IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

VHDL – An Introduction

Santanu Chattopadhyay
Electronics and Electrical Communication Engineering

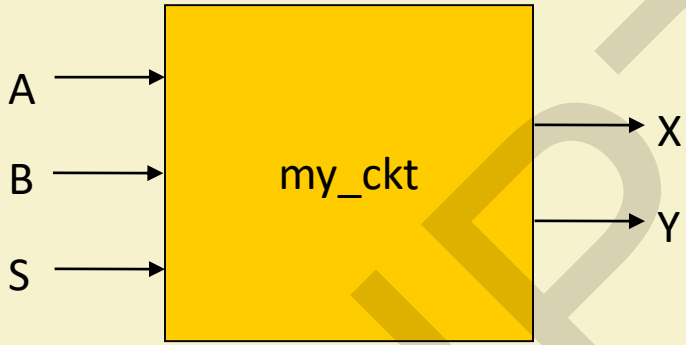
Introduction

- Hardware description languages (HDL)
  - Language to describe hardware
  - Two popular languages
    - VHDL: **V**ery High Speed Integrated Circuits **H**ardware **D**escription **L**anguage
      - Developed by DOD from 1983
      - IEEE Standard 1076-1987/1993/200x
      - Based on the ADA language
    - Verilog
      - IEEE Standard 1364-1995/2001/2005
      - Based on the C language

Applications of HDL

- Model and document digital systems
  - Different levels of abstraction
    - Behavioral, structural, etc.
- Verify design
- Synthesize circuits
  - Convert from higher abstraction levels to lower abstraction levels

