



# VHDL and its Practice (I)

---

**Lecturer: Prof. S. Chang**

**Dept of Electrical and Computer Engineering  
University of Seoul**



# I. Introduction to VHDL

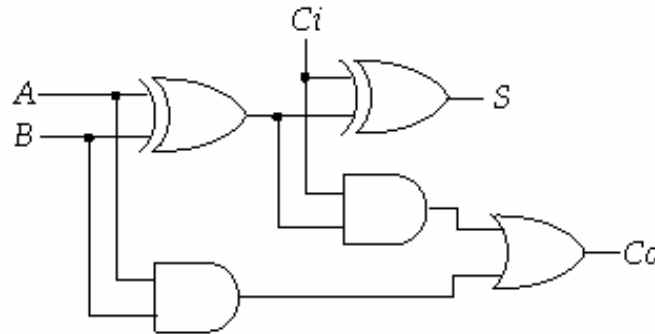
---

## References:

- [http://www.doulos.com/knowhow/vhdl\\_designers\\_guide/](http://www.doulos.com/knowhow/vhdl_designers_guide/)
- <http://www.vhdl.org/>
- <http://www.gmvhdl.com/VHDL.html>
- S. Yalamanchili, “VHDL Starter's Guide”
- Ben Cohen, “VHDL Coding Styles and Methodologies”
- W. F. Lee, “VHDL Coding and Logic Synthesis with Synopsys”

## II. Design of combinational circuits using VHDL

1) 1-bit full adder:



Data flow description:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

standard definitions

entity declarations

```
entity fullAdder is
    port( A,B: in std_logic; -- input bits for this stage
          Ci: in std_logic; -- carry into this stage
          S: out std_logic; -- sum bit
          Co: out std_logic -- carry out of this stage
    );
end fullAdder;
```

comment

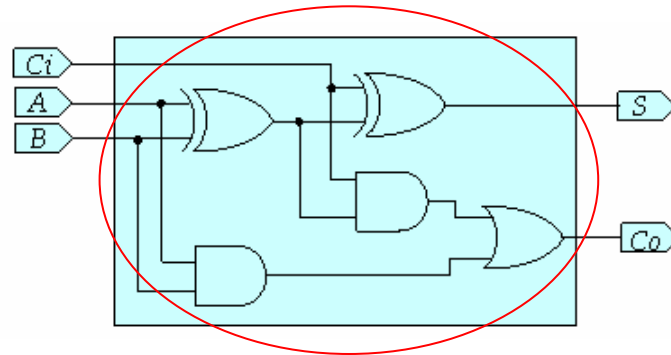
architecture specification

```
architecture a1 of fullAdder is
begin
    S <= A xor B xor Ci;
    Co <= (A and B) or ((A xor B) and Ci);
end a1;
```

