Palestine Technical University-Kadoorie

High Level Digital Design

1214-0420

Term Project

Three 4-bit numbers adder

Prepared by:

Mohammad Saleh AbuSafat

Supervisor:

Dr. Mahmoud Moshref

Project submitted in VHDL Course Requirement for the Bachelor Degree

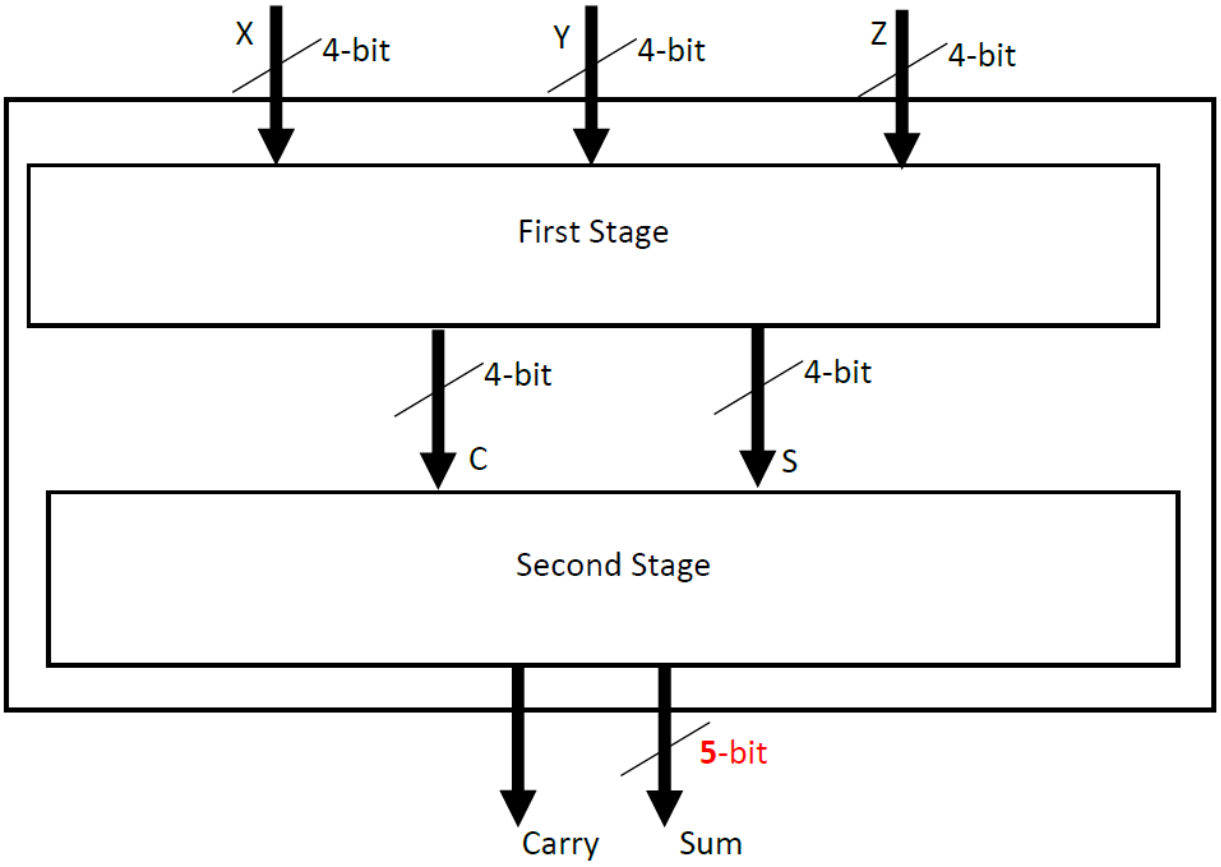
2021/2022

Objective: Design a three 4-bit numbers adder.

Details of the design: Ripple Carry Adder (RCA) performs addition of two n-bit numbers. Therefore, the adder shown in the figure below performs the addition in two stages as follows:

1- In the first stage, we need to reduce the problem of adding three numbers into adding two n-bit numbers. The 8-bits of S and C will be computed at the same time.

2- In the second stage, RCA is used to compute the summation and the carry.



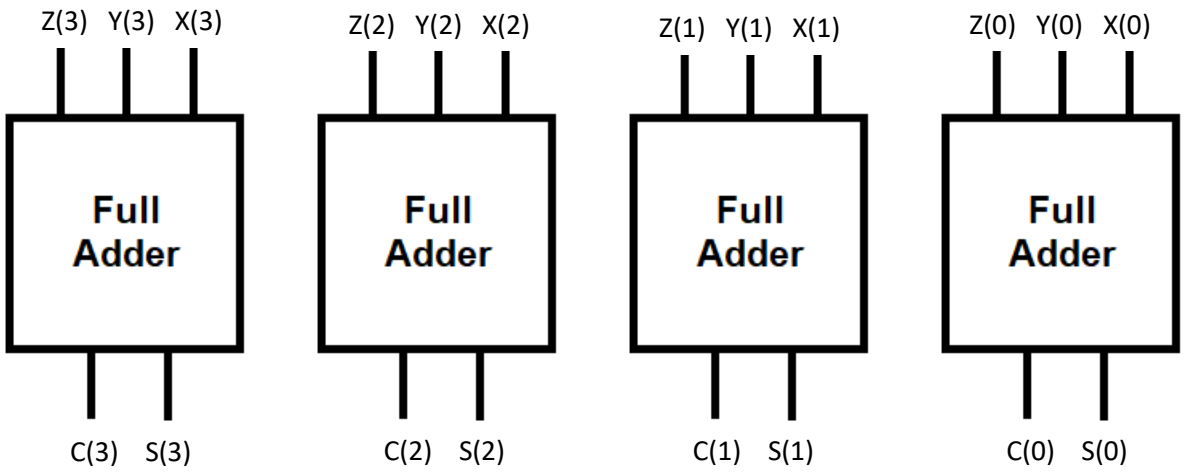
- In the first stage, I will determine C and S vectors, so that each bit in the S vector represents the sum of one bit of each input vectors in the same order, and each bit in the C vector represents the carry of that sum.

This example shows the process:
If we consider that X=”1010”, Y=”1110” and Z=”0001”
We find that S= ”0101” and C= ”1010”.

X	1	0	1	0
Y	1	1	1	0
Z	0	0	0	1
S	0	1	0	1
C	1	0	1	0

For first bit at the right 0+0+1 this will equal 01 (s=1, c=0)

This stage can be done by Full Adder for each bit.

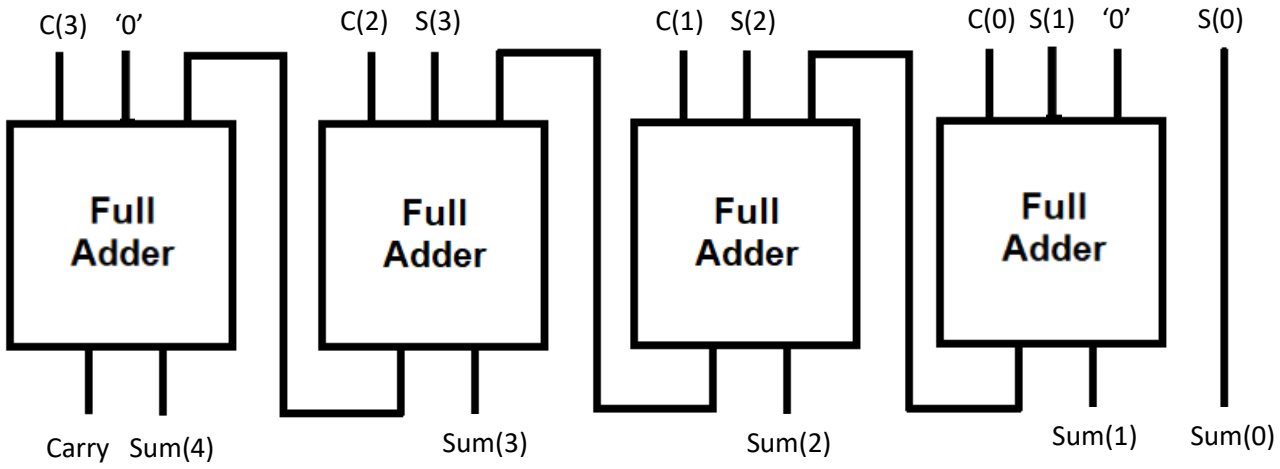


- In the second stage I will sum S vector with C vector but before do this I will add a bit with value zero at the right of C vector to achieve a correct result for sum, to clarify the process, we did the following:

  We determined the S and C vectors from the first stage, suppose we get this result:
  S= “0110” and C= “1011” then
  The calculation will be this

```
    0 1 1 0 +
  1 0 1 1 0
-----------------------------
0 1 1 1 0 0
carry  Sum
```

  This stage can be done by RCA.



- To implement this application on VHDL we will first create entity for Half and full Adder

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity HA is
5   port(a,b : in std_logic;
6         sum,cout : out std_logic);
7   end entity;
8
9   architecture HA_arch of HA is
10  begin
11  sum<= a xor b;
12  cout<= a and b;
13  end architecture;
14
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity FA is
5   port(a, b, cin : in std_logic;
6         sum, cout: out std_logic);
7   end entity;
8
9   architecture FA_arch of FA is
10
11  component HA is
12  port(a,b: in std_logic;
13       sum,cout: out std_logic);
14  end component;
15
16  signal sum1, cout1, cout2 :std_logic;
17
18  begin
19
20  HF1 : HA port map(a,b,sum1,cout1);
21  HF2 : HA port map(sum1,cin,sum,cout2);
22
23  cout<= cout1 or cout2;
24
25  end architecture;
```

And then we create our application:

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

entity stage1 is
port(x, y, z : in std_logic_vector(3 downto 0);
       s, c : out std_logic_vector(3 downto 0));
end entity;

architecture stage1_arch of stage1 is

component FA is
port(a, b, cin : in std_logic;
      sum, cout: out std_logic);
end component;

begin

FA1 : FA port map(x(0),y(0),z(0),s(0),c(0));
FA2 : FA port map(x(1),y(1),z(1),s(1),c(1));
FA3 : FA port map(x(2),y(2),z(2),s(2),c(2));
FA4 : FA port map(x(3),y(3),z(3),s(3),c(3));

end architecture;
```

