Hardware Description Language - Introduction

- HDL is a language that describes the hardware of digital systems in a textual form.
- It resembles a programming language, but is specifically oriented to describing hardware structures and behaviors.
- The main difference with the traditional programming languages is HDL's representation of extensive parallel operations whereas traditional ones represents mostly serial operations.
- The most common use of a HDL is to provide an alternative to schematics.

HDL – Introduction (2)

- When a language is used for the above purpose (i.e. to provide an alternative to schematics), it is referred to as a *structural description* in which the language describes an interconnection of components.
- Such a structural description can be used as input to logic simulation just as a schematic is used.
- Models for each of the primitive components are required.
- If an HDL is used, then these models can also be written in the HDL providing a more uniform, portable representation for simulation input.

HDL – Introduction (3)

- HDL can be used to represent logic diagrams, Boolean expressions, and other more complex digital circuits.
- Thus, in top down design, a very high-level description of a entire system can be precisely specified using an HDL.
- This high-level description can then be refined and partitioned into lower-level descriptions as a part of the design process.

HDL – Introduction (4)

- As a documentation language, HDL is used to represent and document digital systems in a form that can be read by both humans and computers and is suitable as an exchange language between designers.
- The language content can be stored and retrieved easily and processed by computer software in an efficient manner.
- There are two applications of HDL processing: ***Simulation*** and ***Synthesis***

Logic Simulation

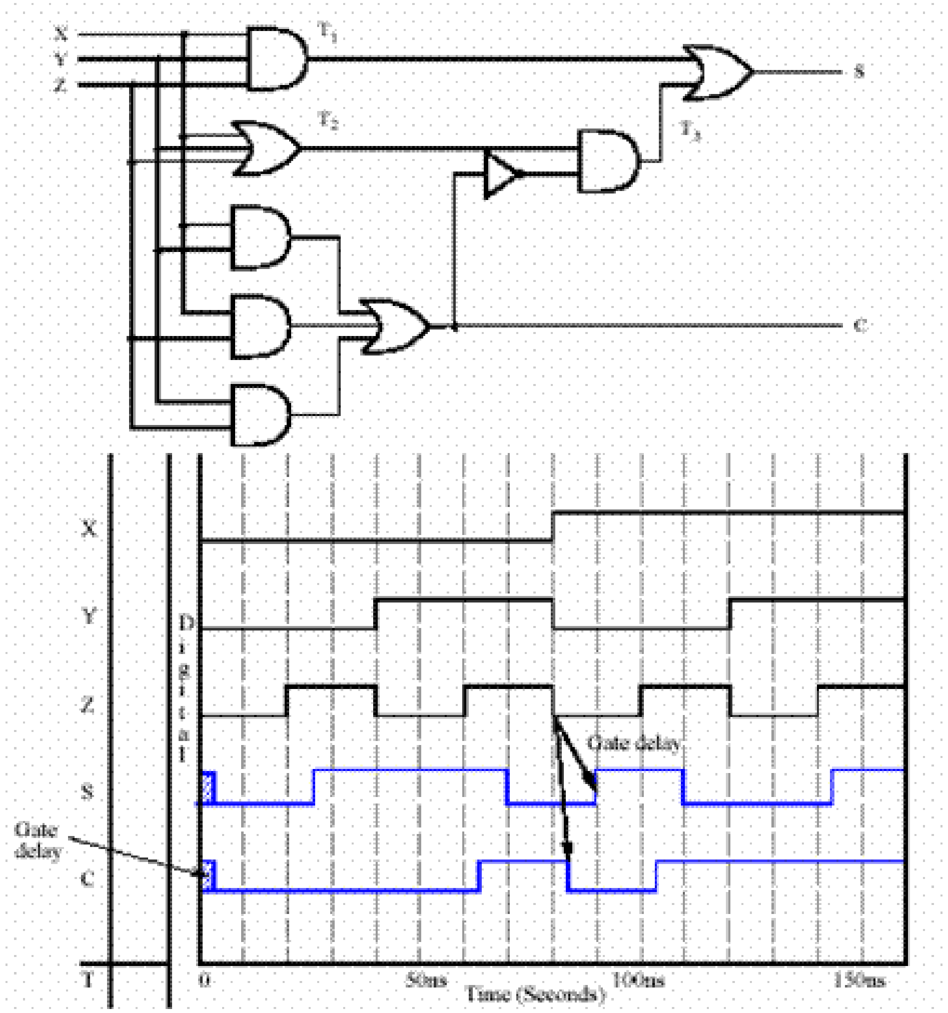
- A simulator interprets the HDL description and produces a readable output, such as a timing diagram, that predicts how the hardware will behave before its is actually fabricated.
- Simulation allows the detection of functional errors in a design without having to physically create the circuit.

