LAB 2
INTRODUCTION TO VHDL

CMPS301: COMPUTER ARCHITECTURE COURSE

EXERTED FROM DR. ADNAN SHAOUT- *THE UNIVERSITY OF MICHIGAN-DEARBORN*

What we learned last time

- How to create Entity

- How to implement Architecture

- What different modeling could be used (Dataflow , behavioral, structural)

- Code is Concurrent not sequential

- "WHEN … ELSE" statement is concurrent only

- Signals are wires

- Different data types

- How to reuse previously created entity (component instantiation a.k.a port mapping)

Objectives

- Review Conditional Concurrent statements

- Others Keyword

- Learn about Generic entities

- Learn about For-Generate and if-Generate
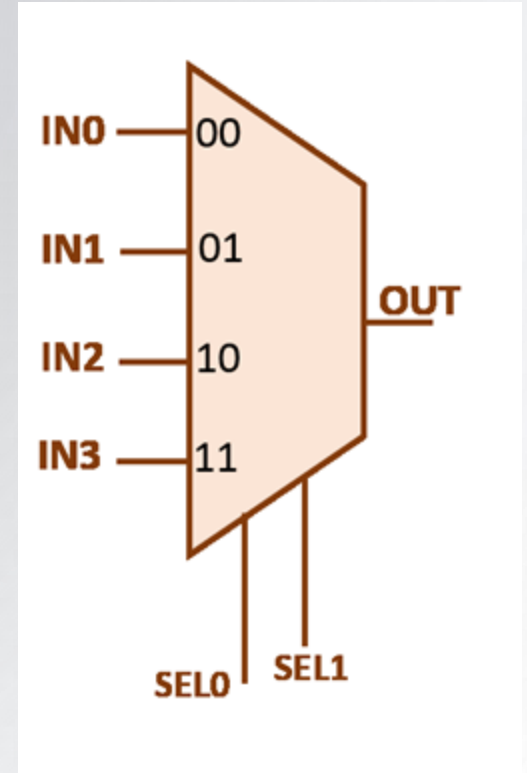
Remember Mux4x1

ENTITY mux **IS**

PORT(in0,in1,in2,in3: IN std_logic;

      sel : IN std_logic_vector (1 DOWNTO 0);

      out1: OUT std_logic);

END mux4;

