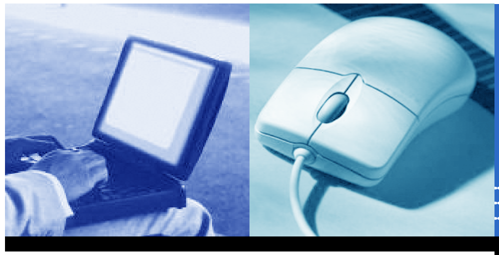
EE461 Verilog-HDL

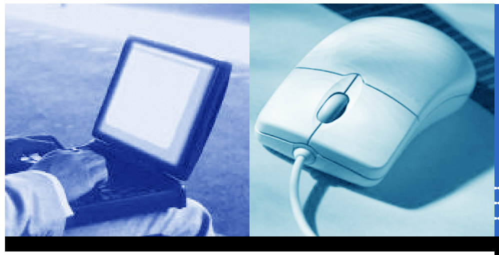
Week 11 Verilog 2001

Alex Yang, Engineering School, NPU



Week 11 Outlines

- Port Declaration
- Variable Initialization
- Parameter: Module Port Parameter/
  /Sized Parameter/Parameter Passing
- Bit Width Variable
- Vector Bits Selection
- Multi-Dimentional Array
- Signed Variable and Shift Operation
- Power Expression
- Conditional Compile Directive
- Generate Block

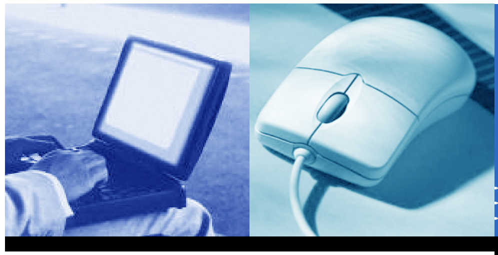


Port Declaration

- Compare Verilog-1995 and Verilog-2001

```
module verilog1995(
                        a_i,
                        b_i,
                        c_i,
                        x_o
);
    input           a_i;
    input           b_i;
    input           c_i;
    output[3:0]     x_o;

    reg[3:0]        x_o;
    ... ...
endmodule
```

```
module verilog2001(
            input wire      a_i, b_i,
            input wire           c_i,
            output reg [3:0]     x_o
);

   ... ...
endmodule
```

