CS223: Hardware Lab

Digital Circuit Design using (a) Breadboard and ICs and (b) FPGA

A. Sahu

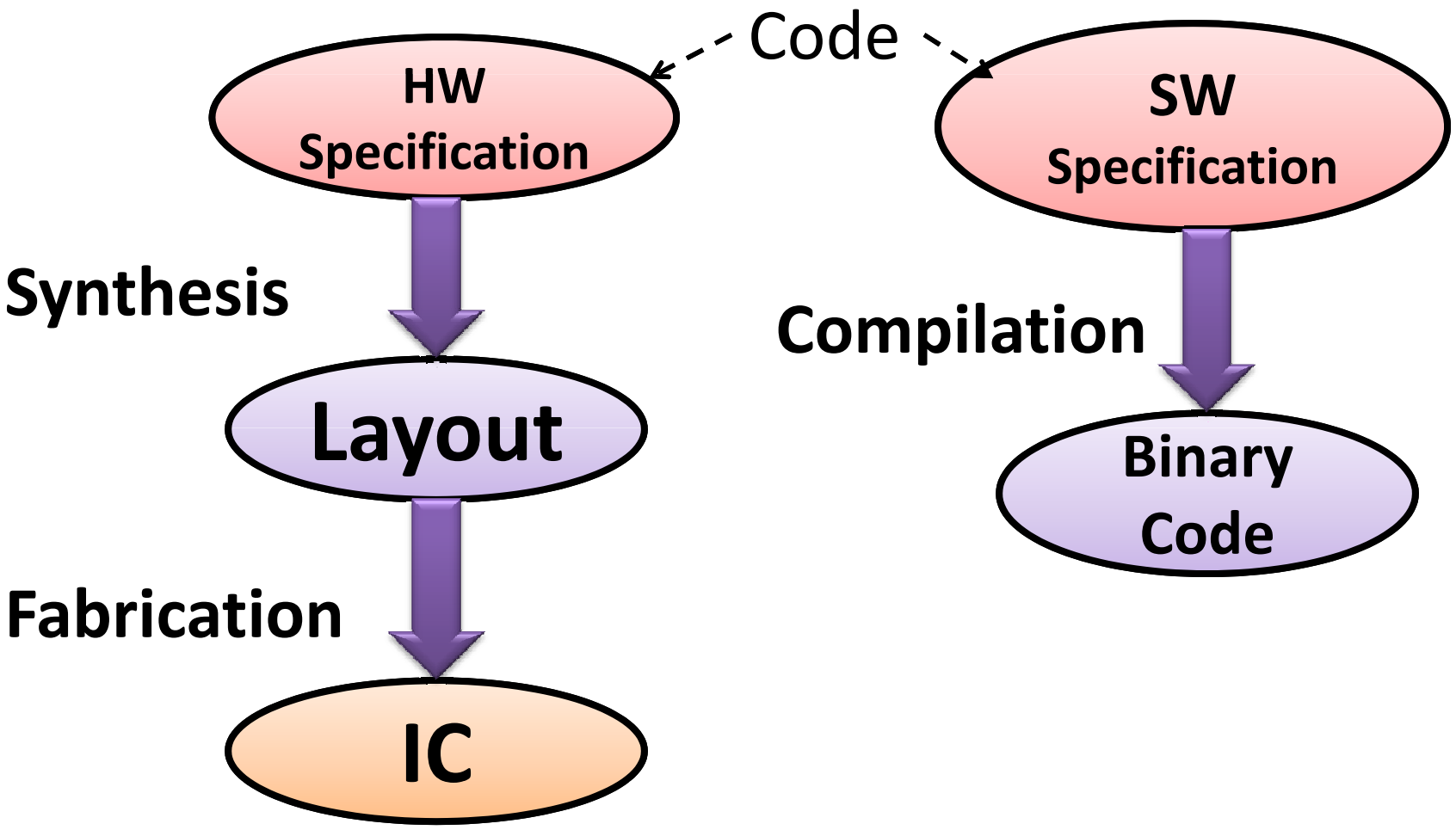
Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

