Cs 343
Spring 2022
Adder Lab 3A

TASKS TO DO:

1. Design in VHDL Half adder using two processes
2. Design 1-bit Full-adder using Half adder as a component
3. Design N=4 bit adder using 1bit Full adder as a component
4. Design N=4 bit a add/sub component that performs addition when the operations code=0, and subtraction when the operations code=1
5. Create a package where you put all components for future use(Your last name is part of package name)
6. Design N – bit add-sub unit **using behavioral VHDL model**
7. You have to design a circuit to output OVERFLOW , ZERO, NEGATIVE flags for N-bit add/sub.
8. Verify all your designs in simulation using waveforms in ModelSim for N=4, and N=32 bits using Most positive, Most negative integer as a first operand, and integers 1 and/or 2 as a second operand. You have to demonstrate that flags are set correctly in appropriate cases.
9. Create N-bit adder/subtractor unit using lpm and compare waveforms with your design
10. Create a Test-Bench file in vhdl to test Add_SUB unit for n=16 bits. Please demonstrate that the test-bench detects an error (intentionally created) in your design and prints out simulation time, expected operand 1 and operand result value, actual result value, and values of operand 1 and operand 2 that caused the error.

Complete REPORT, VIDEO IS REQUIRED FOR THIS EXERCISE.

Cs 343
Spring 2022
Adder Lab 3A

What to Submit:

1. VHDL code for tasks 1,2,3,4,5 printout
2. VHDL code for tasks 6,7 printout
3. Waveforms for task 8 in Model-Sim for N=4, and N=32

In waveforms You must use the following operands

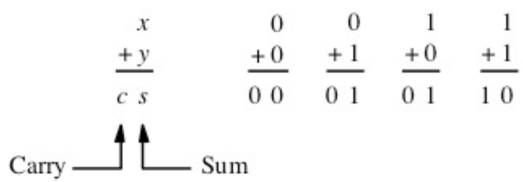
a. Most Positive N bit integer + 1
b. Most Positive N bit integer - 1
c. Most Negative N bit integer + 1
d. Most Negative N bit integer – 1
e. Most Positive N bit integer- Most Negative N bit integer
f. Most Positive N bit integer+ Most Negative N bit integer
g. Most Positive N bit integer- Most Positive N bit integer

The output waveform signal values have to be in HEX, all flag values have to be shown in each case. For each case a-g you have to give a one sentence explanation.

4. VHDL code for task 9. and waveforms for operand cases 3a, 3b, ….,3g.
5. Task 10: Printout of two simulations: 1. that demonstrates error was detected and printouts all parameters described in Task 10, and 2. after the code was corrected the printout is no errors.

6. WHEN TO SUBMIT? March 6, 11:00PM. Thank you for your efforts.

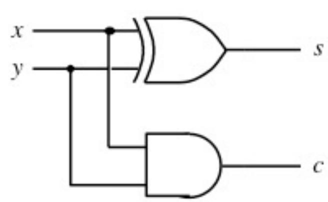
HALF-Adder



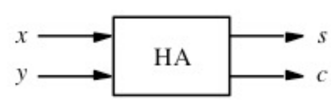
(a) The four possible cases

x	y	Carry c	Sum s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(b) Truth table

