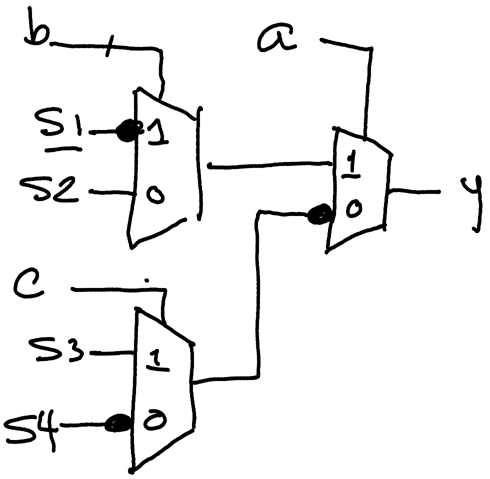
HDL Idioms

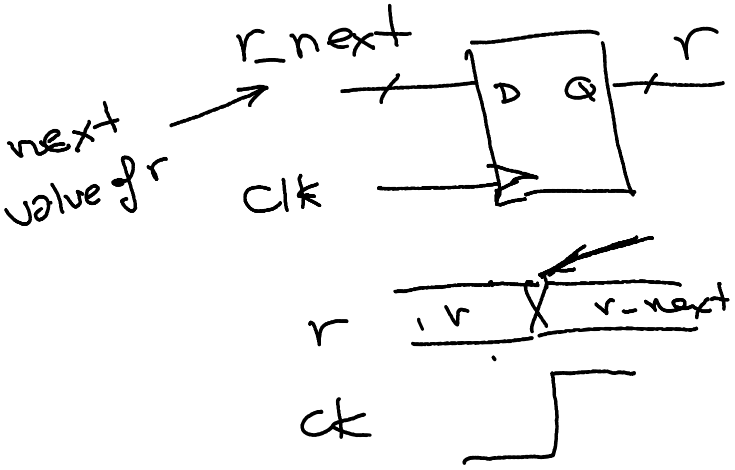
Exercise 1: Draw the schematic corresponding to: `y = a ? ( b ? s1 : s2 ) : ( c ? s3 : s4 )`

a = 0   b = 1

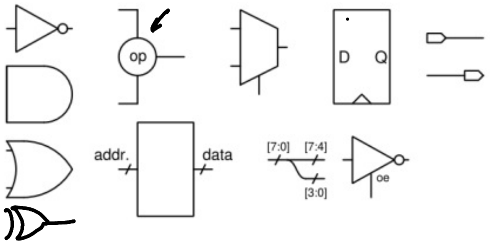
c = 0



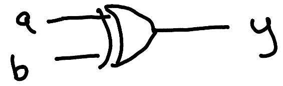
Exercise 2: Which signal is the next value of the register? When does it become the current state?



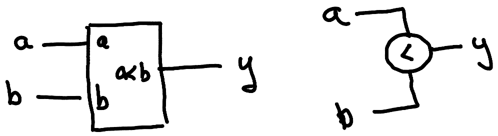
Exercise 3: Using the schematic symbols shown below, convert each of the following System Verilog expressions into a schematic.



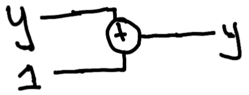
```
y = a ^ b ;
```



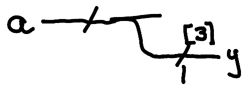
```
y = a < b ;
```



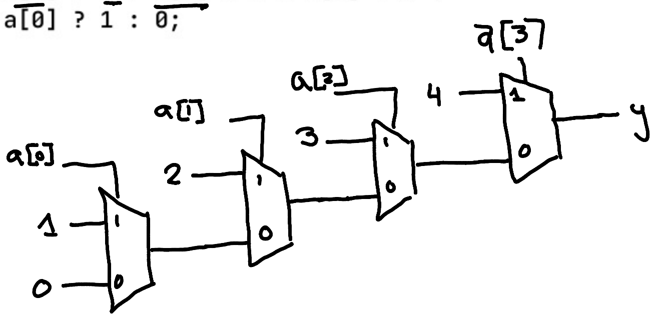
```
y = y+1 ;
```

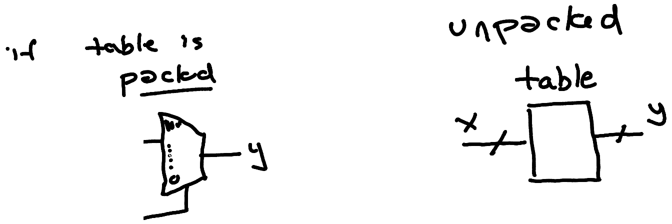


```
y = a[3] ;
```



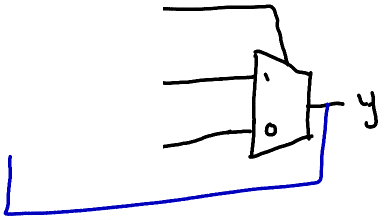
```
y = a[3] ? 4 : a[2] ? 3 : a[1] ? 2 :
    a[0] ? 1 : 0;
```