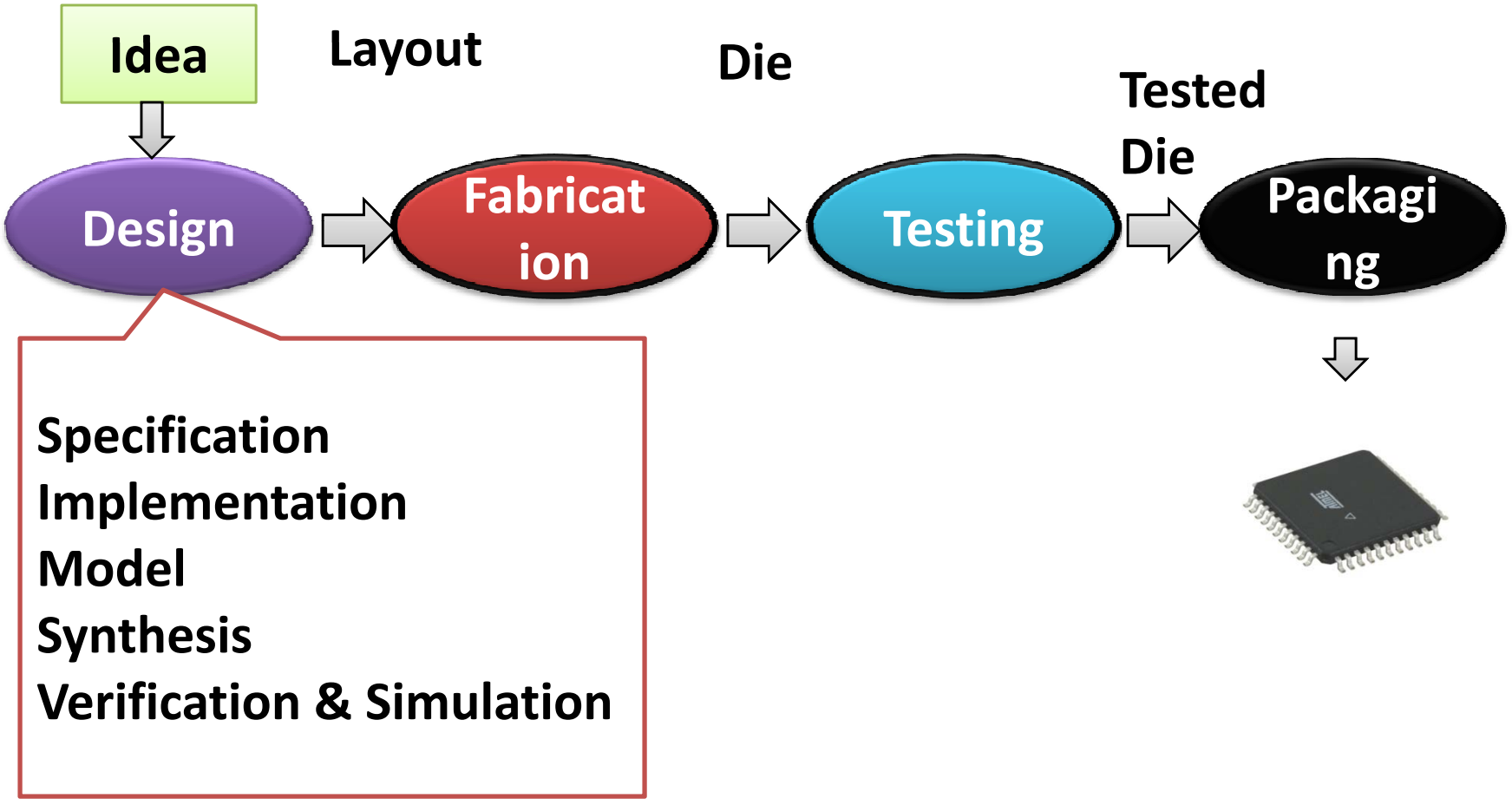
Outline

- Course Administrative
  - Assignments, Grading, Timing, Rules
- ASIC / FPGA Design process
- What is VHDL : Requirement of VHDL
- VHDL
  - basic language concepts
  - basic design methodology
  - examples

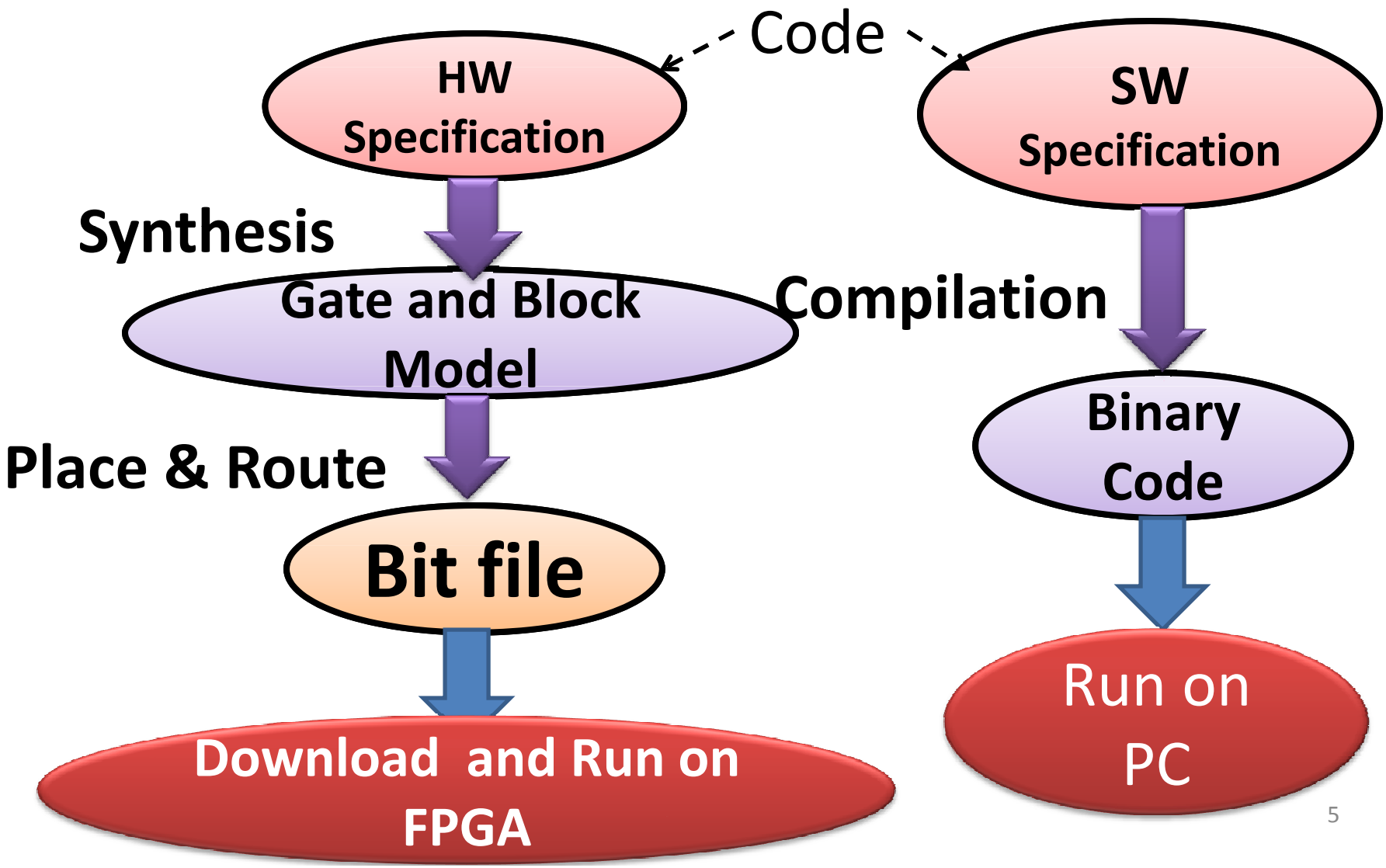
Design Flow: Hardware Vs Software



IC Design Process



Design Flow: Hardware Vs Software

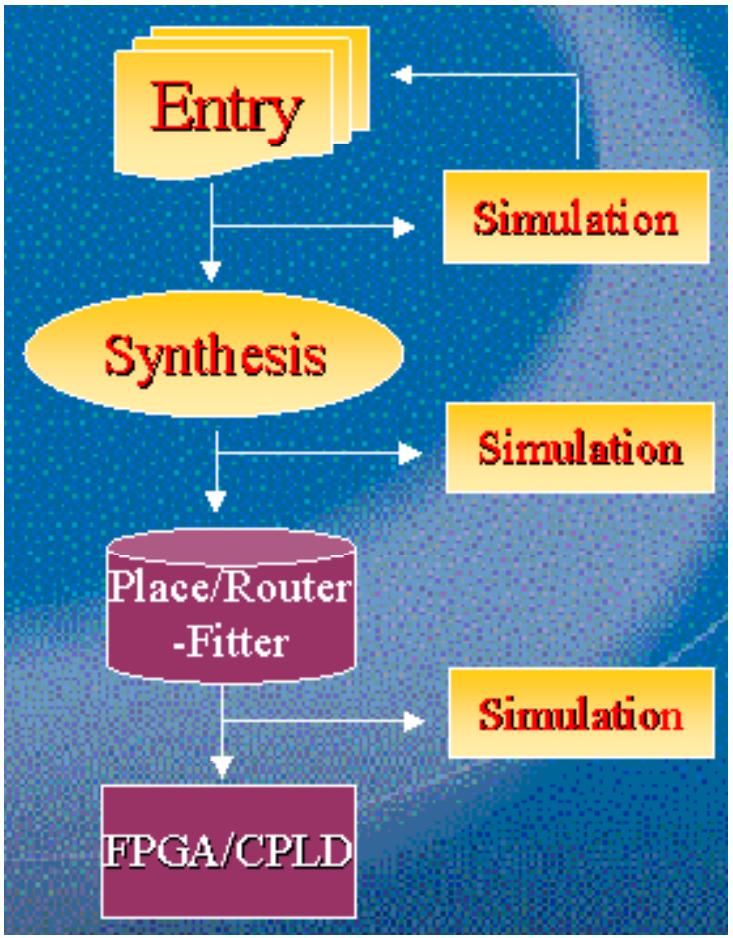


How to compile an run a HDL program

- Use GHDL, Free Linux and Window Version
  - Free, easy, command line
- Use Xilinx ISE or Vivado
  - WebPack: Downloadable, GUI
  - **We have licensed version in our Lab CS223**