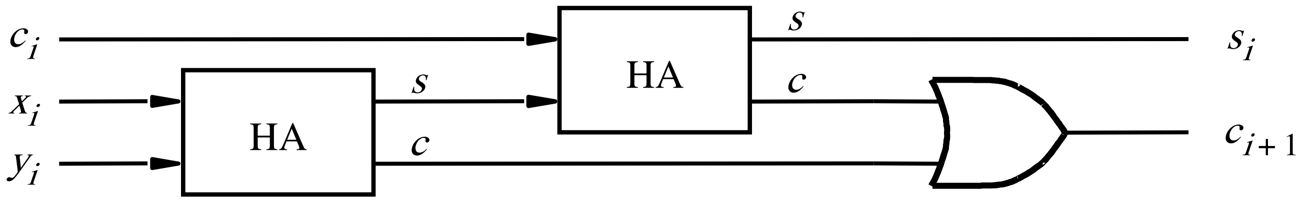


(c) Circuit

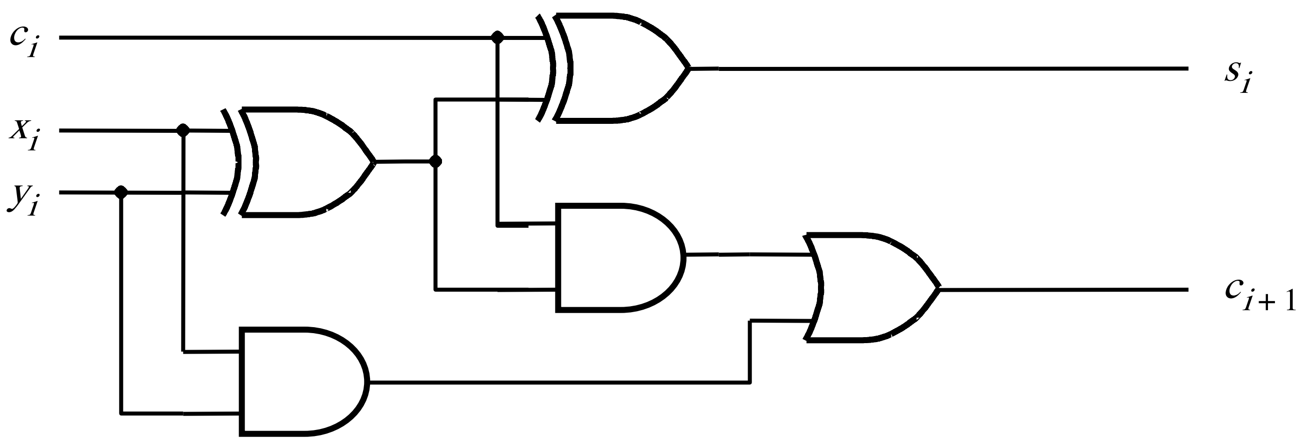


(d) Graphical symbol

Full Adder Circuit

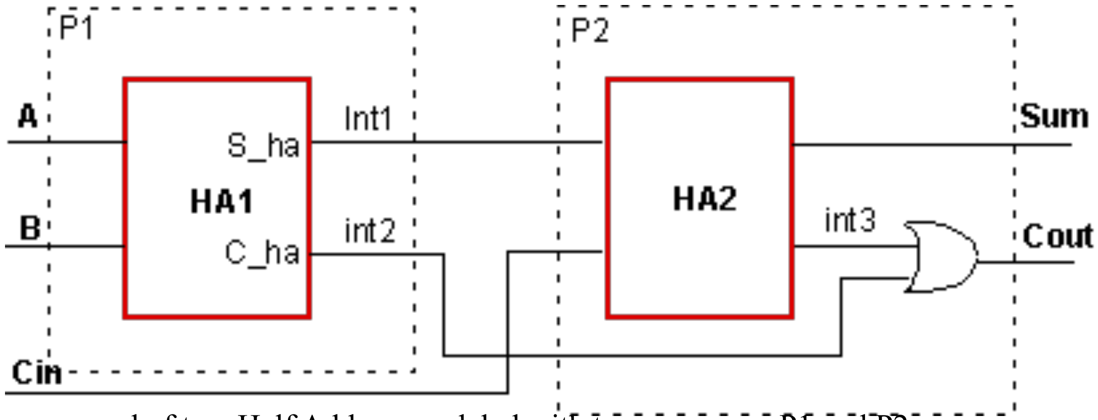


(a) Block diagram



(b) Detailed diagram

Behavioral Modeling of an adder : Sequential Statements

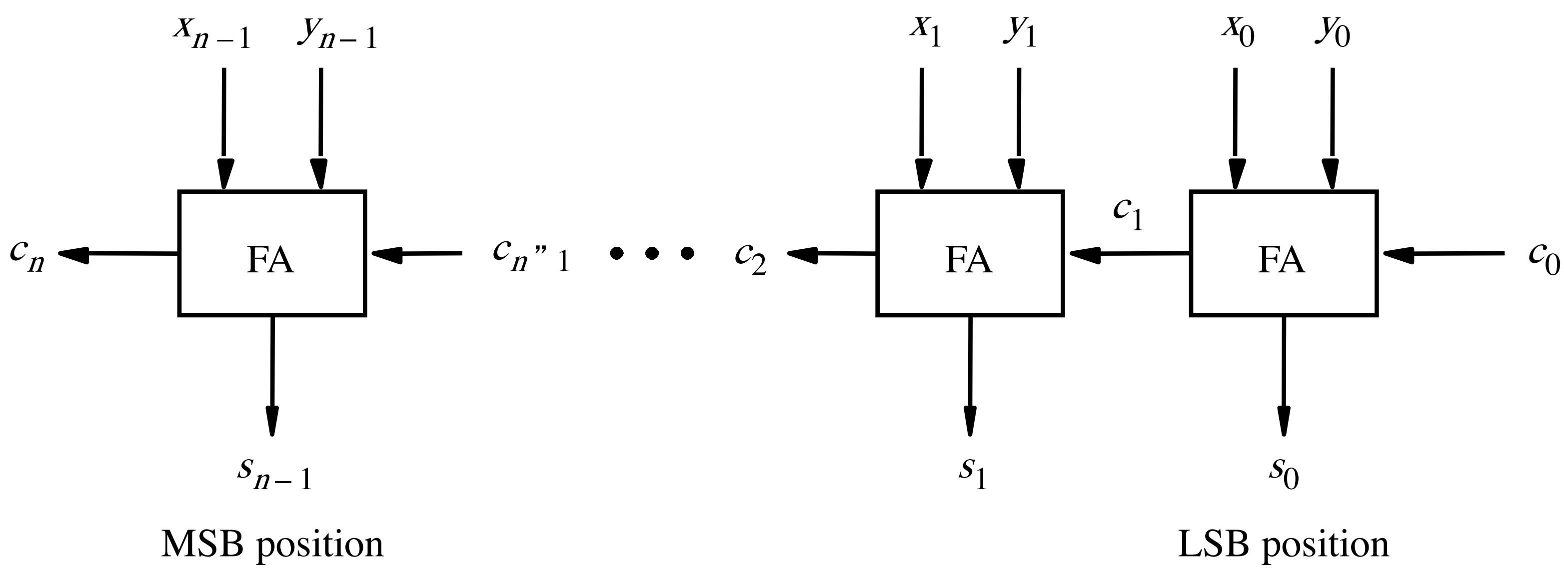


Full Adder composed of two Half Adders, modeled with two processes P1 and P2.

```
library ieee;
use ieee.std_logic_1164.all;
entity FULL_ADDER is
     port (A, B, Cin : in std_logic;
           Sum, Cout : out std_logic);
end FULL_ADDER;

architecture BEHAV_FA of FULL_ADDER is
signal int1, int2, int3: std_logic;
begin
-- Process P1 that defines the first half adder
P1: process (A, B)
     begin
           int1<= A xor B;
           int2<= A and B;
     end process;
-- Process P2 that defines the second half adder and the OR -- gate
P2: process (int1, int2, Cin)
     begin
          Sum <= int1 xor Cin;
           int3 <= int1 and Cin;
           Cout <= int2 or int3;
     end process;
end BEHAV_FA;
```

Of course, one could simplify the behavioral model significantly by using a single process.

An *4*-bit ripple-carry adder



Denote by Δt time 1-Bit Full adder computes addition of two bits and carry.

