Lecture 1 - Combinational Logic Design with VHDL

Exercise 1: A chip has an input labelled $\overline{OE}$ that is used to turn on ("enable") its output. Is this input an active-high or active-low signal? Will the output be enabled if the input is high? Will the output be enabled if the input is 1?

- $\overline{OE}$ – active low
- no
- 1 < if 1 is a truth value: yes
  if 1 is a logic level: no

Exercise 2: Fill in the last two rows.

a	b	c	p
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

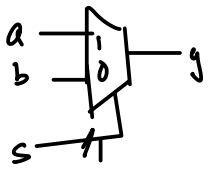
Exercise 3: Write out the sum-of-products equation for p. Evaluate the expression for the first two lines in the table.

$$p = \bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}c + ab\bar{c}$$

p = 111 + 100 + 010 + 001 = 1+0+0+0 = 1

p= 110 + 101 + 011 + 000 = 0+0+0+0 = 0

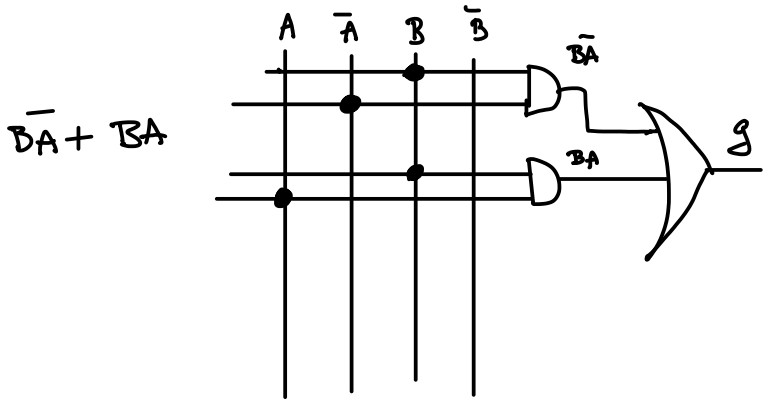
Exercise 4: Write out the truth table and the canonical (unsimplified) sum-of-products expression for a 2-to-1 multiplexer.



ab	s	y
00	0	0
00	1	0
01	0	1
01	1	0
10	0	0
10	1	1
11	0	1
11	1	1

y=a*bs* +ab*s + abs* + abs

Exercise 5: Fill in the last line of the table. Draw the schematic of a circuit that implements the logic function for the 'g' segment.



Exercise 6: Write a VHDL description for the circuit that would generate the 'a' and 'b' outputs for the 7-segment LED driver shown previously.

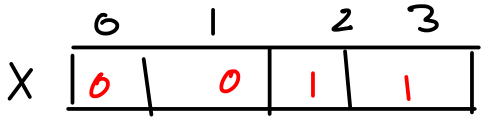
and
or
→ not

$$a = \bar{A}\bar{B} + \bar{A}B + AB$$
$$b = 1$$

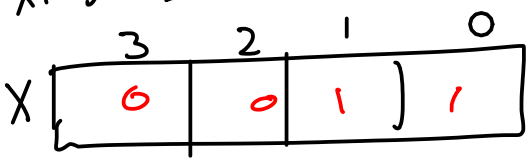
$$a = \bar{x}\bar{y} + \bar{x}y + xy$$
$$b = 1$$

```
a <= (not X and not Y) or (not X and Y) or (X and Y);
b <= '1';
```

Exercise 7: If x is declared as `bit_vector (0 to 3)` and in an architecture the assignment `x<="0011"` is made, what is the value of `x(3)`? What if x had been declared as `bit_vector (3 downto 0)`?

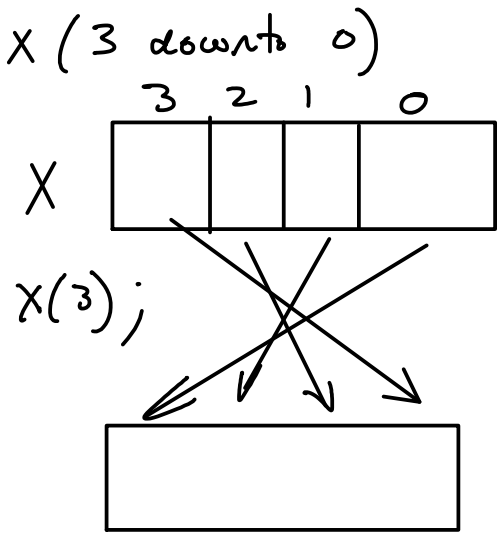


X(3) = 1



X(3) = 0

Exercise 8: Write a VHDL description that uses '&' to assign y a bit-reversed version of a 4-bit vector x.




```
y <= X(0) & X(1) & X(2) & X(3);
```

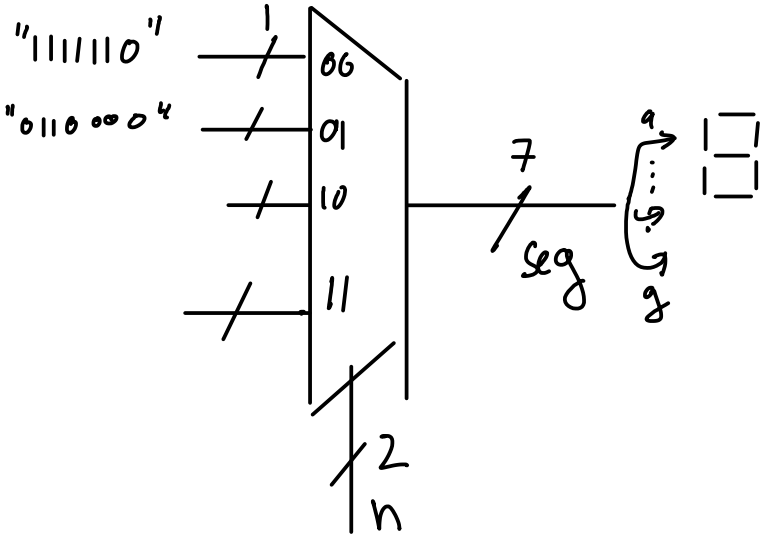
shift left:

```
y <= X(2) & X(1) & X(0) & '0';
y <= X(2 downto 0) & '0';
```

Exercise 9: Write a VHDL description for a 2-to-4 decoder using a 2-bit input and a 4-bit output.

```
with s select y <=
   "0001" when "00",
   "0010" when "01",
   "0100" when "10",
   "1000" when others ;
```

implementing the 7-segment decoder using a multiplexer:



a mux is like a switch: