Additional VHDL

ELEC 418

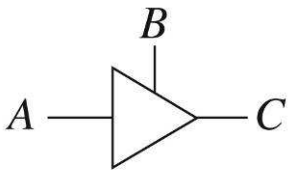
Advanced Digital Systems

Dr. Ron Hayne

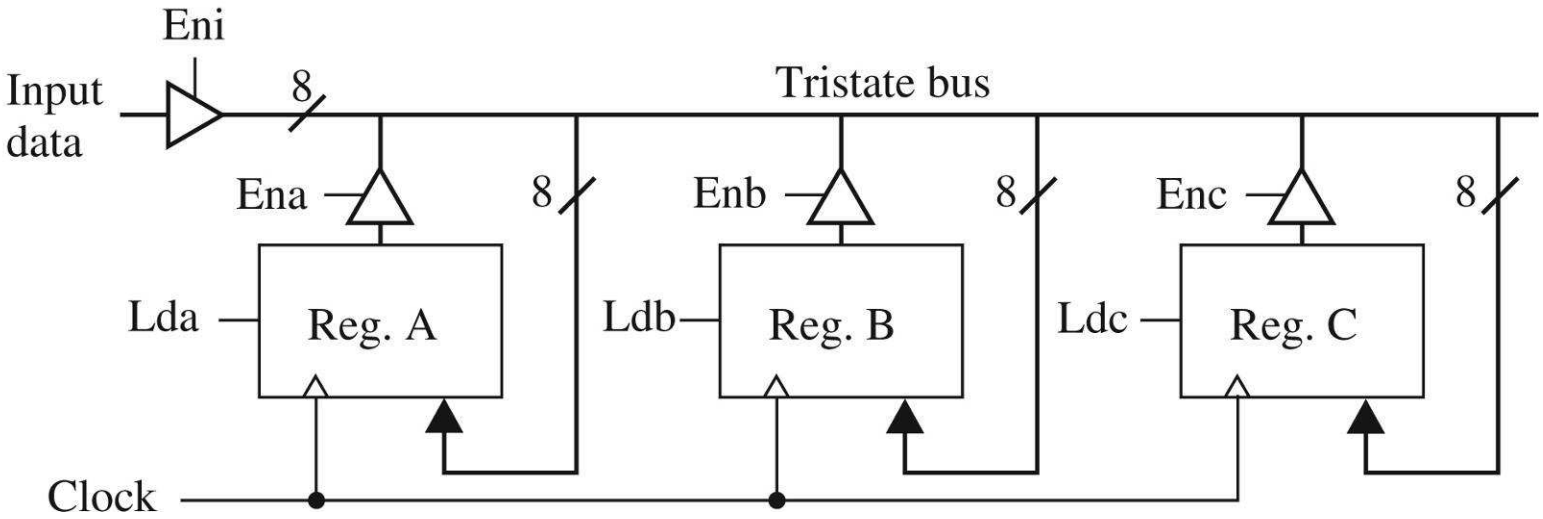
Images Courtesy of Thomson Engineering

THE CITADEL
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Tristate Logic and Busses



