• RTL Verilog coding using continuous assignment.
• RTL Verilog coding using procedural assignment.

Prepared By-
Mohammed Abdul Kader
Lecturer, EEE, IIUC
RTL Verilog Coding of Digital Circuit

Structural Verilog code is not practicable for designing complex circuit. In that case, RTL Verilog code is used to design digital circuit. RTL Verilog code of a digital design can be written in two ways:

1) Using continuous assignment structures.
2) Using Procedural assignment structures.

RTL Coding using continuous assignment

- Modeling of digital circuit using continuous assignment structure is higher level of abstraction than structural Verilog coding.
- It is named as continuous assignment because the assignments written in this procedure are evaluated continuously whereas in procedural assignment structure execution of a statement waits for the clock or other parameter. The expression (in continuous assignment structure) is evaluated whenever any of the operands changes.
- RTL coding using continuous assignment is usually used to model combinational circuit which is more complex than can be handled by structural modeling.
- The declaration ‘module’, ‘input’, ‘output’ and ‘endmodule’ are same as they are used in structural Verilog code. The difference is the keyword ‘assign’ is used to write the continuous assignment.
Examples RTL Verilog coding using continuous Assignment

**Example 1:** Write Verilog code to represent the digital circuit that follows the following logic equations.

\[
M = (AB + CD)(\overline{AB} + \overline{CD}) \tag{1}
\]

\[
N = \overline{A}B(CDE + \overline{DE}) \tag{2}
\]

\[
P = (CD + BE)(A + \overline{DB}) \tag{3}
\]

**Verilog Code:**

```verilog
module ckt (A, B, C, D, E, M, N, P);

input A, B, C, D, E;

output M, N, P;

assign M = ((A & B) | (C & D)) & ((\neg A & \neg B) | (\neg C & \neg D));

assign N = (A & \neg B) & ((C & D & E) | (\neg D & \neg E));

assign P = \neg ((C & D) | (B & E)) & (A | (\neg D & B));

endmodule
```

**Exercise 1:** Write Verilog code to represent the digital circuit that follows the following logic equations.

\[
M = AB + BC + \overline{CD}
\]

\[
N = \overline{AB} + \overline{BC} + \overline{CD}
\]
Examples RTL Verilog coding using continuous Assignment (Cont.)

Example 2: A single bit adder (full adder).

Verilog Code:

```
module adder_single (In1, In2, Cin, Sum, Cout);
input In1, In2, Cin;
output Sum, Cout;
assign {Cout, Sum}=In1+In2+Cin;
endmodule
```

Exercise 2: Design a 4-bit adder circuit using continuous assignment structure of Verilog code.
Example 3: Write Verilog code to design 4-bit adder-subtractor circuit using continuous assignment structure.

Verilog Code:

```verilog
module add_sub(Sum, Cout, A, B, sel);
    input [3:0] A, B;
    input sel; // sel=0 addition; sel=1 subtraction
    output [3:0] Sum; // 4 bit sum or diff
    output Cout; // carry for add, comp of borrow for subtract
    assign {Cout, Sum} = sel ? (A~B+4’b0001) : (A+B); // 2’s complement of B=~B+0001
endmodule
```
Example 3: Write Verilog code to design a 2/1 MUX using continuous assignment structure.

```
module mux_21 (A, B, S, Y);
    input A, B, S;
    output Y;
    assign Y = S ? A : B;
endmodule
```
RTL Verilog Code using Procedural Assignment

• Structural Verilog coding and Verilog coding using continuous assignment structure are used to design simple digital circuit. This approach is practical when functional complexities are not much (gate count within few hundred).

• With the increase of functional complexities, the above two approaches are not suitable approach. In designing complex digital circuit, procedural assignment approach is used.

• In this approach, the functional behavior of the circuit is described using keyword ‘always’.

• Sensitivity list is used with the ‘always’ statement. Sensitivity list indicates ‘output is sensitive to what?’. It contains the list of specific inputs, which have effect on the outputs.

• Whenever any event (change) occurs in any of the parameter in the sensitivity list, the always loop will be executed.

How to write RTL Verilog code using Procedural assignment

• The declaration ‘module’, ‘input’, ‘output’ and ‘endmodule’ are same as used in structural Verilog coding and RTL Verilog coding using continuous assignment structure.

• It is important to remember that all the outputs have to be declare as register using the keyword ‘reg’.

• The keyword ‘always’ is used to describe the behavior of the circuit. Sensitivity list has been placed inside the first bracket. The keyword ‘begin and ‘end’ indicates start and completion of the always block respectively. However, the use of ‘begin and ‘end’ is optional. It is just increase the clarity of the code. **Within the always block, it is described ‘What does the circuit always do: which is basically the behavior of the circuit.’**
Example 1: RTL Verilog code using procedural assignment to design a 2/1 MUX

Verilog Code:

```verilog
module mux_21P(A, B, S, Y);
input A, B, S;
output Y;
reg Y; // all the outputs need to be declared as register

always @(A or B or S) begin
    if(S==0)
        Y=A;
    else
        Y=B;
end
endmodule
```
Example 2: RTL Verilog code using procedural assignment to design a 4/1 MUX

Verilog Code:

```verilog
define module mux_41P(A, B, C, D, S, Y);
  input A, B, C, D;
  input [1:0] S;
  output Y;
  reg Y;
  always @(A or B or C or D or S) begin
    case (S)
      0: Y = A;
      1: Y = B;
      2: Y = C;
      3: Y = D;
      default: Y = 0;
    endcase
  end
endmodule
```

Diagram:

![Diagram of 4/1 MUX with inputs A, B, C, D, and select lines S[1:0] and output Y.]}
Example of RTL Verilog Coding using Procedural Assignment (Continued)

Example 3: RTL Verilog code using procedural assignment to design a 2/4 Decoder

Verilog Code:

```verilog
module dec2_4(E, A, B, Out);
    input A, B, E;
    output [3:0] Out;
    reg [3:0] Out;
    always @(A or B or E) begin
        if(E==0) Out=0;
        else begin
            if (A==0 & B==0)Out=4'b0001;
            else if (A==0 & B==1)Out=4'b0010;
            else if (A==1 & B==0)Out=4'b0100;
            else Out=4'b1000;
        end
    end
endmodule
```
Example of RTL Verilog Coding using Procedural Assignment (Continued)

Example 4: RTL Verilog code using procedural assignment to design a 7-segment Decoder (for common anode display).

Verilog Code:

module decoder(en, in, out);
input en;
input [3:0] in;
output [6:0] out;
reg [6:0] out;
always @(en or in) begin
  if(en==0) out=0;
  else begin
    case(in)
      4'b0000: out=7'b1000000;
      4'b0001: out=7'b1111001;
      4'b0010: out=7'b0100100;
      4'b0011: out=7'b0110000;
      4'b0100: out=7'b0011001;
      4'b0101: out=7'b0010010;
      4'b0110: out=7'b0000010;
      4'b0111: out=7'b1111000;
      4'b1000: out=7'b0000000;
      4'b1001: out=7'b0010000;
      default: out=7'b1111111;
    endcase
  end
endmodule