## II. Design of combinational circuits using VHDL



entity declarations



architecture specification

Design using  
internal signals

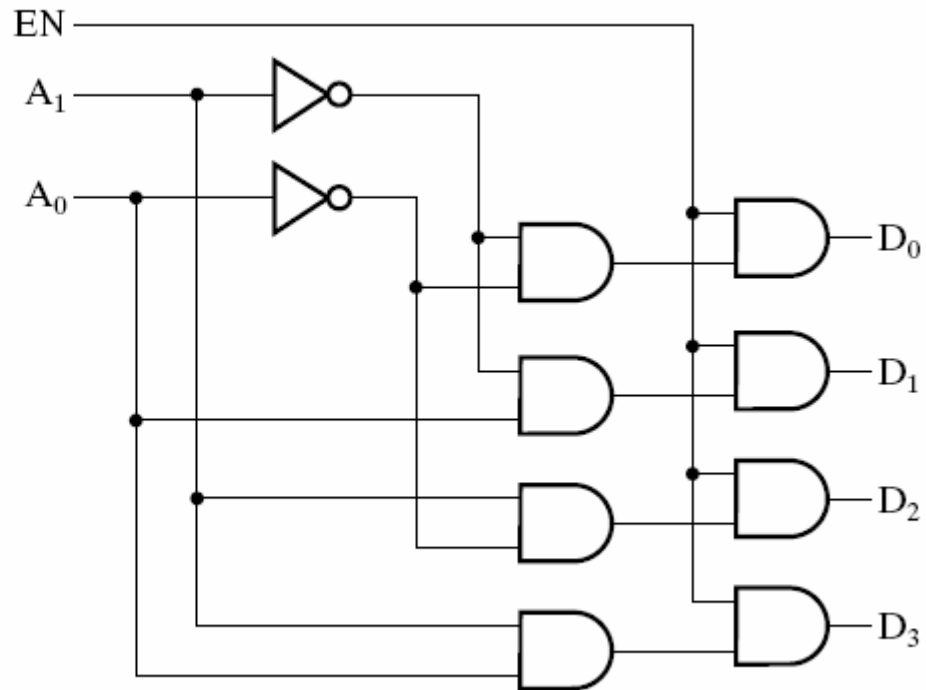
```
architecture a1 of fullAdder is
  signal X: std_logic;
begin
  X <= A xor B;
  S <= X xor Ci;
  Co <= (A and B) or (X and Ci);
end a1;
```

## II. Design of combinational circuits using VHDL

2) 2-to-4-line decoder with enable:

EN	A <sub>1</sub>	A <sub>0</sub>	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(a)



(b)



## II. Design of combinational circuits using VHDL

---

Data flow description:

```
library ieee ;
use ieee.std_logic_1164.all ;
entity decoder_2_to_4_w_enable is
    port(EN, A0, A1: in std_logic;
         D0, D1, D2, D3: out std_logic);
end decoder_2_to_4_w_enable;

architecture dataflow_1 of decoder_2_to_4_w_enable is

    signal A0_n, A1_n: std_logic;
begin
    A0_n <= not A0;
    A1_n <= not A1;
    D0 <= A0_n and A1_n and EN;
    D1 <= A0 and A1_n and EN;
    D2 <= A0_n and A1 and EN;
    D3 <= A0 and A1 and EN;
end dataflow_1;
```

## II. Design of combinational circuits using VHDL

Structural  
description:

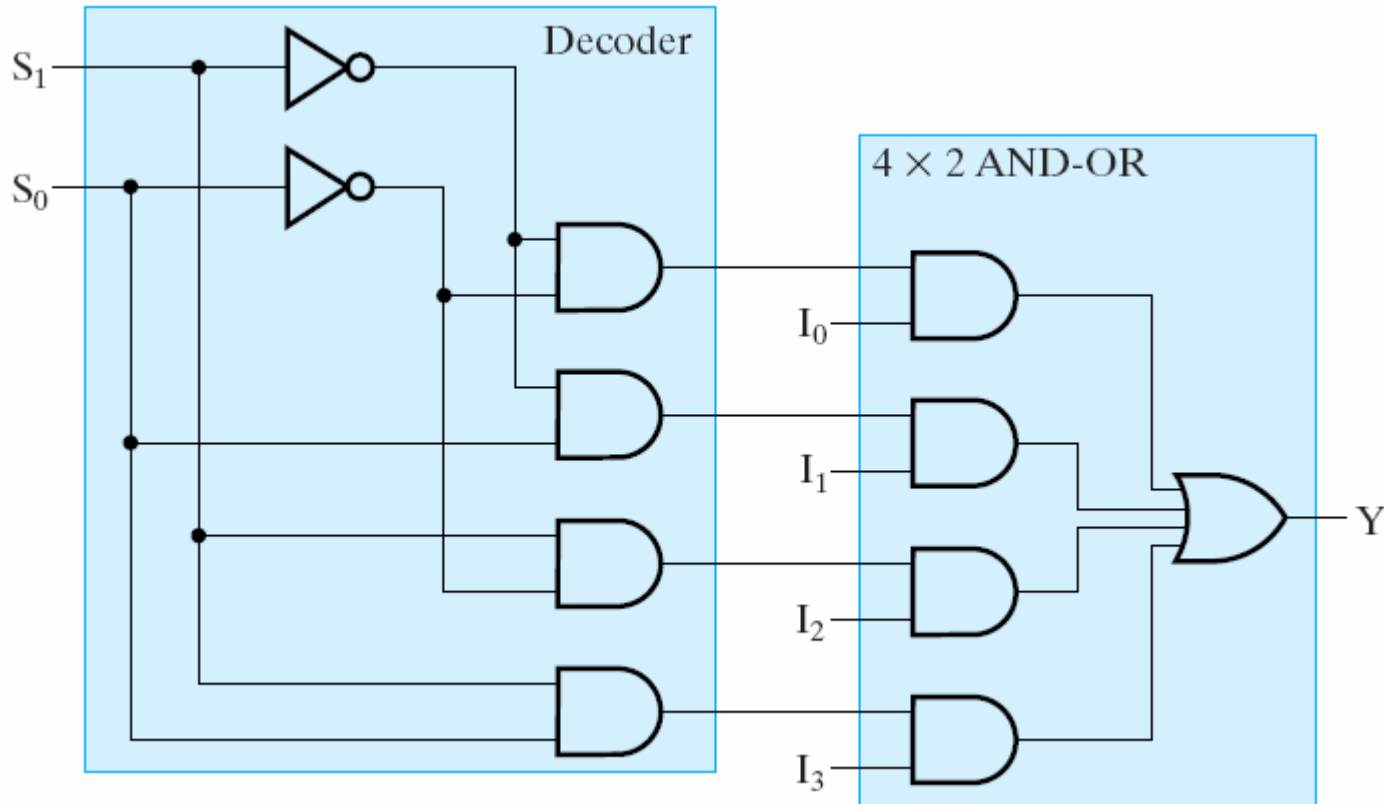
```
library ieee ;
use ieee.std_logic_1164.all ;
entity decoder_2_to_4_w_enable is
    port(EN, A0, A1: in std_logic;
         D0, D1, D2, D3: out std_logic);
end decoder_2_to_4_w_enable;

architecture structural_1 of decoder_2_to_4_w_enable is
    component NOT1
        port(in1: in std_logic;
             out1: out std_logic);
    end component;
    component AND2
        port(in1, in2: in std_logic;
             out1: out std_logic);
    end component;
    signal A0_n, A1_n, N0, N1, N2, N3: std_logic;
begin
    g0: NOT1 port map (in1 => A0, out1 => A0_n);
    g1: NOT1 port map (in1 => A1, out1 => A1_n);
    g2: AND2 port map (in1 => A0_n, in2 => A1_n, out1 => N0);
    g3: AND2 port map (in1 => A0, in2 => A1_n, out1 => N1);
    g4: AND2 port map (in1 => A0_n, in2 => A1, out1 => N2);
    g5: AND2 port map (in1 => A0, in2 => A1, out1 => N3);
    g6: AND2 port map (in1 => EN, in2 => N0, out1 => D0);
    g7: AND2 port map (in1 => EN, in2 => N1, out1 => D1);
    g8: AND2 port map (in1 => EN, in2 => N2, out1 => D2);
    g9: AND2 port map (in1 => EN, in2 => N3, out1 => D3);
end structural_1;
```

component instantiation

## II. Design of combinational circuits using VHDL

3) A single bit 4-to-1-line multiplexer:





## II. Design of combinational circuits using VHDL

---

Data flow description:

```
library ieee;
use ieee.std_logic_1164.all;
entity multiplexer_4_to_1_we is
    port (S : in std_logic_vector(1 downto 0);
          I : in std_logic_vector(3 downto 0);
          Y : out std_logic);
end multiplexer_4_to_1_we;

architecture function_table of multiplexer_4_to_1_we is
begin
    Y <=      I(0) when S = "00" else
              I(1) when S = "01" else
              I(2) when S = "10" else
              I(3) when S = "11" else
              'X';
end function_table;
```