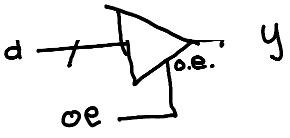


```
y = table[x] ;
```

```
if ( y < b )
  y = y+1 ;
else
  y = y-1 ;
```

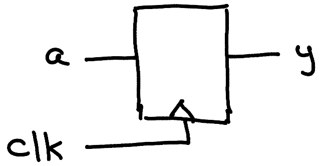


```
y = oe ? d : 16'hzzzz ;
```

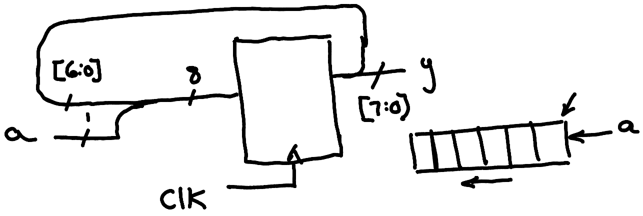


Exercise 4: Using the schematic symbols shown above, convert each of the following System Verilog expressions into a schematic.

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

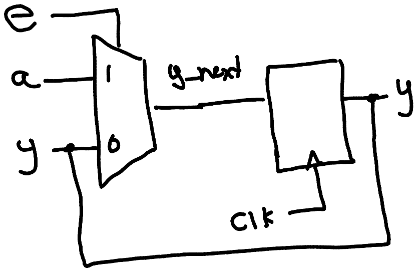


```
always_ff@(posedge clk)
  y[7:0] = {y[6:0],a} ;
```



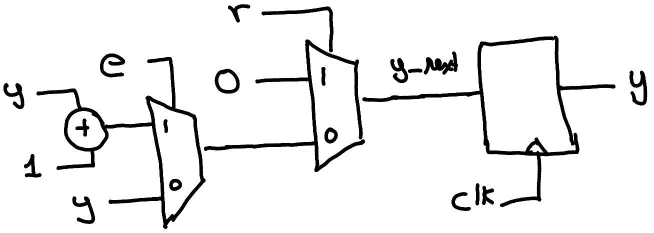
```
always_ff@(posedge clk)
   y = y_next ;

always_comb
  if ( e )
     y_next = a ;
  else
     y_next = y ;
```

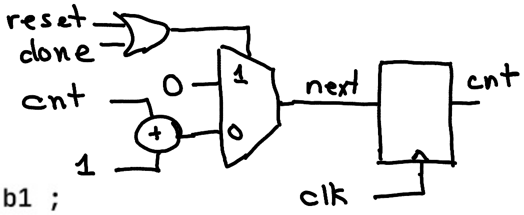


```
always_ff@(posedge clk)
   y = y_next ;

always_comb
  if ( r )
    y_next = '0 ;
  else
    if ( e )
       y_next = y+1'b1 ;
    else
       y_next = y ;
```



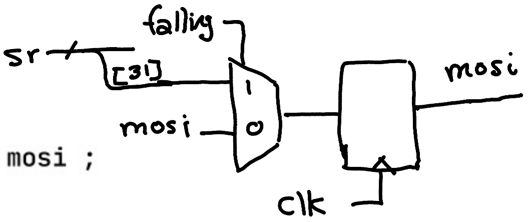
```
alway_ff@(posedge clk)
      cnt = next ;
```



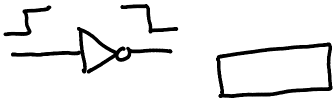
```
next = ( reset || done ) ? '0 : cnt+'b1 ;
```

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

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

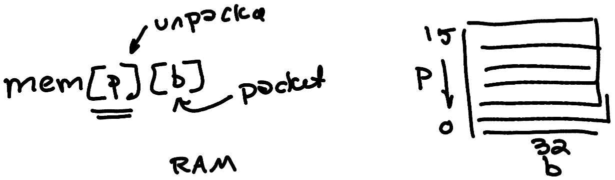


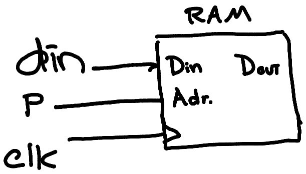
```
always_ff@(posedge clk)
   cnt = cnt_next ;
```



