logic design 2

Lab1

Name & IDs :

كريم فتحي عبد العزيز محمد مصطفى (20011116)

احمد محمد حسن محمد صالح (20010174)

عمر محمود عبد الوهاب السيد(20011027)

محمد احمد علي رياض علي (20011457)

فارس محمد انور رزق (20011065)

محمد عمرو عبد الفتاح (20011675)

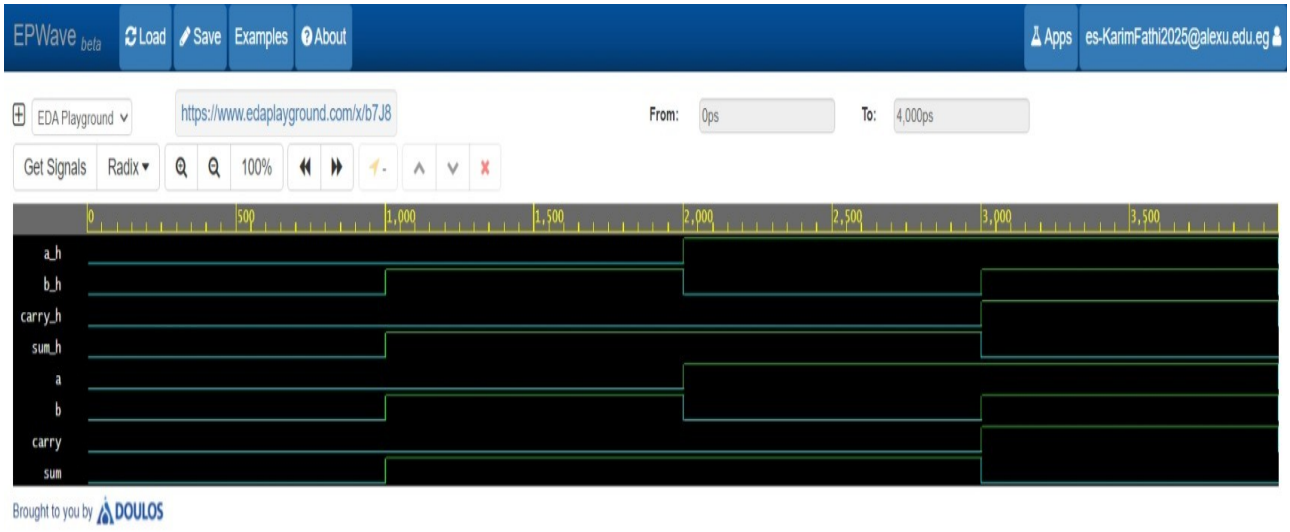
1. problem statement:

It is required to use VHDL to design and test a 4 bit ripple adder, which adds 2 4-bit inputs introducing sum and carry signals. You should implement the following modules:

1. Half adder.
2. Full adder using the half adder.
3. 4-bit ripple adder/subtractor using the full adder.

2.simulation samples:

1. half adder:

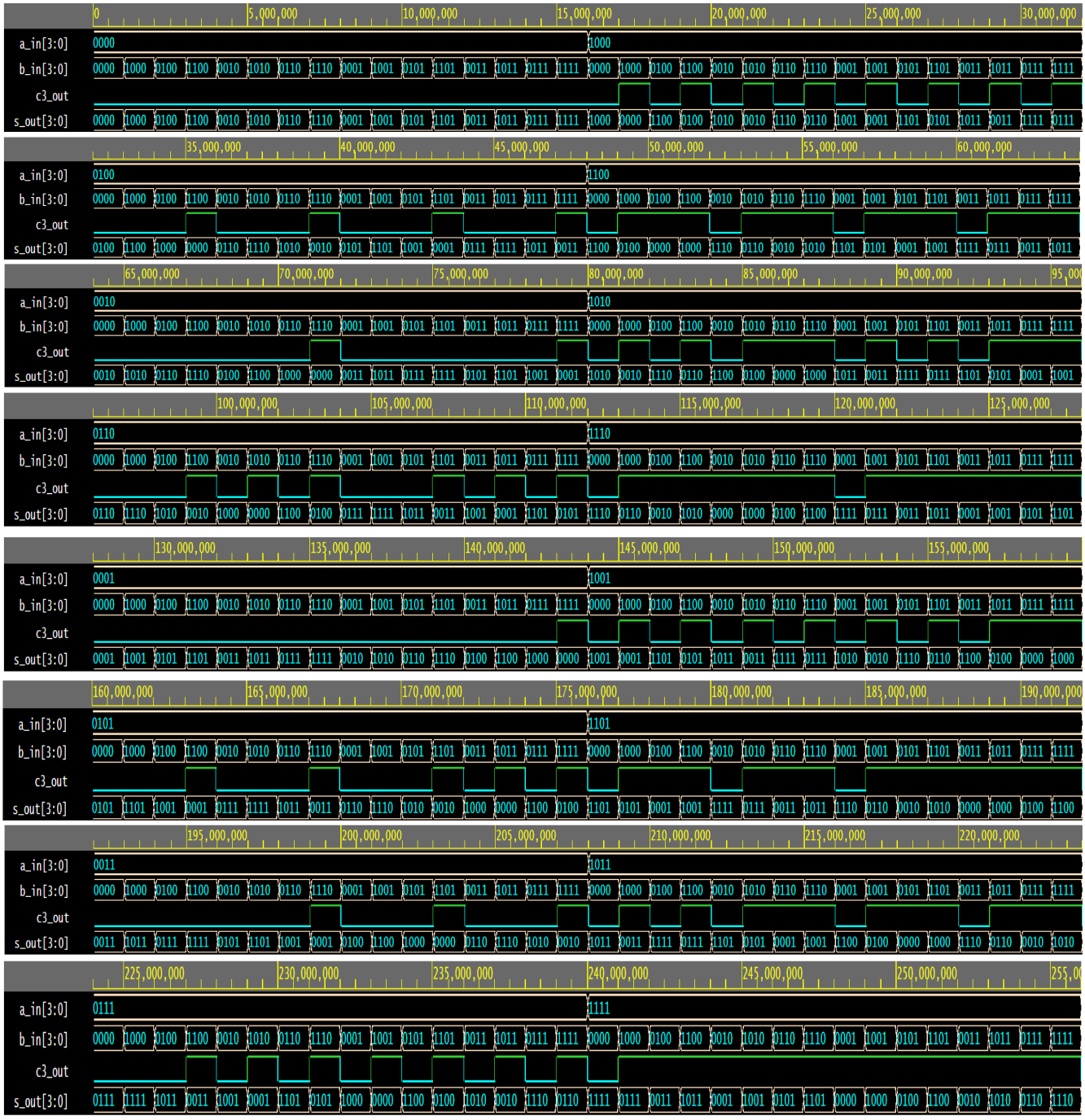


2. full adder:

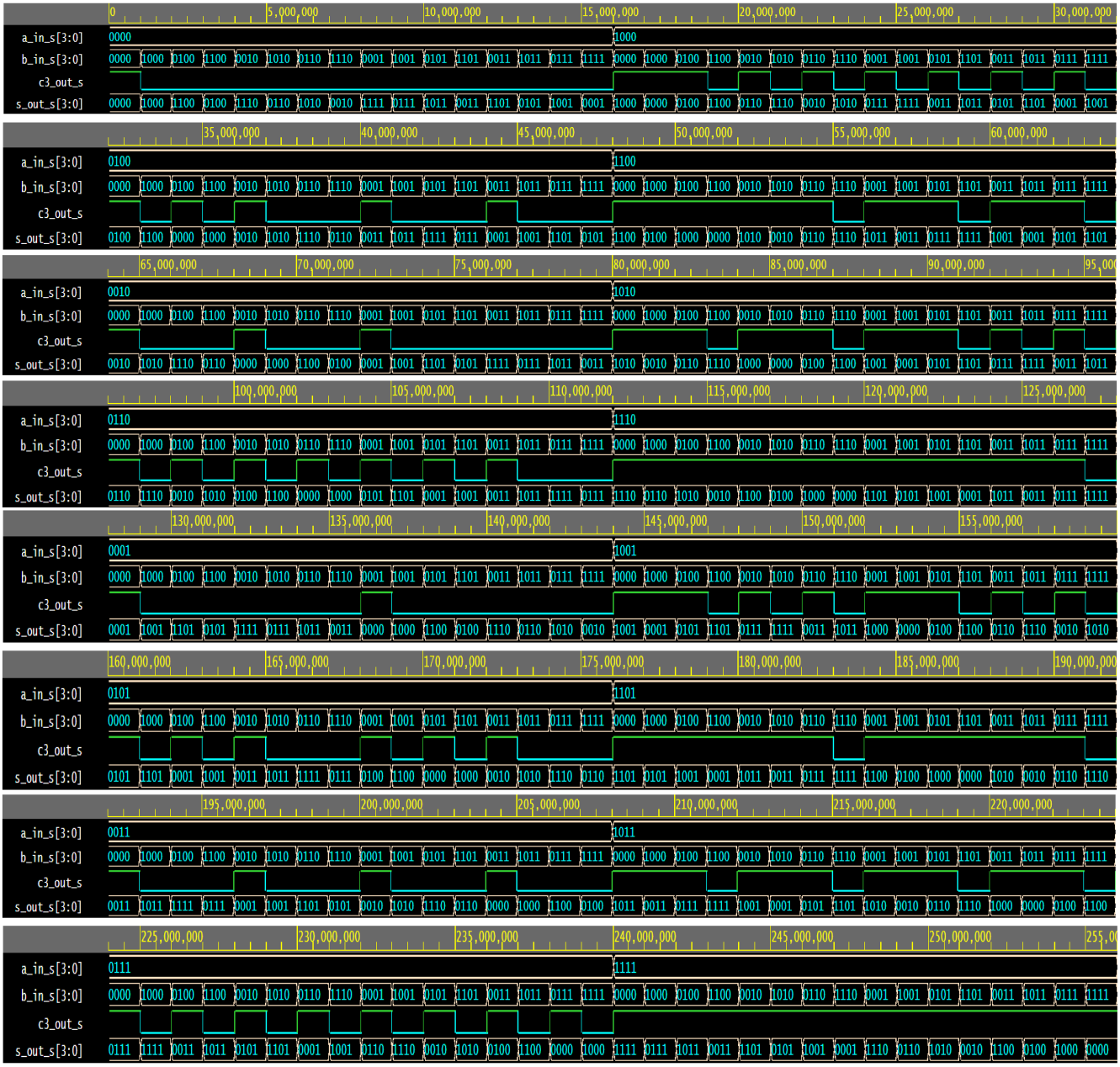

EPWave beta
Load
Save
Examples
About
Apps
es-KarimFathi2025@alexu.edu.eg
EDA Playground
https://www.edaplayground.com/x/b7J8
From:
0ps
To:
8,000ps
Get Signals
Radix
100%
0
1,000
2,000
3,000
4,000
5,000
6,000
7,000
a
b
carry
cin
sum
Brought to you by DOULOS

3. ripple adder/subtractor:

1- adding:



2-subtracting:



3.code snippets:

1. half adder:

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

entity half_adder is
port(
  a : in std_logic;
  b : in std_logic;
  sum : out std_logic;
  carry : out std_logic);
end half_adder;

architecture half_adderfun of half_adder is
begin


sum <= a xor b;
carry <= a and b;

end half_adderfun;
```

It’s test bench:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.NUMERIC_STD.all;
entity testbench_half_adder is

end testbench_half_adder;

architecture tb_half of testbench_half_adder is

component half_adder is
port(
  a : in std_logic;
  b : in std_logic;
  sum : out std_logic;
  carry : out std_logic);
end component;

signal a_h,b_h,sum_h,carry_h : std_logic;

begin
DUT : half_adder port map(a_h,b_h,sum_h,carry_h);
process
begin
for i in 0 to 3 loop
      (a_h,b_h) <= STD_LOGIC_VECTOR(TO_UNSIGNED(i,2));
      wait for 1 NS;
 end loop;
a_h <= '0';
b_h <= '0';
assert false report "test done" severity note;
wait ;
end process;

end tb half;
```

2. full adder:

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

entity full_adder is
port(
  A2 : in std_logic;
  B2 : in std_logic;
  C2 : in std_logic;
  sum_2 : out std_logic;
  carry_2 : out std_logic);
end full_adder;

architecture full_adder_fun of full_adder is
component half_adder is
 port(
  a : in std_logic;
  b : in std_logic;
  sum : out std_logic;
  carry : out std_logic);
end component;
signal sig_1,sig_2,sig_3:std_logic;

begin

eq_bit0 : half_adder port map(A2,B2,sig_1,sig_2);
eq_bit1 : half_adder port map( sig_1, C2 , sum_2, sig_3);
carry_2 <= sig_3 or sig_2;
end full_adder_fun;
```

It’s test bench:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.NUMERIC_STD.all;
entity testbench_full_adder is

end testbench_full_adder;

architecture tb_full of testbench_full_adder is

component full_adder is
port(
  A2 : in std_logic;
  B2 : in std_logic;
  C2 : in std_logic;
  sum_2 : out std_logic;
  carry_2 : out std_logic);
end component;

signal a,b,cin,sum,carry : std_logic;

begin
DUT : full_adder port map(a,b,cin,sum,carry);

process
begin

for i in 0 to 7 loop
      (a,b,cin) <= STD_LOGIC_VECTOR(TO_UNSIGNED(i,3));
      wait for 1 NS;
 end loop;

a <= '0';
b <= '0';
cin <= '0';

assert false report "test done" severity note;
wait ;
end process;
end tb_full;
```

3. ripple adder/subtractor:

1-adding:

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

entity ripple_adder is
port(
  a : in std_logic_vector(3 downto 0);
  b : in std_logic_vector(3 downto 0);
  s : out std_logic_vector(3 downto 0);
  c3 : out std_logic);
end ripple_adder;


architecture ripple_adder_fun of ripple_adder is
component full_adder is
port(
  A2 : in std_logic;
  B2 : in std_logic;
  C2 : in std_logic;
  sum_2 : out std_logic;
  carry_2 : out std_logic);
end component;

signal c0,c1,c2 :std_logic;



begin

F_A0 : full_adder port map(a(0),b(0),'0',s(0),c0);
F_A1 : full_adder port map(a(1),b(1),c0,s(1),c1);
F_A2 : full_adder port map(a(2),b(2),c1,s(2),c2);
F_A3 : full_adder port map(a(3),b(3),c2,s(3),c3);

end ripple_adder_fun;
```

It’s test bench:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.NUMERIC_STD.all;
entity testbench is

end testbench;

architecture tb of testbench is

component ripple_adder is
port(
  a : in std_logic_vector(3 downto 0);
  b : in std_logic_vector(3 downto 0);
  s : out std_logic_vector(3 downto 0);
  c3 : out std_logic);
end component;

signal a_in : std_logic_vector (3 downto 0);
signal b_in : std_logic_vector (3 downto 0);
signal s_out : std_logic_vector (3 downto 0);
signal c3_out : std_logic;

begin
DUT : ripple_adder port map(a_in,b_in,s_out,c3_out);

process
begin

for i in 0 to 15 loop
     (a_in(0),a_in(1),a_in(2),a_in(3) ) <= STD_LOGIC_VECTOR(TO_UNSIGNED(i,4));
    for j in 0 to 15 loop
     (b_in(0),b_in(1),b_in(2),b_in(3) ) <= STD_LOGIC_VECTOR(TO_UNSIGNED(j,4));
      wait for 1 NS;
    end loop;
 end loop;

a_in <= (3 downto 0 => '0');
b_in <= (3 downto 0 => '0');

assert false report "test done" severity note;
wait ;
end process;
end tb;
```

2-subtracting:

```
entity ripple_subtractor is
port(
  a_s : in std_logic_vector(3 downto 0);
  b_s : in std_logic_vector(3 downto 0);
  s_s : out std_logic_vector(3 downto 0);
  c3_s : out std_logic);
end ripple_subtractor;

architecture ripple_subtractor_fun of ripple_subtractor is
component full_adder is
port(
  A2 : in std_logic;
  B2 : in std_logic;
  C2 : in std_logic;
  sum_2 : out std_logic;
  carry_2 : out std_logic);
end component;

signal c0_s,c1_s,c2_s,x1,x2,x3,x4 :std_logic;

begin
x1 <= b_s(0) xor '1';
x2 <= b_s(1) xor '1';
x3 <= b_s(2) xor '1';
x4 <= b_s(3) xor '1';


F_S0: full_adder port map(a_s(0),x1,'1',s_s(0),c0_s);
F_S1: full_adder port map(a_s(1),x2,c0_s,s_s(1),c1_s);
F_S2: full_adder port map(a_s(2),x3,c1_s,s_s(2),c2_s);
F_S3: full_adder port map(a_s(3),x4,c2_s,s_s(3),c3_s);

end ripple_subtractor_fun;
```

It’s test bench:

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

entity testbench_s is
end testbench_s;

architecture tb_s of testbench_s is

component ripple_subtractor is
port(
  a_s : in std_logic_vector(3 downto 0);
  b_s : in std_logic_vector(3 downto 0);
  s_s : out std_logic_vector(3 downto 0);
  c3_s : out std_logic);
end component;

signal a_in_s : std_logic_vector (3 downto 0);
signal b_in_s : std_logic_vector (3 downto 0);
signal s_out_s : std_logic_vector (3 downto 0);
signal c3_out_s : std_logic;

begin
DUT_s : ripple_subtractor port map(a_in_s,b_in_S,s_out_s,c3_out_s);

process
begin

for i in 0 to 15 loop
(a_in_s(0),a_in_s(1),a_in_s(2),a_in_s(3) ) <= STD_LOGIC_VECTOR(TO_UNSIGNED(i,4));
    for j in 0 to 15 loop
(b_in_s(0),b_in_s(1),b_in_s(2),b_in_s(3) ) <= STD_LOGIC_VECTOR(TO_UNSIGNED(j,4));
      wait for 1 NS;
    end loop;
end loop;

a_in_s <= (3 downto 0 => '0');
b_in_s <= (3 downto 0 => '0');

assert false report "test done" severity note;
wait ;
end process;
end tb_s;
```