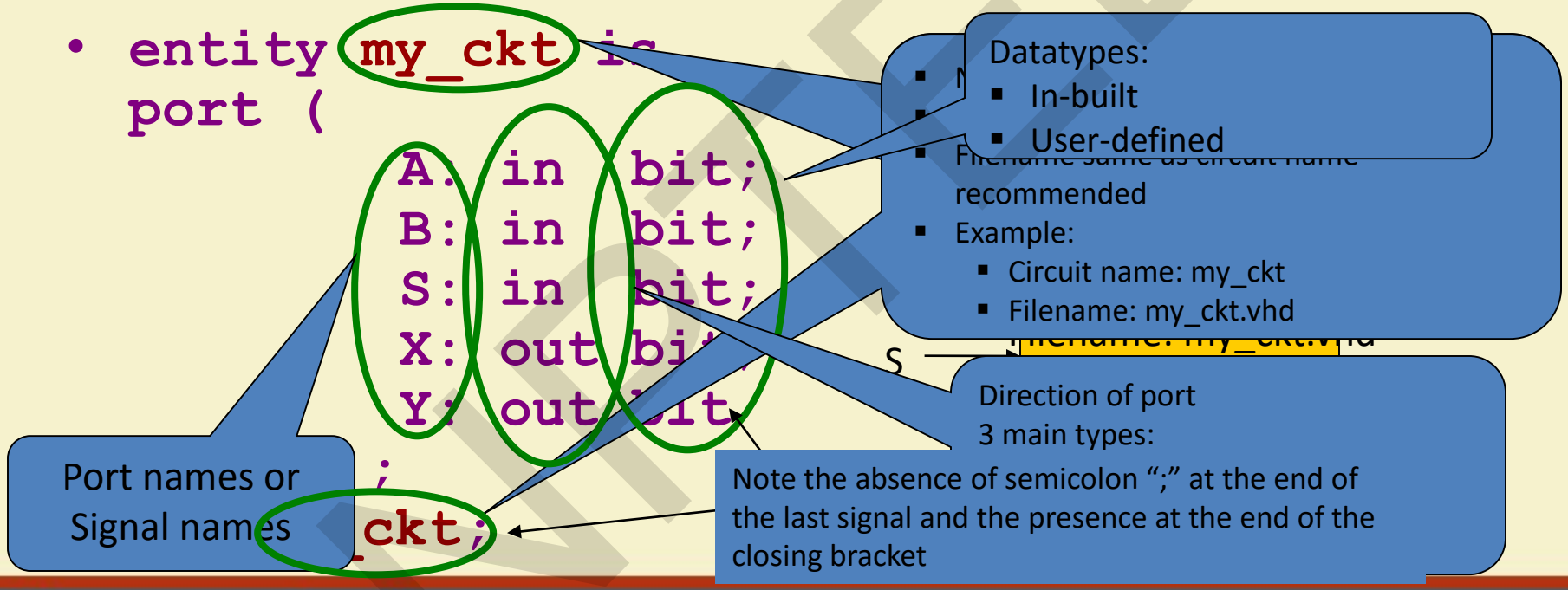
Input-Output specification of circuit



- Example: my_ckt
  - Inputs: A, B, C
  - Outputs: X, Y
- VHDL description:

```
entity my_ckt is
port (
        A: in bit;
        B: in bit;
        S: in bit;
X: out bit;
        Y: out bit);
end my_ckt ;
```

VHDL entity


entity my_ckt is
port (
A: in bit;
B: in bit;
S: in bit;
X: out bit;
Y: out bit
);
end my_ckt;
Datatypes:
In-built
User-defined
recommended
Example:
Circuit name: my_ckt
Filename: my_ckt.vhd
Direction of port
3 main types:
Port names or
Signal names
Note the absence of semicolon ";" at the end of the last signal and the presence at the end of the closing bracket

Built-in Datatypes

- Scalar (single valued) signal types:
  - **bit**
  - **boolean**
  - **integer**
  - Examples:
    - **A: in bit;**
    - **G: out boolean;**
    - **K: out integer range -2**4 to 2**4-1;**
- Aggregate (collection) signal types:
  - **bit_vector**: array of bits representing binary numbers
  - **signed**: array of bits representing signed binary numbers
  - Examples:
    - **D: in bit_vector(0 to 7);**
    - **E: in bit_vector(7 downto 0);**
    - **M: in signed (4 downto 0);**
      --signed 5 bit_vector binary number

User-defined datatype

- Construct datatypes arbitrarily or using built-in datatypes
- Examples:
  - `type temperature is (high, medium, low);`
  - `type byte is array(0 to 7) of bit;`

Functional specification

- Example:
  - Behavior for output X:
    - When S = 0
      X <= A
    - When S = 1
      X <= B
  - Behavior for output Y:
    - When X = 0 and S =0
      Y <= ‘1’
    - Else
      Y <= ‘0’

VHDL Architecture

- VHDL description (sequential behavior):

```
architecture arch_name of my_ckt is
begin
 p1: process (A,B,S)
 begin
  if (S='0') then
   X <= A;
  else
   X <= B;
  end if;

  if ((X = '0') and (S = '0')) then
   Y <= '1';
  else
   Y <= '0';
  end if;

 end process p1;
end;
```

Error: Signals defined as output ports can only be driven and not read

VHDL Architecture

```
architecture behav_seq of my_ckt is

    signal Xtmp: bit;

    begin
     p1: process (A,B,S,Xtmp)
       begin
       if (S='0') then
        Xtmp <= A;
       else
        Xtmp <= B;
       end if;

       if ((Xtmp = '0') and (S = '0')) then
        Y <= '1';
       else
        Y <= '0';
       end if;

       X <= Xtmp;
     end process p1;
end;
```

Signals can only be defined in this place before the **begin** keyword

General rule: Include all signals in the sensitivity list of the process which either appear in relational comparisons or on the right side of the assignment operator inside the process construct.

In our example:

Xtmp and **S** occur in relational comparisons

A, **B** and **Xtmp** occur on the right side of the assignment operators

VHDL Architecture

- VHDL description (concurrent behavior):

```
architecture behav_conc of my_ckt is
   signal Xtmp: bit;
begin
   Xtmp <= A when (S='0') else
           B;
   Y <= '1' when ((Xtmp = '0') and (S = '0')) else
        '0';
   X <= Xtmp;
end ;
```

Signals vs Variables

- Signals
  - Signals follow the notion of ‘event scheduling’
  - An event is characterized by a (time,value) pair
  - Signal assignment example:

    `X <= Xtmp;` means

    Schedule the assignment of the value of signal `Xtmp` to signal `X` at (Current time + delta)

    where delta: infinitesimal time unit used by simulator for processing the signals

Signals vs Variables

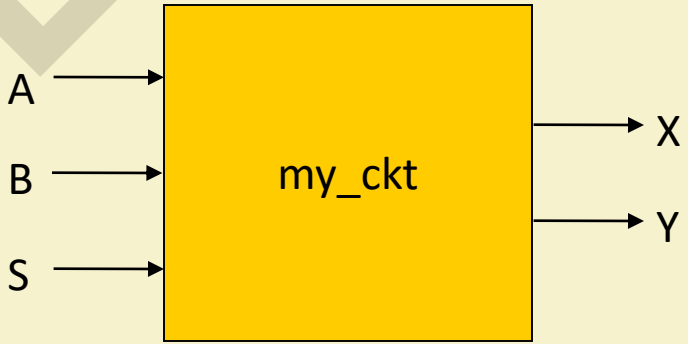
- Variables
  - Variables do not have notion of ‘events’
  - Variables can be defined and used only inside the **process** block and some other special blocks.
  - Variable declaration and assignment example:

```
process (…)
variable K : bit;
 begin
   …
  -- Assign the value of signal
   K := L;
  …
end process;
```

Variables can only be defined and used inside the **process** construct and can be defined only in this place

Simulation

- Simulation is modeling the output response of a circuit to given input stimuli
- For our example circuit:
  - Given the values of A, B and S
  - Determine the values of X and Y
- Many types of simulators used
  - Event driven simulator is used popularly
  - Simulation tool we shall use: ModelSim



Simulation

```
architecture behav_seq of my_ckt is
     signal Xtmp: bit;

     begin
      p1: process (A,B,S,Xtmp)
       variable XtmpVar: bit;
      begin

       if (S='0') then
        Xtmp <= A;
       else
        Xtmp <= B;
       end if;

       if ((Xtmp = '0') and (S = '0')) then
        Y <= '1';
       else
        Y <= '0';
       end if;

       X <= Xtmp;
       XtmpVar := Xtmp;
      end process p1;
end;
```

Time 'T'	A	B	S	Xtmp	Y	XtmpVar	X
0-	U	U	U	'X'	'X'	'X'	'X'
0	0	1	0	'X'	'X'	'X'	'X'
0+d	0	1	0	0	0	0	'X'
0+2d	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0
1+d	0	1	1	1	0	0	0
		1	1	1	0	0	1

Scheduled events

Scheduled events executed:
Xtmp = 0
Y = 1
X = 0

Assignments executed:
XtmpVar = 0

Scheduled events list:
(empty)

X = ('0',0+2d)

Synthesis

- Synthesis:
  Conversion of behavioral level description to structural level netlist
  - Abstract behavioral description maps to concrete logic-level implementation
  - For ex. Integers at behavioral level mapped to bits at structural level
- Structural level netlist
  - Implementation of behavioral description
  - Describes interconnection of gates
- CAD tools are used to perform synthesis

Structural level netlist

- Behavior of our example circuit:
  - Behavior for output X:
    - When S = 0
      X <= A
    - When S = 1
      X <= B
  - Behavior for output Y:
    - When X = 0 and S =0
      Y <= 1
    - Else
      Y <= 0

- Logic functions
  - Sbar = ~ S
  - Xbar = ~ X
  - X = A*(Sbar) + B*S
  - Y = (Xbar)*(Sbar)

Structural level netlist

```
architecture behav_conc of my_ckt is
   -- component declarations

   signal Sbar, Xbar, W1, W2: bit;

begin

   G1: not port map(Sbar,S);
   G2: and port map(W1,A,Sbar);
   G3: and port map(W2,B,S);
   G4: or  port map(X,W1,W2);

   G5: not port map(Xbar,X);
   G6: and port map(Y,Xbar,Sbar);

end ;
```

- Gate level VHDL descriptions (**and, or,** etc) are described separately
- Design in which other design descriptions are included is called a “hierarchical design”
- A VHDL design is included in current design using port map statement

