CHAPTER 16
Automated design of digital ICs

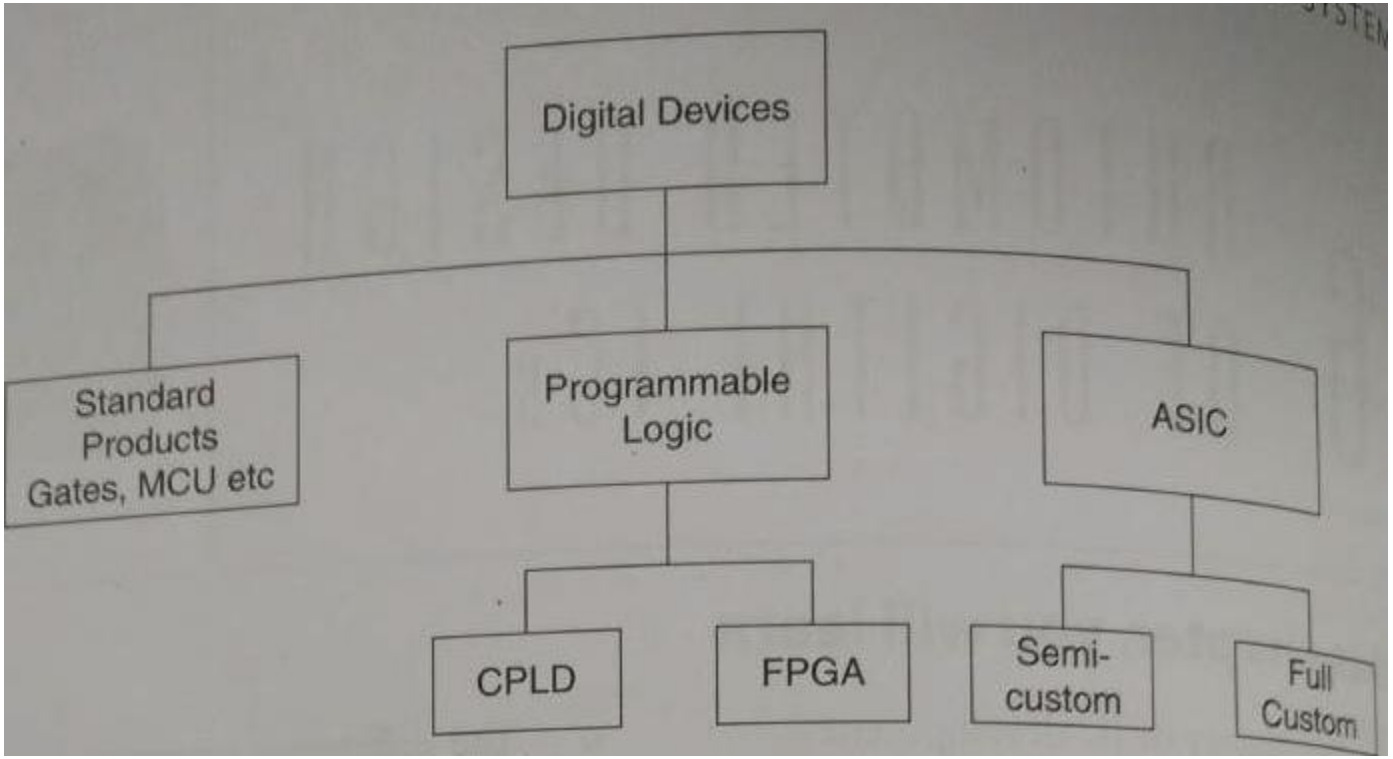
History of IC design

- The semi-conductor industry has started its evolution with advent of diodes and transistors(in 1970s) gradually leading to development of first ICs.
- SSI- few logic gates on single package(1 to 10)

gradual development of IC technology led to MSI, LSI and now VLSI

- No. of transistors in one package keeps on increasing as new design technology emerges

Types of Digital ICs

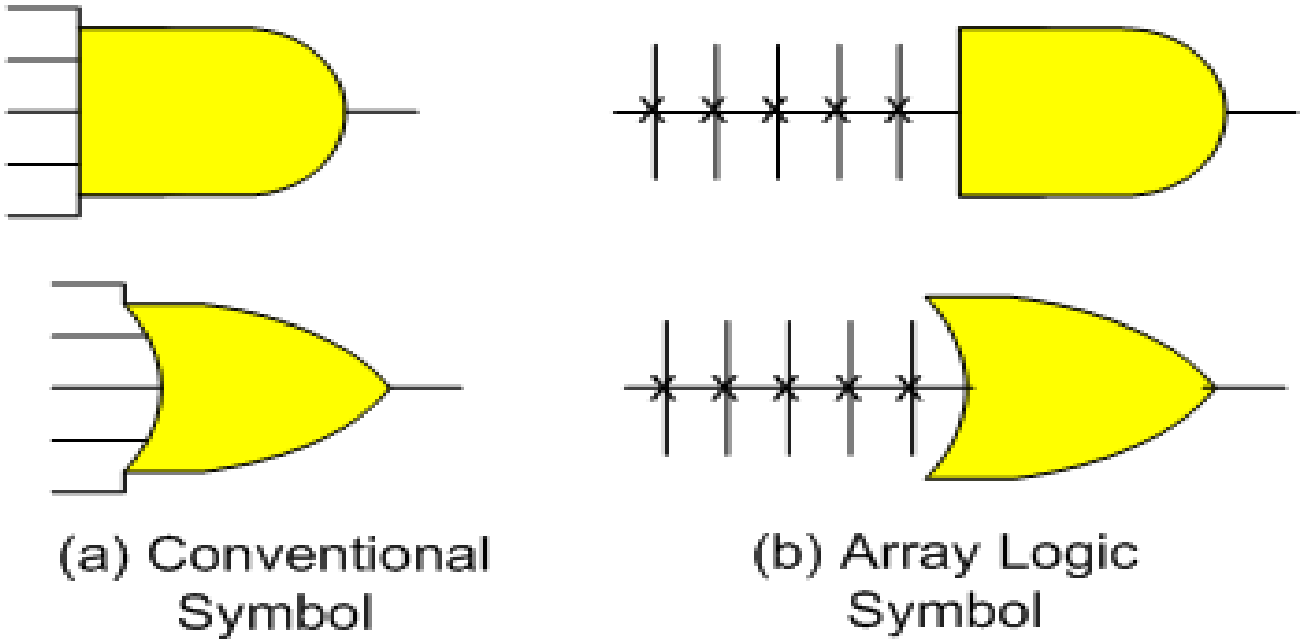


Standard Products

- ICs of gates, counters, multiplexers, MCUS
- Manufactured in bulk and sold in large quantities
- Standard design

Programmable Devices

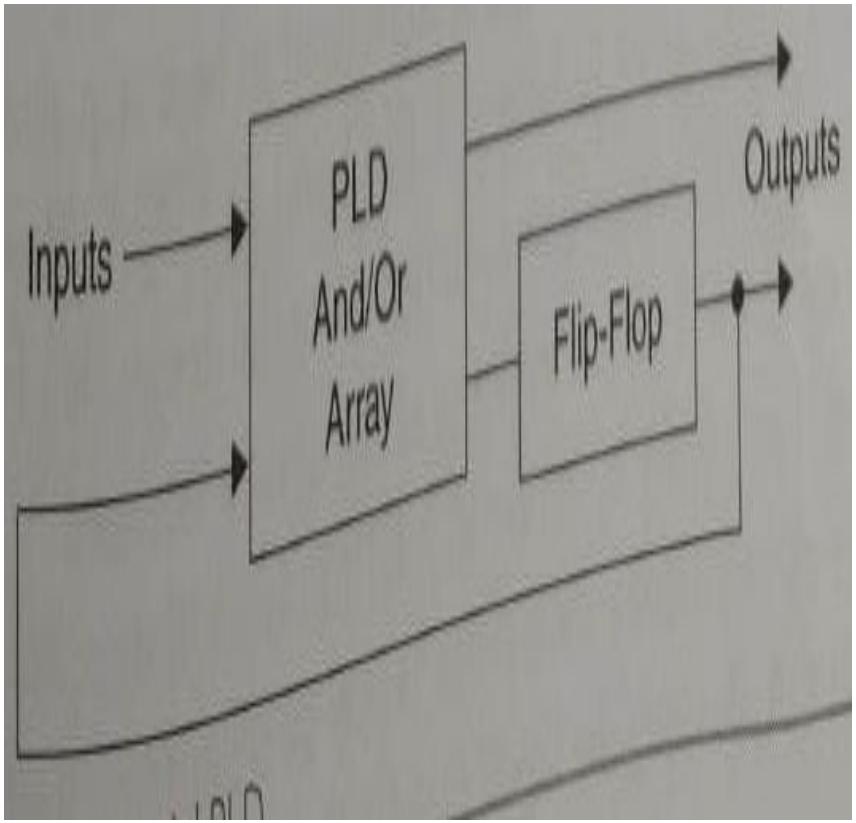
- They are ICs that are programmable and classified as PLD or FPGAs