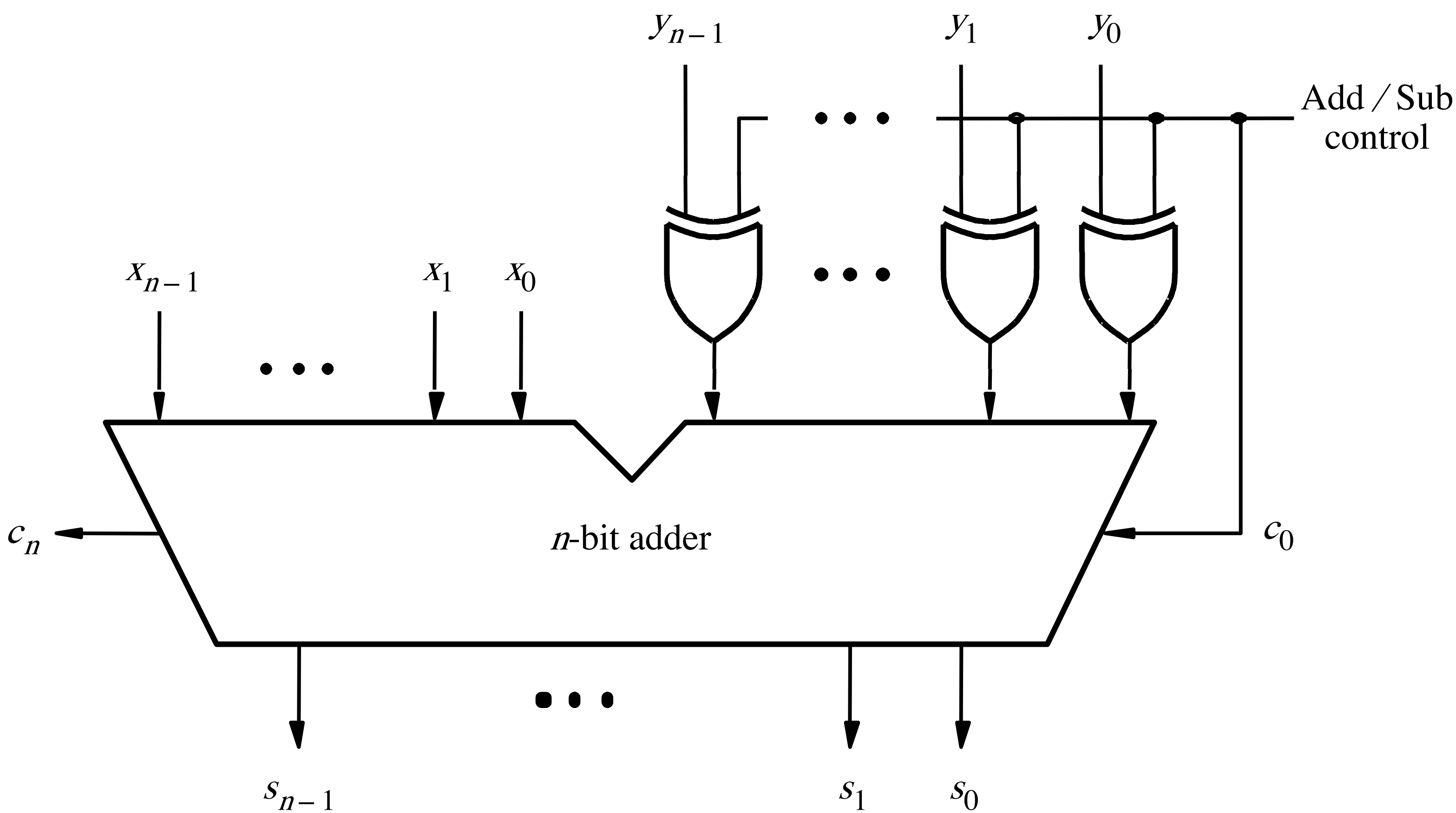
Question: How much time it takes to compute the sum of two 4 bits words ?