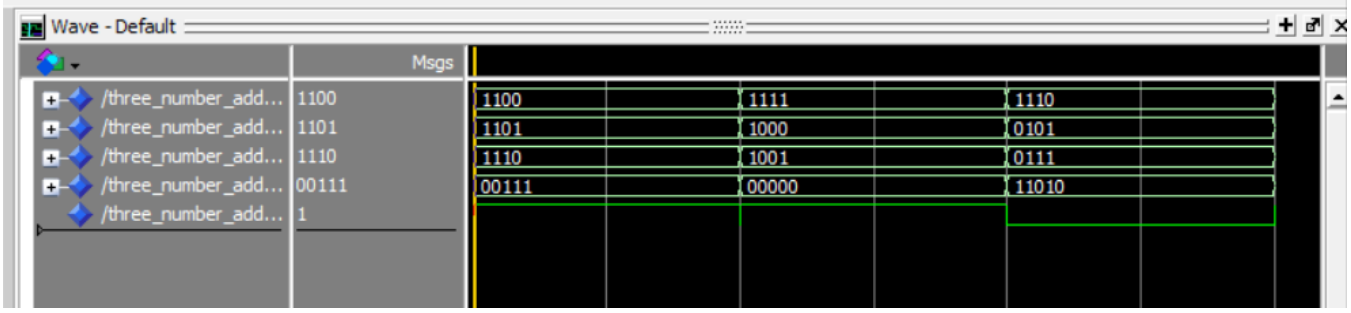
```
library ieee;
use ieee.std_logic_1164.all;

entity stage2 is
port(s, c: in std_logic_vector(3 downto 0);
      sum: out std_logic_vector(4 downto 0);
    carry: out std_logic);
end entity;

architecture stage2_arch of stage2 is

component FA is
port(a, b, cin : in std_logic;
      sum, cout: out std_logic);
end component;

signal cout1, cout2, cout3: std_logic;

begin

sum(0) <= s(0);
FA1 : FA port map(s(1),c(0) ,'0'  ,sum(1),cout1);
FA2 : FA port map(s(2),c(1) ,cout1,sum(2),cout2);
FA3 : FA port map(s(3),c(2) ,cout2,sum(3),cout3);
FA4 : FA port map(c(3),cout3,'0'  ,sum(4),carry);

end architecture;
```

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity Three_Number_Adder is
5  port(x, y, z : in std_logic_vector(3 downto 0);
6            sum : out std_logic_vector(4 downto 0);
7          carry : out std_logic);
8  end entity;
9
10 architecture Three_Number_Adder_arch of Three_Number_Adder is
11
12 component stage1 is
13 port(x, y, z : in std_logic_vector(3 downto 0);
14           s, c : out std_logic_vector(3 downto 0));
15 end component;
16
17 component stage2 is
18 port(s, c: in std_logic_vector(3 downto 0);
19         sum: out std_logic_vector(4 downto 0);
20       carry: out std_logic);
21 end component;
22
23 signal s, c: std_logic_vector(3 downto 0);
24
25
26 begin
27
28 stage1_o: stage1 port map (x, y, z, s, c);
29 stage2_o: stage2 port map (s, c, sum, carry);
30
31 end architecture;
```

Test Bench: For simplicity, the Test bench can be conducted using three different tuples only (e.g., T1= (X= 1100, Y= 1101, Z=1110), T2= (X= 1111, Y= 1000, Z= 1001), T3= (X= 1110, Y= 0101, Z= 0111)).

- To test my solution:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity Three_Number_Adder_TB is
5  end entity;
6
7  architecture Three_Number_Adder_TB_arch of Three_Number_Adder_TB is
8
9  component Three_Number_Adder is
10 port(x, y, z : in std_logic_vector(3 downto 0);
11              sum : out std_logic_vector(4 downto 0);
12            carry : out std_logic);
13 end component;
14
15 signal x_TB, y_TB, z_TB: std_logic_vector(3 downto 0);
16 signal sum_TB: std_logic_vector(4 downto 0);
17 signal carry_TB: std_logic;
18
19 begin
20
21 Three_Number_Adder_1: Three_Number_Adder port map(x_TB, y_TB, z_TB, sum_TB, carry_TB);
22
23 STIMULUS : process
24 begin
25
26 x_TB <= "1100" ; y_TB <= "1101" ; z_TB <= "1110" ; wait for 20 ns;
27 x_TB <= "1111" ; y_TB <= "1000" ; z_TB <= "1001" ; wait for 20 ns;
28 x_TB <= "1110" ; y_TB <= "0101" ; z_TB <= "0111" ; wait for 20 ns;
29
30 end process;
31 end architecture;
```



done