HDL Idioms

Exercise 1: Using the schematic symbols shown below, convert each of the following System Verilog expressions into a schematic.

Assign `y = a ^ b ;`

a^b → y

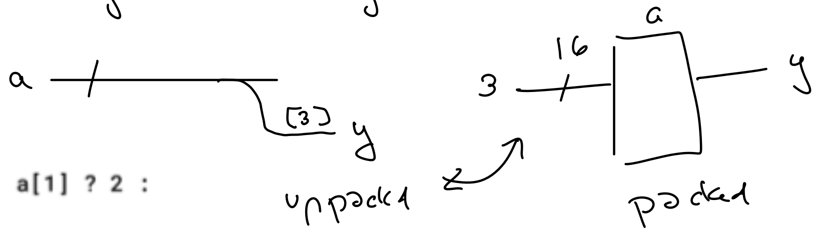
`y = a < b ;`

a<b → y

`y = y+1 ;`

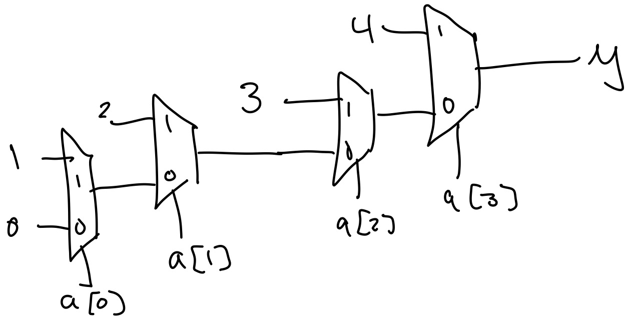
y+1 → y

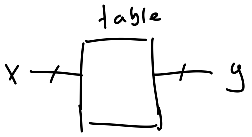
`y = a[3] ;`



↓

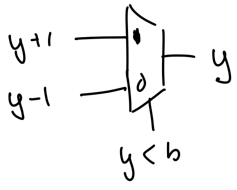
```
y = a[3] ? 4 : a[2] ? 3 : a[1] ? 2 :
    a[0] ? 1 : 0;
```





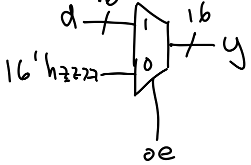
`y = table[x] ;`

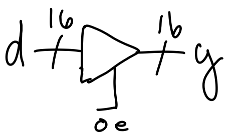
`y = y < b ? y+1 : y-1 ;`



`y = oe ? d : 16'hzzzz ;`

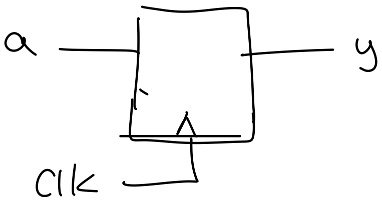
0, 1, Z, X ↖ undefined



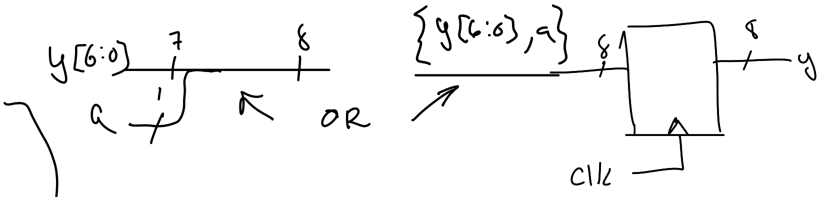


Exercise 2: Using the schematic symbols shown above, convert each of the following System Verilog expressions into a schematic.

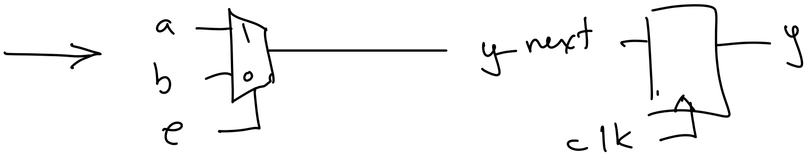
```
always_ff@(posedge clk)
  y = a ;
```



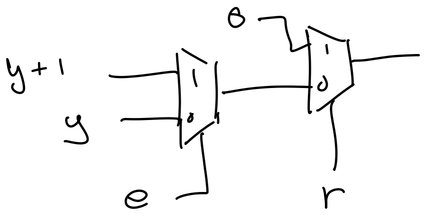
```
always_ff@(posedge clk)
  y[7:0] = {y[6:0],a} ;
```



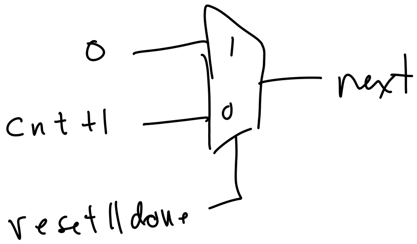
```
always_ff@(posedge clk)
  y = y_next ;

assign y_next = e ? a : y ;
```



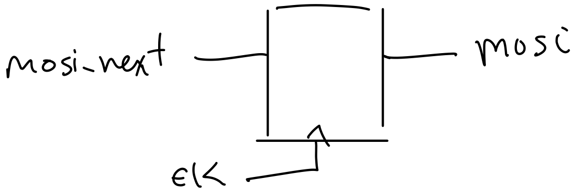
```
always_ff@(posedge clk)
  y = y_next ;
assign y_next = r ? '0 : e ? y+1'b1 : y ;
```



```
assign next = ( reset || done ) ? '0 : cnt+'b1 ;
```



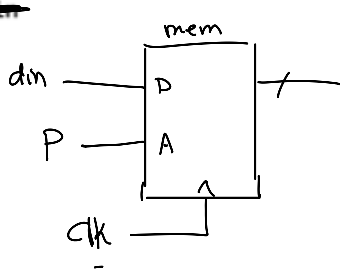
```
always_ff@(posedge clk)
    mosi = mosi_next ;

assign mosi_next = falling ? sr[31] : mosi ;
```



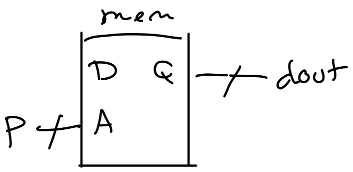
```
always_ff@(posedge clk)
   cnt = cnt_next ;
```

dimensions: packed unpacked

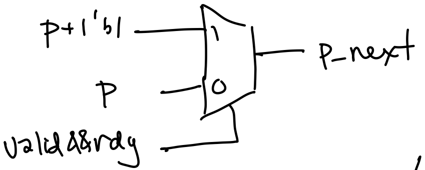
```
// logic [31:0] mem [15:0]
always_ff@(posedge clk) ~~begin~~
   mem[p] <= din ;
```



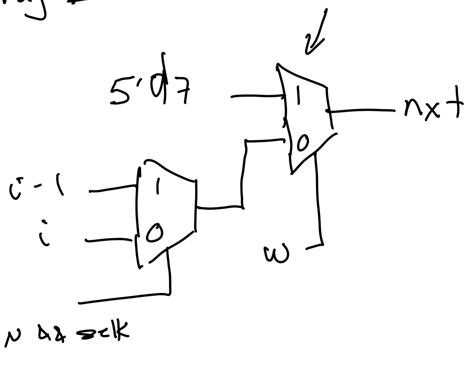
```
// logic [31:0] mem [15:0]
dout = mem[p] ;
```



```
p_next = valid && rdy ? p + 1'b1 : p  ;
```