SUBTRACTOR FROM ADDER

BORROW 10-1 =1

1101	-3	1101
-	-	+
1111	-1	0001
---------	---------	------
1110	-2	1110

Adder/Subtractor Unit

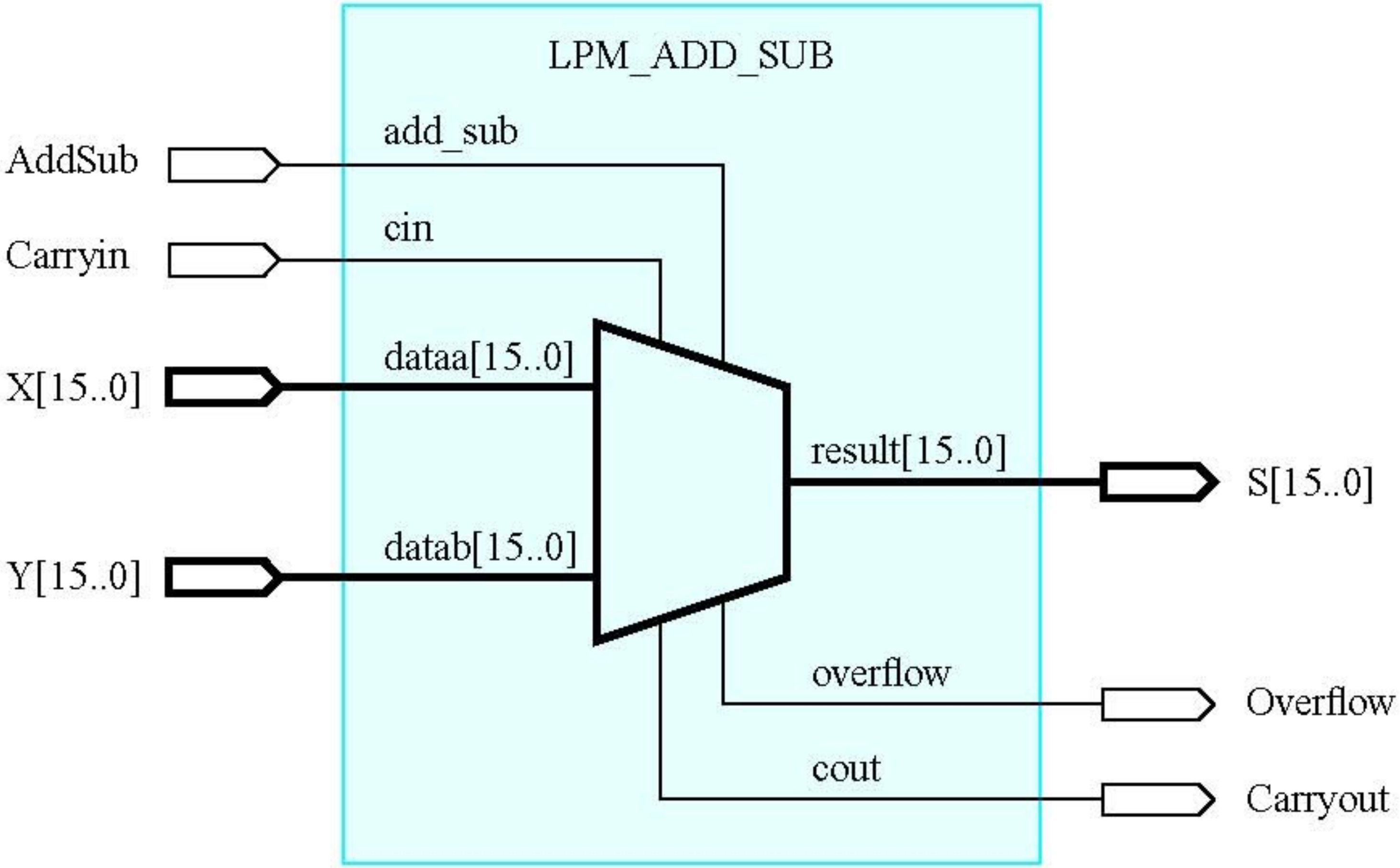


Add /Sub control is =1 then the output of the XOR gate is inverted Y (1's complement of Y).
C_0=1 in the case of subtraction we need to add 1 to form 2's complement of Y.

Recall: XOR performs module 2 operation!!

Design N-Bit adder/SUB using VHDL

Schematic using an LPM adder/subtractor



Examples of Adder Implementation in VHDL

VHDL code for the 1-Bit full-adder

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY fulladd IS
	PORT ( Cin, x, y : IN STD_LOGIC ;
			s, Cout : OUT STD_LOGIC ) ;
END fulladd ;

ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
	s <= x XOR y XOR Cin ;
	Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y) ;
END LogicFunc ;
```

Structural Model

VHDL code for a four-bit adder

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY adder4 IS
      PORT ( Cin : IN STD LOGIC ;
                 x3, x2, x1, x0 : IN STD_LOGIC ;
                 y3, y2, y1, y0 : IN STD_LOGIC ;
                 s3, s2, s1, s0 : OUT STD_OGIC ;
                 Cout           : OUT STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
      SIGNAL c1, c2, c3 : STD_LOGIC ;
      COMPONENT fulladd
            PORT ( Cin, x, y : IN STD_LOGIC ;
                       s, Cout   : OUT STD_LOGIC ) ;
      END COMPONENT ;
BEGIN
      stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1 ) ;
      stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
      stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
      stage3: fulladd PORT MAP (
            Cin => c3, Cout => Cout, x => x3, y => y3, s =>
s3 ) ;
END Structure ;
```

Declaration of a package.