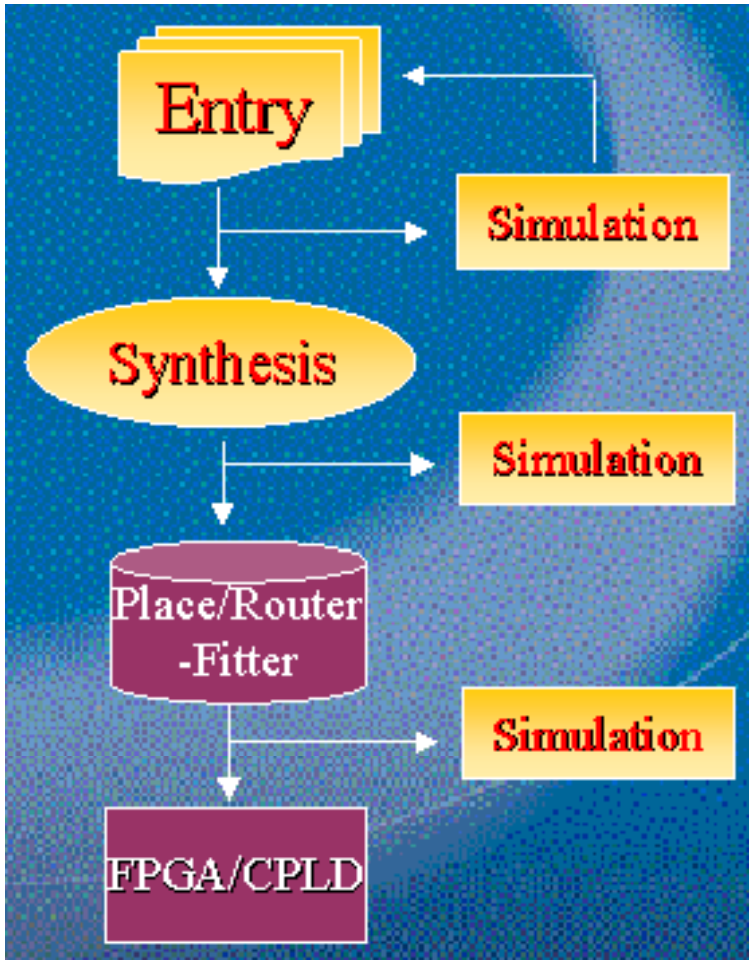
Design Flow for H/w Part

- Almost the same for all digital circuit design
- **Synthesis**
  - Different particularly in Technology mapping
    - LUT-technology mapping
    - Specific to target technology (device)



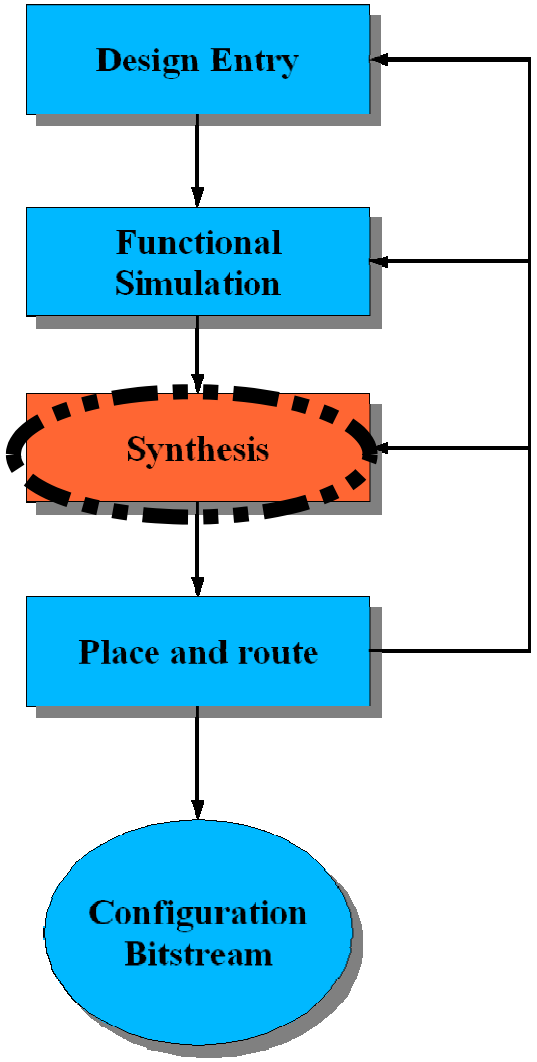
Design Flow for H/w Part

- **Design Entry**
  - Schematic Netlist
  - HDL
  - Waveform
  - State Diagram



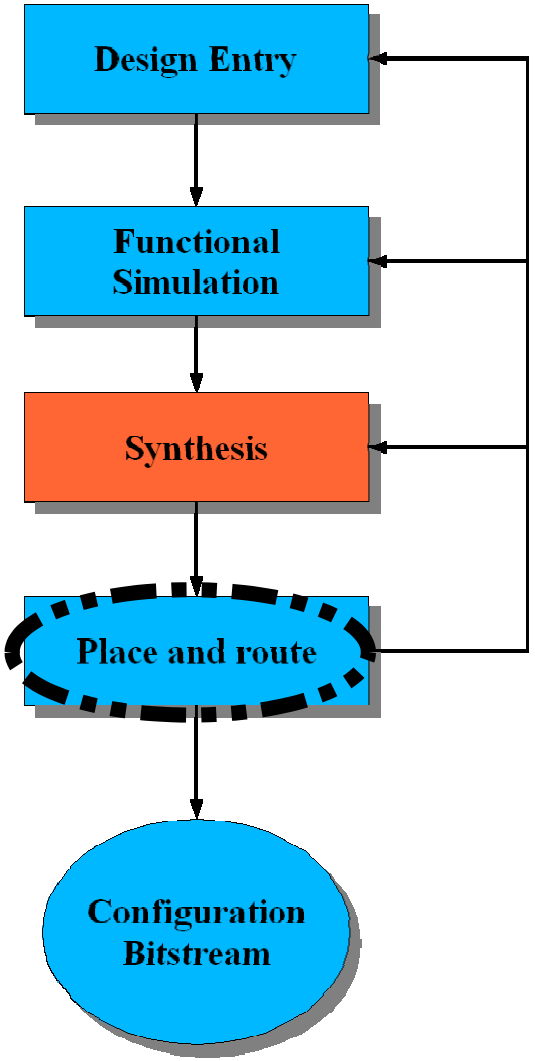
Synthesis

- **Compilation and optimization:**
  - All non-synthesizable data types and operations → synthesizable code
  - Translated into a set of Boolean equations
  - Then minimized (Technology-independent optimization)
- **Technology mapping:**
  - Assign functional modules to library elements.
  - On FPGAs:
    - Mapping control logic and datapath to LUTs and BLEs
    - Mapping optimized datapath to on-chip dedicated circuit structures)
  - Technology-dependent optimization



