Digital Logic Design using Verilog and FPGA devices
Part 1

An Introductory Lecture Series

By

Chirag Sangani

Electronics Club
IIT Kanpur

Terminology

- Verilog: A “Hardware Description Language”.
- FPGA: “Field Programmable Gate Array”. Physically, it is an IC on a circuit board.
- Synthesizer: A tool to convert Verilog code into a useful format (?), analogous to a compiler.

Verilog

- It is NOT a programming language, although its syntax is similar to that of C.
- C describes a “Computer Program” which is a sequence of instructions carried out by the computer processor on an input to give the desired output.

Verilog

- Verilog describes an electronic circuit (for our purposes, the circuit is digital in nature).
- The Verilog code is synthesized (analogous to compiled) to give the circuit logic diagram.
- This circuit can then either be simulated on a computer, or can be fabricated into an actual circuit in the form of an IC.

FPGA

- An FPGA is a device that allows you to implement your synthesized circuit in the real world.
- Physically, it is an IC with a large number of lookup tables and a complicated wire network that can be programmed to work as any desired digital circuit.

Development process

$$\begin{tikzcd}
\text{Design on Paper} \arrow[d] \\
\text{Verilog Code} \arrow[d, "\text{Synthesis}"] \\
\text{Synthesized Format} \arrow[d, "\text{Verification}"] \\
\text{On FPGA}
\end{tikzcd}$$

Verilog Examples: Repeater

```
module Repeater(
    input wire A,
    output wire B);

assign B = A;

endmodule
```

Verilog Examples: Bus Inverter

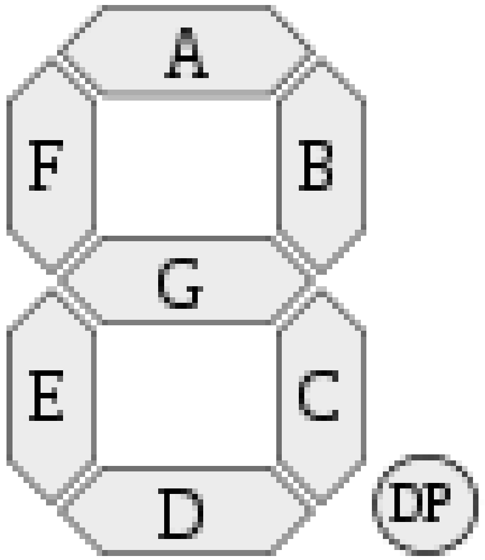
```
module BusInverter(
    input wire [31:0] A,
    output wire [31:0] B
);

    assign B = ~A;

endmodule
```

Advanced Example: Seven Segment Decoder

- A seven segment decoder receives a 4-bit unsigned number as an input and gives an output in a particular manner.
- This output can be used to drive the segments of a seven-segment display.

Advanced Example: Seven Segment Decoder

I3	I2	I1	I0	A	B	C	D	E	F	G
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	1	1	1	1	0	0	0	0	0
1	1	0	0	0	1	1	0	0	0	1
1	1	0	1	1	0	0	0	0	1	0
1	1	1	0	0	1	1	0	0	0	0
1	1	1	1	0	1	1	1	0	0	0



Advanced Example: Seven Segment Decoder

```
module SevenSegmentDisplay(
     input wire [3:0] inp,
     output wire[6:0] out
);

assign out[0] = ~inp[3] & ~inp[2] & ~inp[1] &
inp[0]
| ~inp[3] & inp[2] & ~inp[1] & ~inp[0]
| inp[3] & ~inp[2] & inp[1] & inp[0]
| inp[3] & inp[2] & ~inp[1] & inp[0];

endmodule
```

Introducing the `always` Block

- The `always` block allows for the implementation of sequential and combinatorial circuits.
- It allows to shift our programming focus from being logic-based to behavior-based.

Anatomy of an `always` block

```
always @(#sensitivity list#)
begin
    #actions#
end
```

The Sensitivity List

```
// Run continuously.
always

// Run when any variable changes its value.
always @(*)

// Run when the variables `a' or `b' change their
value.
always @(a,b)

// Run when a positive edge is detected on CLK.
always @(posedge CLK)
```

Example: Counter

```
module Counter(
      input wire CLK,
      output reg [31:0] OUT
);

initial
      OUT <= 0;

always @(posedge CLK)
      OUT <= OUT + 1;

endmodule
```

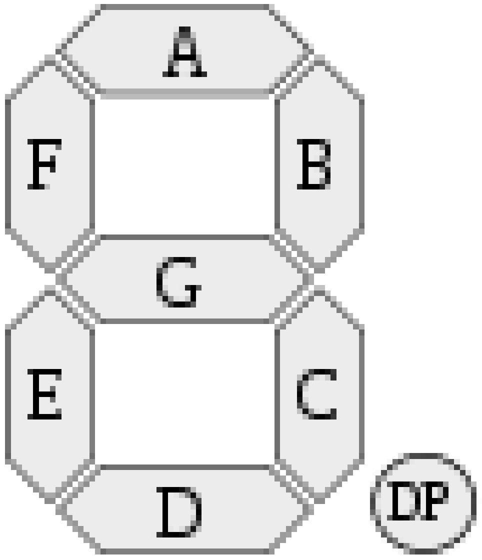
Points to Note

- Output is `reg` instead of `wire`. Why?
- The operator used is <= instead of =. Why?
- An `initial` block is added. What is it and why?

```
module Counter(
      input wire CLK,
      output reg [31:0] OUT
);

initial
      OUT <= 0;

always @(posedge CLK)
      OUT <= OUT + 1;

endmodule
```

Revisiting the Seven Segment Decoder

```
module SevenSegmentDecoder2(
        input wire [3:0] inp,
        output reg [6:0] out);

initial
        out <= 7’b0;

always @(*)
begin
        case (inp)
                4’b0001:out <= 7’b0000110;
                default:out <= 7’b0111111;
        endcase
end

endmodule
```



References

- R. Haskell, D. Hanna: "Introduction to Digital Design Using Digilent FPGA Boards – Block Diagram / Verilog Examples"; available at http://www.digilentinc.com/Data/Textbooks/Intro_to_Digital_Design-Digilent-Verilog_Online.pdf
- C. Sangani, A. Kasina: "Digital Design Using Verilog and FPGAs: An Experiment Manual"; available at http://www.chiragsangani.com/projects/electronics/FPGADesignManual