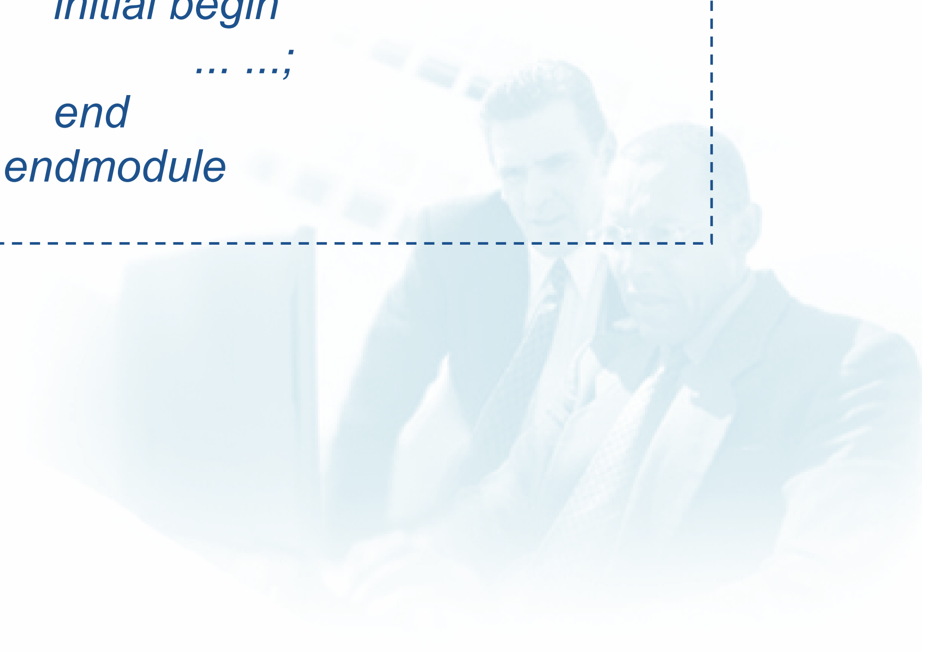


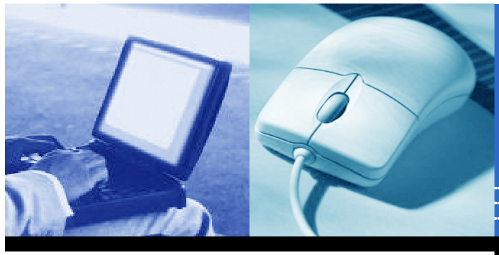
Variable Initialization

- Compare Verilog-1995 and Verilog-2001

```
module verilog1995TB;
    reg[3:0]              x_o;
    ... ...;
    initial begin
              x_o = 1;
              ... ...;
    end
endmodule
```

```
module verilog2001TB;
    reg[3:0]              x_o = 1;
    ... ...;
    initial begin
              ... ...;
    end
endmodule
```



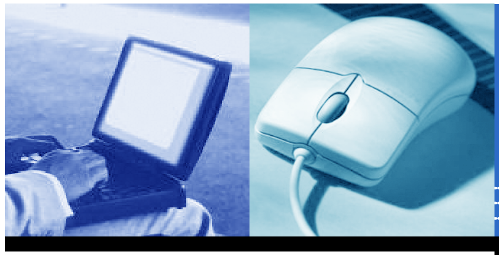


Parameter

- Module Port Parameter
- Compare Verilog-1995 and Verilog-2001

```
module verilog1995(
                                  a_i,
                                  b_i,
                                  c_i,
                                  x_o
);
    parameter          pBitW=4;
    parameter          pBitS=pBitW-1;
    input              a_i;
    input              b_i;
    input              c_i;
    output[3:0]        x_o;
    reg[pBitS:0]       x_o;
    ... ...
endmodule
```

```
module verilog2001
 #(parameter pBitW = 4, pBitS =pBitW-1)
(
     input wire                a_i, b_i,
     input wire                      c_i,
     output reg[pBitS:0]             x_o
);

    ... ...
endmodule
```

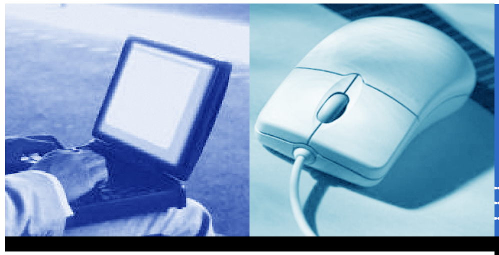


Parameter

- Sized Parameter
- Compare Verilog-1995 and Verilog-2001

```
module verilog1995( ... ...);
    parameter           pIdle = 0,
                        pOneSt = 1,
                        pTwoSt = 2,
                        pThreeSt =3,
                        pFourSt = 4,
                        pFiveSt = 5;
    ... ...
endmodule
```

```
module verilog2001
 parameter [2:0]     pIdle = 0,
                     pOneSt = 1,
                     pTwoSt = 2,
                     pThreeSt =3,
                     pFourSt = 4,
                     pFiveSt = 5;
    ... ...
endmodule
/* Available format
   parameter signed     ....
   parameter integer    ....
   parameter real       ....
   parameter time       ....
   parameter realtime   ....
*/
```

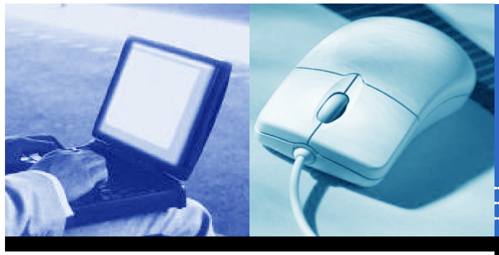


Parameter

- Parameter Passing
- Compare Verilog-1995 and Verilog-2001

```
module verilog1995( ... ...);
 ... ...
 parameter              SYNC= 1;
 parameter              WIDTH= 10;
 parameter              SIZE= 1024;
    ... ...

verilog1995  uVerilog1995 #(1, 12, 4096) (... ...);

endmodule
```

```
module verilog2001... ...
 parameter              SYNC= 1;
 parameter              WIDTH= 10;
 parameter              SIZE= 1024;
    ... ...

verilog2001  uVerilog2001 #(
                .SIZE (4096),
                .WIDTH (12),
                .SYNC (1)
) (... ...);

endmodule
```