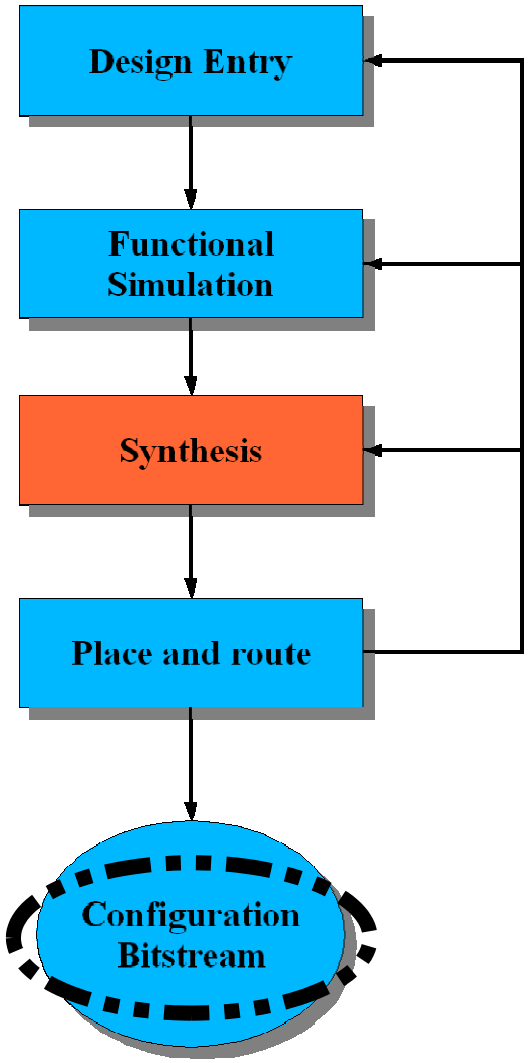
Physical Design: Place and Route

- **Place:**
  - Assign locations to the components
  - In hierarchical architectures:
    - May need a separate clustering step: to group BLEs into logic blocks
    - Clustering: prior to placement or during placement
- **Route:**
  - Provide comm. paths to the interconnections.
- **Important factors:**
  - Clock freq, Power Consumption
  - Routing congestion

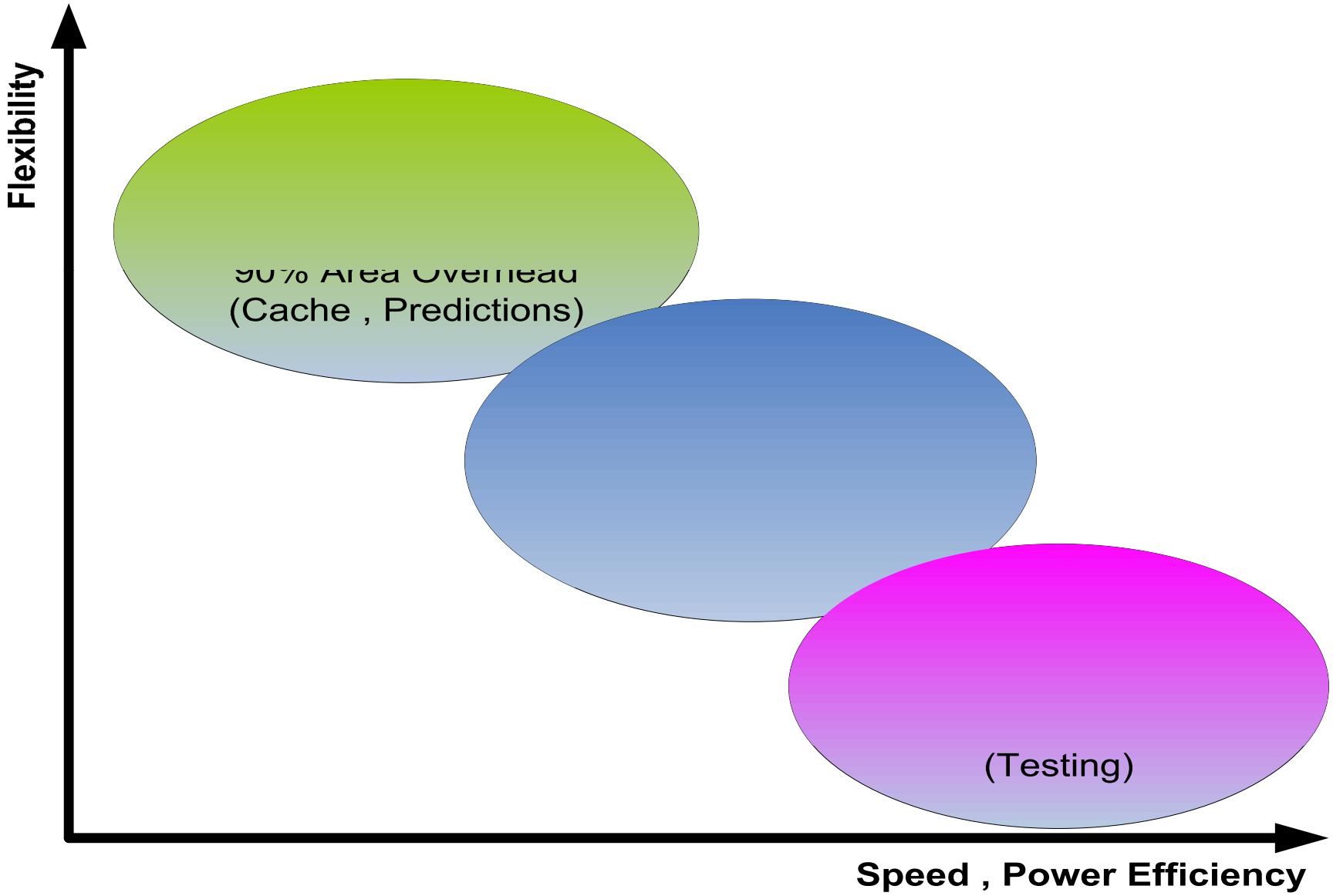


Configuration Bitstream

- **Bitstream:**
  - LUT contents,
  - Multiplexer control lines,
  - Interconnections,
  - ….