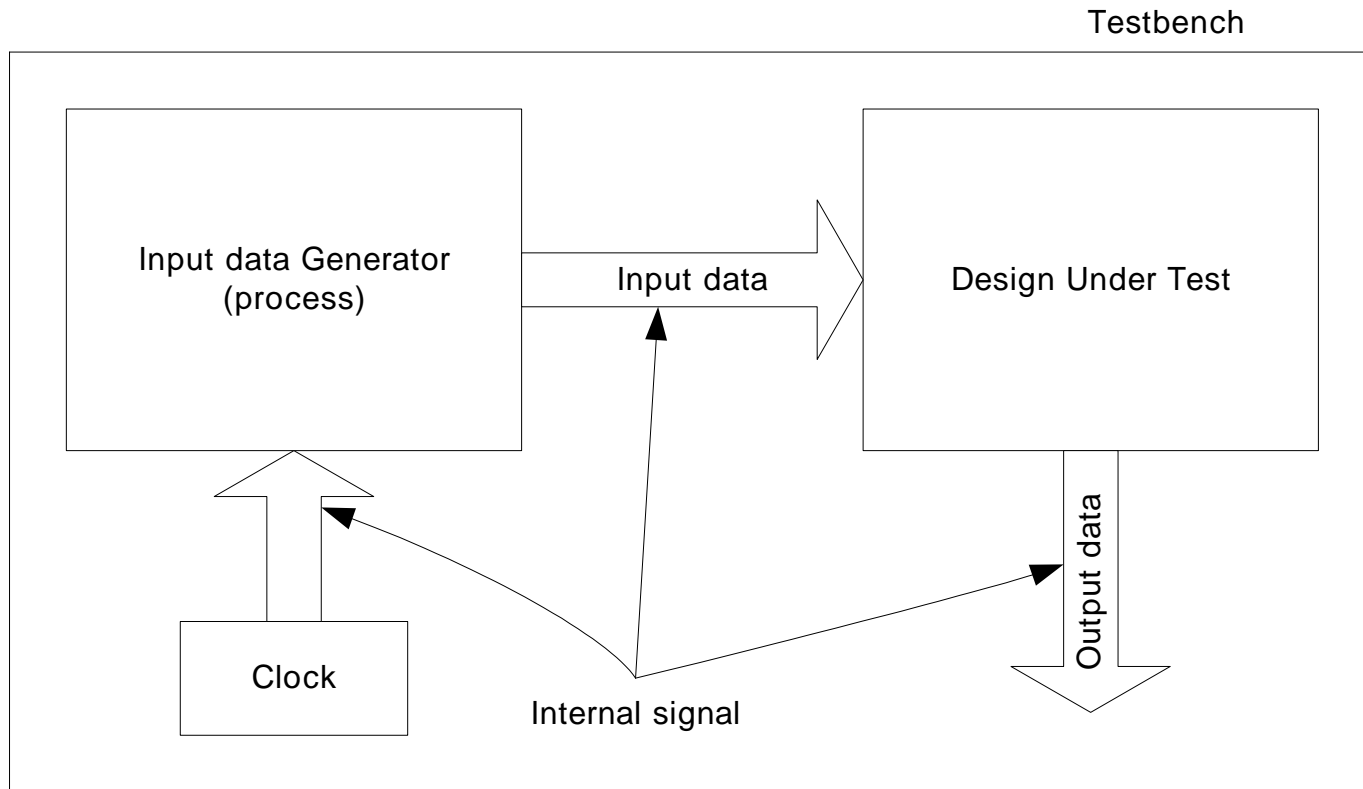
## II. Design of combinational circuits using VHDL