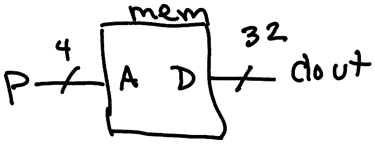
packed unpack

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



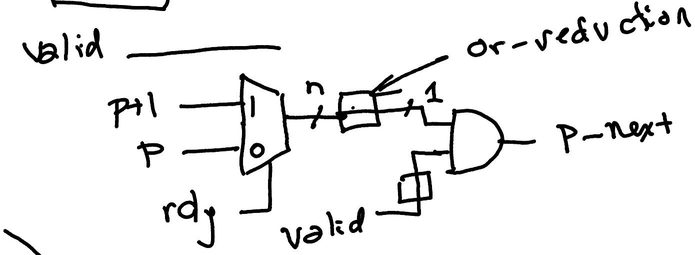


```
// logic [31:0] mem [15:0]
dout = mem[p] ;
```

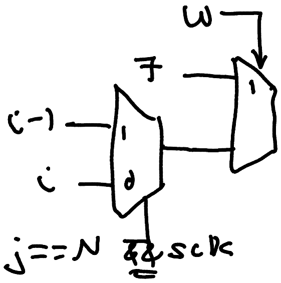


assign

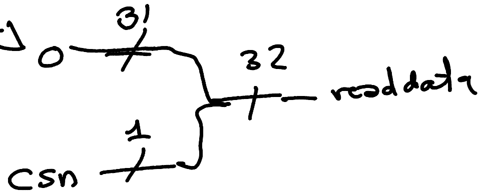
```
p_next = valid && rdy ? p + 1'b1 : p  ;
```



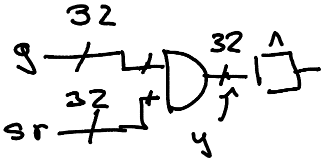
```
// i, j are logic[4:0]; w, sclk are logic
nxt = w ? 5'd7 : ( j==N && sclk ) ? i-1 : i ;
```



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



```
crc = ^ (g&sr) ; // g and sr are logic[31:0]
```

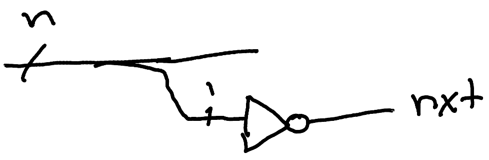


crc = y[31]^y[30]^y[29]...^y[0]

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

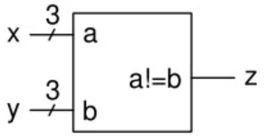
& - bitwise } AND
&& - logical

~ - bitwire
! - logical



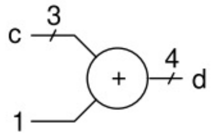
Exercise 5: Write System Verilog that would generate each of the following schematics. Include any required signal declarations (using `logic`).

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

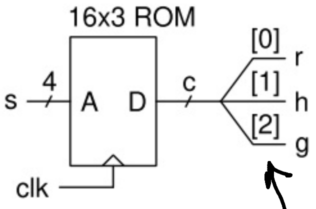


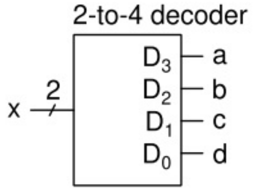
```
logic [2:0] c; logic [3:0] d;
assign d = c + 4'd1;
```

1

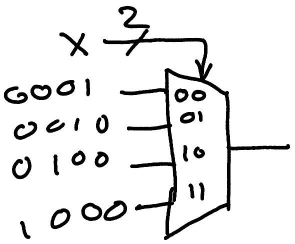


```
logic [2:0] ROM [0:15];        logic [2:0] c;
                               logic[3:0] s;
always_ff@(posedge clk)
   c = ROM[s];
assign {g,h,r} = c;
```





00 → 0001
01 → 0010
10 → 0100
11 → 1000



```
always_comb
unique case (x)
   0 : {a,b,c,d} = 4'b0001;
   1 :       ⋮
   2 :
default:
   endcase
```

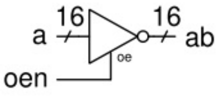
```
assign {a,b,c,d} = 1 << x ;
```

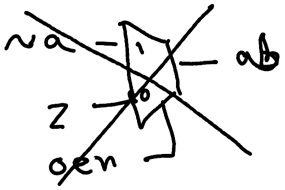
left-shift.