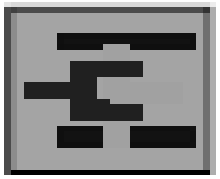


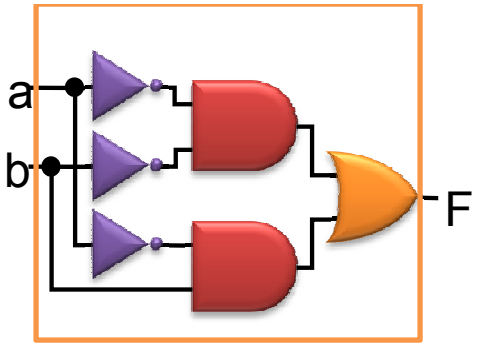
Comparison


Flexibility
90% Area Overhead
(Cache , Predictions)
(Testing)
Speed , Power Efficiency

Hardware Specification

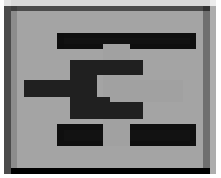
- Layout editor
  - directly enter layout
  - Up to ~10^2 of unique transistors
  - Complex circuits
  - Memory, aided by generators
- Schematic Capture
  - Enter gates and interconnections
  - Up to ~10^4 transistors
- Hardware Description Languages
  - Enter text description
  - 10^7 transistors

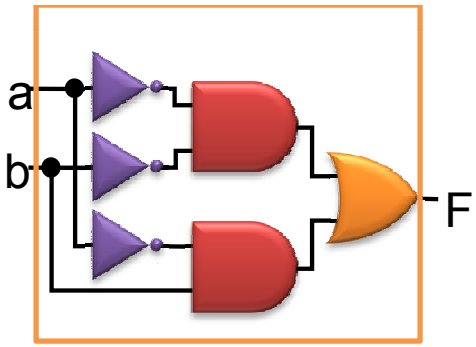





```
Entity ..
If (x < y) then
Y=x and z;
....
```

Hardware Specification



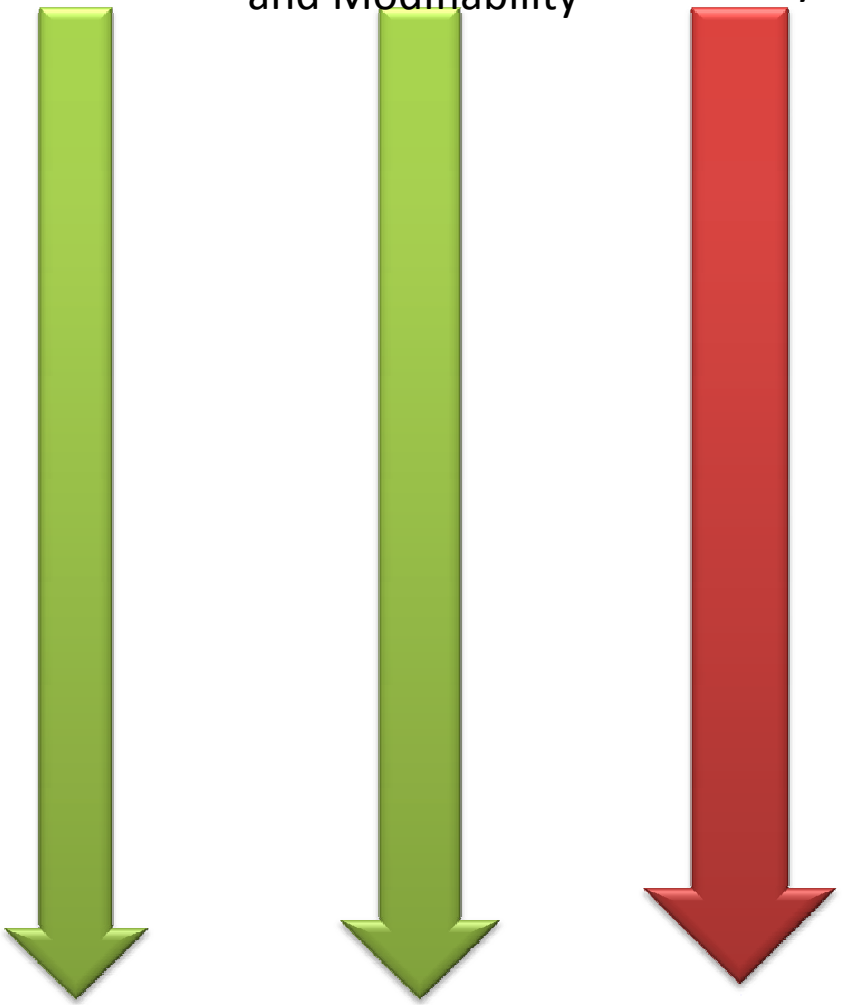



```
Entity ..
If (x < y) then
Y=x and z;
....
```

Complexity

Maintainability and Modifiability

Optimal Efficiency

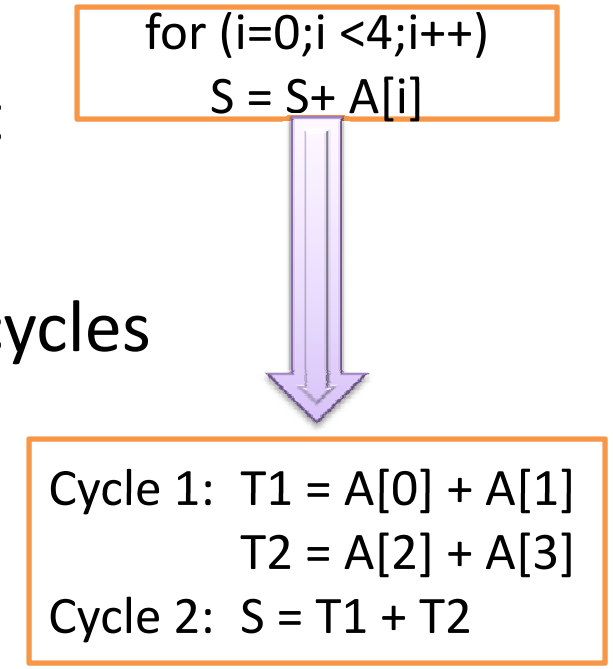


Model

- Representation of abstract view of the System
- Varying abstractions
  - functional only
  - timing only
  - functional + timing

Modeling: level of detail

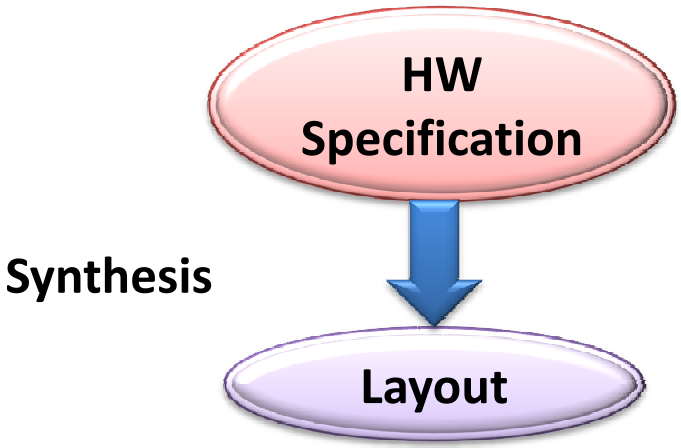
- Behavioral Level
  - no clock cycle level commitment
- Register-Transfer Level (RTL)
  - Operations committed to clock cycles