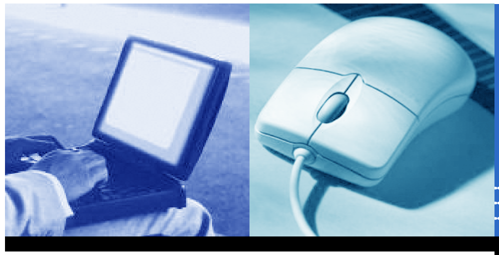


Bit Width Variable

- Compare Verilog-1995 and Verilog-2001

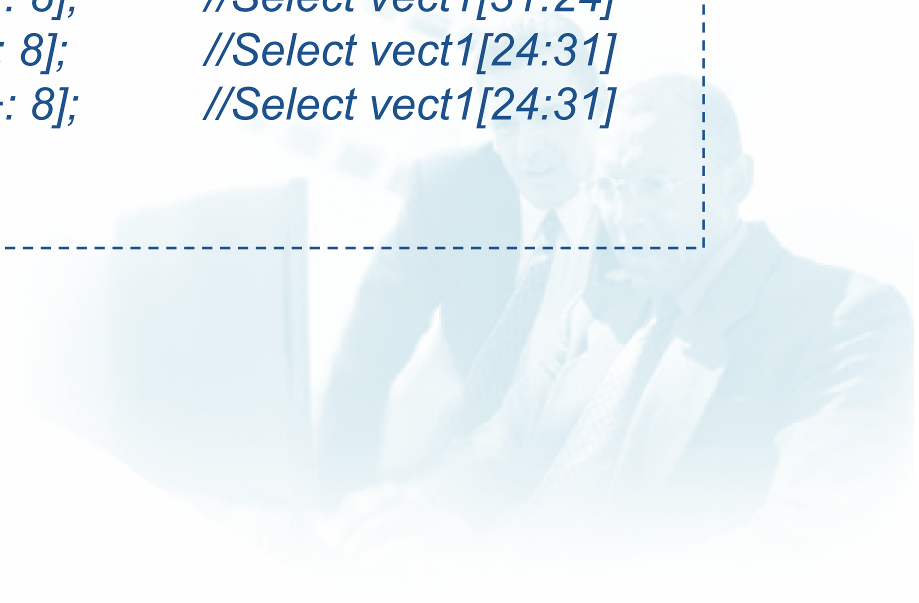
```
module verilog1995( ... ...);
 ... ...
 parameter          SIZE= 1024;
 parameter          ADDR= 10;

 input[ADDR-1:0]    adrBus;
   ... ...



endmodule
```

```
module verilog2001( ... ...);
 ... ...
 parameter          SIZE= 1024;
 parameter          ADDR= 10;

 input[log2(SIZE)-1:0] adrBus;
 ... ...
function integer log2;
 input  integer  depth;
 ... ...
endfunction
endmodule
```



Vector Bits Selection

```
module verilog2001( ... ...);
 ... ...
 wire[7:0]            byte1, byte2,byte3,byte4;
 reg[63:0]            vect1;
 reg[0:63]            vect2;
 ... ...
assign                byte1 = vect1[31 -: 8];      //Select vect1[31:24]
assign                byte2 = vect1[24 +: 8];      //Select vect1[31:24]
assign                byte3 = vect2[31 -: 8];      //Select vect1[24:31]
assign                byte4 = vect2[24 +: 8];      //Select vect1[24:31]
... ...
endmodule
```

