EEE205 – Digital Electronics (II)

Lecture 4

Dr. Ming Xu

Dept of Electrical & Electronic Engineering

XJTLU

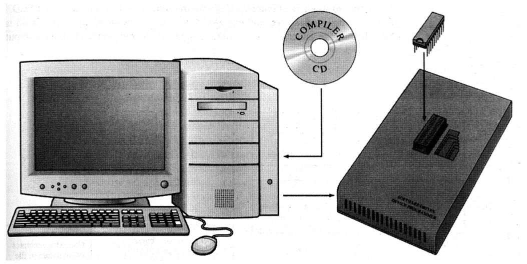
In This Session

- PLD Programming
- Hardware Description Languages (HDLs)

PLD Programming

The three components required:

- A computer
- A CAD system
- A programmer or a JTAG interface that is connected with the computer through a cable.



PLD Programming

The CAD System

- Design entry
- Synthesis and optimization
- Simulation
- Physical design
- Programming

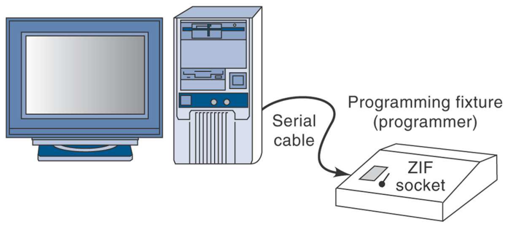
Examples

- Xilinx ISE
- Altera Quartus II
- Altera MAX+plus II

PLD Programming

Programming by a Programmer

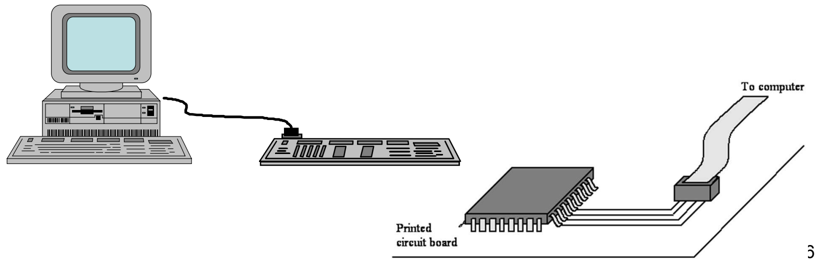
- The PLD is removed from its circuit board and placed into a programmer for configuration
- It is mainly used for SPLDs.



PLD Programming

In-System Programming (ISP)

- The PLD is attached to its circuit board while being programmed through a 4-wire JTAG port.
- There must be dedicated pins on the PLD for the JTAG port.
- It is mainly used for CPLDs and FPGAs.



PLD Programming

The design entry methods may include:

- Schematic capture
- Truth Tables
- Hardware Description Languages (HDLs)

An Overview of HDLs

- A **H**ardware **D**escription **L**anguage (HDL) is a computer language that is used to describe hardware.
- Unlike schematic capture, HDL based designs are highly portable and independent of technology.

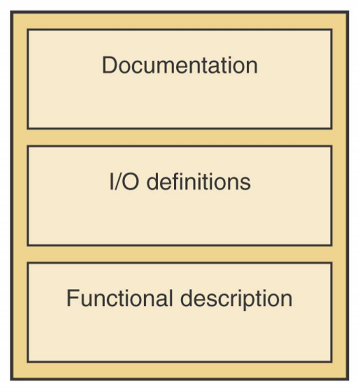
An Overview of HDLs

Examples

- Verilog HDL (**Veri**fy **Log**ic, IEEE standard)
- VHDL (**V**ery High Speed Integrated Circuit HDL, IEEE standard)
- AHDL (**A**ltera HDL)
- ABEL
- CUPL

AHDL will be introduced in this module.

AHDL Overview



Format of AHDL files:

- Documentation is lines of comments.
- I/O definitions
  - Mode – whether a port is input, output, or both.
  - Type – the number of bits
- Functional description is the definition of the circuit's operation.

AHDL Overview

Example: an AND gate

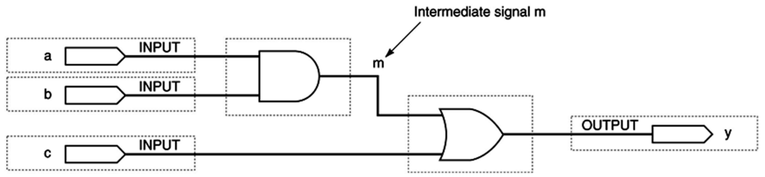
```
SUBDESIGN and_gate
(
   a, b      :INPUT;
   y         :OUTPUT;
)
BEGIN
    y = a & b;
END;
```

AHDL Overview

- **SUBDESIGN** names the circuit block, in this case: and_gate
- The definitions of inputs (**INPUT**), outputs (**OUTPUT**) or bidirectional port (**BIDIR**) are enclosed in parenthesis.
- The description of operation is between the **BEGIN** and **END** keywords.
- Boolean operators: **!** For NOT, **&** for AND, **#** for OR, and **$** for XOR.

Intermediate Signals

- There are points in the circuit that are neither inputs nor outputs for the block.
- They are available only within this block rather than to other block.
- They are called **buried nodes** by AHDL.



Intermediate Signals

AHDL Buried Nodes

```
%  Intermediate variables in AHDL (Figure 3-49)
   MAY 23, 2005          %
SUBDESIGN fig3_50
(
   a,b,c      :INPUT;   -- define inputs to block
   y          :OUTPUT;  -- define block output
)
VARIABLE
   m        :NODE;      -- name an intermediate signal
BEGIN
   m = a & b;            -- generate buried product term
   y = m # c;            -- generate sum on output
END;
```

Intermediate Signals

AHDL Buried Nodes

- Comments are enclosed by a pair of **%** characters or after two dashes.
- The optional **VARIABLE** Section is used to declare and/or generate variables, e.g. buried nodes, primitives, etc.
- NODE designates the type of the variable. Other types include TRI_STATE_NODE, (primitives) LATCH, DFF, JKFF.

AHDL Overview

AHDL vs. Programming Languages

- All the statements between BEGIN and END are evaluated constantly and **concurrently** (at the same time).
- The order in which they are listed makes no difference.
- In the contrast, a computer programming language follows instructions in **sequential** order.