Other VHDL resources

- VHDL mini-reference *by Prof. Nelson*
  - http://www.eng.auburn.edu/department/ee/mgc/vhdl.html
- VHDL Tutorial: Learn by Example *by Weijun Zhang*
  - http://esd.cs.ucr.edu/labs/tutorial/

8085 Microprocessor

Santanu Chattopadhyay
Electronics and Electrical Communication Engineering

Basic Concepts of Microprocessors

- Differences between:
  - Microcomputer – a computer with a microprocessor as its CPU. Includes memory, I/O etc.
  - Microprocessor – silicon chip which includes ALU, register circuits & control circuits
  - Microcontroller – silicon chip which includes microprocessor, memory & I/O in a single package.

What is a Microprocessor?

- The word comes from the combination micro and processor.
  - Processor means a device that processes whatever. In this context processor means a device that processes numbers, specifically binary numbers, 0's and 1's.
    - To process means to manipulate. It is a general term that describes all manipulation. Again in this content, it means to perform certain operations on the numbers that depend on the microprocessor's design.

What about micro?

- Micro is a new addition.
  - In the late 1960's, processors were built using discrete elements.
    - These devices performed the required operation, but were too large and too slow.
  - In the early 1970's the microchip was invented. All of the components that made up the processor were now placed on a single piece of silicon. The size became several thousand times smaller and the speed became several hundred times faster. The "Micro"Processor was born.

Was there ever a “mini”- processor?

- No.
  - – It went directly from discrete elements to a single chip. However, comparing today’s microprocessors to the ones built in the early 1970’s you find an extreme increase in the amount of integration.

- So, What is a microprocessor?

Definition of the Microprocessor

- The microprocessor is a programmable device that takes in numbers, performs on them arithmetic or logical operations according to the program stored in memory and then produces other numbers as a result.

Definition (Contd.)

- Lets expand each of the underlined words:
  - Programmable device: The microprocessor can perform different sets of operations on the data it receives depending on the sequence of instructions supplied in the given program.
  - By changing the program, the microprocessor manipulates the data in different ways.
  - Instructions: Each microprocessor is designed to execute a specific group of operations. This group of operations is called an instruction set. This instruction set defines what the microprocessor can and cannot do.

Definition (Contd.)

- Takes in: The data that the microprocessor manipulates must come from somewhere.
  - It comes from what is called “input devices”.
  - These are devices that bring data into the system from the outside world.
  - These represent devices such as a keyboard, a mouse, switches, and the like.

Definition (Contd.)

- Numbers: The microprocessor has a very narrow view on life. It only understands binary numbers.
- A binary digit is called a bit (which comes from **b**inary dig**it**).
- The microprocessor recognizes and processes a group of bits together. This group of bits is called a "word".
- The number of bits in a Microprocessor's word, is a measure of its "abilities".

Definition (Contd.)

- Words, Bytes, etc.
  - The earliest microprocessor (the Intel 8088 and Motorola's 6800) recognized 8-bit words.
  - Later microprocessors (8086 and 68000) were designed with 16-bit words.
    - A group of 8-bits were referred to as a "half-word" or "byte".
    - A group of 4 bits is called a "nibble".
    - Also, 32 bit groups were given the name "long word".
  - Today, all processors manipulate at least 32 bits at a time and there exists microprocessors that can process 64, 80, 128 bits

Definition (Contd.)

- Arithmetic and Logic Operations:
  - Every microprocessor has arithmetic operations such as add and subtract as part of its instruction set.
    - Most microprocessors will have operations such as multiply and divide.
    - Some of the newer ones will have complex operations such as square root.
  - In addition, microprocessors have logic operations as well. Such as AND, OR, XOR, shift left, shift right, etc.
  - Again, the number and types of operations define the microprocessor's instruction set and depends on the specific microprocessor.

Definition (Contd.)

- Stored in memory :
  - First, what is memory?
    - Memory is the location where information is kept while not in current use.
    - Memory is usually measured by the number of bytes it can hold.
    - KB, MB, GB etc.

Definition (Contd.)