Architecture course

When … ELSE / With … Sel

```vhdl
ARCHITECTURE a_mux OF mux
BEGIN

    out1 <=     in0  WHEN  sel = "00"
      ELSE in1 WHEN   sel = "01"
      ELSE in2 WHEN   sel = "10"
      ELSE in3;


END a_mux;
```
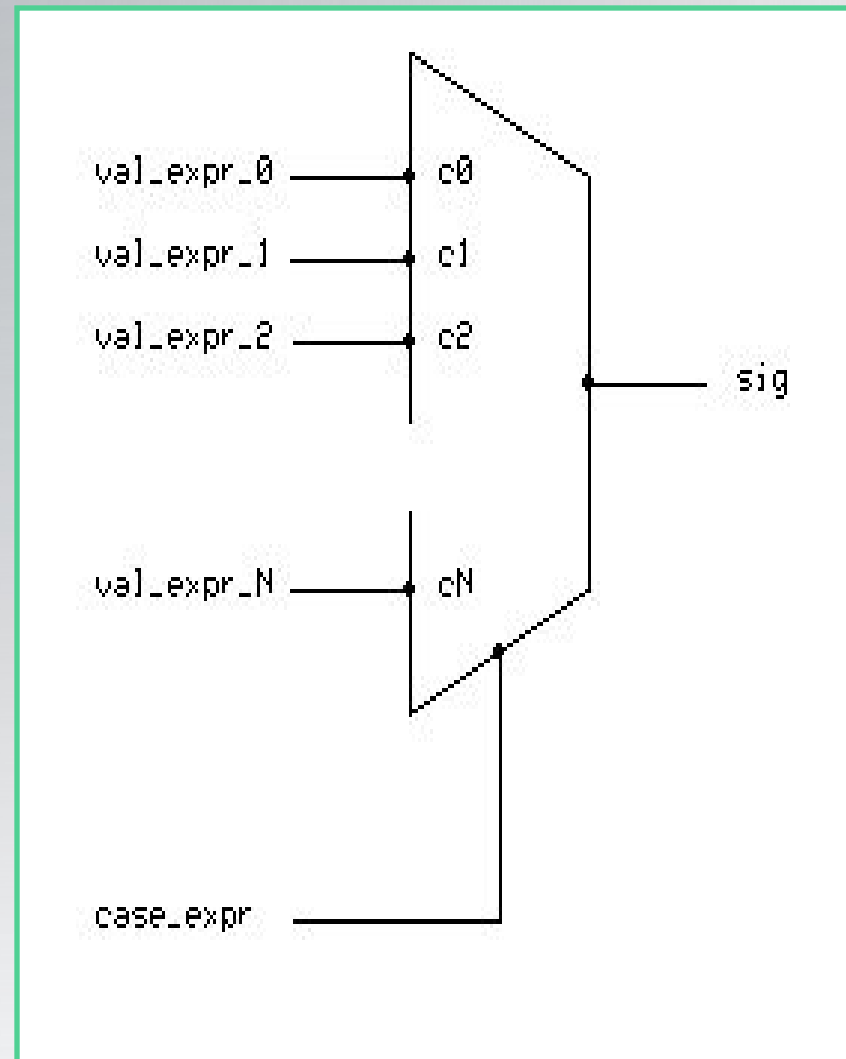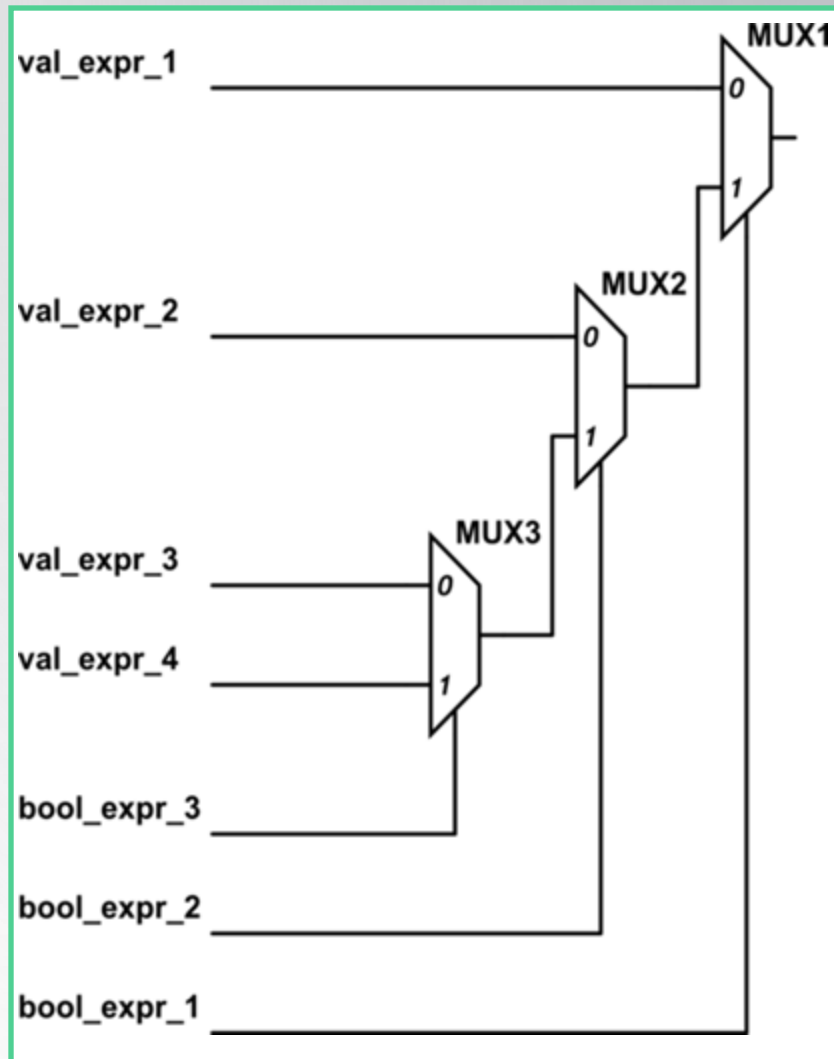
```vhdl
ARCHITECTURE b_mux OF mux
BEGIN
with sel select
out1 <=
        in0 when "00",
        in1 when "01",
        in2 when "10",
        in3 when "11";

END b_mux;
```