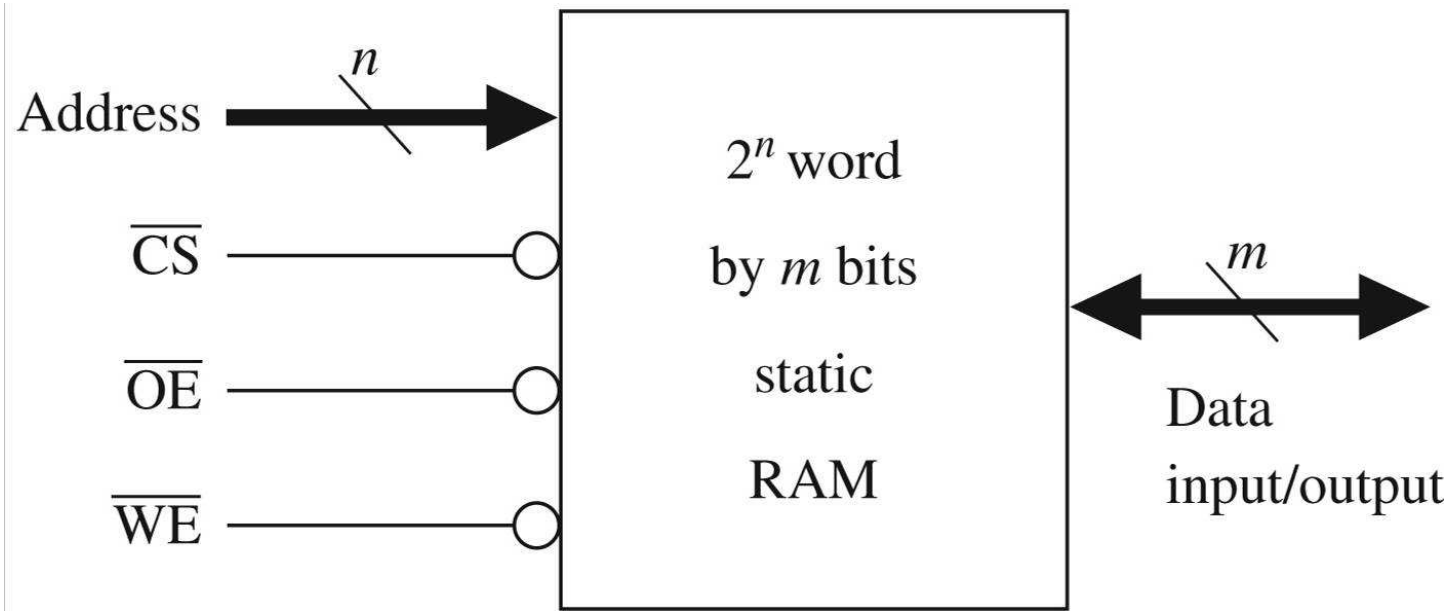
B	A	C
0	0	Hi-Z
0	1	Hi-Z
1	0	0
1	1	1



IEEE 1164 Standard Logic

- **9-Valued Logic System**
  - 'U' Uninitialized
  - 'X' Forcing Unknown
  - '0' Forcing 0
  - '1' Forcing 1
  - 'Z' High Impedance
  - 'W' Weak Unknown
  - 'L' Weak 0
  - 'H' Weak 1
  - '-' Don't Care

SRAM Model



$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	Mode	I/O pins
H	X	X	not selected	high-Z
L	H	H	output disabled	high-Z
L	L	H	read	data out
L	X	L	write	data in

VHDL Model

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

entity RAM6116 is
  port(Cs_b, We_b, Oe_b: in std_logic;
       Address: in std_logic_vector(7 downto 0);
       IO: inout std_logic_vector(7 downto 0));
end RAM6116;
```

VHDL Model

```
architecture simple_ram of RAM6116 is
type RAMtype is array(0 to 255) of
                  std_logic_vector(7 downto 0);
signal RAM1: RAMtype;
begin
  IO <= RAM1(conv_integer(Address)) when Cs_b = '0'
                                     and Oe_b = '0'
                                     and We_b = '1'
        else "ZZZZZZZZ";
```

VHDL Model

```
  process(We_b, Cs_b)
  begin
    if Cs_b = '0' and rising_edge(We_b) then
      RAM1(conv_integer(Address)) <= IO;
    end if;
  end process;
end simple_ram;
```

Generics

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

entity Businv is
  generic(width: positive);
  port(x: in std_logic_vector(width-1 downto 0);
       y: out std_logic_vector(width-1 downto 0));
end Businv;

architecture Behave of Businv is
begin
  y <= not x;
end Behave;
```

Generic Instantiation

```
entity Businv_8 is
  port(in8: in std_logic_vector(7 downto 0);
       out8: out std_logic_vector(7 downto 0));
end Businv_8;

architecture Structure of Businv_8 is
component Businv
  generic(width: positive);
  port(x: in std_logic_vector(width-1 downto 0);
       y: out std_logic_vector(width-1 downto 0));
end component;
begin
  B1: Businv generic map(8) port map(in8, out8);
end Structure;
```

