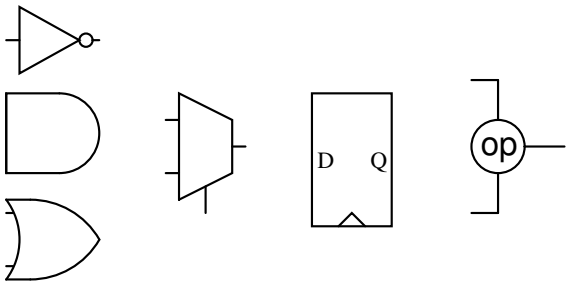
Common HDL Constructs

For efficient logic design it is necessary to be able to visualize the hardware that would be generated by an HDL description. This lecture describes some common HDL constructs and their corresponding hardware implementations.
After this lecture you should be able to convert back and forth between simple digital logic circuits and the corresponding Verilog descriptions.

Using the following schematic symbols:



draw schematics for each of the following HDL constructs:

```
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 ;
```

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

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

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

```
 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] ;
```

```
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]
```

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