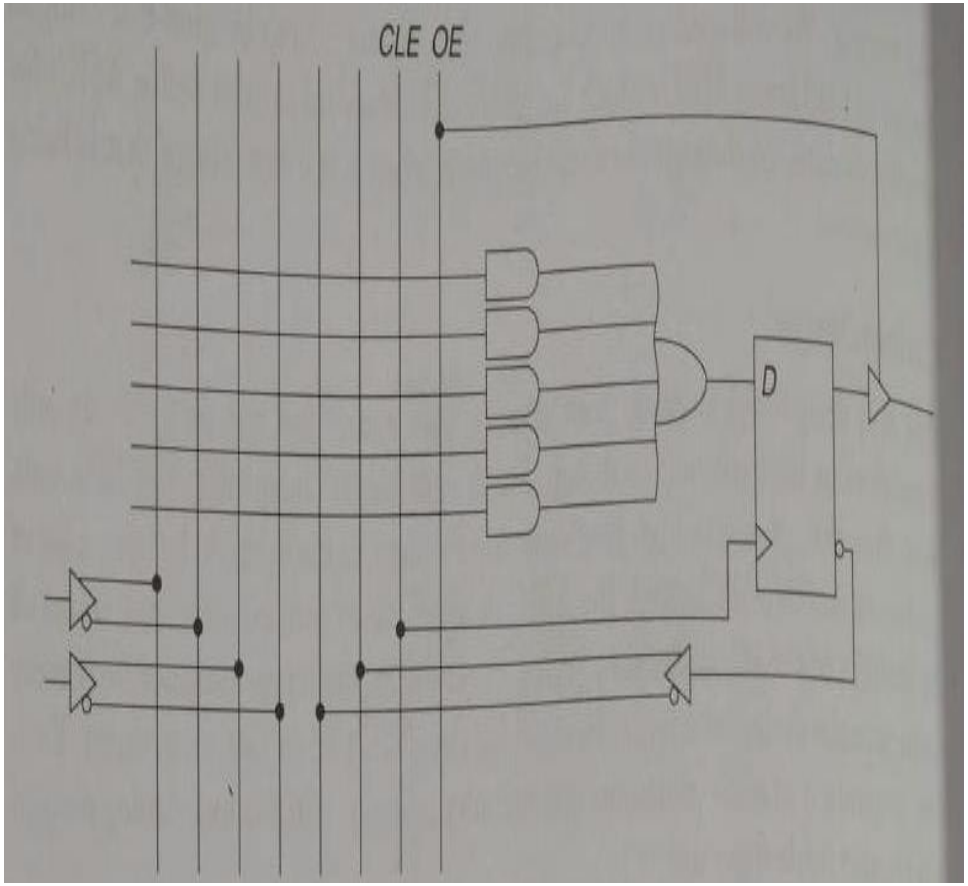
Programmable logic devices

- **programmable logic device** (**PLD**) is an electronic component used to build reconfigurable(i.e. it can be configured by the user to perform different functions) digital circuits.
- It consist of array of AND gates and OR gates. Before the PLD can be used in a circuit it must be programmed (reconfigured) by using a specialized program.
- One of the simplest programming technologies is to use fuses. In the original state of the device, all the fuses are intact. Programming the device involves blowing those fuses along the paths that must be removed in order to obtain the particular configuration of the desired logic function.
- PLDs are used to implement combination logic together with sequential logic.

- Basic building block of PLD is macrocell.
- PLD contain no. of macrocells which are programmable.
- When PLD has flip flop at its output it becomes sequential PLD.



sequential PLD

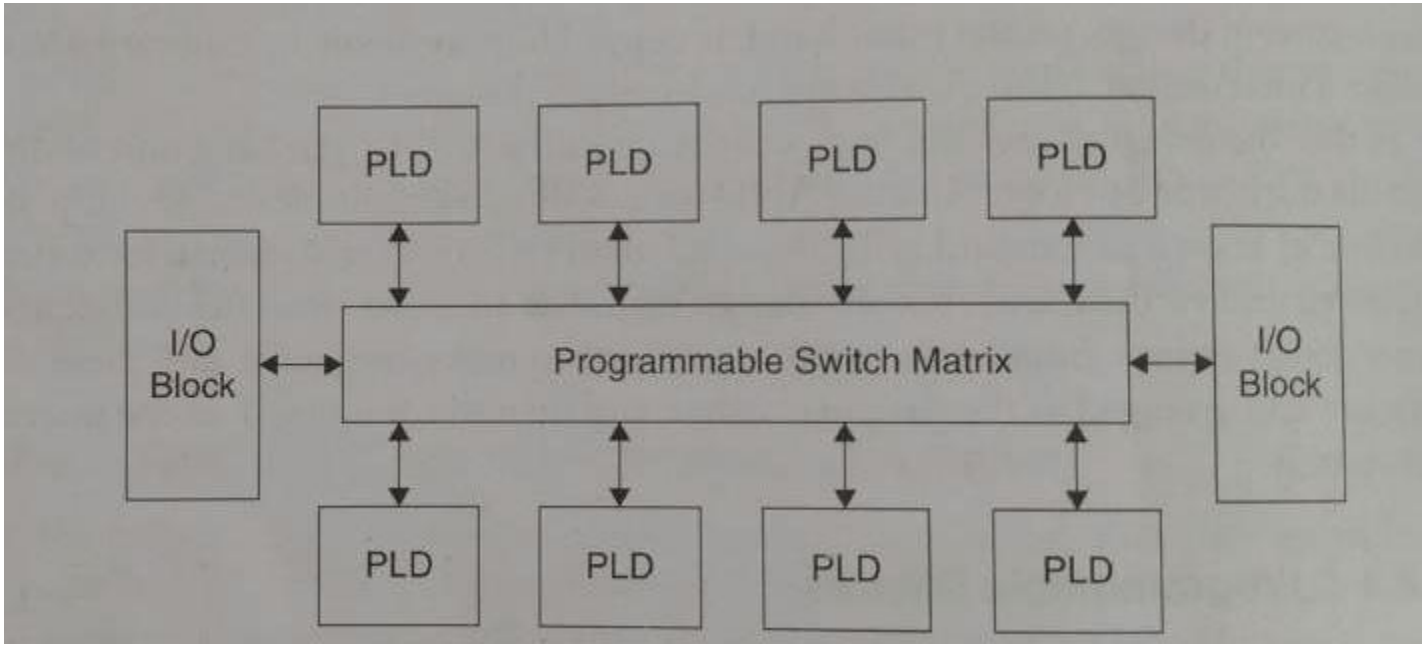


Macrocell

CPLD

- When small PLD blocks are combined, they become CPLD.
- I/O block indicate the presence of programmable logic behind each I/o pin.
- The switch matrix is a programmable interconnect.
- It receives input from I/O block and direct it to individual macrocells
- Selected outputs from macrocells are sent to output pins as needed.

