks and resets

- **Digital ASICs must be designed to be synchronous when possible.** This must be considered, when VHDL for synthesis is written. The main benefits of synchronous design are:

  - Timing problems are avoided. Only the propagation of signals to the next register during one clock cycle must be verified.
  - Most of the problems with hazards are avoided.
  - The X-states and glitches in gated and multiplexed clocks are avoided.
  - It is easier to test a synchronous circuit than an asynchronous one.
  - Static timing analysis is possible.

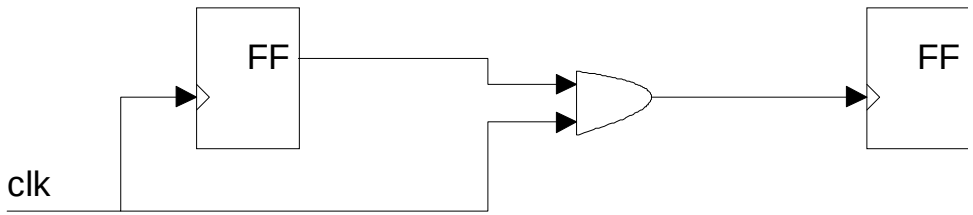
Guidelines for clocks and resets

- partition **the design so that all the logic in a single module uses a single clock and a single reset.**
- Isolating clock and reset generation logic in a separate module allows the other modules to use the standard timing analysis and scan insertion techniques. It also makes it easier to develop specific test strategies for the clock/reset generation logic.



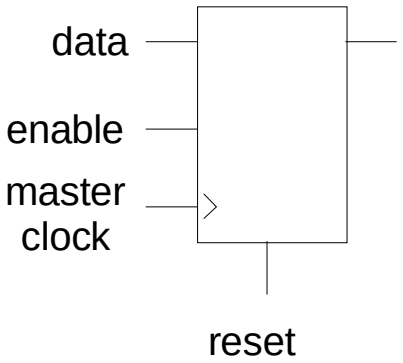
Guidelines for clocks and resets

- **Avoid gated clocks**
- Clock gating circuits tend to be technology specific and timing dependent. Improper timing of a gated clock can generate a false clock or glitch, causing a flip-flop to clock in the wrong data. Also, the skew of different local clocks can cause hold time violations.
- Gated clocks also cause limited testability because the logic clocked by a gated clock cannot be made part of a scan chain.



Guidelines for clocks and resets

- If your design requires gated clocks, use preferably vendor provided gated flip-flop elements using master clock.



Guidelines for clocks and resets

- **Avoid internally generated resets**
- Make sure your registers are controlled only by a single reset signal.
- Avoid internally generated, conditional resets if possible. Generally, all the registers in the macro should be reset at the same time. This approach makes analysis and design much simpler and easier.
- If conditional reset is required, create a separate signal for the reset signal, and isolate this in a separate module. This approach results in more readable code and improves synthesis results.

Design partitioning

- **Good partitioning in the design provides several advantages including:**

Easy handling of design requires sensible hierarchy

Better synthesis results

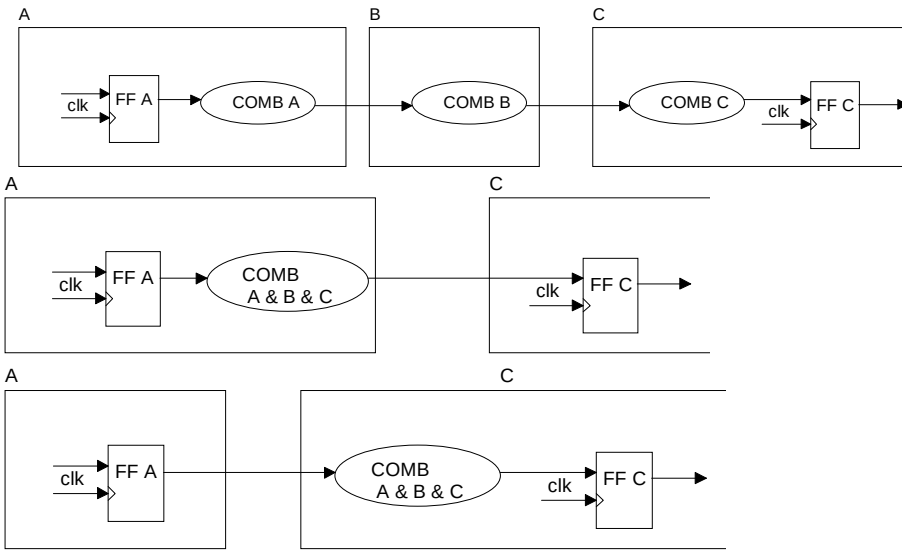
Faster synthesis compile runtimes

Ability to use simpler synthesis strategies to meet timing

Reusability

Design partitioning

- **Locate related combinational logic in a single module and single process if possible**
- The synthesis tool has more flexibility in optimising a design when related combinational logic is located in the same module. This is because synthesis tools cannot move logic across hierarchical boundaries during compile operation.