When … ELSE / With … Sel

Intro to VHDL

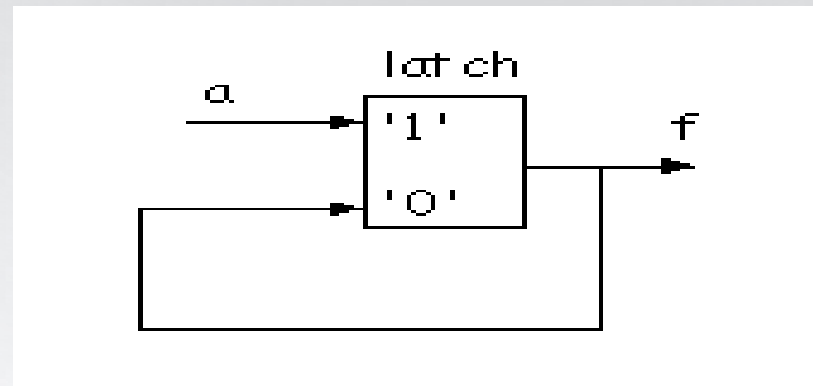# When … ELSE / With … Sel

What happens if we missed a case ?!

F <= a when Sel = "00";
-- what is the Value of F if Sel is not 00 ?!!!!

When … ELSE / With … Sel

What happens if we missed a case ?!

F <= a when Sel = "00";
-- what is the Value of F if Sel is not 00 ?!!!!

A latch will be formed to save previous value

Objectives

- Review Conditional Concurrent statements

- Others Keyword

- Learn about Generic entities

- Learn about For-Generate and if-Generate

"OTHERS" Keyword

Use "OTHERS" Whenever you want to:

Set a bus to several zeros or several 1s but you don't want to write it.

Example set F to 32 zero

F <= "00000000000000000000000000000000";
F <= x"00000000";   -- hexdecimal
F<= (Others => '0');

"OTHERS" Keyword

Use "OTHERS" Whenever you want to:

- Set a bus to several zeros or several 1s but you don't want to write it.

- Set a certain bit to value and others zero

Example

```
F <= "0000000000000000000000010000000";
F <= x"00000080";       -- hexdecimal
F<= (7=>'1', Others => '0');
```

"OTHERS" Keyword

Use "OTHERS" Whenever you want to:

- Set a bus to several zeros or several 1s but you don't want to write it.

- Set a certain bit to value and others zero

- Generic Case in Switch Cases

Example

With S **select**
F <= a **when** "0010",
    b **when** others;