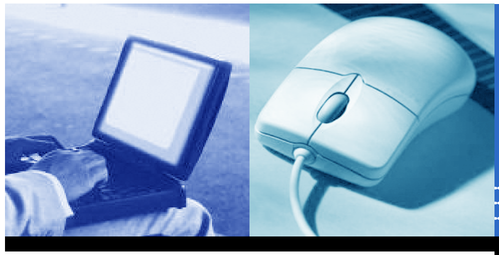


Multi-Dimentional Array

- Compare Verilog-1995 and Verilog-2001

```
module verilog1995( ... ...);
 ... ...
  reg[31:0]       a_r[0:255];
  reg[31:0]       temp;
  reg[7:0]        data;
 ... ...
  temp = a_r[5];
  data = temp[31:24];

 ... ...
endmodule
```

```
module verilog2001( ... ...);
 ... ...
  reg[31:0]       a_r[0:255];
  reg[15:0]       b_r[127:0][0:127]; //[row][col]
  reg[7:0]        data;
 ... ...
  initial begin
          data =  a_r[5][31:24];
  end
 ... ...

endmodule
```



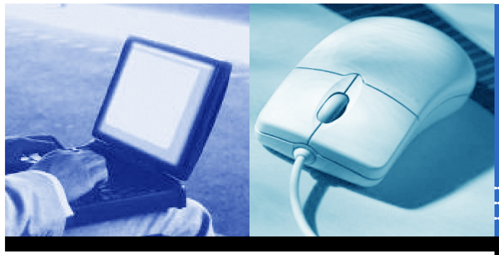
Signed Variable & Shift Op.

- Compare Verilog-1995 and Verilog-2001

```
module verilog1995( ... ...);
 ... ...
   reg[31:0]       a_r;
   reg[31:0]       b_r;
   reg[7:0]        data;
 ... ...
   function[3:0]  foo;
     ... ...
   endfunction
 ... ...
endmodule
```

```
module verilog2001( ... ...);
 ... ...
   reg signed [31:0]        a_r;
   reg[31:0]                  b_r;
   reg[7:0]                 data;
 ... ...
   function signed [3:0]  foo;
     ... ...
   endfunction
 ... ...
endmodule
```

```
module verilog2001( ... ...);
 reg[7:0]     a_r;  wire[7:0] out;
initial begin
   a_r = 8'b1100_1010;
end
assign       out = a_r >>3;          // out = 0001_1001
assign       out = a_r >>>3;         // out = 1111_1001    adding sign bit 1
assign     out = a_r <<< 2;          // out = 0010_1000    adding 0
endmodule
```



Power Expression

```
module verilog2001( ... ...);
 ... ...
 parameter          pAdrW = 12;

 reg[(2**pAdrW)-1:0]          vect2;
 ... ...
assign              byte1 = vect1[31 -: 8];       //Select vect1[31:24]
assign              byte2 = vect1[24 +: 8];       //Select vect1[31:24]
assign              byte3 = vect2[31 -: 8];       //Select vect1[24:31]
assign              byte4 = vect2[24 +: 8];       //Select vect1[24:31]
... ...
endmodule
```

Conditional Compile Directive

- Compare Verilog-1995 and Verilog-2001

In verilog-1995
only `ifdef, `else, `endif and `undef
are supported.

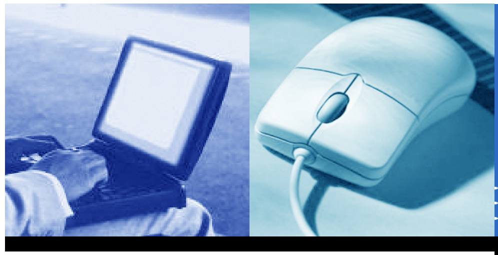
In verilog-2001 two more are added
`ifndef, and `elsif

Generate Block

- Instantiate more submodules or primitives
- Create more tasks/functions or continous assignments
- Combine if-else/case to generate different instants in the different conditions.