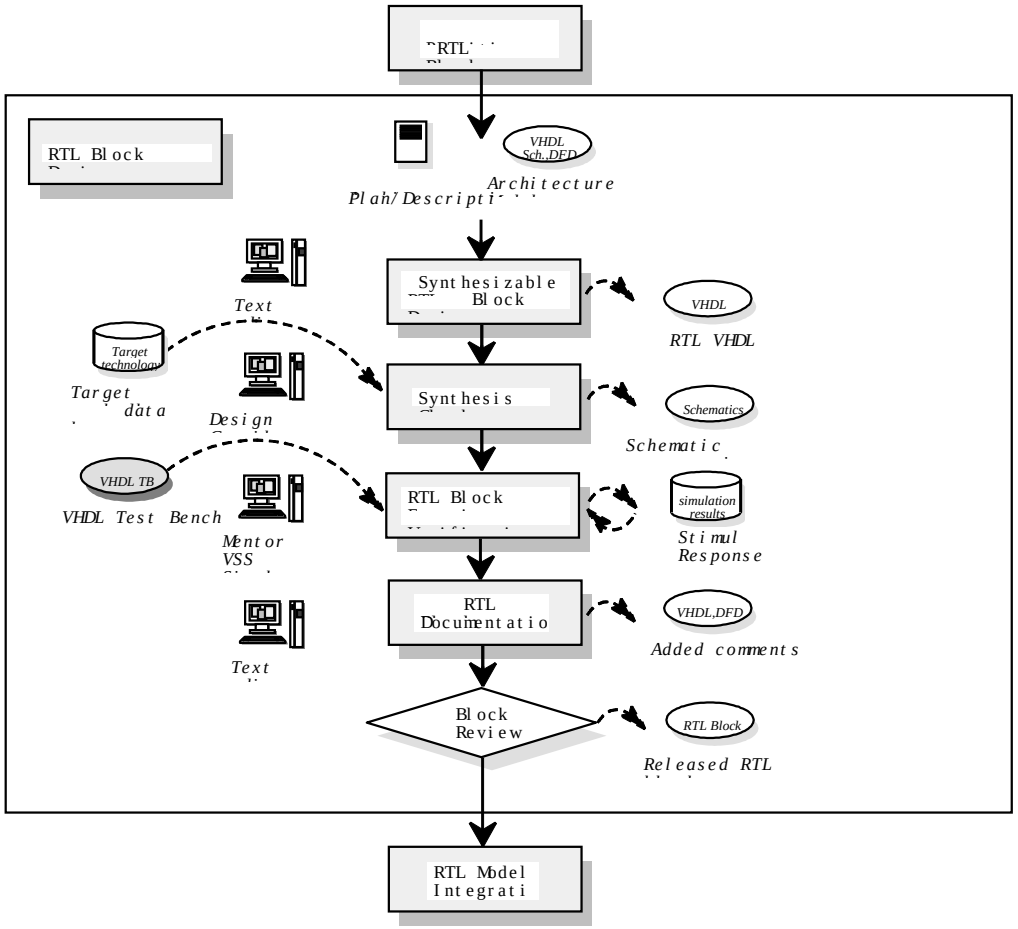


Bad

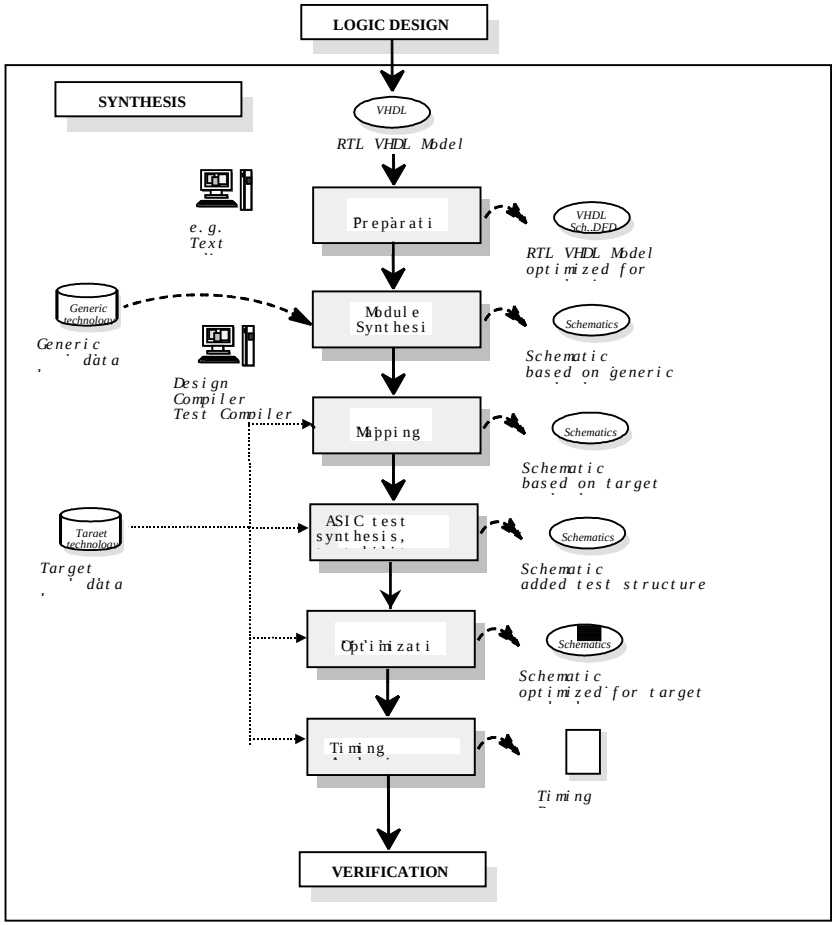
Better

Best

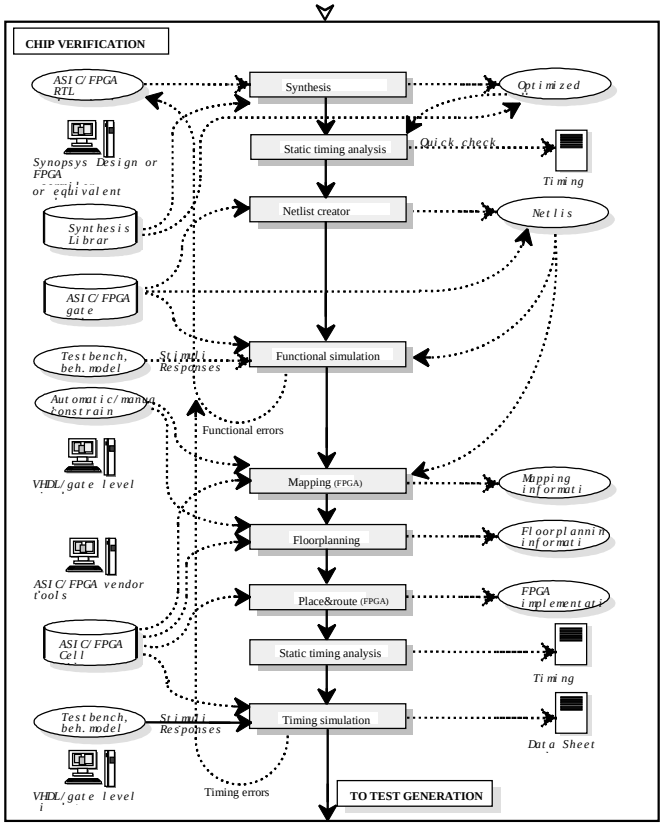
RTL block design



Synthesis


LOGIC DESIGN
SYNTHESIS
VHDL
RTL VHDL Model
e.g. Text
Prepàrati
VHDL Sch.DFD
RTL VHDL Model optimized for
Generic technology
Generic dàta
Module Synthesi
Schematics
Schematic based on generic
Design Compiler Test Compiler
Mapping
Schematics
Schematic based on target
Taraet technology
Target dàta
ASIC test synthesis,
Schematics
Schematic added test structure
Opt'imizati
Schematics
Schematic optimized for target
Timing
Timing
VERIFICATION

