Assignment 1
Introduction to Verilog

ECN 102 - Digital Logic Design
Department of Electronics and Communication Engineering
Indian Institute of Technology, Roorkee

November 29, 2017

Hardware Description Languages

Hardware description language (HDL) is a convienient, device independent way of representing digital logic. HDLs are helpful describing, simulating and verifying digital circuits.

Why not use C/C++, Java...?

C/C++, Java are programming languages which are very good at what they were designed for, that is programming. However describing digital circuits in programming language is difficult and often confusing, which led to development of HDLs.

Which HDL to use?

Today many Hardware description languages are available, each of them are different from each other in terms of functionality, semantic and grammar. In this course however we will be using Verilog 2001.

Using Verilog 2001

As a begininer we are going to describe and simulate simple combinational circuit with Verilog.

Design Flow

Verilog Syntax

Modules

Module is the basic building block in Verilog, modules helps in organizing and structuring designs in logical and human readable form. Module can be considered as analogus to functions in programming languages.

Listing 1: Sample module indicating its structure

```
1 module module_name( ...module_input_and_outputs... );
2   ...
3   // Module functionality
4   ...
5 endmodule;
```

An example of AND gate would be

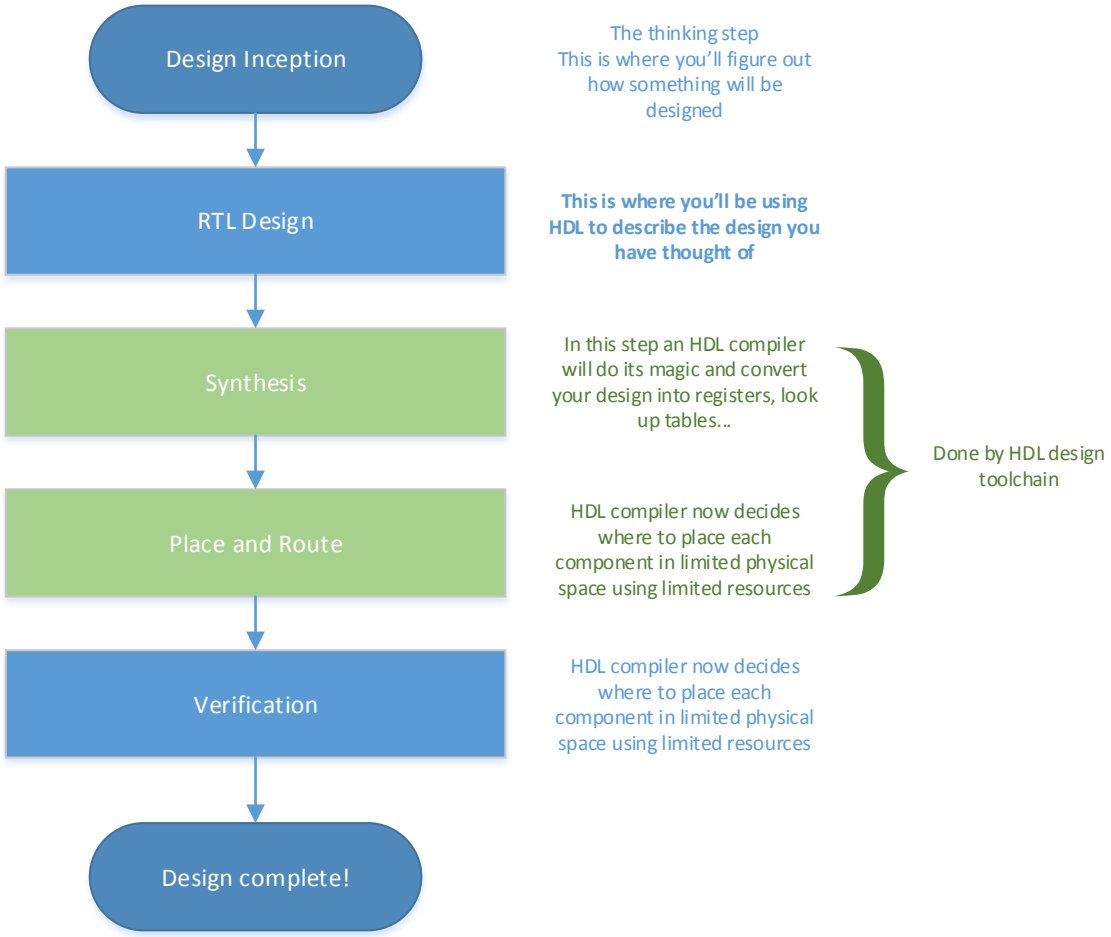

Figure 1: Example binary search tree constructed using names from Table **??** with insertion order and output port information.

Listing 2: Illustrative AND gate module

```
1  module andGate (inputA, inputB, result);
2       ...
3       // AND gate functionality
4       ...
5  endmodule
```

Instantiating modules

The process of creating objects of modules is called instatiation in Verilog. Modules are a

Listing 3: Illustrative AND gate module

```
1  // Define the top-level module called ripple carry
2  // counter. It instantiates 4 T-flipflops. Interconnections are
3  // shown in Section 2.2, 4-bit Ripple Carry Counter.
4  module AndGate3(input1, input2, input3, result);
5    input input1, input2, input3;
6    output result;
7
8    wire result1;
9
```

```
10    AndGate2(input1, input2, result1);
11    AndGate2(result1, input3, result);
12  endmodule
13
14  module AndGate2(input1, input2, result);
15    assign result = input1 & input2;
16  endmodule
```

Comments

Numerical Literals

Constants

Wires

Registers

Always Block

If and Case Statement

Listing 4: Sample Python code – Fibonacci sequence calculated analytically.

```
from math import *

# define function
def analytic_fibonacci(n):
  sqrt_5 = sqrt(5);
  p = (1 + sqrt_5) / 2;
  q = 1/p;
  return int( (p**n + q**n) / sqrt_5 + 0.5 )

# define range
for i in range(1,31):
  print analytic_fibonacci(i)
```

Following Listing 4... Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Problem 2

Listing 5: Sample Bash code.

```
#! /bin/bash
python stage1.py
echo "Stage I done!"
python stage2.py
echo "Stage II done!"
python stage3.py
echo "Stage III done!"
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate

velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.