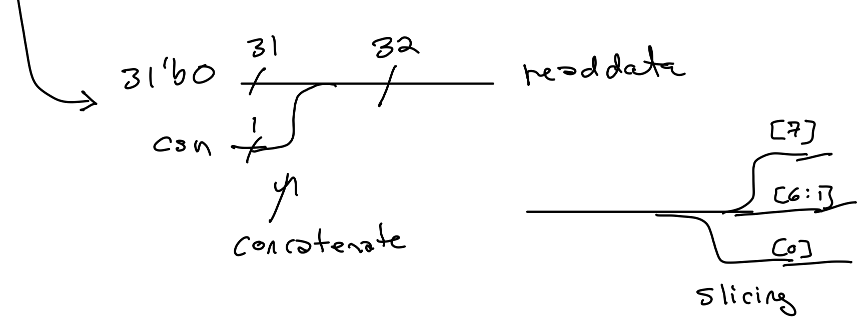


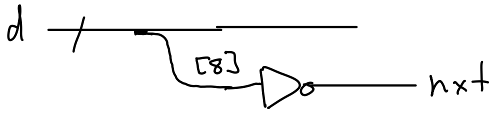
```
// i, j are logic[4:0]; w, sclk are logic
nxt = w ? 5'd7 : ( j==N && sclk ) ? i-1 : i ;
```



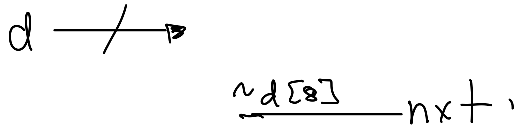
```
readdata = {31'b0,csn} ; // csn is logic
```

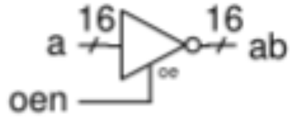
```
nxt = ~d[8] ;
```





OR

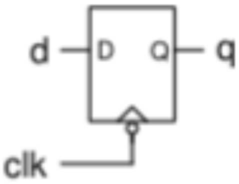


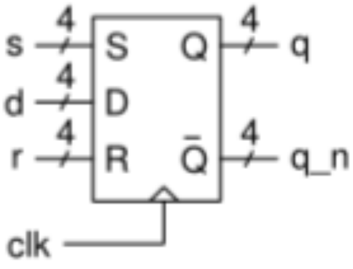


assign ab = oen ? ~a : 4'hzzzz;

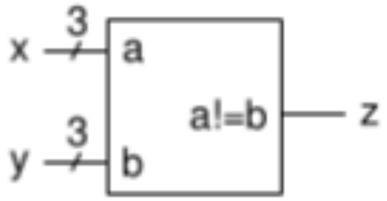
!a   ~a   -1 ^ a

logical   bitwise

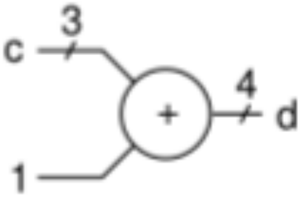




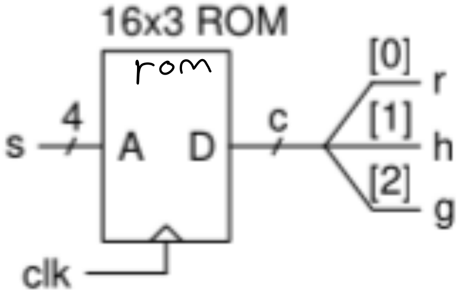
Exercise 3: Write System Verilog that would generate each of the following schematics. Include any required signal declarations (using **logic**).



```
logic [2:0] x, y;
logic z;

assign z = x!=y;
```



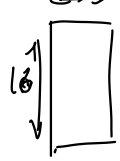
```
logic [2:0] c;
logic [3:0] d
    assign d = c+1'd1;
```



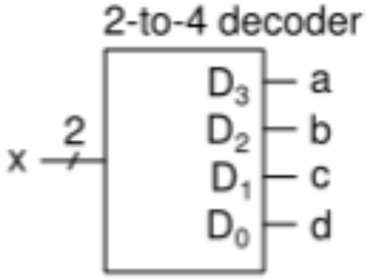
```
logic [3:0] s;
logic [2:0] rom [15:0];
```

packed

unpacked



```
assign {g,h,r} = rom[s];
```



in		output			
		a	b	c	d
0	0 0	0	0	0	1
1	0 1	0	0	1	0
2	1 0	0	1	0	0
3	1 1	1	0	0	0

truth table for decoder

T.T. conversion to Verilog

```
assign {a, b, c d} = in == 2'b00 ? 4'b0001 :
                     in == 2'b01? 4'b0010
                     in == 2'b10? 4'b0100 : 4'b1000;
```

more concise

```
assign {a,b,c,d} = 1 << in;
```