ECE316 Assignment 2
Full 4-bit Adder

Assignment 2. Full 4-bit Adder

Goal of this assignment:

- Be familiar with concurrent port map mechanism in VHDL
- Know how to use component defined in another file

Concurrent Statements	Sequential Statement
Assignment (arithmetic and logic)	Assignment (arithmetic and logic)
When … else	If … else
With ... Select	Case
Port map	While….Loop
	For … Loop

How to Use VHDL Describe Hardware Dataflow

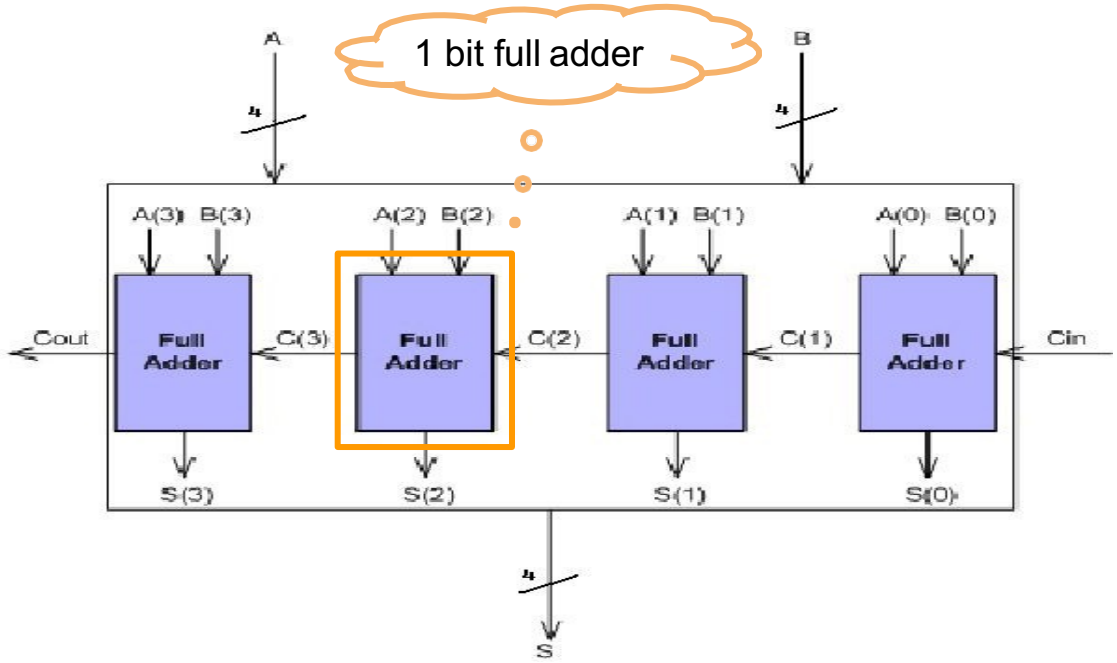
- Dataflow
  - The data flow describes how data moves through the circuit.
  - VHDL describes data flow using **concurrent signal assignment (port map)**
  - Each **concurrent statement** is triggered by changes on its inputs and delivers its outputs
  - There is no correspondence between the order of assignments in the source code.(pay attention to conflict)