(alternative style of code)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

PACKAGE fulladd_package IS
    COMPONENT fulladd
        PORT ( Cin, x, y    : IN STD_LOGIC ;
                  s, Cout  : OUT STD_LOGIC ) ;
    END COMPONENT ;
END fulladd_package ;
```

. A different way of specifying a four-bit adder.

Your Last Name has to be part of package name!

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE work.fulladd_package.all ;

ENTITY adder4 IS
	PORT (		Cin		: IN	STD_LOGIC ;
		x3, x2, x1, x0 : IN	STD_LOGIC ;
		y3, y2, y1, y0 : IN	STD_LOGIC ;
		s3, s2, s1, s0  : OUT	STD_LOGIC ;
		Cout	: OUT	STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
	SIGNAL c1, c2, c3 : STD_LOGIC ;
BEGIN
	stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1 ) ;
	stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
	stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
	stage3: fulladd PORT MAP (
		Cin => c3, Cout => Cout, x => x3, y => y3, s => s3  ) ;
END Structure ;
```

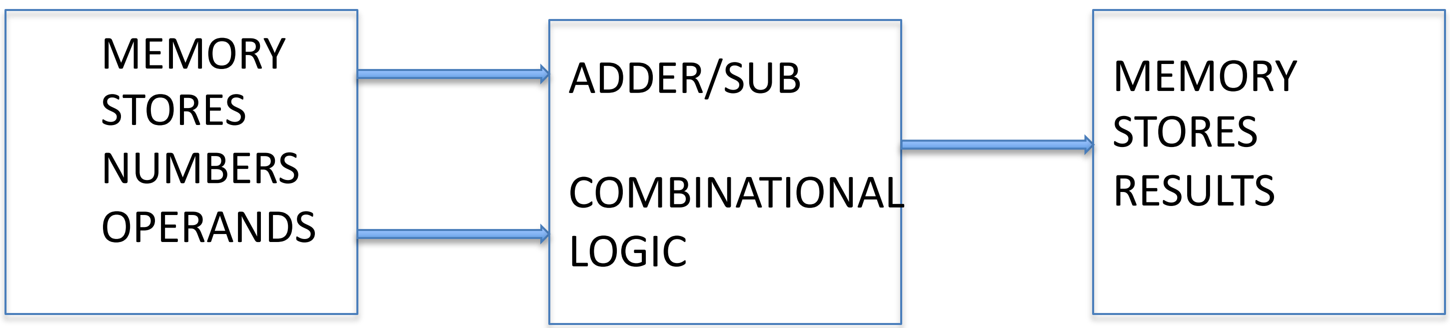
A four-bit adder defined using multibit signals

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE work.fulladd_package.all ;

ENTITY adder4 IS
	PORT (	Cin		: IN	STD_LOGIC ;
		X, Y : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
		S		: OUT	STD_LOGIC_VECTOR(3 DOWNTO 0) ;
		Cout	: OUT  STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
	SIGNAL C : STD_LOGIC_VECTOR(1 TO 3) ;
BEGIN
	stage0: fulladd PORT MAP ( Cin, X(0), Y(0), S(0), C(1) ) ;
	stage1: fulladd PORT MAP ( C(1), X(1), Y(1), S(1), C(2) ) ;
	stage2: fulladd PORT MAP ( C(2), X(2), Y(2), S(2), C(3) ) ;
	stage3: fulladd PORT MAP ( C(3), X(3), Y(3), S(3), Cout ) ;
END Structure ;
```

ADDER/ SUB USAGE

In the future lab

NOT IN THIS ONE



READ FROM MEMORY

WRITE TO MEMORY