FPGA Design Challenge Techkriti 2013

Digital Logic Design using Verilog

Part 1

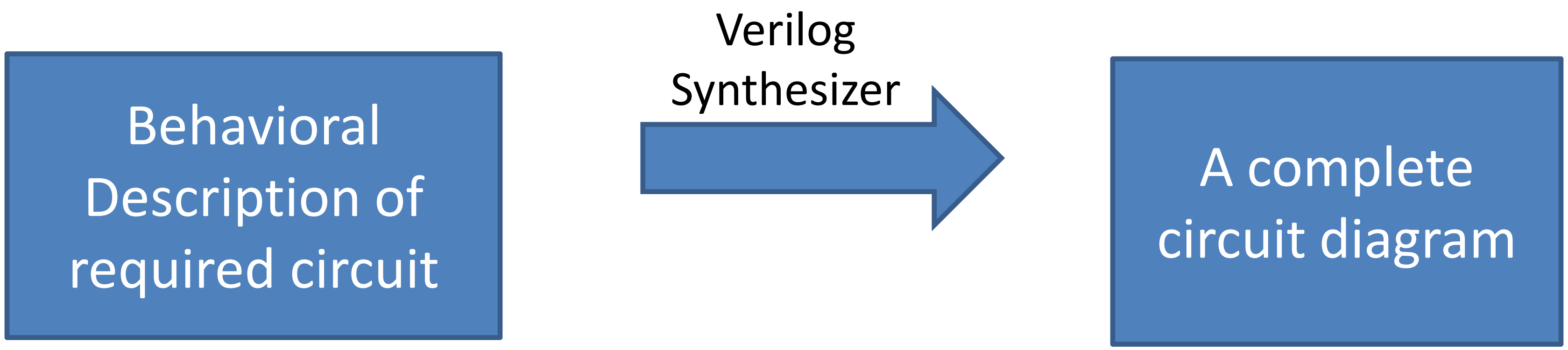
By

Neeraj Kulkarni

Introduction

- Verilog: A Hardware Description Language

  -mostly describes a digital circuit
- Motivation for such a language
- Adds a level of abstraction in digital design
- Basic Idea:


Behavioral Description of required circuit
Verilog Synthesizer
A complete circuit diagram

Introduction to Verilog

- Data types?

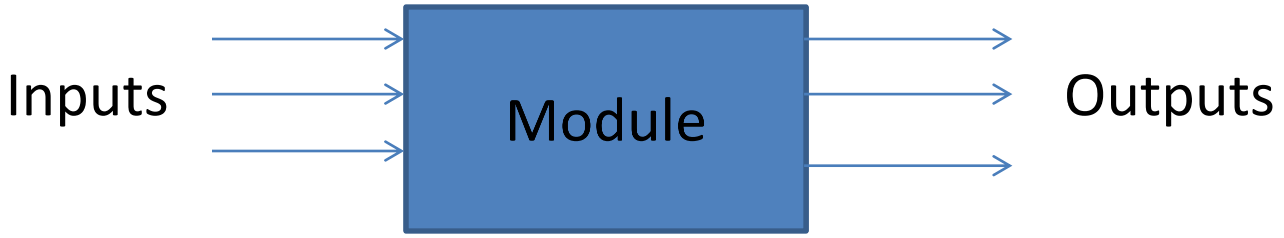
  -Yes, called Drivers in Verilog

- Drivers: wire and register
  – Wire: Connects to points in circuit – wire clk
  – Register (reg) : Stores values – reg A

- Arrays?
  – wire [31:0] in
  – reg [16:0] bus

Modules

- Modules are like ‘Black boxes’.



- Example: Inverter

```
module Inverter(
        input wire A,
        output wire B
);
        assign B = ~A;
endmodule
```

Note:
Input and Output ports of the module should be specified.

Combinational Circuits

- Combinational circuits are **acyclic** interconnections of gates.
  - And, Or, Not, Xor, Nand, Nor ……
  - Multiplexers, Decoders, Encoders ….
  - Adders, Multipliers ….
- How are these gates, muxs etc. abstracted in Verilog?
  - Gates, Add, Multiply … : by simple operators like in C
  - Multiplexers … : by control statements like if-else, case, etc.
- Gate level implementation of above high level operators done by Verilog synthesizer.

Some Operators

Arithmetic	*****	**Multiply**
	/	Division
	+	Add
	-	Subtract
	%	Modulus
	+	Unary plus
	-	Unary minus
Logical	!	Logical negation
	&&	Logical and
	||	Logical or
Relational	>	Greater than
	<	Less than
	==	Equality
Shift	>>	Right shift
	<<	Left shift

Control Statements

- if-else, case :
  - Exactly like C.
  - *Hardware view*: implemented using multiplexers
- for loops, repeat:
  - for-loops are synthesizable only if length of iteration is determined at compile time & finite.
  - repeat -similar to for loop.
  - *Hardware view*: All loops are unrolled during synthesis.

