

**Institute of Engineering and Technology,
DAVV, Indore (M.P.)**



Department of Electronics and Telecommunication

SESSION: 2020-2021

Subject: Circuit Design Using VHDL

Lab Assignment

SUBMITTED TO:

Dr. Vaibhav Neema sir

SUBMITTED BY:

Mayank Jain

Roll No. – 17E7025

- **Dataflow Modelling**

In this modelling style, the flow of data through the entity is expressed using concurrent (parallel) signal. The concurrent statements in VHDL are WHEN and GENERATE.

Besides them, assignments using only operators (AND, NOT, +, *, sll, etc.) can also be used to construct code.

Finally, a special kind of assignment, called BLOCK, can also be employed in this kind of code.

In concurrent code, the following can be used –

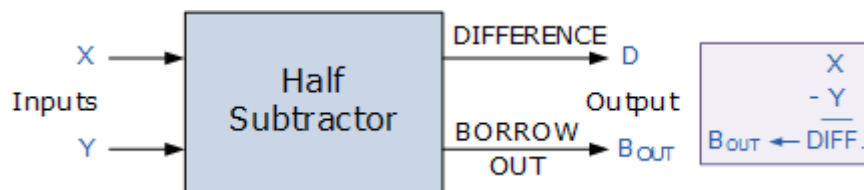
- Operators
- The WHEN statement (WHEN/ELSE or WITH/SELECT/WHEN);
- The GENERATE statement;
- The BLOCK statement.

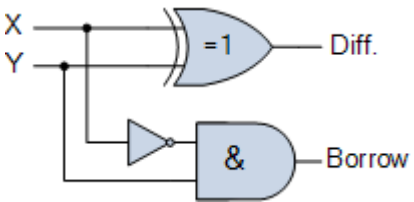
- **2-bit Subtractor**

A **Binary Subtractor** is a decision making circuit that subtracts two binary numbers from each other, for example, $X - Y$ to find the resulting difference between the two numbers. the *binary subtractor* produces a DIFFERENCE, D by using a BORROW bit, B from the previous column.

A half subtractor is a logical circuit that performs a subtraction operation on two binary digits. The half subtractor produces a sum and a borrow bit for the next stage.

Half Subtractor with Borrow-out



Symbol	Truth Table			
	A	B	DIFFERENCE	BORROW
	0	0	0	0
	0	1	1	0
	1	0	1	1
	1	1	0	0

From the truth table of the half subtractor we can see that the DIFFERENCE (D) output is the result of the Exclusive-OR gate and the Borrow-out (E) is the result of the NOT-AND combination. Then the Boolean expression for a half subtractor is as follows.

For the **DIFFERENCE** bit:

$$D = A \text{ XOR } B = A \oplus B$$

For the **BORROW** bit

$$E = \text{not-}A \text{ AND } B$$

One major disadvantage of the *Half Subtractor* circuit when used as a binary subtractor, is that there is no provision for a “Borrow-in” from the previous circuit when subtracting multiple data bits from each other. Then we need to produce what is called a “full binary subtractor” circuit to take into account this borrow-in input from a previous circuit.

- **Design Code**
-- Simple Half Subtractor design

```
library IEEE;
use IEEE.std_logic_1164.all;

entity half_sub is
port(
    a: in std_logic;
    b: in std_logic;
    d: out std_logic;
    e: out std_logic);
end half_sub;

architecture dataflow of half_sub is
begin
    process(a, b) is
    begin
        d <= a xor b;
        e <= not a and b;
    end process;
end dataflow;
```

- **Testbench code**

```
library IEEE;

use IEEE.std_logic_1164.all;

entity testbench is
end testbench;

architecture tb of testbench is

component half_sub is
port(
    a: in std_logic;
    b: in std_logic;
```

```
d: out std_logic;

e: out std_logic);

end component;

signal a_in, b_in, d_out, e_out: std_logic;

begin

DUT: half_sub port map(a_in, b_in, d_out, e_out);

process

begin

    a_in <= '0';

    b_in <= '0';

    wait for 1 ns;

    a_in <= '0';

    b_in <= '1';

    wait for 1 ns;

    a_in <= '1';

    b_in <= '0';

    wait for 1 ns;

    a_in <= '1';

    b_in <= '1';

    wait for 1 ns;

    wait;

end process;

end tb;
```

- **Waveform**

