Overview of this Lecture

• Types of CAD tools
• Why HDLs are needed?
Recap

Gajski’s Chart

Abstraction Levels
Recap – Design Representation

```
module detector (Xin,clk,R,I,Zout);
input Xin,clk,R,I;
output Zout;
reg Y1,Y0;

initial begin
  Y1 = 1'b0;    Y0 = 1'b0;
end

always @(posedge clk or negedge R) begin
  if(R == 1'b0)
    begin
      Y1 = 1'b0; Y0 = 1'b0;
    end
  else begin
    Y1 = Xin; Y0 = I;
  end
end
assign Zout = Y0 & ((Y1 & !Xin) | (Y1 & Xin))
endmodule
```
Need for CAD Tools

- Current systems are very complex.
- Design abstraction and decomposition is done to manage complexity.
- Tools automate the process of converting your design from one abstraction level to another.
- Design Automation Tools improve productivity.
- Different tools are required in different steps.
Classification of CAD Tools

• Editors
  – Allows specification of the design either textually or graphically.

• Simulators
  – Models the response of a system to input stimuli.

• Analyzers
  – Used at different levels to check for correctness and compliance to rules.

• Synthesis
  – Transformation of representation between different abstraction levels.
Flow and Tools

Specifications

X = AB; Y = CD;
Z = X+Y;

Functional Design

Logic Design

Circuit Design

Physical Design

Fabrication

Specifications

System Description Languages

Hardware Description Languages, Schematic Editors

Logic Synthesis Tools
(Tech mappers)

Physical Synthesis Tools
(Place & Route)

Tape out and Manufacture
CAD Tools – 1. Design Entry

• Graphical
  – Silicon Level – To create layouts
    • e.g. Magic
  – Other Levels
    • e.g. ViewLogic, Protel

• Text
  – Natural language specification at system level.
  – Hardware Description Languages at Chip, Register and Gate levels.
    • e.g. VHDL, Verilog
  – Circuit Level
    • e.g. SPICE
Graphical Editors

- **Silicon Level editors are called Layout editors.**
  - Draw rectangles describing metal, poly, diffusion etc
  - Library components are also at the same level.
  - Usually has online Design Rule Checking (DRC).

- **Graphical Editors at other levels are usually called Schematic editors.**
  - Used to create block diagrams and schematics.
  - The process is usually called *Schematic Capture*. 
Schematic Editors

• Can create and display graphical components called “tokens”
• Can “interconnect” these tokens.
• Advantage :
  – Gives a structural representation called “netlist” describing the components used and their interconnections.
  – Also provides a simulation model to find the system’s response for different stimuli.
Important Features in Schematic Editors

1. Library

- A list of library components constituting the primitives.
- Every library component has two models:
  - The schematic with the I/O of the primitive
  - A simulation model describing the behavior
- The library components can be
  - *generic* – allows retargeting
  - *specific* – tied to specific targets e.g. ASIC library components, macros etc.
Important Features in Schematic Editors

2. GUI

• Start with an empty root window.
• Select the library component to be added.
• Label the component and place it in the window.
• Change properties of the component if necessary.
  – e.g. Footprint of a component for board design
• Create wiring
  – either by labeling the pins with a logical name
  – or by “drawing a line” physically to establish interconnection.
Example of Schematic Entry

13

Example of Schematic Entry

- **label**
- **gate primitive**
- **wire**
Text Based Design Entry

• Choose a specific HDL.
• Use text editors to describe the design.
  – e.g. vi, emacs, notepad etc.
  – Some tools have built-in editors
• Enter your design conforming to the language lexicon, syntax and semantics.
• Check for errors.
• “Compile” to get a simulation model.
What Makes HDLs Different?

• Hardware systems are concurrent in nature.
• Hardware systems may be distributed in nature.
  – Many components
  – Different rates for processing data, different clocks.
• Hardware systems are timed.
  – All hardware components have inherent delays and hence managing timing is crucial.
• Traditional software design techniques are insufficient.