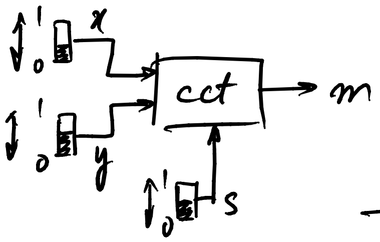
Design example

design a cct that can control an LED (m) from either of two switches x, y. one of x and y is selected by using a 3rd switch



s	m
0	x
1	y

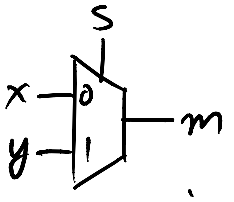
$m = \bar{s}x + sy$

s	x	y	m
0	0	0	0
0	0	1	0
0	1	0	1 ←
0	1	1	1 ←
1	0	0	0
1	0	1	1 ←
1	1	0	0
1	1	1	1 ←

SOP form of $m = \underbrace{\bar{s}x\bar{y} + \bar{s}xy}_{} + \underbrace{s\bar{x}y + sxy}_{}$

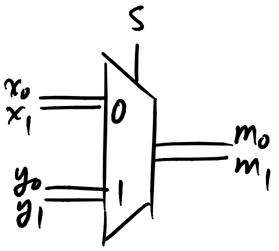
$= \bar{s}x + sy$

This cct is called a 2-to-1 multiplexer (MUX)



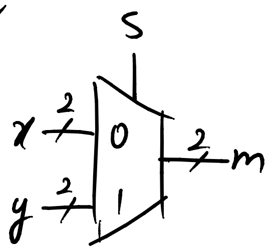
basic building block.

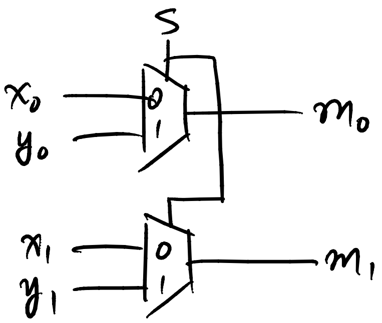
extend it to multibit 2-to-1 mux



2-bit 2-to-1 mux

s	$m_0\ m_1$
0	$x_0\ x_1$
1	$y_0\ y_1$





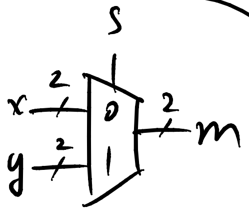
Verilog 2-to-1 mux

```
module mux2to1 (x, y, s, m);
  input x, y, s;
  output m;
  assign m = (~s & x) | (s & y);
endmodule
```

NOT* OR

* ! and ~ have the same result.

~ ⇒ $-\triangleright\!\circ-$



```
module mux2bit_2to1 (X, Y, s, M);
  input [1:0] X, Y;
  input s;
  output [1:0] M;
  assign M[0] = (~s & X[0]) | (s & Y[0]);
```

```
        assign M[1] = (~s & X[1]) | (s & Y[1]);
      end module
```

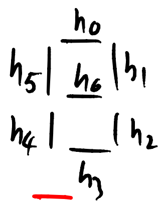
Hierarchical Code (2bit 2-to-1 mux)

```
  module mux2bit_2to1 (X, Y, s, M);
    input [1:0] X, Y;
    input s;
    output [1:0] M;          random name you give
    mux2to1 u1 (X[0], Y[0], s, M[0]);
    mux2to1 u2 (X[1], Y[1], s, M[1]);
  end module                       single bit
```

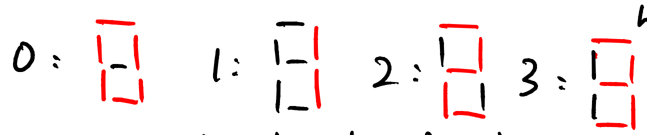
[2:0]	$x_2 x_1 x_0$
[0:2]	$x_0 x_1 x_2$

```
  module mux2to1 (x, y, s, m);
    input x, y, s;
    output m;
    assign m = (~s & x) | (s & y);
  end module    ↑         ↑
               NOT*       OR
```

7-segment display



create a ckt that has inputs $x_1 x_0$ that represent 2 digit number. Display the number on a 7-seg. display



	x_1	x_0	h_0	h_1	h_2	h_3	h_4	h_5	h_6
0	0	0	1	1	1	1	1	1	0
1	0	1	0	1	1	0	0	0	0
2	1	0	1	1	0	1	1	0	1
3	1	1	1	1	1	1	0	0	1

$h_0 = \bar{x}_1\bar{x}_0 + x_1\bar{x}_0 + x_1 x_0 = \bar{x}_1\bar{x}_0 + x_1 = \bar{x}_0 + x_1$

$h_1 = 1$

$h_2 = \bar{x}_1 + x_0$

$h_3 = h_0 = \bar{x}_0 + x_1$

$h_4 = \bar{x}_0$

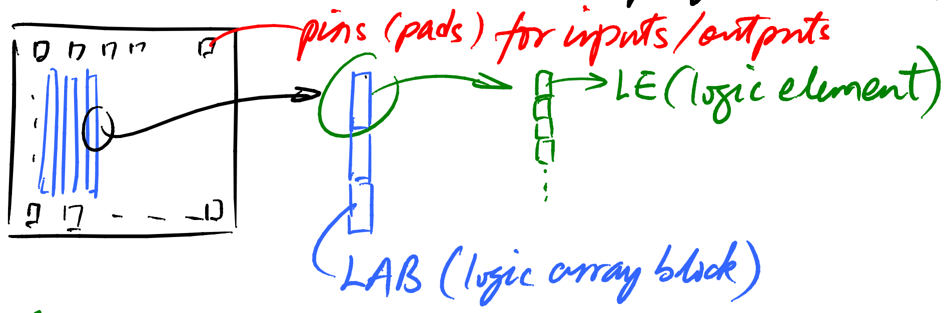
$h_5 = \bar{x}_1\bar{x}_0$

$h_6 = x_1$

```
module seg7 (input x1, x0, output [0:6] H);
  assign H[0] = x1 | ~x0 ;        binary
  assign H[1] = 1'b1;             value "1"
                                  1 bit
```

```
    assign H[2] = ~x1 | x0;
    assign H[3] = x1 | ~x0;
    assign H[4] = ~x0;
    assign H[5] = ~x1 & ~x0;
    assign H[6] = x1;
endmodule
```