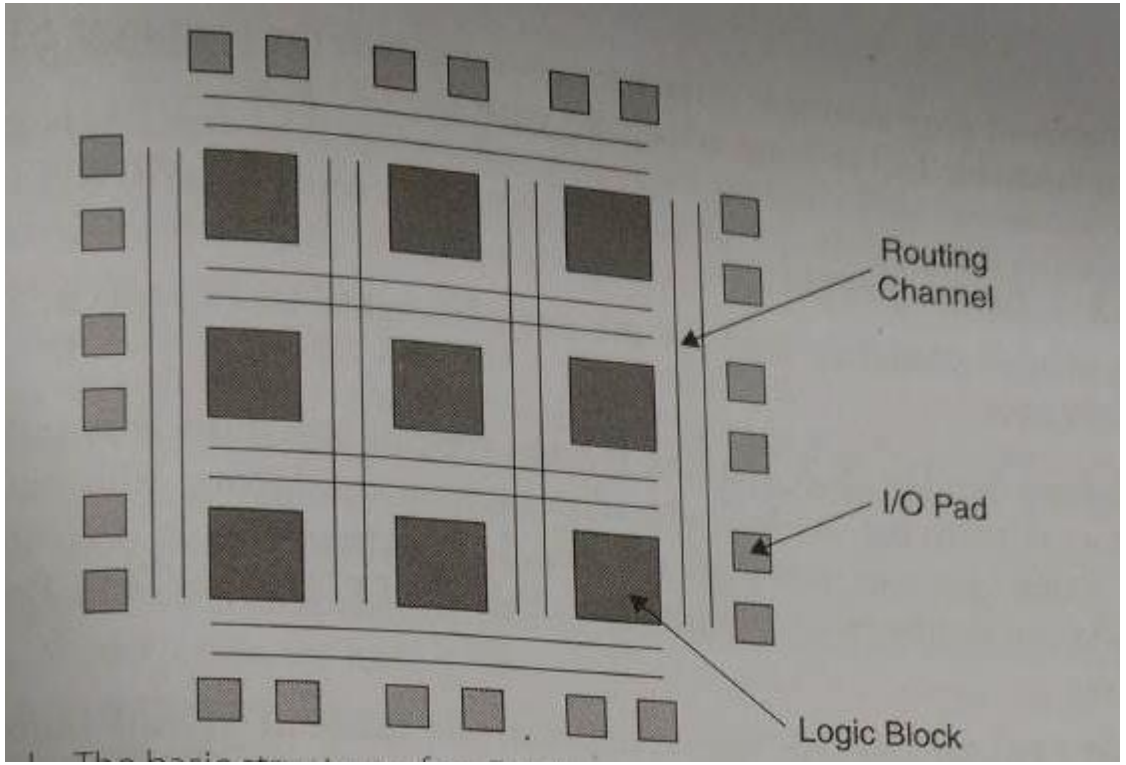
CPLD made up of PLDs



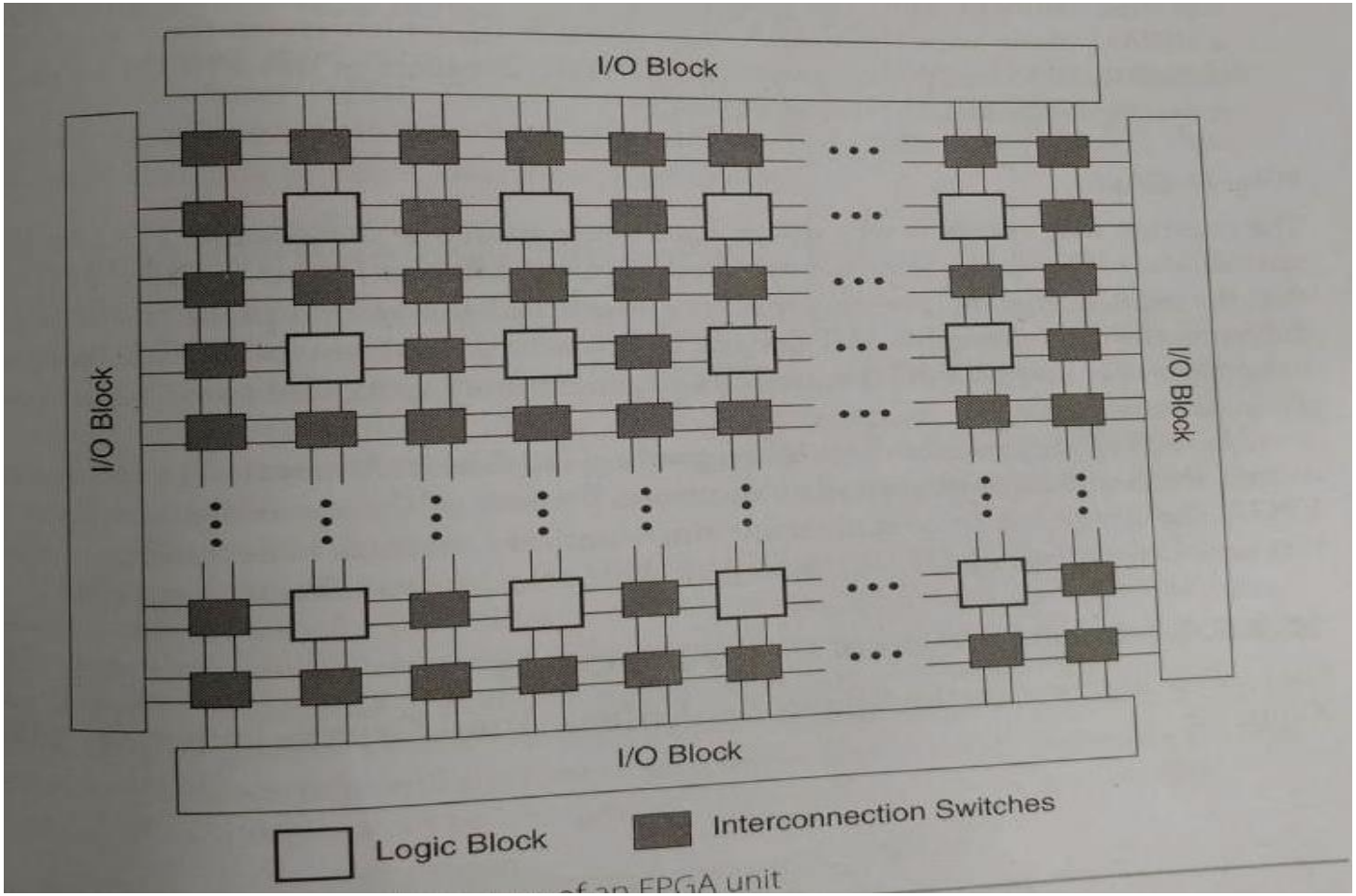
FPGA(Field Programmable Gate Array)

- An FPGA is a device that contains a matrix of reconfigurable gate array logic circuitry.
- The structure is regular array of programmable basic logic cells that can implement combinational as well as sequential logic
- A matrix of programmable interconnects surrounds the basic cell
- Programmable I/O cells are associated with I/O pins
- FPGA provides its user a way to configure:
  - 1. The intersection between the logic blocks and
  - 2. The function of each logic block

Basic Structure of FPGA Unit



Internal Structure of FPGA Unit



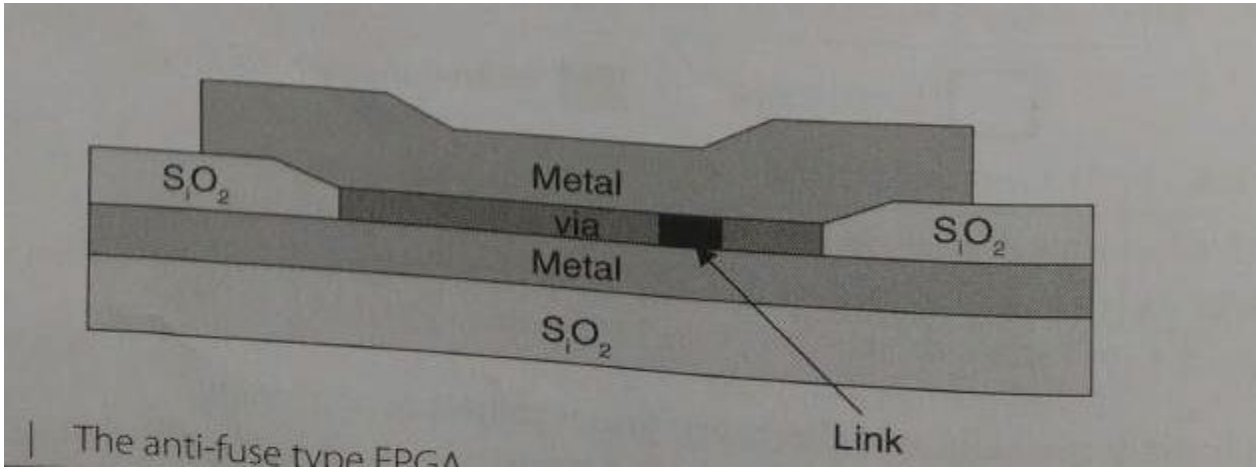
Logic block of an FPGA can be configured in such a way that it can provide functionality as simple as that of transistor or as complex as that of a microprocessor. It can used to implement different combinations of combinational and sequential logic functions. Logic blocks of an FPGA can be implemented by any of the following:

1. Transistor pairs
2. combinational gates like basic NAND gates or XOR gates
3. n-input Lookup tables
4. Multiplexers
5. And-OR structure.

Routing in FPGAs consists of wire segments of varying lengths which can be interconnected via electrically programmable switches. Density of logic block used in an FPGA depends on length and number of wire segments used for routing. Number of segments used for interconnection typically is a tradeoff between density of logic blocks used and amount of area used up for routing.

Types Of FPGA

i. **Anti-fuse:** Anti-fuse doesn't conduct initially, but can be burned to become conducting.
   – Such FPGAs can be programmed only once, they are one time programmable devices.



The anti-fuse type FPGA

Types Of FPGA

SRAM FPGA:

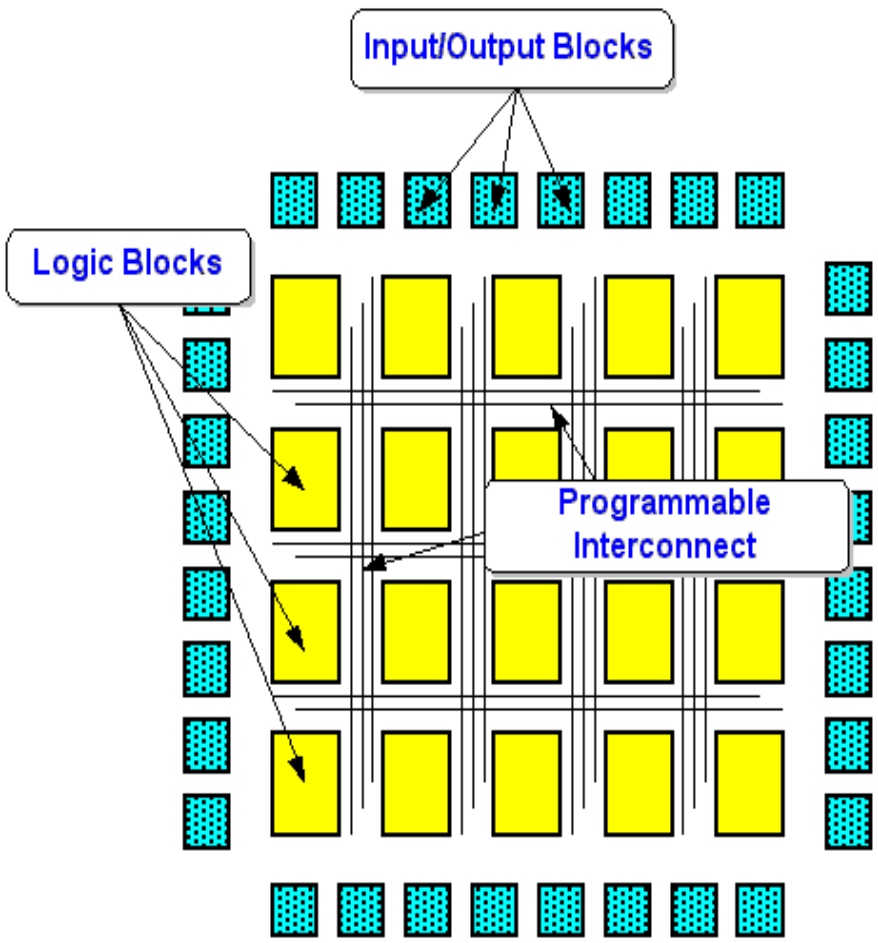
- An interconnect made of SRAM transistors which conduct or does not conduct according to the logic burned on it
- Such FPGA lose their configuration once power is switched OFF.
- To avoid this, there is always a PROM along FPGA chip which stores configuration information.
- When power is switched ON, the contents of PROM are used to re-configure the SRAM based FPGA.

Why FPGA?

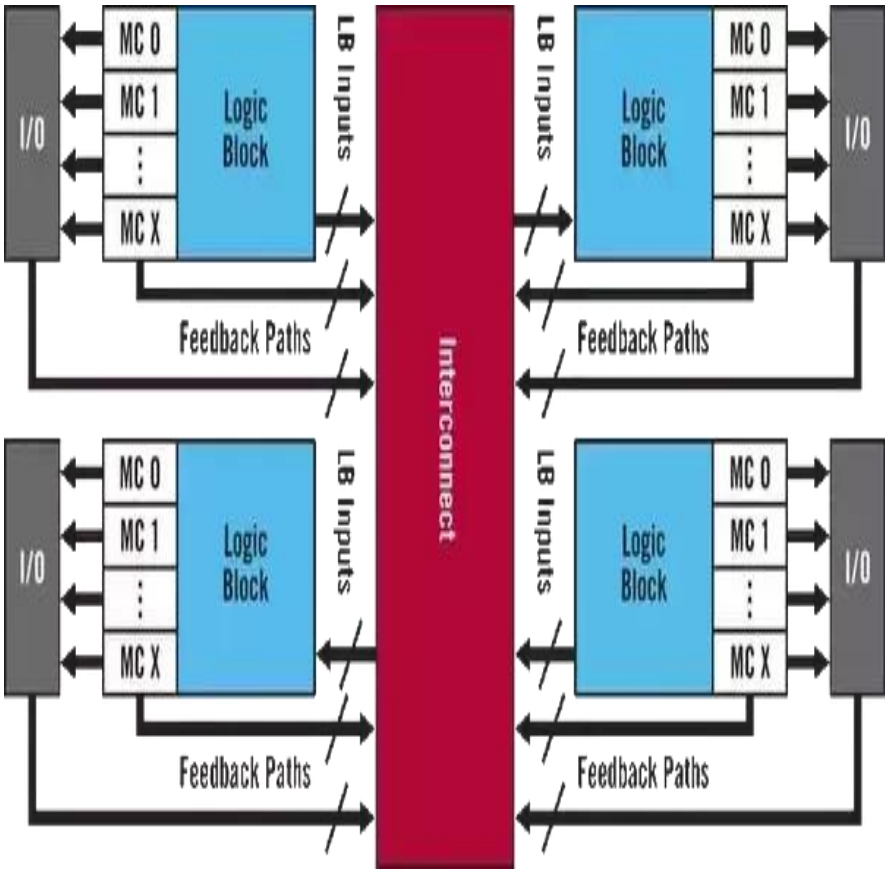
i. '**time to market**' of this design is very small
ii. Flexibility and re-configurability
iii. All parts of product need not to be FPGA-based, but some parts of it may be FPGA based.
iv. Prototyping device: A new design can be tested on an FPGA and inadequacies understood. Because of re-configurability of FPGA, the design can be re-done any times until it converges to becoming a satisfactory one.
    i. Once design is satisfactory, ASIC can be manufactured out of it.

FPGA Vs CPLD

FPGA



CPLD



CPLD Vs FPGA

	CPLD	FPGA
1	Instant-on. CPLDs start working as soon as they are powered up due to EEPROM Flash based macro cells	Since FPGA has to load configuration data from external ROM and setup the fabric before it can start functioning, there is a time delay between power ON and FPGA starts working. The time delay can be as large as several tens of milliseconds.
2	Non-volatile. CPLDs remain programmed, and retain their circuit after powering down. FPGAs go blank as soon as powered-off.	FPGAs uses SRAM based configuration storage. The contents of the memory is lost as soon as power is disconnected.
3	Deterministic Timing Analysis. Since CPLDs are comparatively simpler to FPGAs, and the number of interconnects are less, the timing analysis can be done much more easily.	Size and complexity of FPGA logic can be humongous compared to CPLDs. This opens up the possibility less deterministic signal routing and thus causing complicated timing scenarios. Thankfully implementation tools provided by FPGA vendors have mechanisms to assist achieving deterministic timing. But additional steps by the user is usually necessary to achieve this.