Generate Block

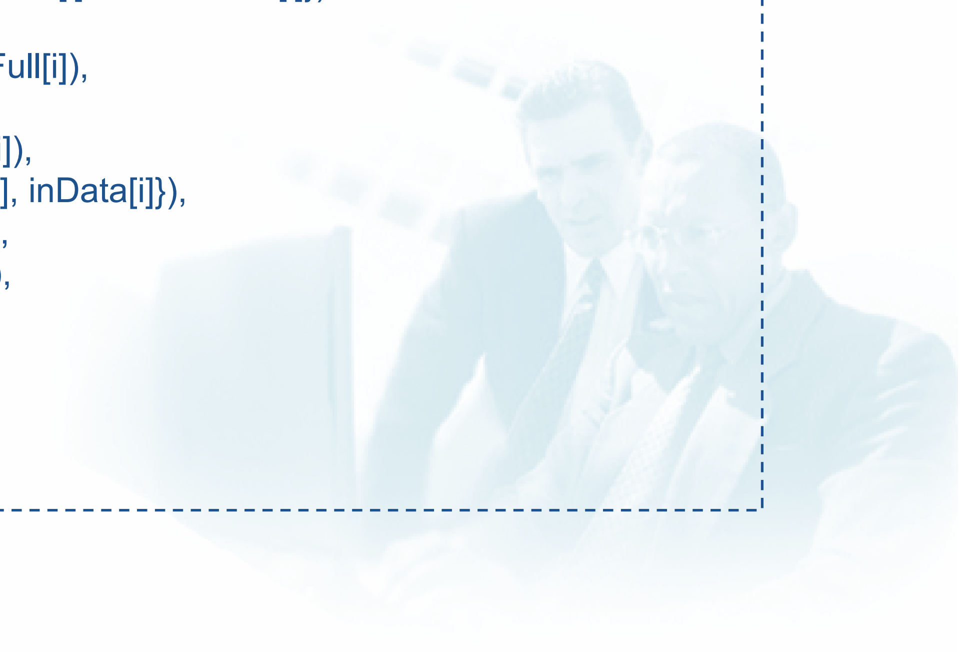
- Usage:
  - generate for
  - generate if
  - generate case

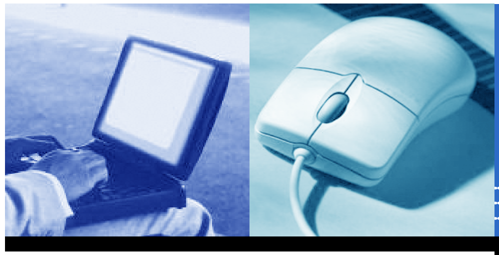


Generate Block

- Examples: generate for

```
module verilog2001(... ...);
... ...
 generate
        genvar i;                                        //Declare loop variable
        for(i=0; i<8; i=i+1) begin: FIFO                 //Must use named begin-end block

          FIFO  uFIFO #(   .WIDTH   (DATAWIDTH + CTRLWIDTH),  .MAXDEPTHBITS    (2) )
              (
               .dout                        ({fifoOutCtrl[i], fifoOutData[i]}),
               .full                        (),
               .nearlyFull                  (nearlyFull[i]),
               .progFull                    (),
               .empty                       (empty[i]),
               .din                         ({inCtrl[i], inData[i]}),
               .wrEn                        (inWr[i]),
               .rdEn                        (rdEn[i]),
               .rst                         (reset),
               .clk                         (clk)    );
        end
  endgenerate
endmodule
```

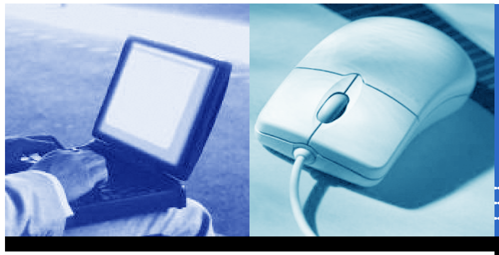




Generate Block

- Examples: generate if

```
module verilog2001(... ...);
... ...
        generate
             if (REGWIDTH == WRITEWIDTH) begin : newDataAGen
                assign newDataA = mergeUpdate ? mergeWrData : heldWrDataA;
             end
             else begin
                assign newDataA = mergeUpdate ?
                   {{(REGWIDTH - WRITEWIDTH - 1){mergeWrDataSign}}, mergeWrData} :
                   {{(REGWIDTH - WRITEWIDTH){heldWrDataSignA}}, heldWrDataA};
             end
          endgenerate
endmodule
```



Generate Block

- Examples:

```
module verilog2001(
                                        input           clk,
                                        input[7:0]      datain,
                                        output[7:0]     dataout,
                                        output          finish
);
   wire[7:0]           mem[31:0];
   wire[32*8-1:0]      xxx;
  //reg[7:0] i;
  generate
              genvar i;
              for(i=0;i<=31;i=i+1)begin
                          assign mem[i]= 8'b0;
              end
  endgenerate
endmodule
```