Syntax-Control Statements

```
for (i = 0; i < n; i = i +1)
 begin
            ........
 end
```

```
case(address)
   0 :  .......
   1 :  .......
   2 :  ........
   default : ......
endcase
```

```
if ( ....... )
begin
   ...........
end
else begin
    ...........
end
```

```
repeat (18)
begin
     ...........

end
```

assign statement

- Continuous assignment statement.
- Used for modeling only combinational logic.

```
module BusInverter(
        input wire A,
        output wire B
);
        assign B = ~A;
endmodule
```

- Basically B is shorted to ~A.
- RHS should have variable of wire type.

Example-1 bit Full Adder

```
module full_adder(
            input wire a,
            input wire b,
            input wire cin,
            output wire sum,
            output wire carry
);
    assign sum = a & ~b & ~cin | ~a
& b & ~cin |~a & ~b & cin | a & b &
cin;
    assign carry = a & b | a & cin | b
& cin;
endmodule
```

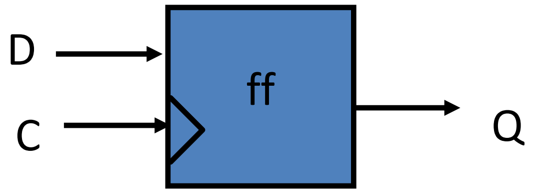
Gate Level Description

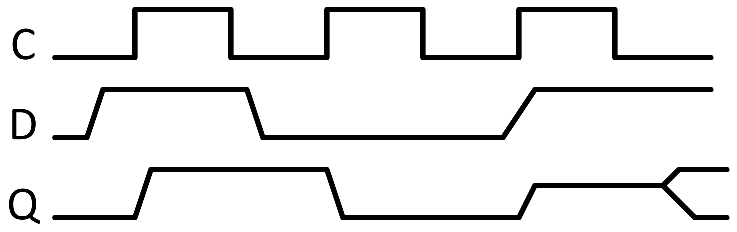
```
module full_adder(
            input wire a,
            input wire b,
            input wire cin,
            output wire sum,
            output wire carry
);
       assign { carry, sum } = a+b+cin;
endmodule
```

Behavioral Description

Sequential Circuits

- Circuits containing state elements are called *sequential circuits.*
- The simplest synchronous state element: Edge-Triggered D Flip-Flop





- How do you implement such an element in Verilog?

always@ Block

- It is an abstraction provided in Verilog to mainly implement sequential circuits.
- Also used for combinational circuits.

- Structure of always block:

```
always @(#sensitivity list#)
 begin
          ……….                //No assign statements inside always@
end
```

- Execution of always block depends on the sensitivity list.