4	Lower idle power consumption. Newer CPLDs such as CoolRunner-II use around 50 uA in idle conditions.	Relatively higher idle power consumption.
5	Might be cheaper for implementing simpler circuits	FPGAs are much more capable compared to CPLDs but can be more expensive as well.
6	More "secure" due to design storage within built in non-volatile memory.	FPGAs that use external memory can expose the layout plan externally. Many FPGA vendors offer mechanisms such as encryption to combat this. Design specific protection mechanisms also can be implemented.
7	Very small amount of logic resources.	Massive amount logic and storage elements, with which incredibly complex circuits can be designed. FPGAs have thousands times more resources! This point alone makes FPGAs more popular than CPLDs.
8	Power down and reprogramming is always required in order to modify design functionality.	FPGAs can change their circuit even while running(Since it is just a matter of updating LUTs with different content) This is called Partial Reconfiguration, and is very useful when FPGAs need to keep running a design and at the same time update the it with different design as per requirement. This feature is widely used in Accelerated Computing.

Manufactures of FPGA and CPLDs

- Actel, Altera, Atmel, Cypress, Lattice, Quicklogic, Xilinx

ASIC

- Manufactured for specific applications
- A video codec, audio codec, controller for specific devices are examples of ASIC.
- Not manufactured in bulk
- Product manufacturer fix the product specifications
- ASIC designer design them according to specification

ASIC

- Two different manufacturer give **two different specs for audio codec**
- ASIC is built for **one and only one customer**
- ASIC is likely to be **proprietary by nature** and not available to general public
- Specific design ,entails more work and more time-consuming and expensive
- Once a product become popular ASIC used in them have to be manufactured in bulk and become a standard product
- Such ASIC are called ASSP(Application Specific Standard product).

Classification of ASIC

- ICs for specific application.
- Classified as
  - Custom design ASIC
  - Semi custom design ASIC

Custom Design

- Design starts from scratch.
- Designer has only transistors as his basic unit of design.
- Designs bigger circuit with basic unit.
- This gives very high flexibility and freedom to the designer but also a lot of hard work and time is involved.
- ASIC designer customizes all the features of IC
- Each new generation micro processor and memory is designed in this way

Semi-Custom Design

- Semi-Custom design is easier.
- Units available in library are used to make a final design.
- Designer use logic cells from a library.
- Basic unit of design is not a transistor but logic circuits(AND gates, OR gates, and flip-flops)
- Standard cells are designs which are already been tested and verified
- Easier time and faster design ensures
- Small cells are combined to make ‘megacells’ and

Semi-Custom Design

- These could be stacked and arranged as designer wishes and designer does the interconnections only

FPGA vs ASIC

FPGA

1. Reconfigurable circuit. FPGAs can be reconfigured with a different design. They even have capability to reconfigure a part of chip while remaining areas of chip are still working! This feature is widely used in accelerated computing in data centres.

2. Design is specified generally using hardware description languages (HDL) such as VHDL or Verilog.

3. Easier entry-barrier. One can get started with FPGA development for as low as USD $30.

4. Suited for low-volume mass production.

5.Less energy efficient, requires more power for same function which ASIC can achieve at lower power.

ASIC

1. **Permanent circuitry**. Once the application specific circuit is taped-out into silicon, it cannot be changed. The circuit will work same for its complete operating life.

2. **Same as for FPGA**. Design is specified using HDL such as Verilog, VHDL etc.

3. **Very high entry-barrier** in terms of cost, learning curve, liaising with semiconductor foundry etc. Starting ASIC development from scratch can cost well into millions of dollars.

4. Suited for **very high-volume mass** production.

5. Much **more power efficient** than FPGAs. Power consumption of ASICs can be very minutely controlled and optimized.

FPGA vs ASIC

FPGA

6. Limited in operating frequency compared to ASIC of similar process node. The routing and configurable logic eat up timing margin in FPGAs.

7. Analog designs are not possible with FPGAs. Although FPGAs may contain specific analog hardware such as PLLs, ADC etc, they are not much flexible to create for example RF transceivers.
8. FPGAs are highly suited for applications such as Radars, Cell Phone Base Stations etc where the current design might need to be upgraded to use better algorithm or to a better design. In these applications, the high-cost of FPGAs is not the deciding factor. Instead, programmability is the deciding factor.
9. Preferred for prototyping and validating a design or concept. Many ASICs are prototyped using FPGAs themselves! Major processor manufacturers themselves use FPGAs to validate their System-on-Chips (SoCs). It is easier to make sure design is working correctly as intended using FPGA prototyping.

ASIC

6. ASIC fabricated using the same process node can run at much **higher frequency** than FPGAs since its circuit is optimized for its specific function.

7. ASICs can have **complete analog circuitry**, for example WiFi transceiver, on the same die along with microprocessor cores. This is the advantage which FPGAs lack.

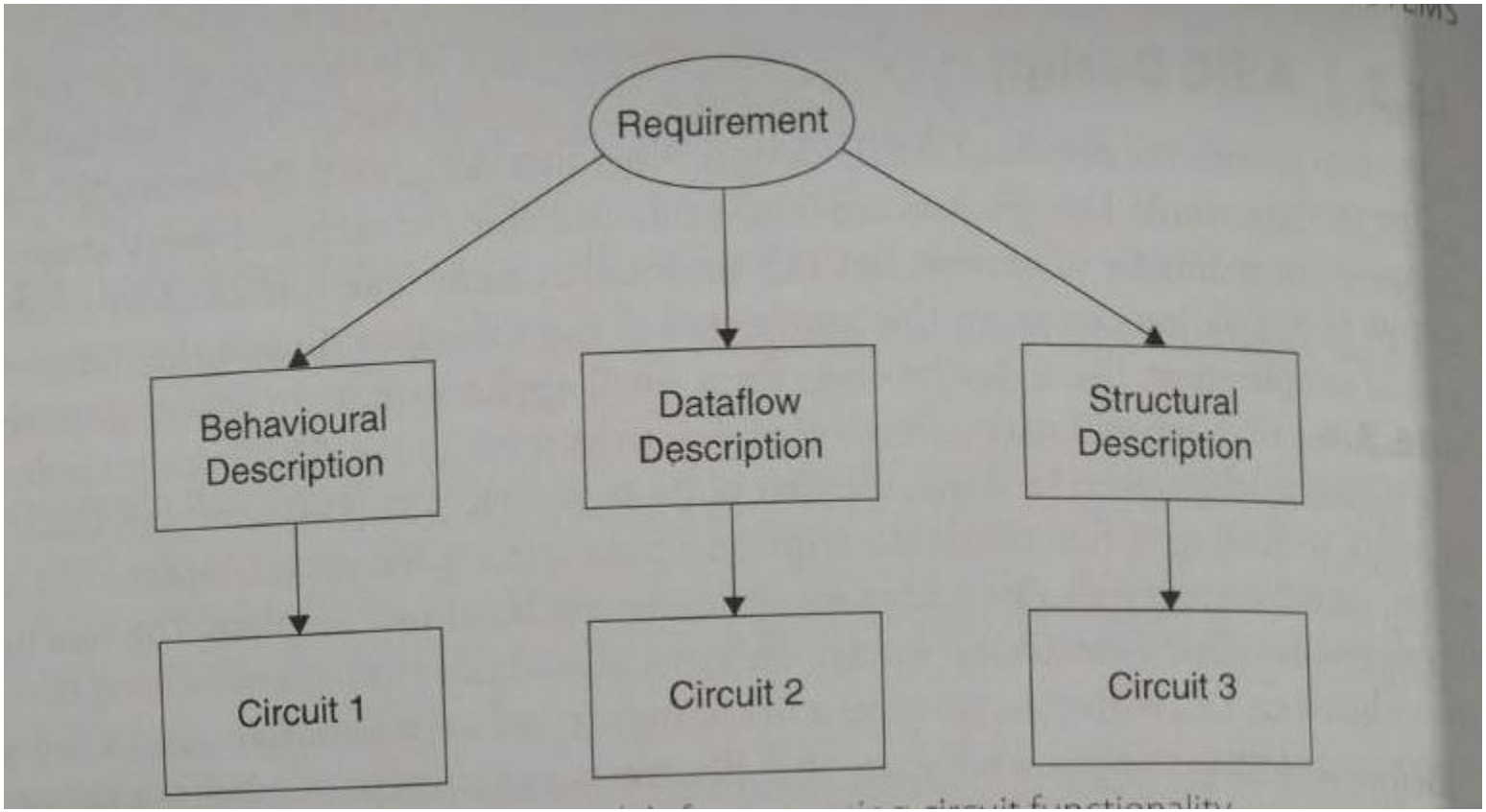
8. ASICs are definitely not suited for application areas where the design might need to be upgraded frequently or **once-in-a-while.**

9. It is not recommended **to prototype** a design using ASICs unless it has been absolutely validated. Once the silicon has been taped out, almost nothing can be done to fix a design bug (exceptions apply).

