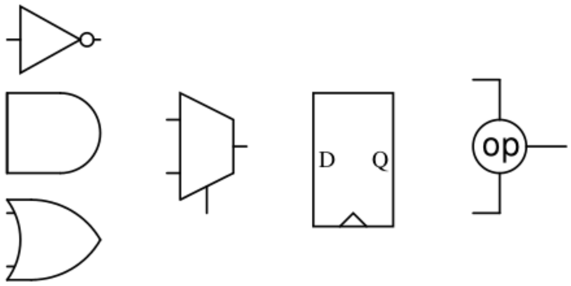
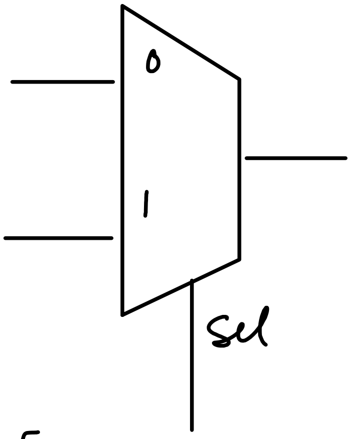
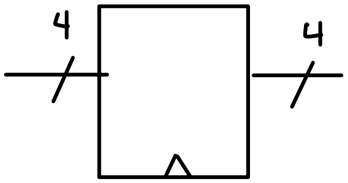
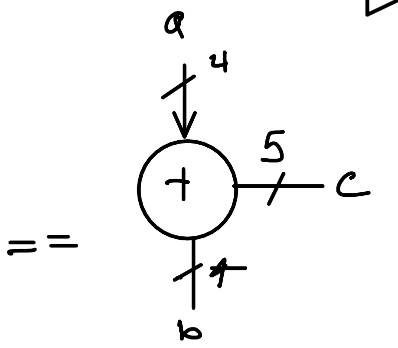








a ,b logi

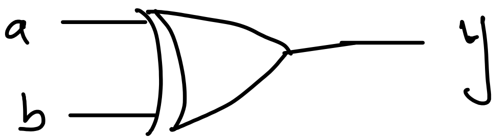
```
y = a ^ b ;
```

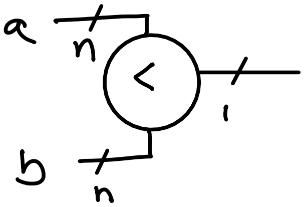
n

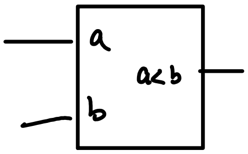
```
y = a < b ;
```

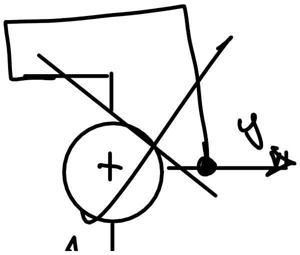
n

~~y = y+1;~~

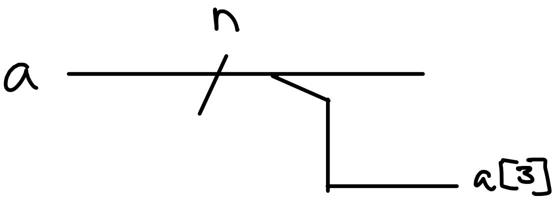




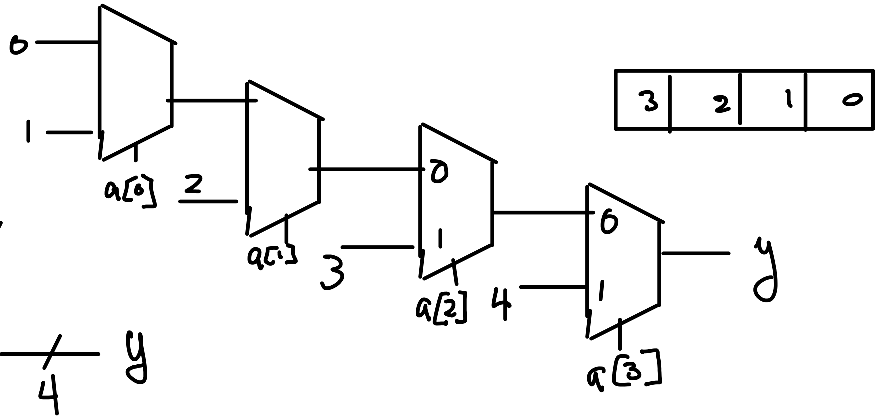




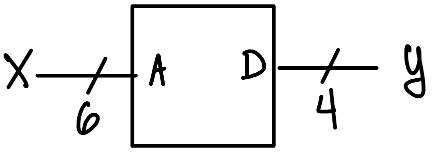
```
y = a[3] ;

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

y = table[x] ;
```