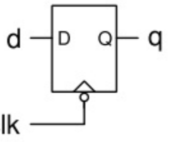
input	output
address	data
0	4
1	8
2	3
3	0
⋮	⋮
15	9

```
case (a)
→{x,y}:
→{1,0}:
```

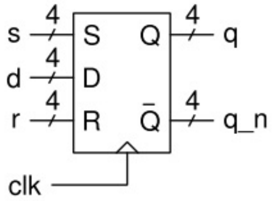
```
assign ab = oen ? ~a : 16bz ;
                        (or 'z )
```



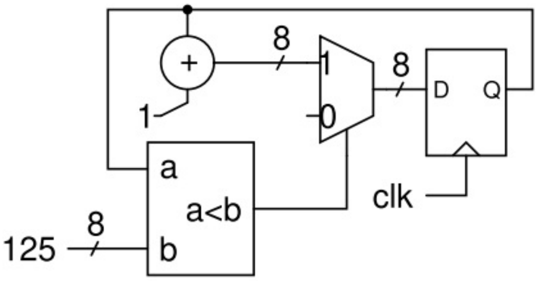




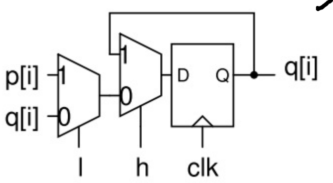
```
alway-ff@(negedge clk)
      q = d ;
```



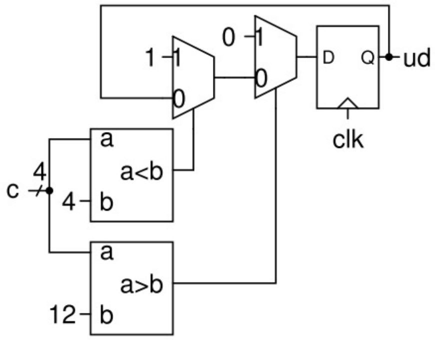
```
alway-ff@ (posedge clk) begin
    q = d ;
    q = q | s ;      } bitwise
    q = q & ~r ;       masking
     q-n = ~q ;
end
```



```
assign  q_next = h ? q : l ? p : q ;
always_ff@ (posedge clk)
        q =  q_next;
```

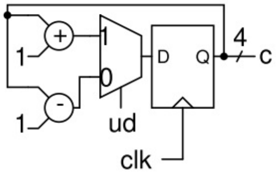


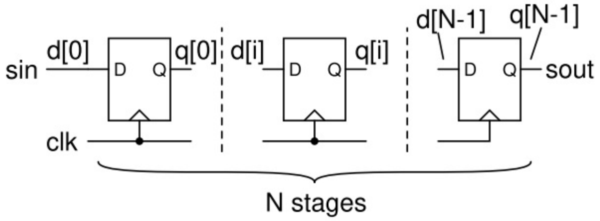
```
assign ud_next =  c > 12 ? 0 :
        c < 4 ? 1 : ud;
always_ff@ (posedge clk)
        ud = ud_next;
```

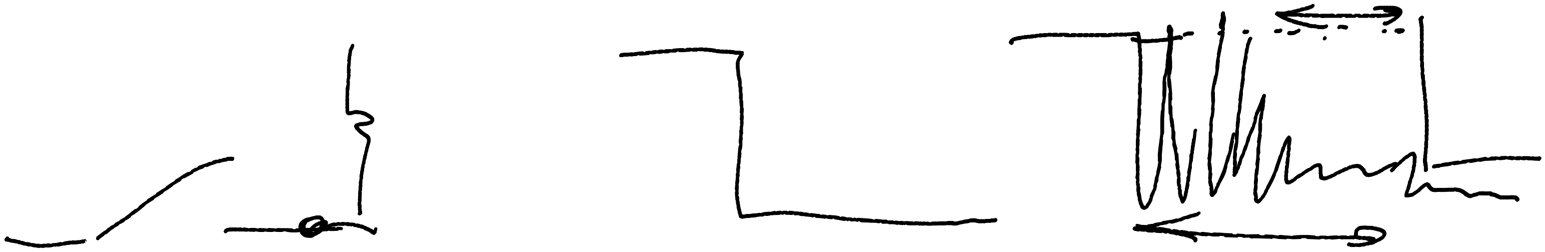


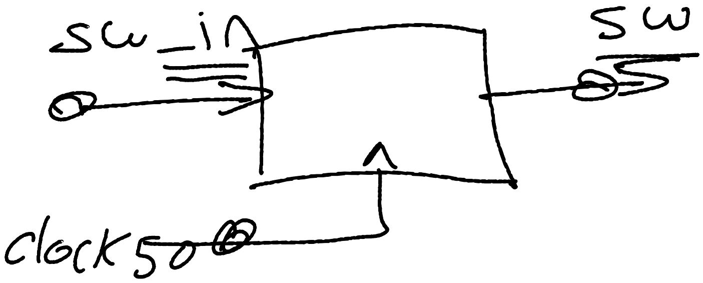
```
always_ff@ (posedge clk)
        c =  c_next;

assign  c_next = ud ? c+1 : c-1 ;
```








sw_in
sw
clock50