Objectives

- Review Conditional Concurrent statements

- Others Keyword

- Learn about Generic entities

- Learn about For-Generate and if-Generate
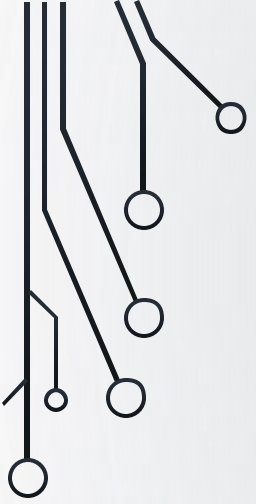
Let's build a full adder ☺
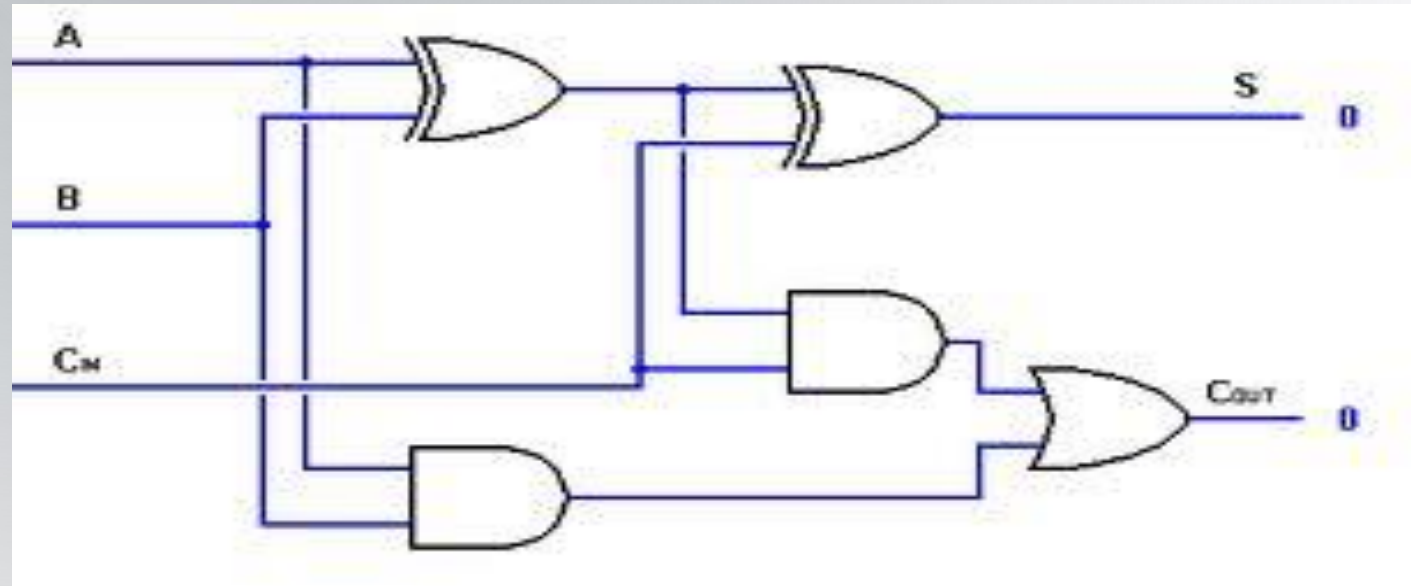
1bit-adder example

```vhdl
ENTITY my_adder IS     -- single bit adder
        PORT( a,b,cin : IN std_logic;
                s,cout : OUT std_logic);
END my_adder;

ARCHITECTURE a_my_adder OF my_adder IS

BEGIN
        s <= a XOR b XOR cin;
        cout <= (a AND b) or (cin AND (a XOR b));
END a_my_adder;
```

Now how to build 4-bits full adder from 1-bit full adder?
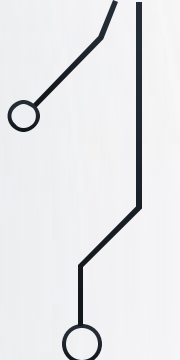
Carry-Propagate Adder

4 Bit-full Adder

```vhdl
ENTITY my_nadder IS

PORT   (a, b : IN std_logic_vector(3 DOWNTO 0) ;
         cin : IN std_logic;
         s : OUT std_logic_vector(3 DOWNTO 0);
         cout : OUT std_logic);


END my_nadder;
```

```vhdl
ARCHITECTURE a_my_nadder OF my_nadder IS

    COMPONENT my_adder IS

        PORT( a,b,cin : IN std_logic; s,cout : OUT std_logic);

     END COMPONENT;

    SIGNAL temp : std_logic_vector(4 DOWNTO 0);
BEGIN

    Temp(0) <= cin;

    f0: my_adder PORT MAP(a(0),b(0),temp(0),s(0),temp(1));

    f1: my_adder PORT MAP(a(1),b(1),temp(1),s(1),temp(2));

    f2: my_adder PORT MAP(a(2),b(2), temp(2),s(2),temp(3));

    f3: my_adder PORT MAP(a(3),b(3), temp(3),s(3),temp(4));

    Cout <= temp(4);

END a_my_nadder;
```

Architecture course

From Specific to Generic entities

```
ENTITY my_nadder IS

PORT   (a, b : IN std_logic_vector(3 DOWNTO 0) ;
         cin : IN std_logic;
         s : OUT std_logic_vector(3 DOWNTO 0);
         cout : OUT std_logic);


END my_nadder;
```

Now how can we change 8-bits
full adder to generic n-bit full adder?

