

# ECE241 Lab6 Background

---

In this lab you will implement simple two-level logic using VHDL concurrent statements. Follow the two examples below, plus Examples 4.3, 4.4 and 4.6 in the text.

These examples implement the truth table:

| a2 | a1 | a0 | y |
|----|----|----|---|
| 0  | 0  | 0  | 0 |
| 0  | 0  | 1  | 1 |
| 0  | 1  | 0  | 1 |
| 0  | 1  | 1  | 0 |
| 1  | 0  | 0  | 1 |
| 1  | 0  | 1  | 0 |
| 1  | 1  | 0  | 1 |
| 1  | 1  | 1  | 0 |

**Table 1: truth table for examples**

You will use “conditional assignments”, in which an output is assigned a value based on logical conditions. For example:

```
y <= '1' when x='0' else '0';
```

Implements an inverter with input x and output y. Notice that in std\_logic, bit values are surrounded by apostrophes.

This code implements a 2:1 multiplexer, with inputs a0 and a1, and select signal s.

```
y <= a0 when s = '0' else a1;
```

You can include logic functions like ‘and’, ‘or’, ‘xor’ and ‘not’ in the statements. Study the example below, where the text following the keyword ‘when’ and the next three lines each implements a minterm.

```
-- VHDL example with conditional assignments
```

```
--
```

```
library ieee;
```

```
use ieee.std_logic_1164.all;
```

```
entity lab6a is
```

```
    port (a2, a1, a0: in std_logic;
```

```
          y: out std_logic);
```

```
end lab6a;
```

```
architecture demo of lab6a is
```

```
begin
```

```
    y <= '1' when ((a2 = '0' and a1 = '0' and a0 = '1') or  
                  (a2 = '0' and a1 = '1' and a0 = '1') or  
                  (a2 = '1' and a1 = '0' and a0 = '0') or  
                  (a2 = '1' and a1 = '1' and a0 = '0')) else '0';
```

```
end demo;
```

For large designs like the 7-segment decoder, this becomes tedious, so we use a shorthand to represent minterms: a *standard logic vector*. In port statement of the sample below, we combine a2, a1 and a0 into one three-bit signal:

```
    a: in std_logic_vector(2 downto 0);
```

The value of a standard logic vector must be surrounded by quotation marks instead of apostrophes, like this:

```
    a <= "110";
```

In VHDL syntax, we can access an individual array element, or bit, with an index number enclosed in (). So the statement above is the same as:

```
    a(2) <= '1';
```

```
    a(1) <= '1';
```

```
    a(0) <= '0';
```

You can implement the truth table in Table 1 using vectors, as shown below. If you implement your 7-segment decoder this way, you can save a lot of typing and have much more readable (and less error-prone) code.

```
-- VHDL example with conditional assignments
--   using vectors to represent minterms
--
```

```
library ieee;
use ieee.std_logic_1164.all;
```

```
entity lab6b is
  port (a: in std_logic_vector(2 downto 0);
        y: out std_logic);
end lab6b;
```

```
architecture demo of lab6b is
begin
  y <= '1' when (a = "001" or a = "010" or a = "100" or a = "110") else '0';
end demo;
```