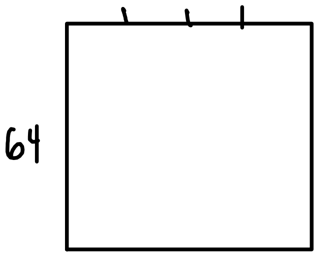


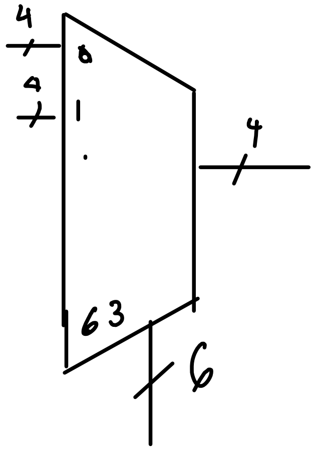
number of el.



logic $\underbrace{[3:0]}_{\text{width}}$ table [5:0];





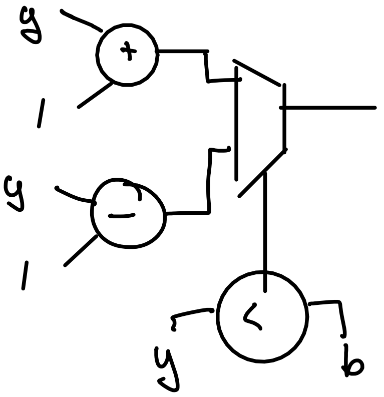


case (x)
1 : y = ___ ;
2 ;
3 ;

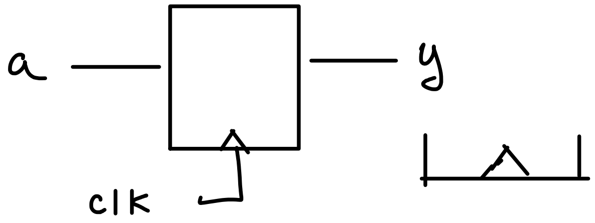
casez(y)

array → ROM
case → N-mux
if/else → 2-way mux

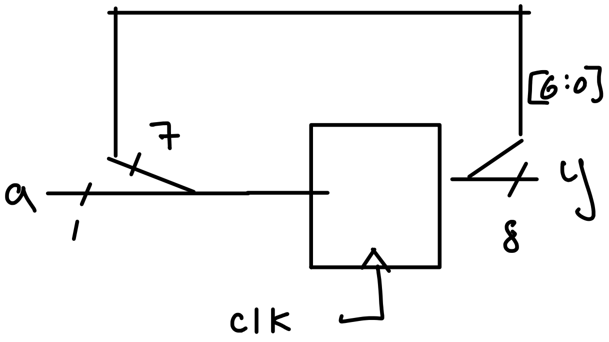
```
if ( y < b )
z ~~y~~ = y+1 ;
else
z ~~y~~ = y-1 ;
```

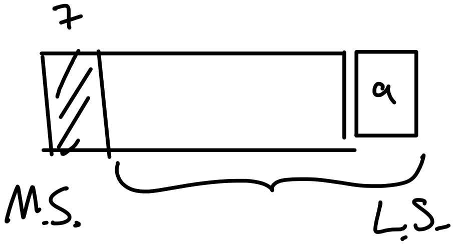


```
 always@(posedge clk)
   y<= a ;
```



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





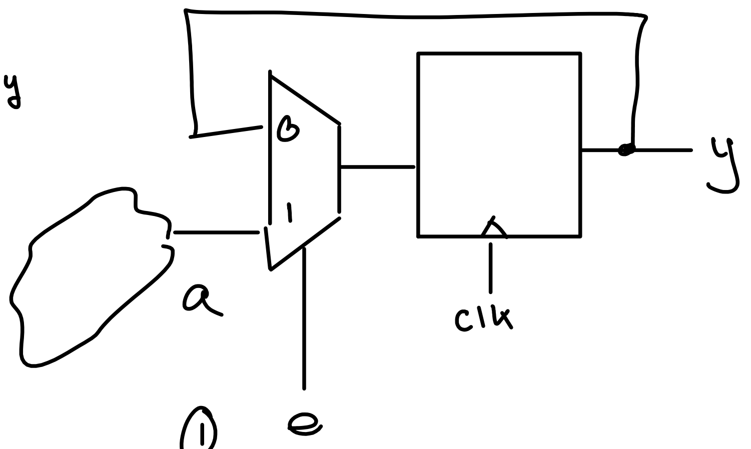
```
always@(posedge clk) begin
  if ( e )
     y <= a ;
  else
     y <= y ;
```