Time Modeling

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

entity xor2 is
  generic(gate_delay: time := 2 ns);
  port(in1, in2: in std_logic;
       z: out std_logic);
end xor2;

architecture behave of xor2 is
begin
  z <= in1 xor in2 after gate_delay;
end behave;
```

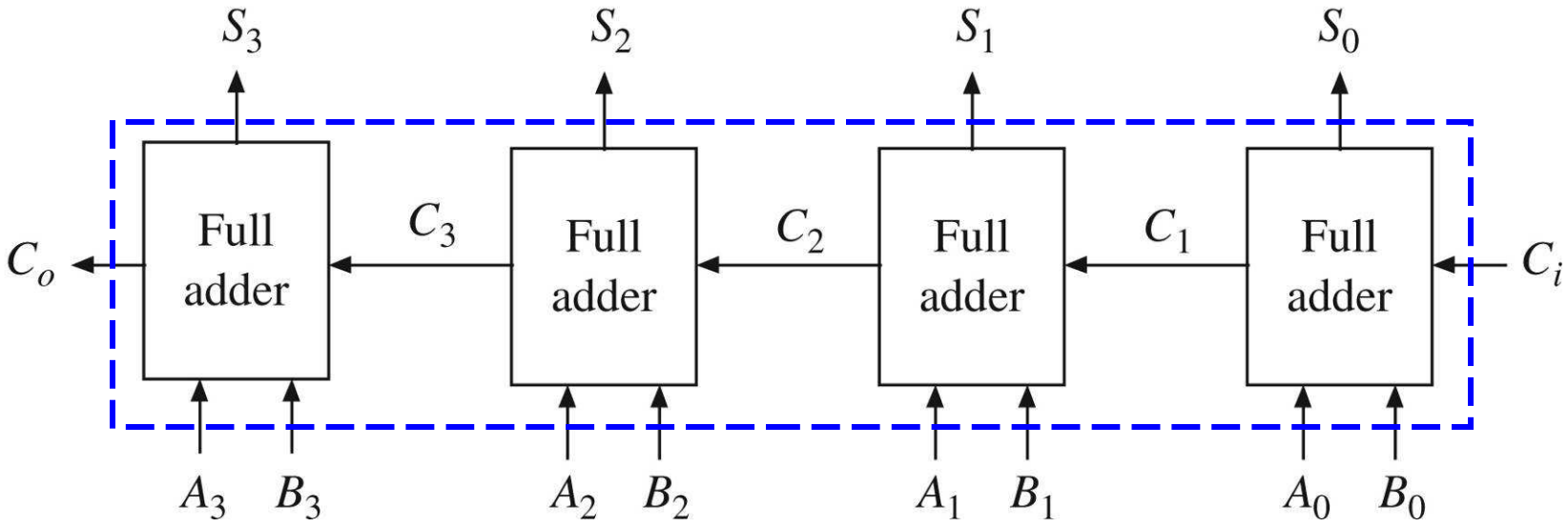
Named Association

```
component FA is
  port(X, Y, Cin: in std_logic;
       Cout, Sum: out std_logic);
end component;

FA0: FA port map(A(0), B(0), '0', open, S(0));

FA0: FA port map(Sum => S(0),
                 X   => A(0),
                 Y   => B(0),
                 Cin => '0');
```

Regular Structures (Generate)




```
entity Adder4 is
  port(A, B: in std_logic_vector(3 downto 0);
       Ci: in std_logic;
       S: out std_logic_vector(3 downto 0);
       Co: out std_logic);
end Adder4;
```

Generate Statements

```
architecture Structure of Adder4 is
  component FullAdder
    port(X, Y, Cin: in std_logic;
         Cout, Sum: out std_logic);
  end component;
  signal C: std_logic_vector(4 downto 0);
begin
  C(0) <= Ci;
  Co <= C(4);
  FullAdd4: for i in 0 to 3 generate
  begin
    FAx: FullAdder port map (A(i), B(i), C(i),
                             C(i+1), S(i));
  end generate FullAdd4;
end Structure;
```

Summary

- IEEE 1164 Standard logic
- Generics
- Named Association
- Generate Statements