The Sensitivity List

- Run continuously. (mostly used in Test Benches)
```
always
```
- Run when any variable changes its value.
```
always @(*)          //for combinational ckts
```
- 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)       //for sequential ckts
```

initial block

- An initial block is executed only once when simulation starts

- This is useful in writing test benches

- If we have multiple initial blocks, then all of them are executed at the beginning of simulation

Example- Counter

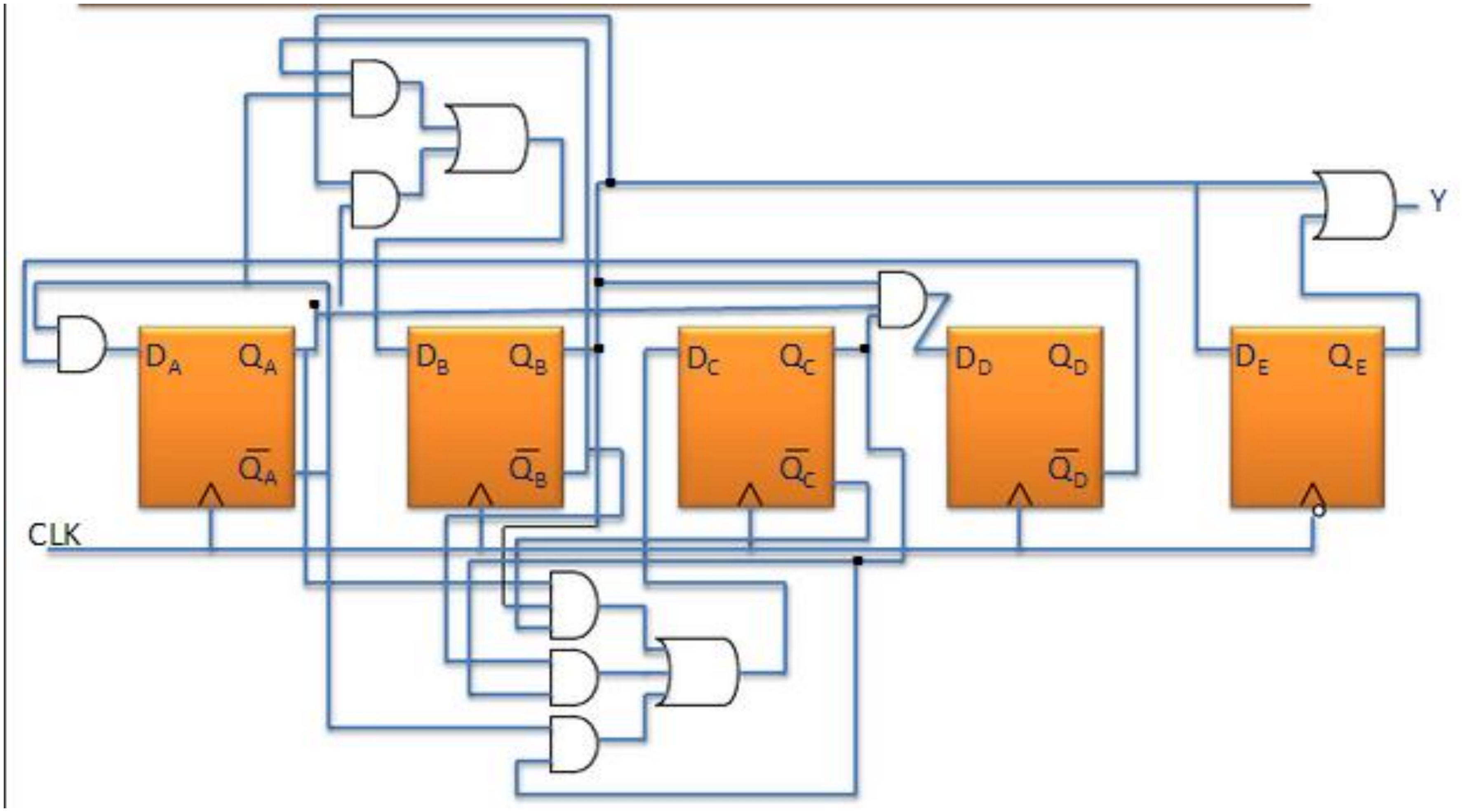
```
module Counter(
        input wire CLK,
        output reg [31:0] OUT
);
        initial
                OUT <= 0;
        always @(posedge CLK)
                OUT <= OUT + 1;
endmodule
```

Note the ‘<=‘ sign for register assignment

Divide by 9 counter
In Verilog

```
module Counter(
        input wire CLK,
        output reg [4:0] OUT
);
        initial
                OUT <= 4’b0;
        always @(posedge CLK)
        begin
                if(OUT==4’b1000)
                        OUT <= 4’b0;
                else
                        OUT <= OUT + 1;
        end
endmodule
```

Divide by 9 counter
By hand analysis



Designing this is messy!

Blocking and Non-Blocking Statements

- Non-blocking assignments happen in parallel.

```
always @ ( #sensitivity list # ) begin
        B <= A ;
        C <= B ;                              (A,B) = (1,2) -> (B,C) = (1,2)
end
```

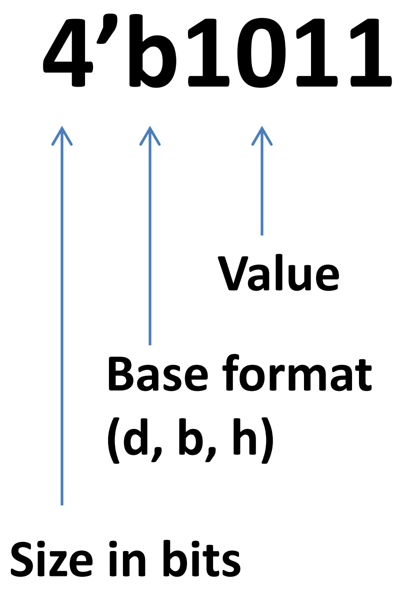
- Blocking assignments happen sequentially.

```
always @ ( #sensitivity list # ) begin
        B = A ;
        C = B ;                               (A,B) = (1,2) -> (B,C) = (1,1)
end
```

Points to Note

- Use always@(*) block with blocking assignments for combinational circuits.
- Use always@(posedge CLK) block with non-blocking assignments for sequential circuits.
- Do not mix blocking and non-blocking assignments.

Extras- Bit Literals



- If no size given, number is assumed to be 32 bits.
- If <size> is smaller than value
  - MSB of value truncated
- If <size> is greater than value
  - MSB of ‘value’ filled with zeros
- e.g. - hexadecimal: 4’hB
- If no base given, number assumed to be decimal. e.g. - 11

More to come

- Parameterized Modules
- Modular Circuits
- Test Benches
- FPGA Design Challenge, Techkriti 13
  - Problem Statement Discussion

Questions?