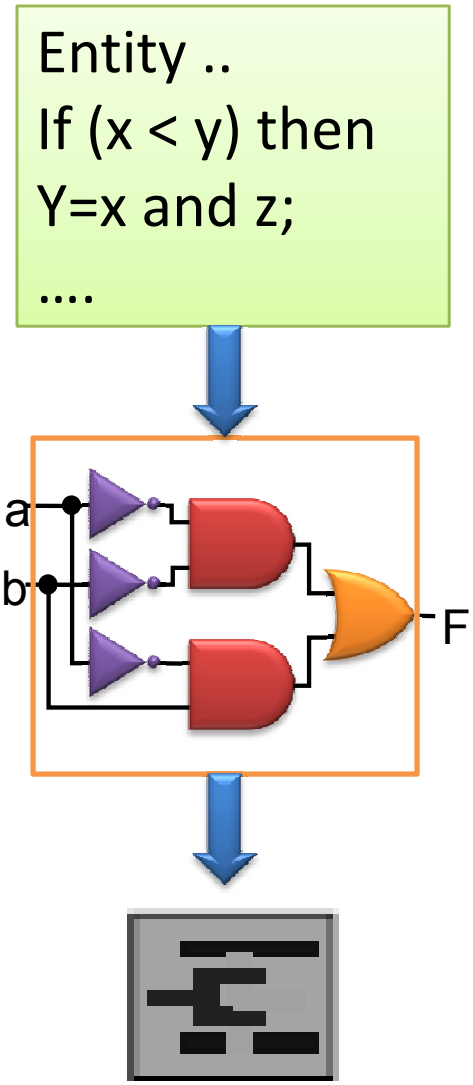


- Gate level
  - structural netlist

Synthesis

- HDL → Layout
  - HDL → Gates
  - Gates → Layout

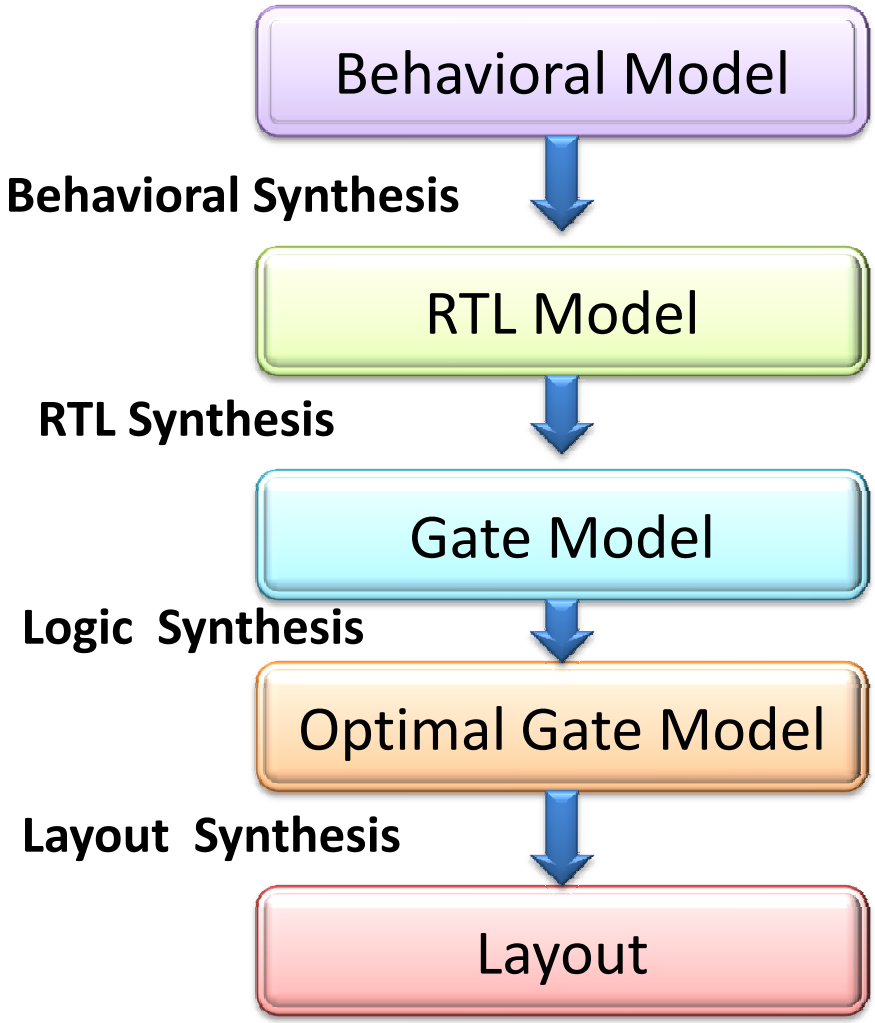




Synthesis

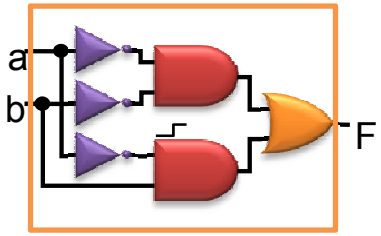
- Behavioral Synthesis (Process & Sequential)
  - Behavioral HDL → RTL HDL
  - No notion of clock to Clocked
- RTL Synthesis
  - RTL HDL → Gates
- Layout Synthesis
  - Gates → Layout

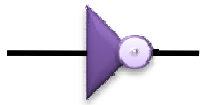
Design Flow

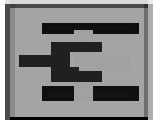



```
for (i=0;i <4;i++)
S = S+ A[i]
```

```
Cycle 1:  T1 = A[0] + A[1]
          T2 = A[2] + A[3]
Cycle 2:  S = T1 + T2
```







VHDL

~~Very Hard Difficult Language~~

VHSIC Hardware
Description Language

VHSIC --
Very High Speed Integrated Circuits

Modeling Digital Systems

- VHDL : Coding models of digital system
- Reasons for modeling
  - requirements specification
  - documentation
  - testing using simulation, formal verification
  - synthesis
- Goal
  - most ‘reliable’ design process, with minimum cost and time
  - avoid design errors!

VHDL

- VHDL is NOT C
- There are some similarities, as with any programming language
- But syntax and logic are quite different; so get over it !!