Structural  
description:

```
library ieee;
use ieee.std_logic_1164.all;
entity multiplexer_4_to_1_st is
    port(S: in std_logic_vector(0 to 1);
         I: in std_logic_vector(0 to 3);
         Y: out std_logic);
end multiplexer_4_to_1_st;

architecture structural_2 of multiplexer_4_to_1_st is
    component NOT1
        port(in1: in std_logic;
             out1: out std_logic);
    end component;
    component AND2
        port(in1, in2: in std_logic;
             out1: out std_logic);
    end component;
    component OR4
        port(in1, in2, in3, in4: in std_logic;
             out1: out std_logic);
    end component;
    signal S_n: std_logic_vector(0 to 1);
    signal D, N: std_logic_vector(0 to 3);
begin
    g0: NOT1 port map (S(0), S_n(0));
    g1: NOT1 port map (S(1), S_n(1));
    g2: AND2 port map (S_n(1), S_n(0), D(0));
    g3: AND2 port map (S_n(1), S(0), D(1));
    g4: AND2 port map (S(1), S_n(0), D(2));
    g5: AND2 port map (S(1), S(0), D(3));
    g6: AND2 port map (D(0), I(0), N(0));
    g7: AND2 port map (D(1), I(1), N(1));
    g8: AND2 port map (D(2), I(2), N(2));
    g9: AND2 port map (D(3), I(3), N(3));
    g10: OR4 port map (N(0), N(1), N(2), N(3), Y);
end structural_2;
```

### III. Testbench - abstract



### III. Testbench – example 1(1/2)

```
-- testbench of " 4 to 1 mux "
```

```
-- HEADER -----
```

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

```
-- ENTITY -----
```

```
entity mux4to1_test is  
end mux4to1_test;
```

*Port ƒ*

```
-- ARCHITECTURE -----
```

```
architecture mux4to1_test_arc of mux4to1_test is
```

```
component multiplexer_4_to_1_we  
  port (  
    S : in std_logic_vector(1 downto 0);  
    I : in std_logic_vector(3 downto 0);  
    Y : out std_logic);  
end component;
```

*Design under test*

### III. Testbench – example 1(2/2)

```
constant cycle      : time := 100 ns;
```

```
signal sel          : std_logic_vector (1 downto 0);  
signal int_input    : std_logic_vector(3 downto 0);  
signal y            : std_logic;  
signal clk          : std_logic:='0';
```

*Testbench*     *DUT*

```
begin
```

```
blk : multiplexer_4_to_1_we port map(  
  S  => sel,  
  I  => int_input,  
  Y  => y  
);
```

*Input Data Generate*

```
clk <= not (clk) after cycle /2;
```

*clock*

```
-- test input gen  
data_gen : process (clk)  
variable data : std_logic_vector (5 downto 0) := "000000";  
begin  
  if (clk = '1' and clk'event ) then  
    data := unsigned (data) + 1;  
    int_input <= data(3 downto 0);  
    sel <= data (5 downto 4);  
  end if;  
end process data_gen;
```

```
end mux4to1_test_arc;
```

### III. Testbench – example 2

```
-- testbench of " 4 to 1 mux "  
-- HEADER -----  
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.std_logic_arith.all;  
  
-- ENTITY -----  
entity mux4to1_test is  
end mux4to1_test;  
  
-- ARCHITECTURE -----  
architecture mux4to1_test_arc of mux4to1_test is  
  
component multiplexer_4_to_1_we  
  port (  
    S : in std_logic_vector(1 downto 0);  
    I : in std_logic_vector(3 downto 0);  
    Y : out std_logic);  
end component;  
  
signal int_s      : std_logic_vector (1 downto 0) := "00";  
signal int_i      : std_logic_vector (3 downto 0) := "0000";  
signal y          : std_logic;  
  
begin  
  blk : multiplexer_4_to_1_we port map(  
    S      => int_s,  
    I      => int_i,  
    Y      => y  
  );  
  
  -- test input gen  
  int_i <= unsigned(int_i) + 1 after 100 ns;  
  int_s <= unsigned(int_s) + 1 after 300 ns;  
  
end mux4to1_test_arc;
```

*Data generate*  
- No clk  
- only delay