Unsigned 4-bit Adder Function Specification

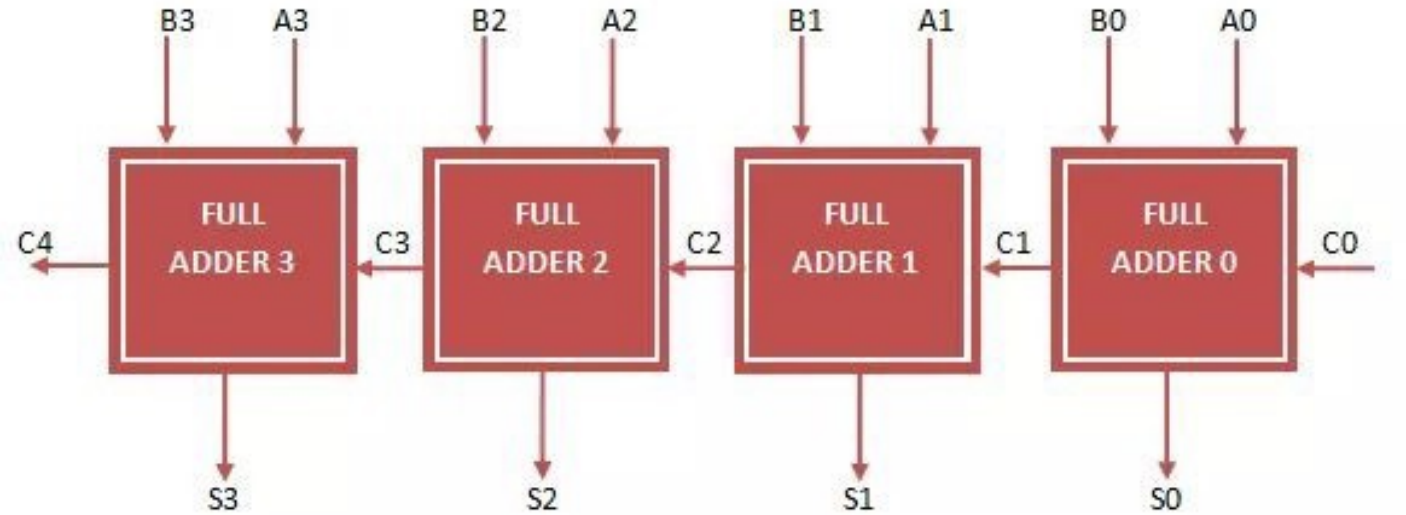
- Unsigned Full 4-bit Adder (4-bit Adder) adds two 4-bit binary numbers plus Carry-on input and gives 4-bit SUM and a Carry-on output.
- The Main operation of this 4-bit adder is to ripple each carry output to carry input of next single bit addition.
- Each single bit addition is performed with full 1-bit Adder operation (A, B, Cin) input and (Sum, Cout) output.
- The 4-bit Adder VHDL Code can be Easily Constructed by Port Mapping 4 1-bit Full Adder.

Concurrent Statements: Port map Example:

- 4-bit full adder



Full 4-bit adder



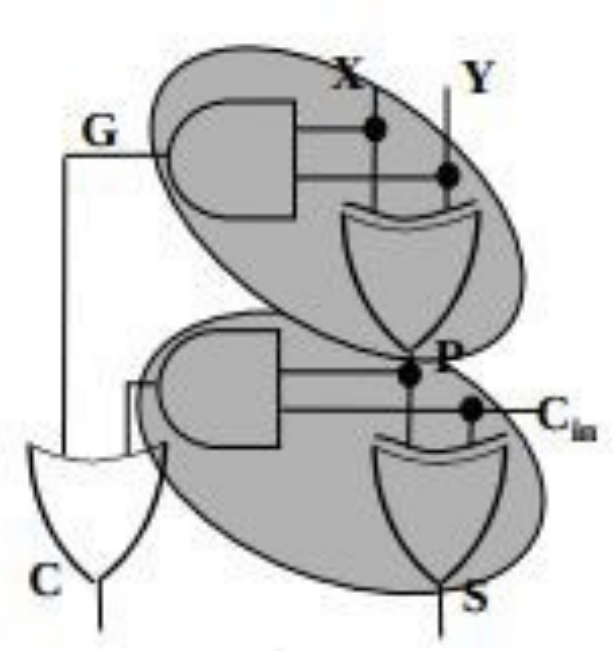
VHDL Operators

Class						
1. Logical operators	and	or	nand	nor	xor	xnor
2. Relational operators	=	/=	<	<=	>	>=
3. Shift operators	sll	srl	sla	sra	rol	ror
4.Addition operators	+	-	&			
5. Unary operators	+	-				
6. Multiplying op.	*	/	mod	rem		
7. Miscellaneous op.	**	abs	not			