From Specific to Generic entities

ENTITY my_nadder **IS**

PORT   (a, b : IN std_logic_vector(n-1 DOWNTO 0) ;
           cin : IN std_logic;
           s : OUT std_logic_vector(n-1 DOWNTO 0);
           cout : OUT std_logic);

END my_nadder;

Notice the use of 'n' here
But where did we define "n"

From Specific to Generic entities

Integer another data type

8 is default value and could be override while instantiation

Add generic input

ENTITY my_nadder **IS**
GENERIC (n : integer := 8);
PORT   (a, b : IN std_logic_vector(n-1 DOWNTO 0) ;
        cin : IN std_logic;
        s : OUT std_logic_vector(n-1 DOWNTO 0);
        cout : OUT std_logic);

END my_nadder;

Notice the use of 'n' here
But where did we define "n"

```vhdl
ARCHITECTURE a_my_nadder OF my_nadder IS

    COMPONENT my_adder IS

        PORT( a,b,cin : IN std_logic; s,cout : OUT std_logic);

     END COMPONENT;

    SIGNAL temp : std_logic_vector(4 DOWNTO 0);

BEGIN

temp(0) <= cin;

f0: my_adder PORT MAP(a(0),b(0),temp(0),s(0),temp(1));

f1: my_adder PORT MAP(a(1),b(1),temp(1),s(1),temp(2));

f2: my_adder PORT MAP(a(2),b(2), temp(2),s(2),temp(3));

f3: my_adder PORT MAP(a(3),b(3), temp(3),s(3),temp(4));

Cout <= temp(4);

END a_my_nadder;
```

Architecture course

```vhdl
ARCHITECTURE a_my_nadder OF my_nadder IS

    COMPONENT my_adder IS

        PORT( a,b,cin : IN std_logic; s,cout : OUT std_logic);

     END COMPONENT;

    SIGNAL temp : std_logic_vector(n DOWNTO 0);

BEGIN

temp(0) <= cin;

f0: my_adder PORT MAP(a(0),b(0),temp(0),s(0),temp(1));

f1: my_adder PORT MAP(a(1),b(1),temp(1),s(1),temp(2));

f2: my_adder PORT MAP(a(2),b(2), temp(2),s(2),temp(3));

f3: my_adder PORT MAP(a(3),b(3), temp(3),s(3),temp(4));

Cout <= temp(4);

END a_my_nadder;
```

Change size of signal

Architecture course

```vhdl
ARCHITECTURE a_my_nadder OF my_nadder IS

    COMPONENT my_adder IS

        PORT( a,b,cin : IN std_logic; s,cout : OUT std_logic);

     END COMPONENT;

    SIGNAL temp : std_logic_vector(n DOWNTO 0);

BEGIN

temp(0) <= cin;

loop1: FOR i IN 0 TO n-1 GENERATE

    fx: my_adder PORT MAP(a(i),b(i),temp(i),s(i),temp(i+1));

END GENERATE;

Cout <= temp(n);

END a_my_nadder;
```

Change size of signal

i →is not visible outside the for generate

Architecture course

For….Generate

We use it when we want to generate more than one component of the same type

i →is not visible outside the for generate

loop1: **FOR** i **IN** 0 **TO** n-1 **GENERATE**

   fx: my_adder **PORT MAP**(a(i),b(i),temp(i-1),s(i),temp(i));

END GENERATE;

The (for …generate) is not a Software Sequential loop .. It is like writing this line n times and they execute **concurrently. Generates N hardware**

Generic ....cont.

So after I made my entity generic when I instantiate I do the following

u0: my_nadder GENERIC MAP (16) PORT MAP (......)

I indicate here the value of the generic data

Generic could be more than one parameter like generic (n,m : integer;)
Even could be with different type

Generic ….cont.

- In simulation

- vsim my_nadder –g<genericName>=<value>

- Example

vsim my_nadder –gn=8

Common Errors

During Coding:

- Mix the use of bit & std_logic
- Using port map inside a condition
- Size of Input Vector not Equal Size of Output Vector

During Simulation:

- Simulating without compiling last changes
- Not forcing all input ports at initialization
- Force values on output port/signal

Summary

- Review on (When .. Else / With…Sel)

- Use of "OTHERS" keyword

- Make Generic Entities & use them

- Use "for … Generate"

- Check for "if..Generate"