– Stored in memory:

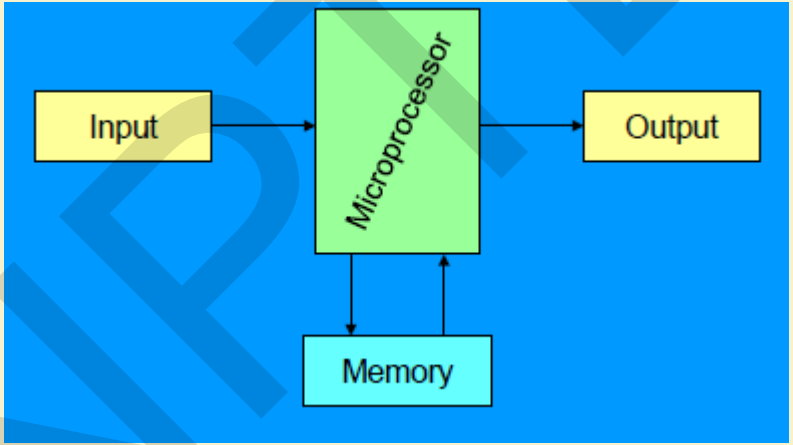
- When a program is entered into a computer, it is stored in memory. Then as the microprocessor starts to execute the instructions, it brings the instructions from memory one at a time.
- Memory is also used to hold the data.
  - The microprocessor reads (brings in) the data from memory when it needs it and writes (stores) the results into memory when it is done.

Definition (Contd.)

- Produces: For the user to see the result of the execution of the program, the results must be presented in a human readable form.
  - The results must be presented on an output device.
  - This can be the monitor, a paper from the printer, a simple LED or many other forms.

A Microprocessor-based system

- From the above description, we can draw the following block diagram to represent a microprocessor-based system:

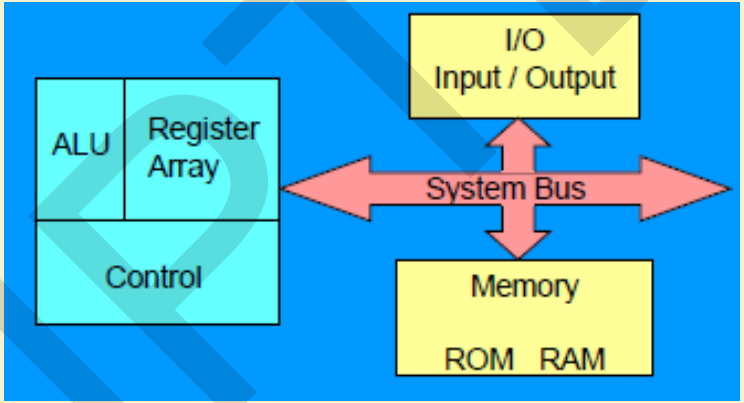


Inside The Microprocessor

- Internally, the microprocessor is made up of 3 main units.
  - The Arithmetic/Logic Unit (ALU)
  - The Control Unit.
  - An array of registers for holding data while it is being manipulated.

Organization of a microprocessor- based system

- Let's expand the picture a bit.