Logic Simulation (2)

- The stimulus that tests the functionality of the design is called a test bench.
- To simulate a digital system
  - Design is first described in HDL
  - Verified by simulating the design and checking it with a test bench which is also written in HDL.

Logic Simulation

- Logic simulation is a fast, accurate method of analyzing a circuit to see its waveforms



Types of HDL

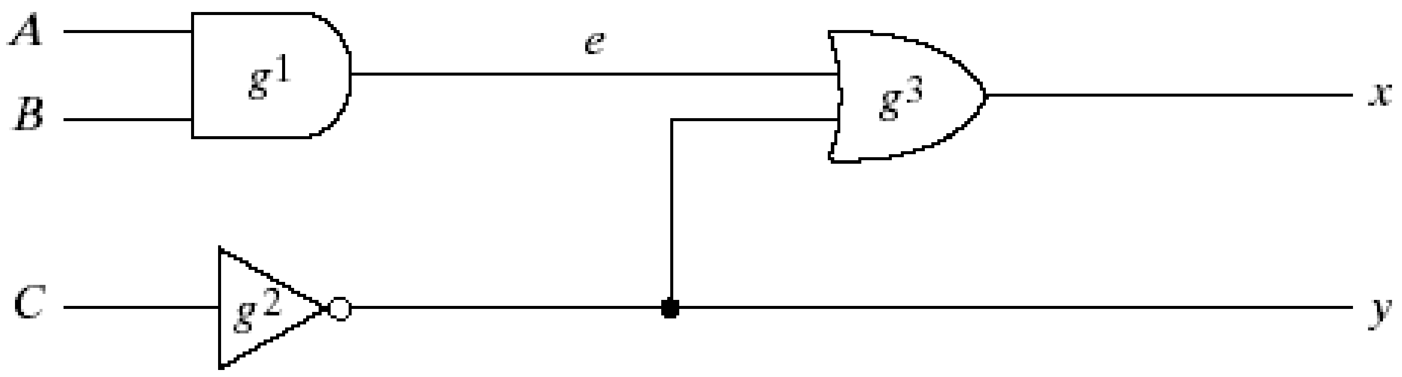
- There are two standard HDL’s that are supported by IEEE.
  - **VHDL** (*Very-High-Speed Integrated Circuits Hardware Description Language*) - Sometimes referred to as VHSIC HDL, this was developed from an initiative by US. Dept. of Defense.
  - **Verilog HDL** – developed by Cadence Data systems and later transferred to a consortium called *Open Verilog International* (OVI).

Verilog

- Verilog HDL has a syntax that describes precisely the legal constructs that can be used in the language.
- It uses about 100 keywords pre-defined, lowercase, identifiers that define the language constructs.
- Example of keywords: *module*, *endmodule*, *input*, *output* *wire*, *and*, *or*, *not* , etc.,
- Any text between two slashes (//) and the end of line is interpreted as a comment.
- Blank spaces are ignored and names are case sensitive.

Verilog - Module

- A *module* is the building block in Verilog.
- It is declared by the keyword *module* and is always terminated by the keyword *endmodule*.
- Each statement is terminated with a semicolon, but there is no semi-colon after *endmodule*.