ASIC Design

- Design is fully automated –there are EDA tools are available for each and every stage.
- First point is to have requirement specified i.e. functional description of circuit we need
- The functional description is represented using different architectural model such as behavioral model, data description model and so on
- Model contains information regarding how circuit behaves
- List of inputs, outputs and a function which maps between the two

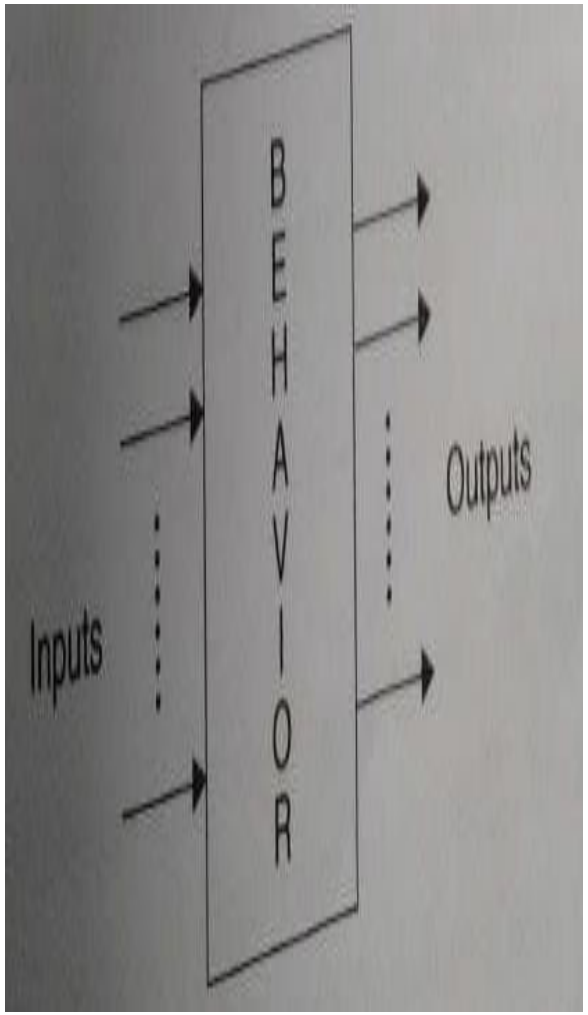
Models for expressing Functionality of circuit



Behavioral Model

Behavioral Modeling:

- Behavior is expressed in terms of truth table or HDL(ex. Varilog and VHDL)
- HDLs are syntactically similar to HLL like C. Ex. It support various construct like -if, for, case statements etc., same as C
- However, C is sequential language, this is not so for HDLs. Concurrency is key in HDLs. Additionally, No HDLs available for designing of analog devices
- It does not directly convert to hardware
- Sequential model is not fully synthesizable



Data Flow Model

- Expressed in terms of a set of concurrent statements which shows the logical relationship between input and output ports.

Structural Model

- Actual hardware components for design are identified
- And interconnection between these components is defined by the statements in the structural model.

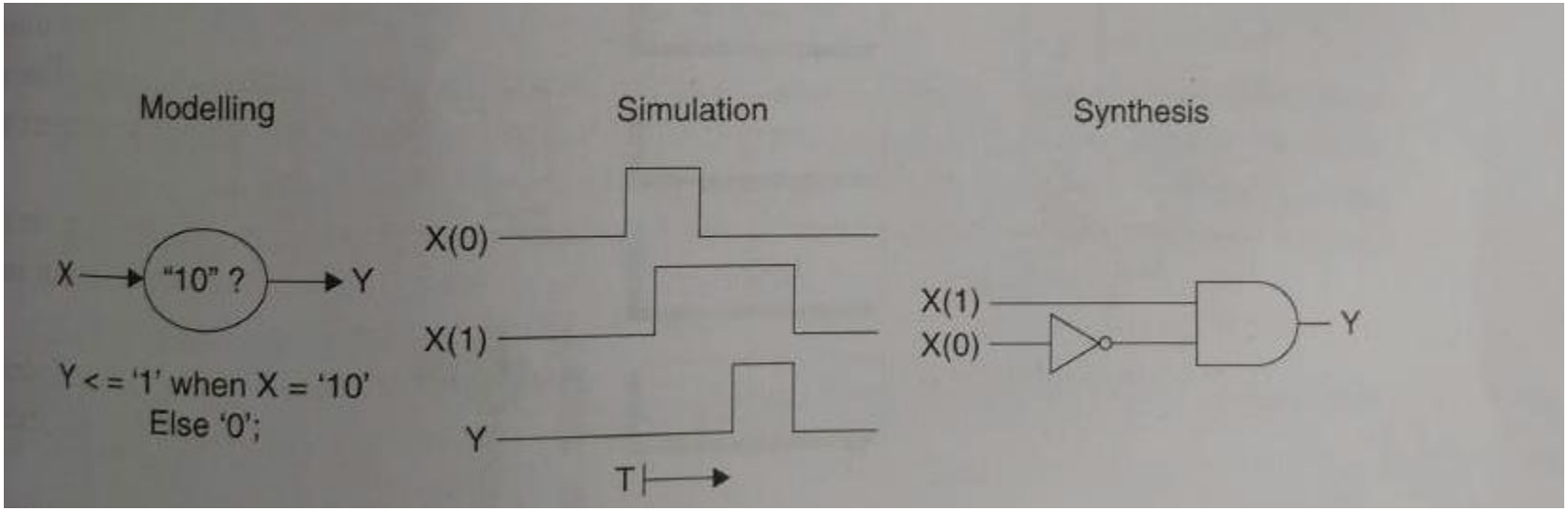
Simulation

- To verify functional correctness of design called functional verification
- Tool called simulator is used or the designer writes the test bench using HDL itself
- Mostly a wave form simulator . Display of inputs and outputs in wave forms

Synthesis

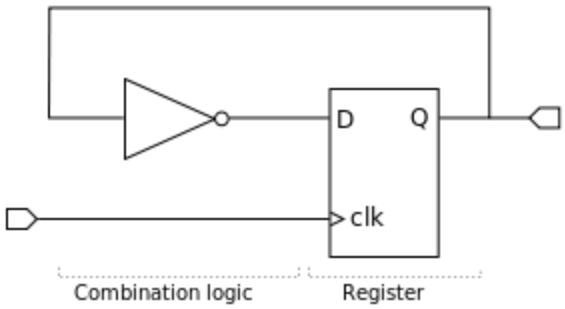
- Synthesis tool is available as a part of automation software
- It converts design to actual
- Synthesis tools convert HDL design to RTL level design and then gate level netlist