Functional block implementation of 1 bit full adder

$S = X \oplus Y \oplus C_{in}$

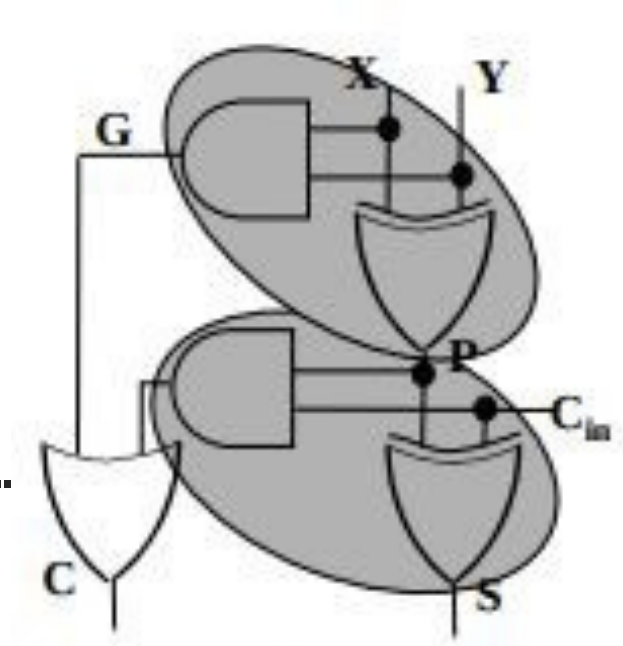
$C = XY + (X \oplus Y)C_{in}$



1-bit Full Adder

```
Entity FullAdder is
      port (X, Y, Cin : in bit;
                  Cout, Sum : out bit);
End Full Adder;

Architecture behav of FullAdder is
begin
----------Please fill in this part----------
End behav;
```



Concurrent Statements: Port map

port map (*signal1, signal2,...signaln*);

instance_name : *component name*
port map (*port1*=>*signal1*, *port2*=> *signal2*,... *port3*=>*signaln*);

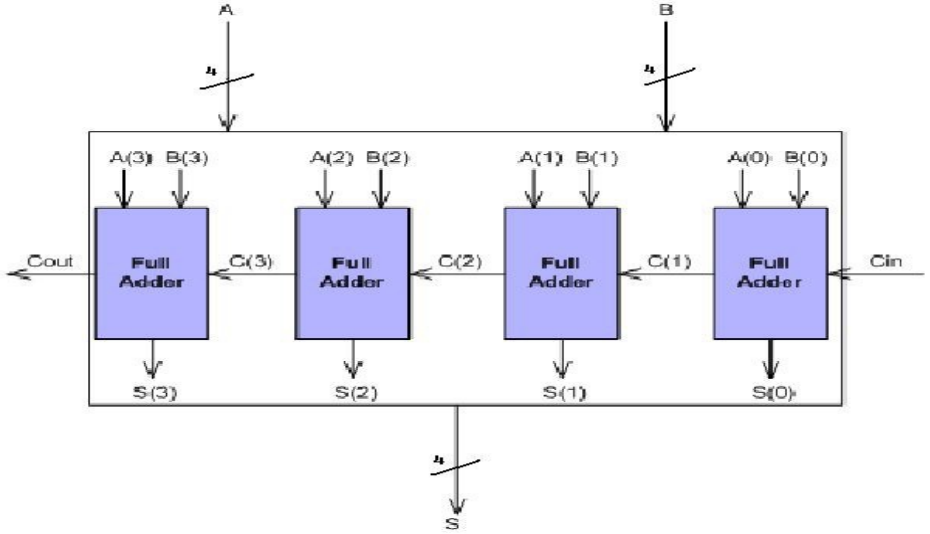
- Describe sub-components of the circuit using VHDL program
- Describe interconnections between sub-components of the circuit using Port Map statement.