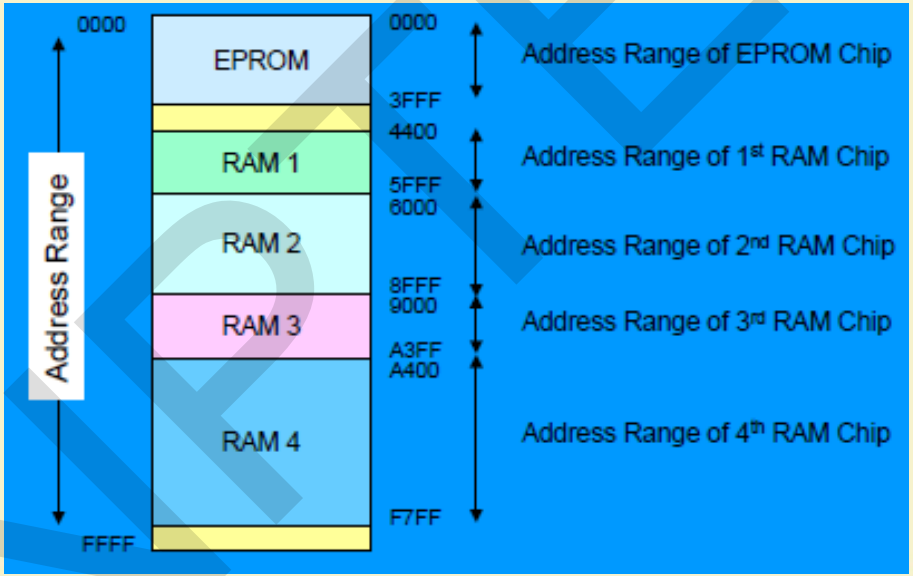


Memory

- Memory stores information such as instructions and data in binary format (0 and 1). It provides this information to the microprocessor whenever it is needed.
- Usually, there is a memory "sub-system" in a microprocessor-based system. This sub-system includes: Registers, ROM, RAM

Memory Map and Addresses

- The memory map is a picture representation of the address range and shows where the different memory chips are located within the address range.



Memory

- To execute a program:
  - the user enters its instructions in binary format into the memory.
  - The microprocessor then reads these instructions and whatever data is needed from memory, executes the instructions and places the results either in memory or produces it on an output device.

The three cycle instruction execution model

- To execute a program, the microprocessor "reads" each instruction from memory, "interprets" it, then "executes" it.
- To use the right names for the cycles:
  - The microprocessor **fetches** each instruction,
  - **decodes** it,
  - Then **executes** it.
- This sequence is continued until all instructions are performed.

Machine Language

- The number of bits that form the "word" of a microprocessor is fixed for that particular processor.
  - These bits define a maximum number of combinations.
- However, in most microprocessors, not all of these combinations are used.
  - Certain patterns are chosen and assigned specific meanings.
  - Each of these patterns forms an instruction for the microprocessor.
  - The complete set of patterns makes up the microprocessor's machine language.

The 8085 Machine Language

- The 8085 (from Intel) is an 8-bit microprocessor.
  - The 8085 uses a total of 246 bit patterns to form its instruction set.
  - These 246 patterns represent only 74 instructions.
  - Because it is very difficult to enter the bit patterns correctly, they are usually entered in hexadecimal instead of binary.
    - For example, the combination 0011 1100 which translates into "increment the number in the register called the accumulator", is usually entered as 3C.

Assembly Language

- Entering the instructions using hexadecimal is quite easier than entering the binary combinations.
  - However, it still is difficult to understand what a program written in hexadecimal does.
  - So, each company defines a symbolic code for the instructions.
  - These codes are called "mnemonics".
  - The mnemonic for each instruction is usually a group of letters that suggest the operation performed.

Assembly Language

- Using the same example from before,
  - 00111100 translates to 3C in hexadecimal (OPCODE)
  - Its mnemonic is: “INR A”.
  - INR stands for “increment register” and A is short for accumulator.
- Another example is: 1000 0000,
  - Which translates to 80 in hexadecimal.
  - Its mnemonic is “ADD B”.
  - “Add register B to the accumulator and keep the result in the accumulator”.

Assembly Language

- It is important to remember that a machine language and its associated assembly language are completely machine dependent.
- For example, Motorolla has an 8-bit microprocessor called the 6800.
  - The 8085 machine language is very different from that of the 6800. So is the assembly language.
  - A program written for the 8085 cannot be executed on the 6800 and vice versa.

“Assembling” The Program

- How does assembly language get translated into machine language?
  - There are two ways:
  - 1st there is “**hand assembly**”.
    - The programmer translates each assembly language instruction into its equivalent hexadecimal code (machine language). Then the hexadecimal code is entered into memory.
  - The other possibility is a program called an “**assembler**”, which does the translation automatically.

8085 Microprocessor Architecture

- 8-bit general purpose µp
- Capable of addressing 64 k of memory
- Has 40 pins
- Requires +5 v power supply
- Can operate with 3 MHz clock
- 8085 upward compatible

8085 Pin Diagram