Modelling, Simulation and Synthesis



RTL

- Design can be broken up into sequential circuit and combinational circuit

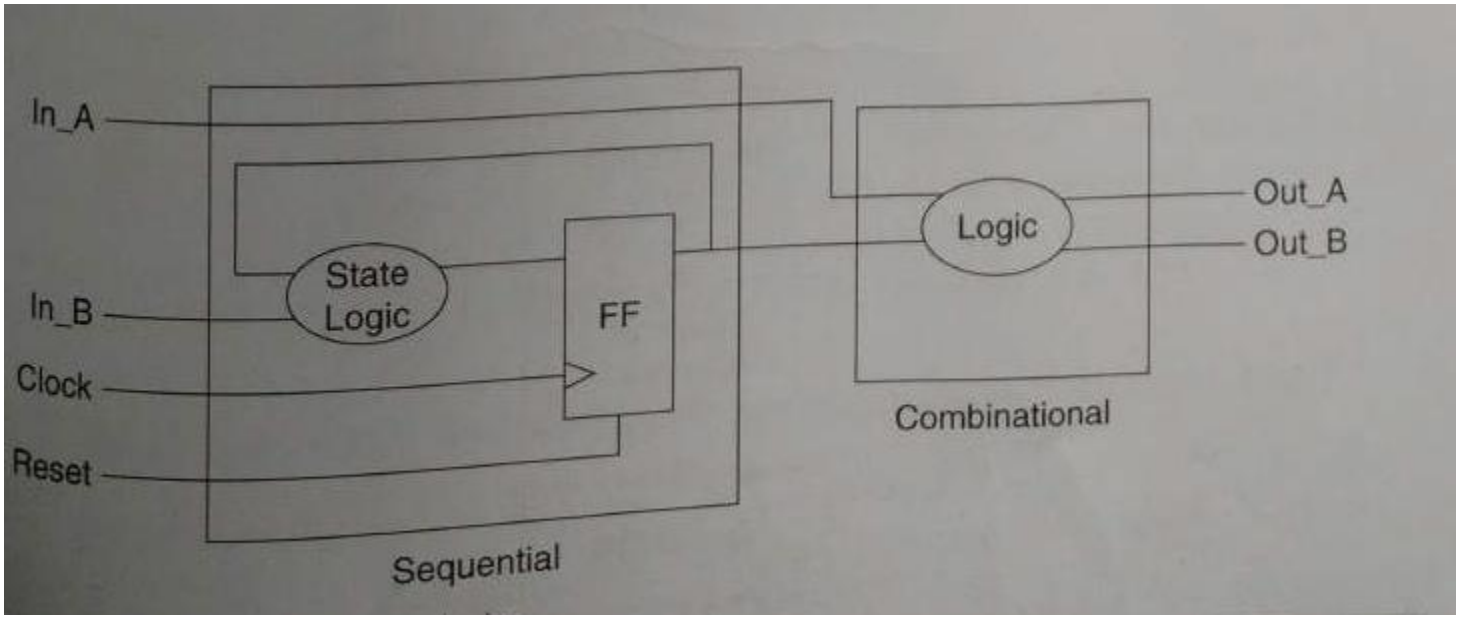



```
D <= not Q;
process(clk)
begin
        if rising_edge(clk)
        then Q <= D;
        end if;
end process;
```

The inverter is connected from the output, Q, of a register to the register's input, D, to create a circuit that changes its state on each rising edge of the clock, clk. In this circuit, the combinational logic consists of the inverter.

Circuit described in VHDL

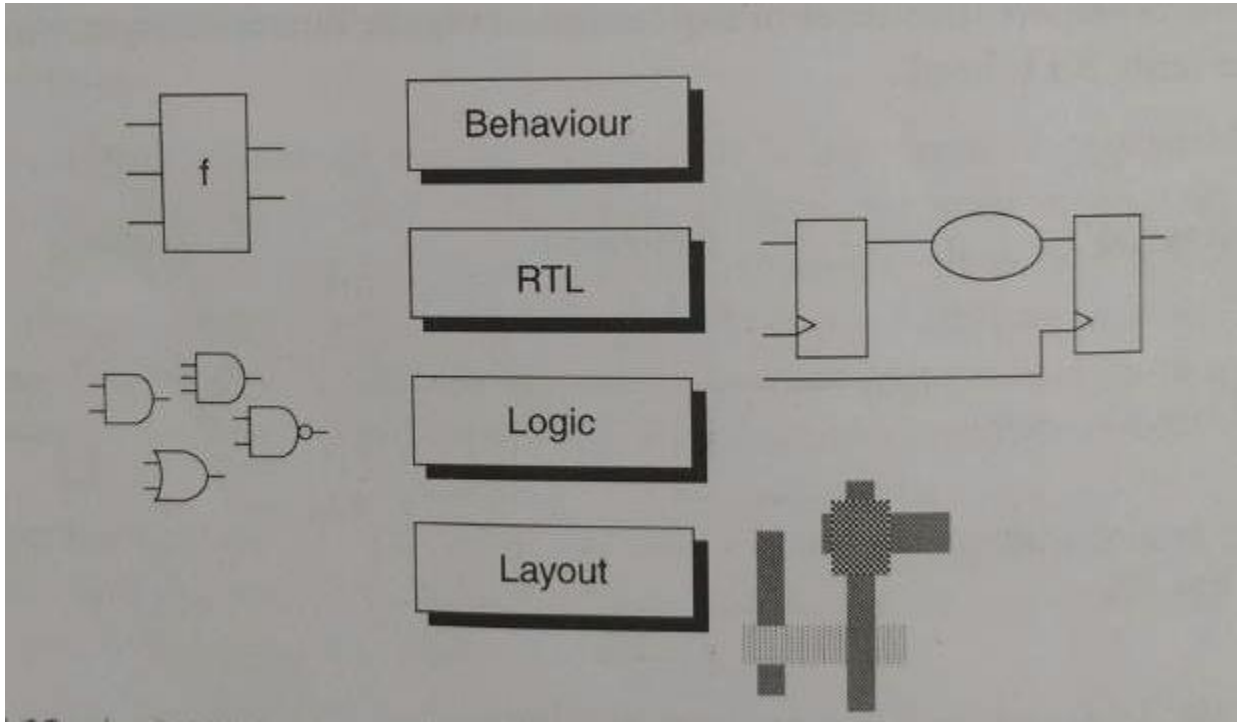
RTL


In_A
In_B
Clock
Reset
State
Logic
FF
Sequential
Logic
Out_A
Out_B
Combinational

Gate Level Netlist

- Dividing RTL further down take us to gates and wires
- Net means ‘electronic wires’
- List of wires is called net list
- At logic level design is expressed as list of logic gates and interconnection between them

Design Expressed at Different Levels of Abstraction



Front End Design Steps

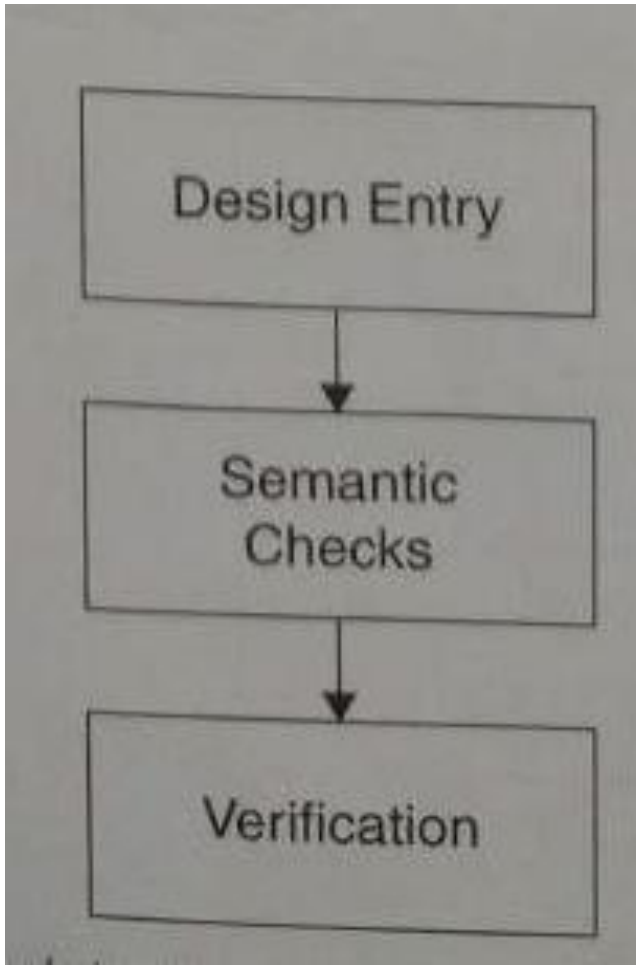
Design Entry:
Entering the design using HDL

Semantic Checks:

1. Design Entry needs to be checked for language correctness
2. This phase can also be used to check the synthesizability of code i.e. whether the code can lead to practical hardware
3. It is important to check synthesizability, as RTL should be mapped to standard cells using synthesis tools

Design Verification:
Verification is the process where the design is tested against specification

* At the end of front end design step, synthesis is performed.



