Basics of Verilog HDL

In this tutorial, different programming styles in Verilog coding will be discussed. Various online tutorials on programming syntax, operators, different commands, assignment strategies and other important topics are already available. Readers can find the references useful for basics. In going through the different tutorials, more on Verilog coding will be explored.

A Verilog code can be written in the following styles:

1. Dataflow style
2. Behavioral style
3. Structural style
4. Mixed style

Each of the programming styles is described below with realization of a simple 2:1 mux.

Dataflow style

In data flow style of modeling, logic blocks are realized by writing their Boolean expressions. The basic 2:1 mux for 1-bit data width can be realized in this style as

```
module mux_df(
   input  a,b,s,
   output y
   );
wire sbar;
assign y = (a&sbar)|(s&b);
assign sbar = ~s;
endmodule
```

Behavioral style

In the behavioral style of modeling instead of writing Boolean expressions, the behavior of the logic block is described. The same mux can be realized in this style of coding as

```
module mux_bh(
   input  a,b,s,
   output y
   );
reg y;
wire s;
always @( s or a or b )
begin
```

```
  if( s == 0)
    y = a;
        else
    y = b;
end
endmodule
```

Structural style

The structural style is a hierarchical design style in which a logic block is realized in terms of its basic sub-blocks. A sub-block can be a logic gate or any complex logic block. In two ways it can be described which are

1. Gate Level Modeling
2. Module Instantiation

Gate Level Modeling

```
module mux_gl(
   input  a,b,s,
   output y
   );
wire  q1,q2,sbar;
not n1( sbar,s);
and a1( q1, sbar, a);
and a2( q2, s, b);
or o1( y, q1, q2);
endmodule
```

Module Instantiation

A 4:1 mux for 1-bit data is realized using basic 2:1 mux in this style. Basic block 2:1 mux can be designed in any of the above mentioned styles.

```
module mux_4_1(
   input a1,a2,a3,a4,
         input [1:0]s,
         output y
   );
wire t1,t2;
```

```
mux_df m1(a1,a2,s[0],t1);
mux_df m2(a3,a4,s[0],t2);
mux_df m3(t1,t2,s[1],y);
endmodule
```

Mixed style

In mixed style, one can mix up the design styles in their program. Consider the same 4:1 mux. In this style one mux is design in gate level modeling, one mux is designed in data flow modeling and the 2nd stage mux is designed using behavioral modeling.

```
module mux_4_1_mix(
   input a1,a2,a3,a4,
         input [1:0] s,
         output y
   );
reg y;
wire t1,t2;
mux_df m1(a1,a2,s[0],t1);
mux_gl m2(a3,a4,s[0],t2);
always @( s[1] or t1 or t2 )
begin
  if( s[1] == 0)
    y = t1;
        else
    y = t2;
end
endmodule
```

A simple 2:1 mux can be designed in other ways also.

Using Conditional statement

```
module mux_cs(
   input  a,b,s,
   output y
   );
```

```
wire s;
assign y = (s == 0)? a : b;
endmodule
```

Using Case statement

```
module mux_case(
   input  a,b,s,
   output y
   );
reg y;
always @(s or a or b)
begin
  case(s)
     0 : y = a;
     1 : y = b;
  endcase
end
endmodule
```

It is observed in this tutorial that a logic block can be designed in various programming style. A general question arises that which programming style is better to follow. For simpler logic blocks any design style can be suitable. If only the system behavior is known then behavioral strategy suits better. The behavioral style is time-saving because in that case, a designer need not know what inside that logic block. But for a complete system when optimization is required at every level of design, behavioral style failed to perform. On the other hand, hierarchical design style gives a designer the flexibility to access every node of a design. Optimization and verification can be performed easily.

In the upcoming tutorials, it will be assumed that the basic logic blocks are designed in data flow or behavioral style. Complete system design will be carried out in structural modeling. Readers are encouraged to use structural modeling in designing logic blocks.