→ old value of y

→ new value of y

/⇐y

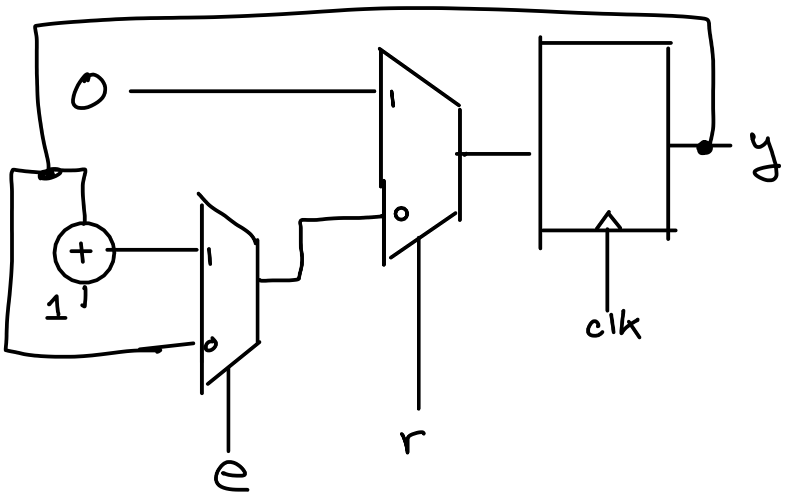
end



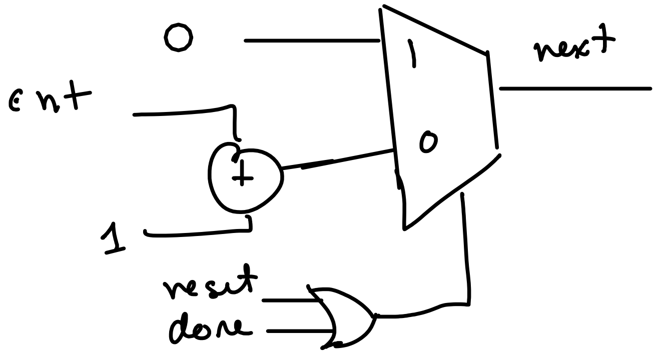
using <= x takes on old value ①

= x " new val ②

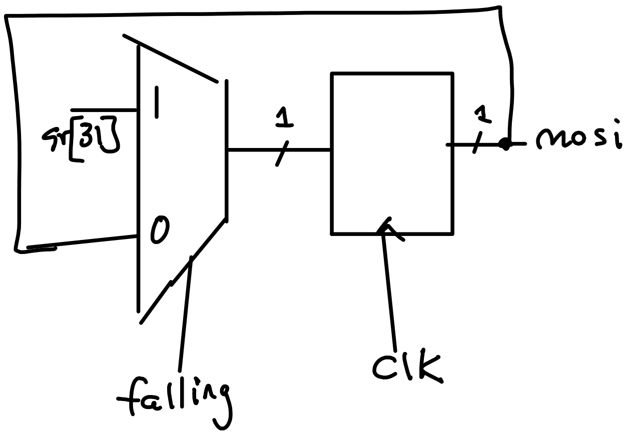
```
always@(posedge clk)
  if ( r )
    y <= '0 ;
  else
    if ( e )
       y <= y+1'b1 ;
    else
       y <= y ;
```



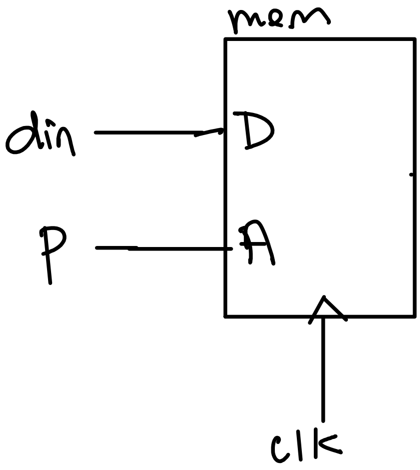
```
next = ( reset || done ) ? '0 : cnt+'b1 ;
```



```
always@(posedge clk)
  if ( falling )
     mosi <= sr[31] ;

always@(posedge clk)
   cnt <= cnt_next ;
```



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



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