Requirement of any HDL -1

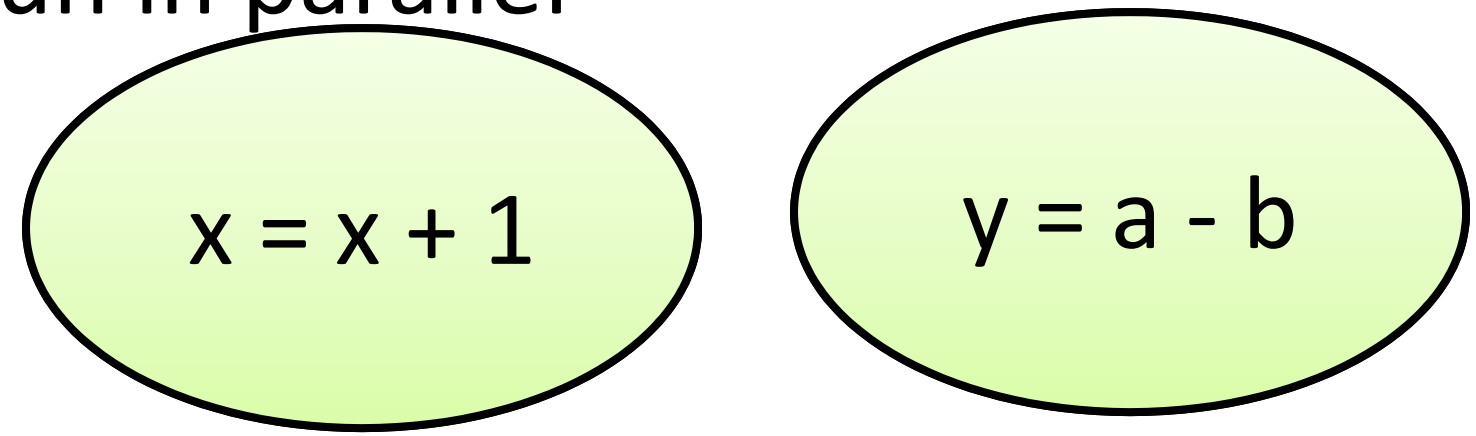
- Time
  - How the behavior of the system changes with time
  - Creating waveforms

Requirement of any HDL -2

- Periodic Signals
  - –Clocks

Requirement of any HDL -3

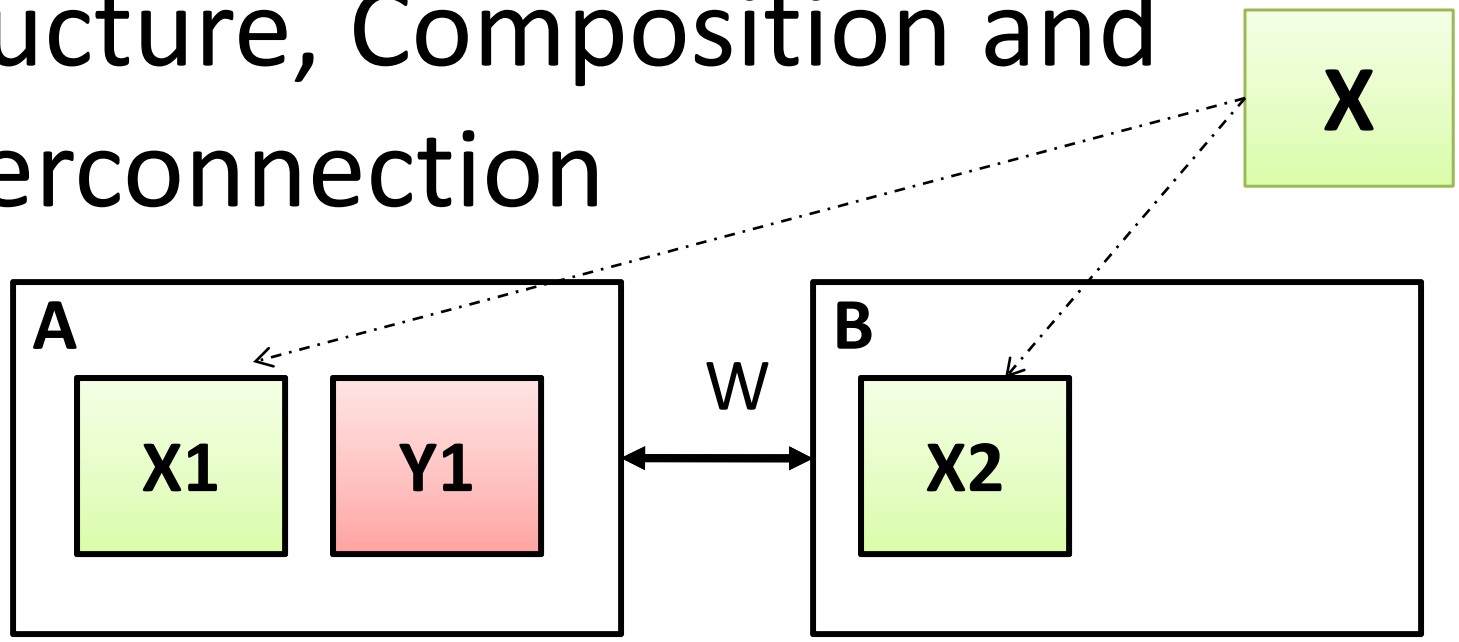
- Concurrency : All HW components run in parallel



- Specify: Processes P1 and P2 execute in parallel

Requirement of any HDL -4

- Structure, Composition and Interconnection



  - Block A consists of two blocks: X1 and Y1
  - Block X is duplicated
  - Wire W connects A and B

Requirement of any HDL -5

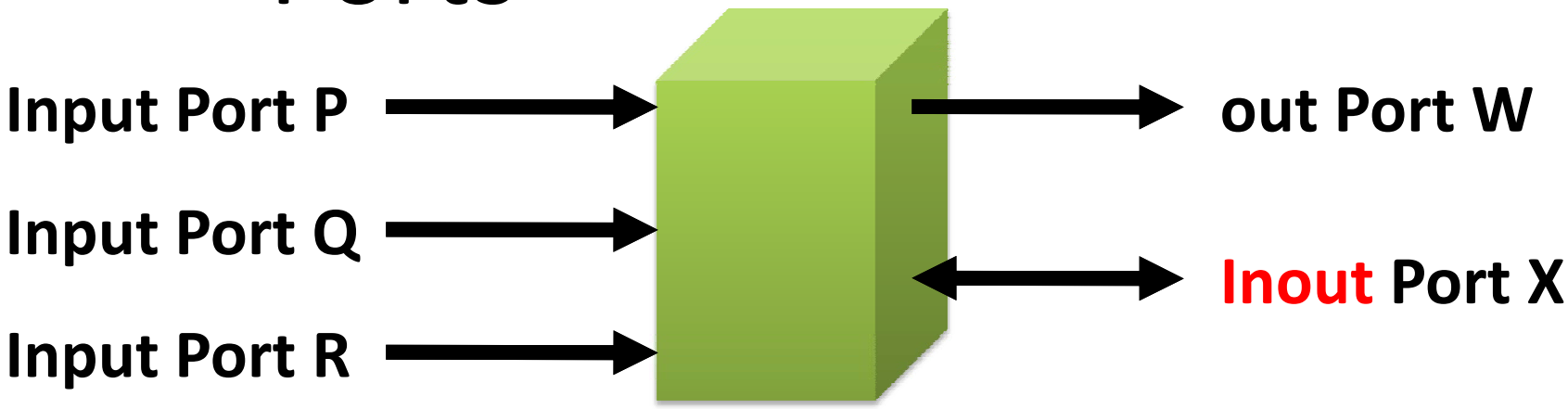
- Bit-true data types
  - –Not so important in SW
  - –Important in HW

int<6:0> var;