Gate level verification

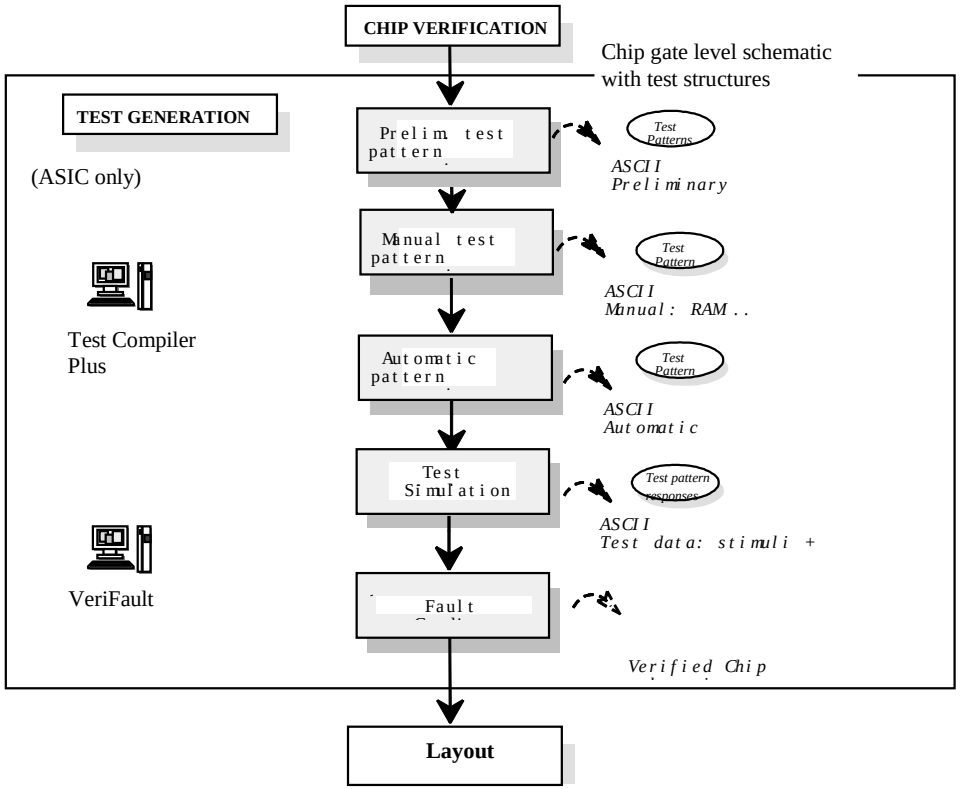


Formal verification

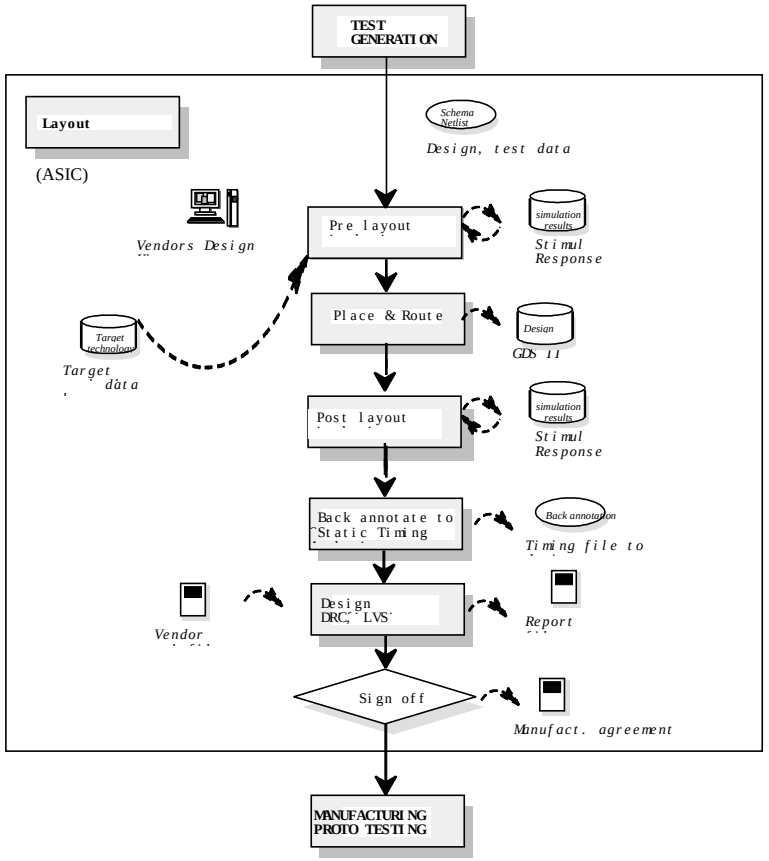
and

static timing analyses

Test generation



Layout and sign off



Prototype testing

- Final verification is done with E/S (engineering samples).
- Physical measurements:

Parametric DC tests

Parametric AC tests

Functional tests

Power consumption

Thermal tests

- Release to system tests
- Final documentation and project completion
- Chips manufacturing

FPGA prototyping

- FPGA, Field Programmable Gate Array
- VHDL coding for FPGAs (design partitioning, limited maximum clock speed, limited amount of interfaces)
- Building prototyping board
- Prototyping in laboratory environment

FPGA prototyping

- Advantages:

VHDL is tested with real clock speed, in real environment

Prototyping environment offers "early ASIC" for other projects

More secure to sign off

- Disadvantages

Requires a lot of time and resources

Expensive