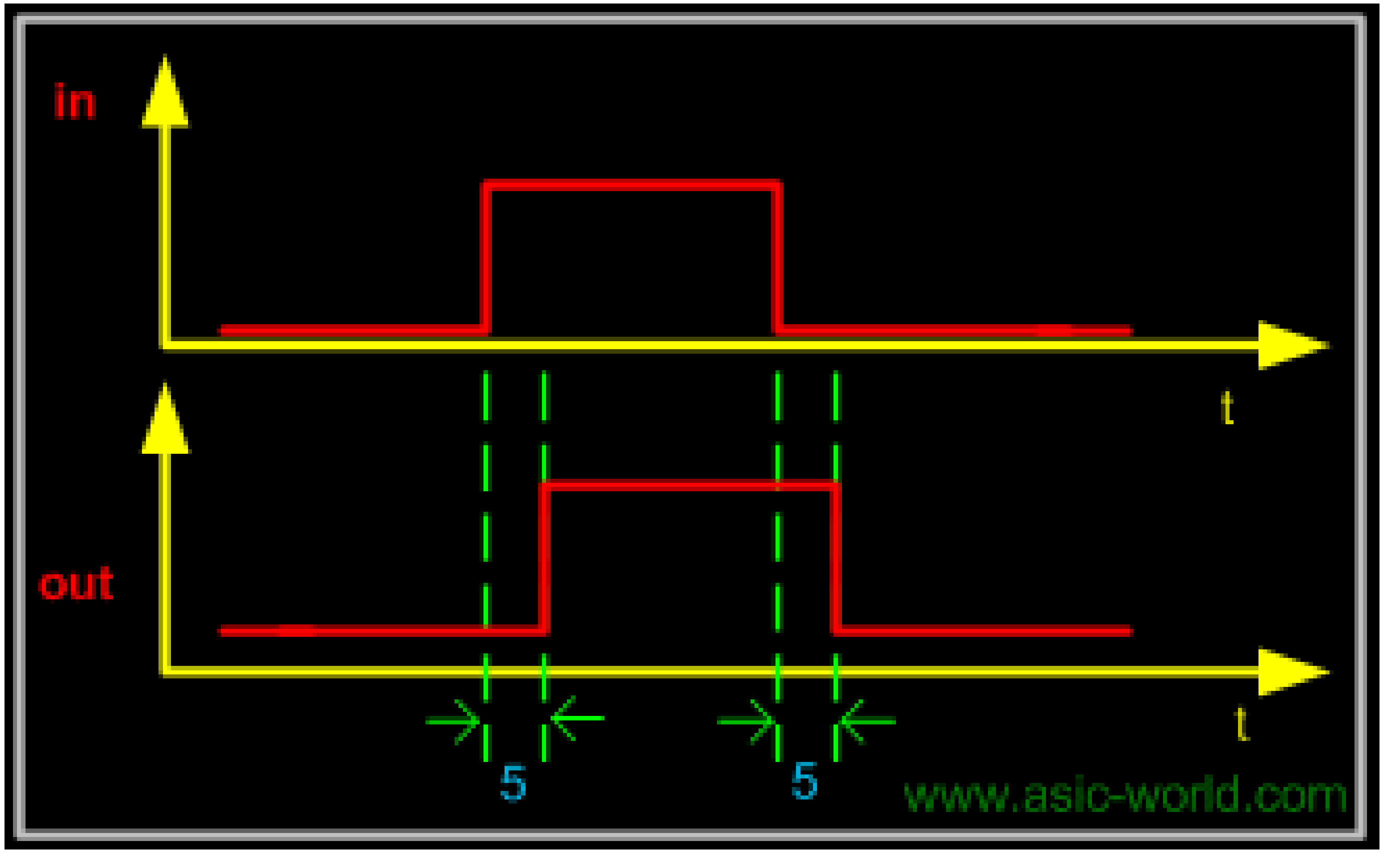


Verilog – Module (2)

HDL Example

```
module smpl_circuit(A,B,C,x,y);
  input  A,B,C;
  output x,y;
  wire   e;
  and g1(e,A,B);
  not g2(y,C);
  or  g3(x,e,y);
endmodule
```

Verilog – Gate Delays



Source: http://www.asic-world.com/verilog/gate3.html

Verilog – Gate Delays

- Sometimes it is necessary to specify the amount of delay from the input to the output of gates.
- In Verilog, the delay is specified in terms of time units and the symbol #.
- The association of a time unit with physical time is made using ***timescale*** compiler directive.
- Compiler directive starts with the "backquote (`)" symbol.

```
    `timescale 1ns/100ps
```

- The first number specifies the *unit of measurement* for time delays.
- The second number specifies the *precision* for which the delays are rounded off, in this case to 0.1ns.