- Specify the bit-width of variables

Requirement of any HDL -6

- Modules and Interfaces
  - Ports



Requirement of any HDL -7

- Electrical Characteristics
  - –Current Levels
  - –Tristating
- Sensitivity
  - –Rising edge/falling edge

Requirement of any HDL -8

- Other programming constructs
  - –Text and File I/O
- Useful in simulation/debugging

Requirement of any HDL -9

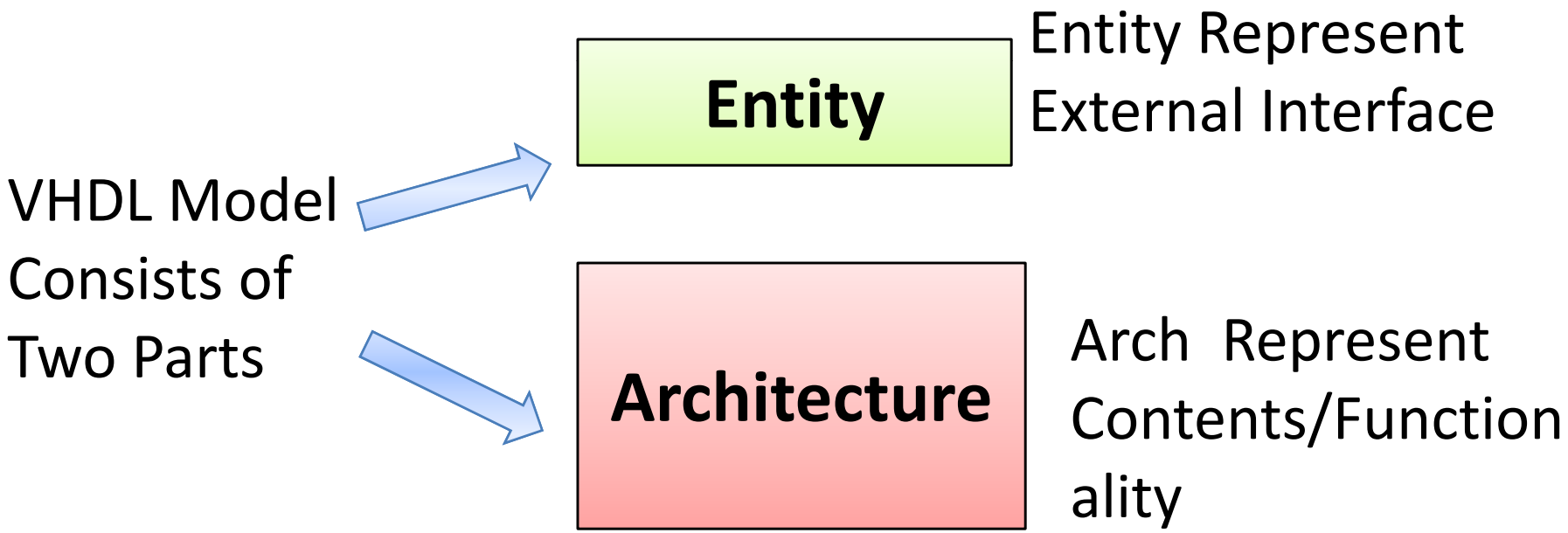
- Bit-true data types
  - –Not so important in SW
  - –Important in HW

int<6:0> var;

- Specify the bit-width of variables

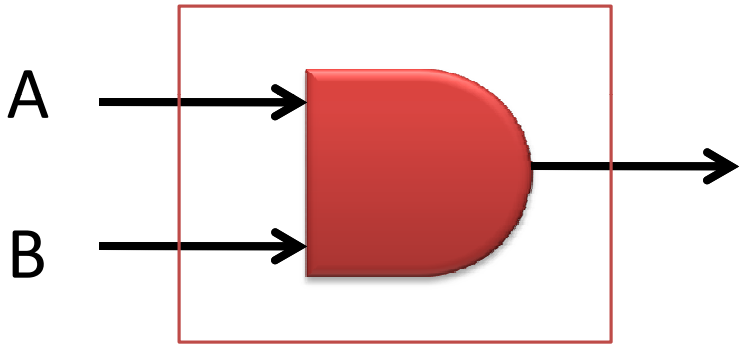
Fundamental VHDL Objects

- Entity and Architecture Pair



VHDL: Entity

- Entity : Represent External Interface




```
ENTITY    and_gate    IS
          PORT (  A:   IN   BIT;
                  B:   IN BIT;
                  Y:   OUT BIT
          );
END and_gate;
```

Model Name

Port Type

Port Direction

Port Name

Entity has Interface: No functionality

VHDL: Architecture, Specifying functionality

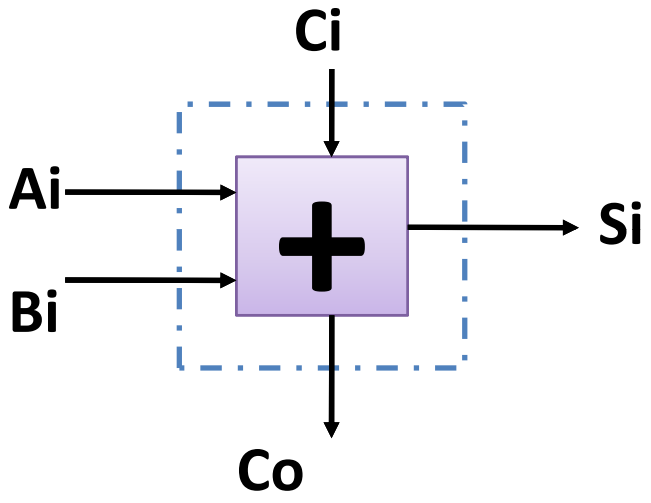
```
ARCHITECTURE data_flow OF and_gate IS

BEGIN
        y <= a AND b;
END data_flow;
```

- May have multiple architectures for given entity
  - different views
  - different levels of detail

Specifying Concurrency

- Concurrent Signal Assignments




```
ARCHITECTURE data_flow   OF full_adder IS
BEGIN
        si <= ai XOR bi XOR ci;
        co <= (ai AND bi) OR (bi AND ci) OR (ai AND ci);
END data_flow;
```

Order of Execution

- Execution independent of Specification

```
ARCHITECTURE data_flow  OF full_adder IS
BEGIN
        si <= ai XOR bi XOR ci;
        co <= (ai AND bi) OR (bi AND ci) OR (ai AND ci);
END data_flow;
```

```
ARCHITECTURE data_flow  OF full_adder IS
BEGIN
        co <= (ai AND bi) OR (bi AND ci) OR (ai AND ci);
        si <= ai XOR bi XOR ci;
END data_flow;
```

Thanks