FPGA (Field programmable gated array)

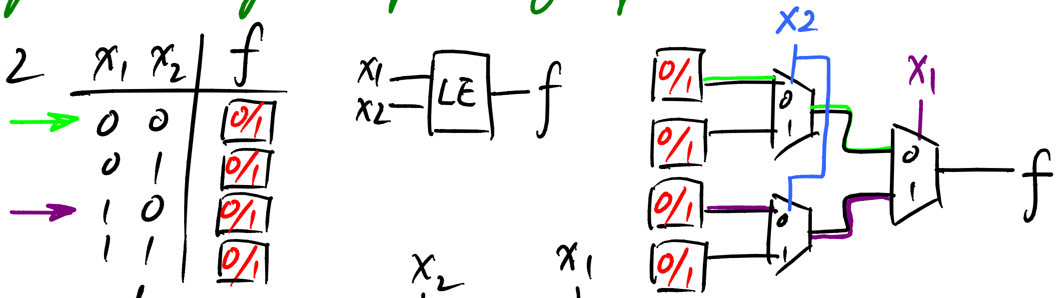


Logic element (LE)

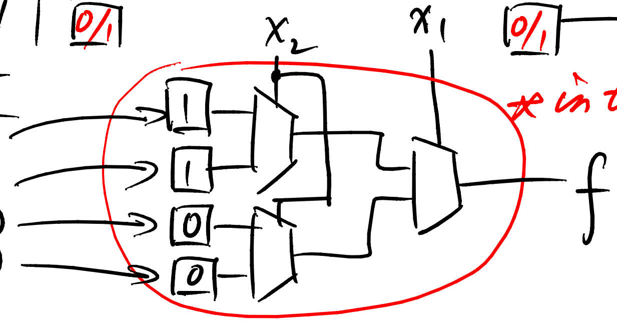
each LE can store a truth table that represents the logic function that is implemented. If the LE has k inputs, then it can represent any k-input logic function

e.g. k=2

x_1	x_2	f
0	0	0/1
0	1	0/1
1	0	0/1
1	1	0/1



x_1	x_2	f
0	0	1
0	1	1
1	0	0
1	1	0

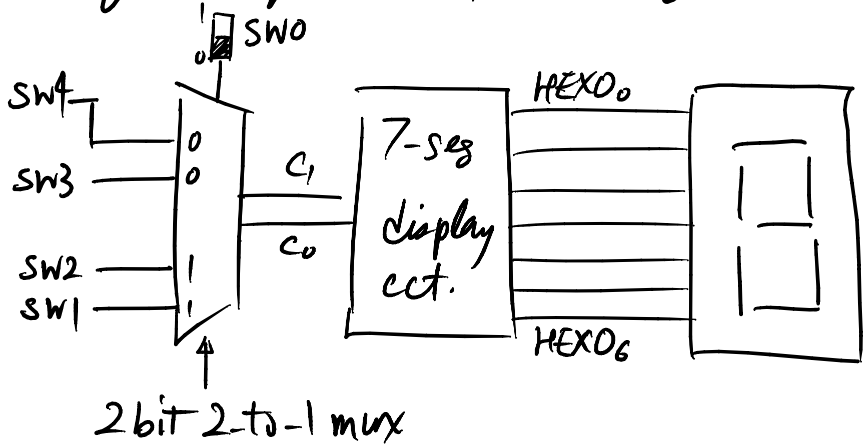


* in this example, the 2-input LUT is used to implement the function $f = \bar{x}_1\bar{x}_2 + \bar{x}_1 x_2$

$\Rightarrow$ This is called look up table (LUT)

DE2. 35,000 4-input LUT.

Design example (cont.) 7-segment display



```
module hier_ex(input [4:0]SW, output [0:6]HEX0);
    wire [1:0]C;
    wire [0:6]H;
    mux2bit_2to1 U1 (SW[4:3], SW[2:1], SW[0], C);
    seg7  U2 (C[1], C[0], H);
    assign HEX0 = ~H;
end module
```

module instantiation (make a copy)

7seg (on DE2 board) lights up when the driving signal is "0"

```
module mux2bit_2to1 (X, Y, S, M);
   input [1:0]X, Y;
   input S;
   output [1:0] M;
   mux2to1 U1 (X[0], Y[0], S, M[0]);
   mux2to1 U2 (X[1], Y[1], S, M[1]);
end module
```

```
module mux2to1 (x, y, s, m);
   input x, y, S;
   output m;
   assign m=(~S&x)|(S&y);
end module
```

```
module seg7 (input x1, x0, output [0:6]H);
     assign H[0] = x1 | ~x0 ;
     assign H[1] = 1'b1 ;
     assign H[2] = ~x1 | x0;
     assign H[3] = x1 | ~x0;
     assign H[4] = ~x0 ;
     assign H[5] = ~x1 & ~x0;
     assign H[6] = x1 ;
end module
```