What to do with netlist obtained after front-end design

Two options

i. One is to burn into FPGA

NOTE: SRAM-FPGA are re-configurable. This design can be overwritten by another one and FGPA gets re-configured as per new design

e.g. if FPGA is counter to count 0 to 9. later if we want to change it to 64-bit shift register, we only need to write the design for shift register using HDL, synthesize it and burn it into FPGA. It now function as shift register.

What to do with netlist obtained after front-end design

- The other option is to make ASIC out of this, this means that it is to be fabricated as a chip.
- For this few additional steps are required which is called the **Back end Design**

Back-end Design

- **Layout :** The layout is a set of patterns in which the transistors and their connections are laid out i.e. finalized

- **Floor Plan: it defines the following**
  - Aspect ratio of chip
  - Placement of cells
  - Position of the I/O pad

- **Routing:** interconnection between and within cells are finalized

Back-end Design

- There are CAD tools for all these steps but manual intervention is sometimes necessary to make the design optimal.

- Once all these steps are over, the design can be sent to fabrication unit which delivers the design in the form of an IC.

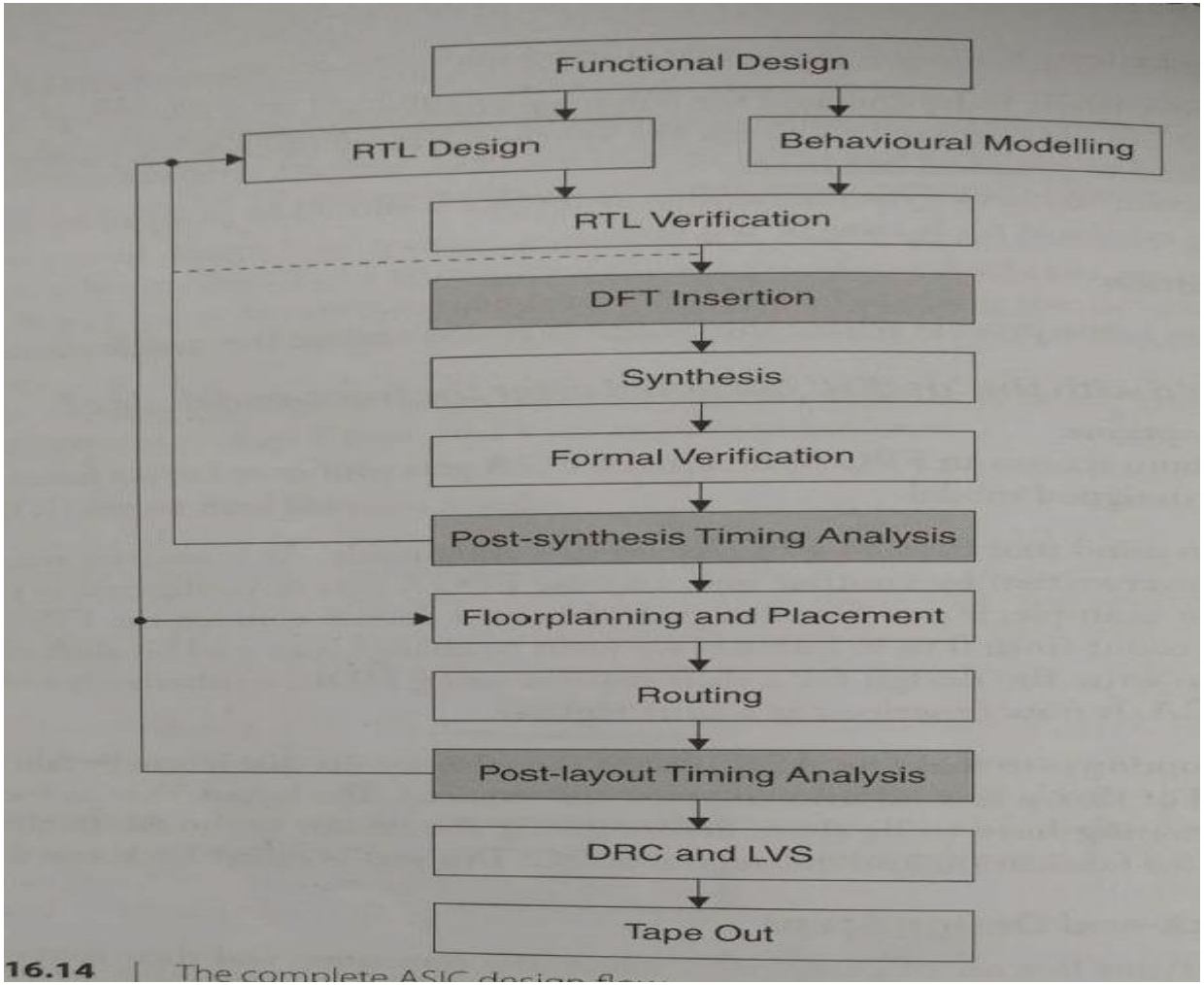
ASIC: The complete Sequence

- In such a step ‘front end’ and ‘back end’ design teams are separate.
- For the front-end design, RTL design and verification are also done by different teams.
- It starts like this.
- A customer is usually a product designer who needs ‘the’ ASIC as one of ICs to be incorporated in his product.

ASIC: The complete Sequence

- The customer lays out his requirement to the designer.
- There is no need for any specific format for this, the designer is made very clear about the requirements and specifications of the product his team is to deliver

Steps in Flowchart



16.14 | The complete ASIC design flow

Steps in Flowchart

- **Functional Design**: this is made using HDL
- **RTL Design**: This is obtained at end of synthesis stage, after a simple simulation check done by the RTL design team.
- **Behavioural model:** The verification team keeps ready a model for the functionality of his design, it could be just a set of statements indicating the expected output for a set of input conditions.

Steps in Flowchart

- **RTL verification**: RTL is verified against the behavioral model prepared by the verification team. This step is also called validation if RTL design does not match the behavioral model, the re-design is done by RTL team. This sort of feedback mechanism is practised until the complete design is validated.

Steps in Flowchart

- **DFT insertion**: DFT stands for **Design for Testability**. This is very important aspect of modern ICs which are very complex. In a big lot of many such chips, it is quite likely that some of them are faulty.
- The process of confirming that an IC is fault-free is called ‘testing or post-silicon validation.
- Extra testing circuits are inserted into the design and these extra test circuits are called DFT circuits. They are added to RTL, after functional verification is over.

Steps in Flowchart

- **Logic Synthesis**: This corresponds to technology mapping, where the RTL is mapped to the standard cells in the library. After this, the gate level netlist is available.
- **Formal Verification:** This stage attempts to prove mathematically that all requirements are met and that certain undesired behaviours like deadlock do not occur. There are formal verification tools to check whether RTL and logic netlist are equivalent

Steps in Flowchart

- **Post-synthesis timing analysis**: timing had not considered at all. Now at this stage, preliminary timing results after synthesis are analysed and checked against the project performance requirements.
- If needed RTL design and synthesis options are modified, and synthesis is repeated as shown in fig. by feedback loop.

Steps in Flowchart

- **Layout Design:** The next two stages are for layout preparation, and includes floor planning, placement and routing. The EDA tools do this, but manual routing is allowed if designer feels it necessary
- **Post-layout Timing Analysis:** once again timing results are again compared with performance requirements. If it doesn't fit, floor plan can be changed or the placement run with other parameters.
- Feedback loop shows that RTL design/layout can be modified again.

Steps in Flowchart

- **DRC:** Design rule Check. This confirms that layout ,made conforms to the rules of foundry in which ASIC is to be fabricated.
- **LVS:** Layout verses Schematic

Another formal verification check between post-synthesis netlist and final layout

Tape out: the process of handing over the resulting layout to semiconductor fabrication plant is called tape out

Suppliers of EDA tools

- Synopsys, cadence, chrysalis, Avanti, Xilinx, Mentor Graphics, Magma, co-ware, altera etc. are some important vendors of EDA tools for digital design.