



Electrical and Computer Engineering EE3622 Embedded System Design

Dennis Silage, PhD
silage@temple.edu



Nested Modules in Verilog

Here is an example of a nested module program which illustrates several concepts in the formulation of a Verilog hardware description language program. The program itself displays the ubiquitous *hello world* string on an LCD display and is designed for the Spartan-3E Starter Board (see www.digilentinc.com). The concepts shown are as follows:

1. The module `lcdtest` is the top module and has an inputs and outputs those signals that interface to peripheral components using the UCF file. `CCLK` is the master clock, `BTN0` is pushbutton 0, `LCDRS`, `LCDRW` and `LCDE` are LCD control signals and `LCDDAT` is a 4-bit LCD data bus.
2. Wires are used to connect the 4-bit signal `lcd` from the nested module `lcd` to the output `LCCDAT` using a continuous assignment statement.
3. Wires are used to connect the 8-bit signal `lcdatin` from the nested module `lcd` to the nested module `genlcd`.
4. 1-bit signals in nested modules need not be declared by the wire construct as only the size of the multi-bit module needs to be declared as in the `lcdtest` top module.
5. Assignment of a multibit output signal such as `LCDDAT` cannot be in vector form since the UCF file is not in vector form either. Four continuous assignment statements are used in the `lcdtest` top module.
6. The external LCD control output signals `LCDRS`, `LCDRW` and `LCDE` in the top module `lcdtest` cannot be passed to a nested module, so the signals `rslcd`, `rwlcd` and `elcd` are used instead with a continuous assignment statement.
7. Register variables are not declared in the top module, only with a nested module as in the module `genlcd`.
8. Each of the nested modules are referenced by their module name and a label such as `M0` in the top module.

The modules `lcd` and `genlcd` are an example of a datapath and controller, respectively. Control signal such as `resetlcd` from the controller module `genlcd` is set (see the FSM state 2) and waits for a response `lcdreset` from the datapath module. The integer `i` in the controller module `genlcd` is used to count the 88 data bits in the *hello world* string so that they can be sent (see the FSM state 10 and 14). Note the construct for parsing the string in FSM state 11:

```
lcddatin[7:0]=strdata[i-:8];
```

A 1 msec local clock signal clk is used to debounce the pushbutton BTN0 to produce the signal debpb0 for the controller module genlcd.

```
// Spartan-3E Starter Board  
// Liquid Crystal Display Test lcdtest.v  
// c 2008 Embedded Design using Programmable Gate Arrays Dennis Silage
```

```
module lcdtest(input CCLK, BTN0, output LCDRS, LCDRW, LCDE,  
               output [3:0] LCDDAT);
```

```
wire [7:0] lcddatin;  
wire [3:0] lcdd;
```

```
assign LCDDAT[3]=lcdd[3];  
assign LCDDAT[2]=lcdd[2];  
assign LCDDAT[1]=lcdd[1];  
assign LCDDAT[0]=lcdd[0];
```

```
assign LCDRS=rslcd;  
assign LCDRW=rwlcd;  
assign LCDE=elcd;
```

```
lcd M0 (CCLK, resetlcd, clearlcd, homelcd, datalcd, addrlcd,  
        lcdreset, lcdclear, lcdhome, lcddata, lcdaddr,  
        rslcd, rwlcd, elcd, lcdd, lcddatin, initlcd);  
genlcd M1 (CCLK, debpb0, resetlcd, clearlcd, homelcd, datalcd,  
          addrlcd, initlcd, lcdreset, lcdclear, lcdhome,  
          lcddata, lcdaddr, lcddatin);  
pbdebounce M2 (clk, BTN0, debpb0);  
clock M3 (CCLK, 25000, clk);
```

```
endmodule
```

```
module genlcd(input CCLK, debpb0, output reg resetlcd,  
              output reg clearlcd, output reg homelcd,  
              output reg datalcd, output reg addrlcd,  
              output reg initlcd, input lcdreset, lcdclear,  
              input lcdhome, lcddata, lcdaddr,  
              output reg [7:0] lcddatin);
```

```
reg [3:0] gstate;           // state register  
reg [87:0] strdata="hello world";
```

```
integer i;
```

```
always@(posedge CCLK)  
begin  
    if (debpb0==1)  
    begin  
        resetlcd=0;  
        clearlcd=0;  
        homelcd=0;  
        datalcd=0;  
        gstate=0;
```

```

end
else
case (gstate)
0: begin
    initlcd=1;
    gstate=1;
    end
1: begin
    initlcd=0;
    gstate=2;
    end
2: begin
    resetlcd=1;
    if (lcdreset==1)
        begin
            resetlcd=0;
            gstate=3;
        end
    end
3: begin
    initlcd=1;
    gstate=4;
    end
4: begin
    initlcd=0;
    gstate=5;
    end
5: begin
    clearlcd=1;
    if (lcdclear==1)
        begin
            clearlcd=0;
            gstate=6;
        end
    end
6: begin
    initlcd=1;
    gstate=7;
    end
7: begin
    initlcd=0;
    gstate=8;
    end
8: begin // DD RAM address 44h
    lcddatin[7:0]=8'b01000100;
    addrlcd=1;
    if (lcdaddr==1)
        begin
            addrlcd=0;
            gstate=9;
        end
    end
9: begin
    initlcd=1;
    gstate=10;

```

```

        end
    10: begin
        initlcd=0;
        i=87;
        gstate=11;
    end
    11: begin
        lcddatin[7:0]=strdata[i-:8];
        datalcd=1;
        if (lcddata==1)
            begin
                datalcd=0;
                gstate=12;
            end
        end
    end
    12: begin
        initlcd=1;
        gstate=13;
    end
    13: begin
        initlcd=0;
        gstate=14;
    end
    14: begin
        i=i-8;
        if (i<0)
            gstate=15;
        else
            gstate=11;
        end
    end
    15: gstate=15;
    default: gstate=15;
endcase

end

endmodule

```