Interface Specification

Name	Bit Length	Mode	Description
A	4	Input	A operand
B	4	Input	B operand
Cin	1	Input	Carry-on Input
Sum	4	Output	Sum of A and B
Cout	1	Output	Carry-on Output

The 4-bit Ripple-Carry Full Adder (Unsigned)

```
Entity Adder4 is
        port    (A, B : in bit_vector (3 downto 0);
                        Cin : in bit;
                        S : out bit_vector (3 downto 0
                        Cout : out bit);
End Adder4;
Architecture struct of Adder4 is
Component FullAdder
        port (X, Y, Cin : in bit;
                Cout, Sum : out bit);
End component;
Signal C : bit_vector(3 downto 1);
Begin
        FA0 : FullAdder port map (A(0), B(0), Cin, C(1), S(0));
       ……..Fill In……..
End struct;
```



Simulation Example:

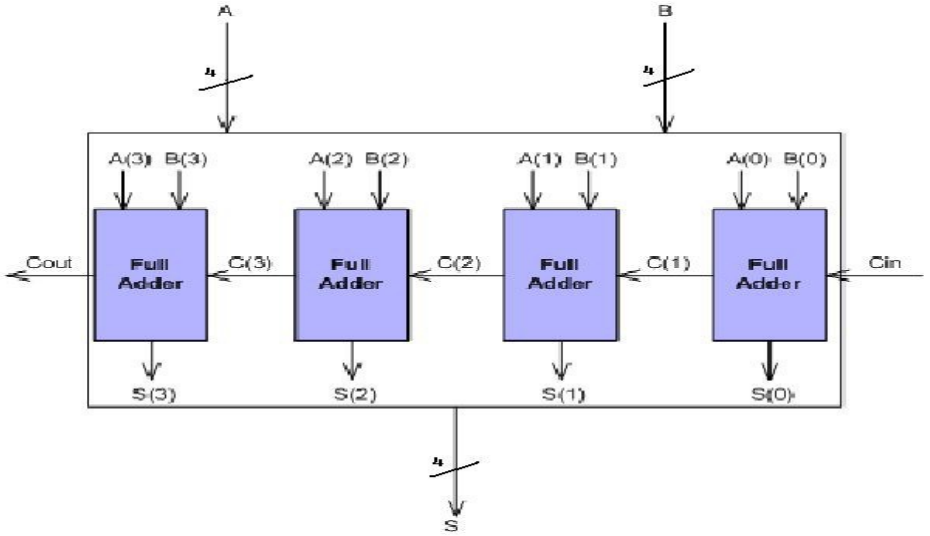
A = 1 1 1 1 (Unsigned)
B = 0 0 0 1 (Unsigned)
Cin = 0

Sum = 0 0 0 1
Cout = 1

Note: If A and B are both signed number, they do not have carry bit (Cout/Cin). For signed addition, it only has overflow bit. Also, for signed number, we always use the Most Significant Bit (MSB) to represent the sign.

Ripple-Carry Performance

Performance:
If each adder takes 5ns to calculate its output:
=> 5ns : C(1) and S(0)
=> 10ns : C(2) and S(1)
=> 15ns : C(3) and S(2)
=> 20ns : Cout and S(3)



Can show in simulation by adding a transport delay in the code:

A <= transport B after 5ns;

Carry Look Ahead Adder

To combat this, we can calculate the all the carries at the same time:

$$S_i = A_i \oplus B_i \oplus C_i$$
$$C_{i+1} = A_i B_i + (A_i + B_i) C_i$$

the carry logic can be separated into:
Generation:

$$G_i = A_i B_i$$

Propagation:

$$P_i = A_i + B_i$$

So:

$$C_{i+1} = G_i + P_i C_i$$

So we can then calculate the carry values for a 4-bit adder as

$$C_1 = G_0 + P_0 C_{in}$$
$$C_2 = G_1 + P_1 C_1$$
$$C_3 = G_2 + P_2 C_2$$
$$C_4 = G_3 + P_3 C_3$$

Carry Look Ahead Adder

By substituting the values of C:

$$C_1 = G_0 + P_0 C_{in}$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{in}$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in}$$