Verilog – Module (4)

```
//Description of circuit with delay
`timescale 1ns/100ps
module circuit_with_delay (A,B,C,x,y);
   input  A,B,C;
   output x,y;
   wire   e;
   and #(30) g1(e,A,B);
   or  #(20) g3(x,e,y);
   not #(10) g2(y,C);
endmodule
```

Verilog – Module (5)

- In order to simulate a circuit with HDL, it is necessary to apply inputs to the circuit for the simulator to generate an output response.
- An HDL description that provides the stimulus to a design is called a **test bench**.
- The ***initial*** statement specifies inputs between the keyword ***begin*** and ***end***.
- Initially ABC=000 (A,B and C are each set to 1'b0 (one binary digit with a value 0).
- **$finish** is a *system task.*

Verilog – Module (6)

```
module circuit_with_delay
(A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and #(30) g1(e,A,B);
   or  #(20) g3(x,e,y);
   not #(10) g2(y,C);
endmodule
```

```
//Stimulus for simple circuit
module stimcrct;
reg A,B,C;
wire x,y;
circuit_with_delay cwd(A,B,C,x,y);
initial
   begin
         A = 1'b0; B = 1'b0; C = 1'b0;
      #100
         A = 1'b1; B = 1'b1; C = 1'b1;
      #100  $finish;
   end
endmodule
```

Verilog – Module (6)

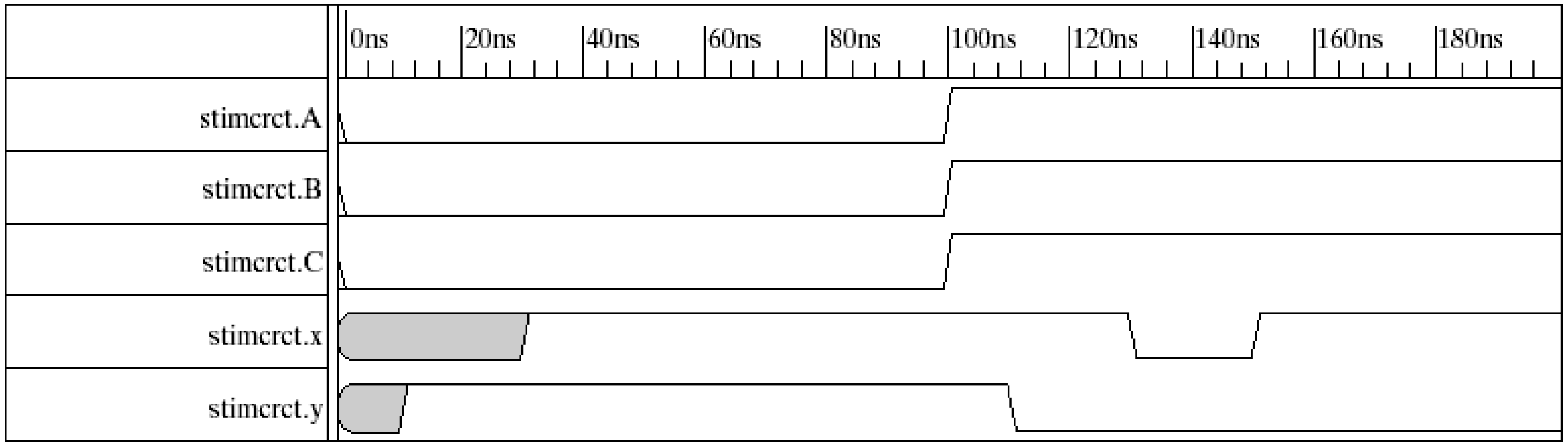


Fig. 3-38 Simulation Output of HDL Example 3-3

In the above example, **cwd** is declared as one instance **circuit_with_delay**. (similar in concept to object<->class relationship)

Verilog – Module (7)

Bitwise operators

Bitwise NOT :	~
Bitwise AND:	&
Bitwise OR:	|
Bitwise XOR:	^
Bitwise XNOR:	~^ or ^~