Where:

$$G_i = A_i B_i$$
$$P_i = A_i + B_i$$

Partial 1-bit Full Adder with G and P

Name	Bit Length	Mode	Description
A	1	Input	A bit
B	1	Input	B bit
Cin	1	Input	Carry-in bit
Sum	1	Output	Sum of A and B
P	1	Output	A or B
G	1	Output	A and B

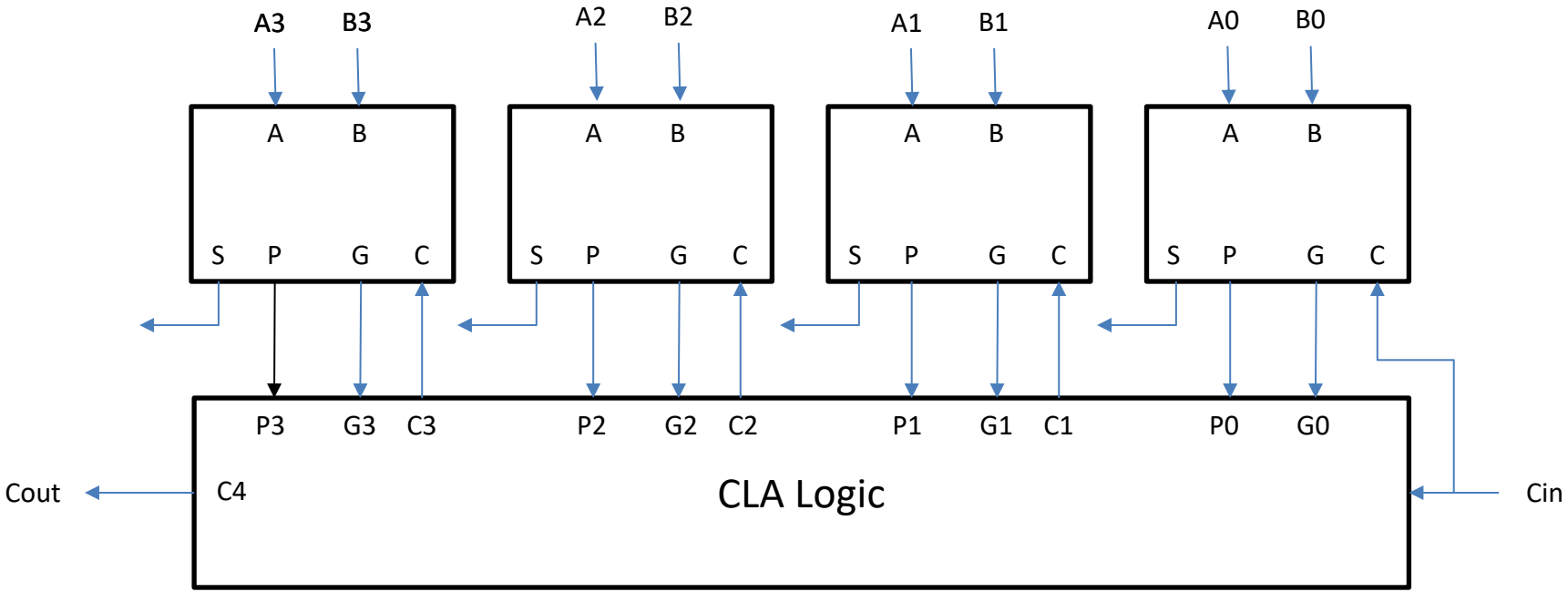
Carry Look Ahead Logic

Name	Bit Length	Mode	Description
P	**4**	**Input**	**Propagaters**
G	**4**	**Input**	**Generators**
Cin	**1**	**Input**	**Carry-in bit**
Cs	**4**	**Output**	**Carry values**

Interface Specification

Name	Bit Length	Mode	Description
A	4	Input	A operand
B	4	Input	B operand
Cin	1	Input	Carry-on Input
Sum	4	Output	Sum of A and B
Cout	1	Output	Carry-on Output

4-bit CLA Adder - Structure



Assignment 2 Due Date

ModelSim Simulation Result + Lab Report
Deadline: Next class day (Blackboard submission)