Appendix (project Advance-Bader Tawafsheh)

```
------------------------ the gates library -------------------------------

library ieee;
use ieee.std_logic_1164.all;
package MY_GATES is
      COMPONENT xor2 IS
            PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
    END COMPONENT xor2;
      COMPONENT xnor2 IS
            PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
    END COMPONENT xnor2;
    COMPONENT and2 IS
            PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
      END COMPONENT and2;
    COMPONENT or2 IS
        PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
    END COMPONENT or2;
      COMPONENT not1 IS
            PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
    END COMPONENT not1;
      COMPONENT nand2 IS
            PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
    END COMPONENT nand2;
      COMPONENT nor2 IS
            PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
    END COMPONENT nor2;
      COMPONENT or3 IS
            PORT (a, b,c: IN STD_LOGIC; d: OUT STD_LOGIC);
    END COMPONENT or3;
end package MY_GATES;

-------------------------------NOT gate 4ns delay----------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY not1 IS
      PORT( a: IN std_logic;
      b: OUT std_logic);
END ENTITY not1;
ARCHITECTURE simple OF not1 IS
BEGIN
      b <= NOT a AFTER 4ns;
END   ARCHITECTURE simple;
-------------------------------NAND gate 5ns delay ------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY nand2 IS
      PORT(a, b: IN std_logic;
      c: OUT std_logic);
END ENTITY nand2;
ARCHITECTURE simple OF nand2 IS
```

```
BEGIN
	c <= a NAND b AFTER 5ns;
END ARCHITECTURE simple;
--------------------------   NOR gate 5ns delay --------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY nor2 IS
	PORT(a, b: IN std_logic;
	c: OUT std_logic);
END ENTITY nor2;
ARCHITECTURE simple OF nor2 IS
BEGIN
	c <= a NOR b AFTER 5ns;
END ARCHITECTURE simple;
---------------------------- AND gate 7ns delay --------------------------


LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY and2 IS
	PORT(a, b: IN std_logic;
	c: OUT std_logic);
END ENTITY and2;
ARCHITECTURE simple OF and2 IS
BEGIN
	c <= a AND b AFTER 7ns;
END ARCHITECTURE simple;
---------------------------- OR gate 7ns delay (A OR B )--------------------

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY or2 IS
	PORT(a, b: IN std_logic;
	c: OUT std_logic);
END ENTITY or2;
ARCHITECTURE simple OF or2 IS
BEGIN
	c <= a OR b AFTER 7ns;
END ARCHITECTURE simple;
------------------------- OR gate 7ns delay (A OR B OR C) -----------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY or3 IS
	PORT(a, b,c: IN std_logic;
	d: OUT std_logic);
END ENTITY or3;
ARCHITECTURE simple OF or3 IS
BEGIN
	d <= a OR b OR c AFTER 7ns;
END ARCHITECTURE simple;
------------------------- XOR gate 11ns delay-------------------------------
```

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY xor2 IS
	PORT(a, b: IN std_logic;
	c: OUT std_logic);
END ENTITY xor2;
ARCHITECTURE simple OF xor2 IS
BEGIN
	c <= a XOR b AFTER 11ns;
END ARCHITECTURE simple;
----------------------- XNOR gate 9ns delay ------------------------------


LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY xnor2 IS
	PORT(a, b: IN std_logic;
	c: OUT std_logic);
END ENTITY xnor2;
ARCHITECTURE simple OF xnor2 IS
BEGIN
	c <= a XNOR b AFTER 9ns;
END ARCHITECTURE simple;
---------------------------Full Adder-------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY gates;
USE gates.MY_GATES.ALL;

ENTITY FA IS
	PORT(a, b, c_in: IN std_logic;
	sum, c_out: OUT std_logic := '0') ;
END ENTITY FA;
ARCHITECTURE simple OF FA IS
SIGNAL n1, n2, n3, n4, n5: std_logic;
BEGIN
	g1: xor2 PORT MAP (a, b, n1);
	g2: xor2 PORT MAP (c_in, n1, sum);
	g3: and2 PORT MAP (a, b, n2);
	g4: and2 PORT MAP (b, c_in, n3);
	g5: and2 PORT MAP (a, c_in, n4);
	g6: or2 PORT MAP (n2, n3, n5);
	g7: or2 PORT MAP (n4, n5, c_out);
END ARCHITECTURE simple;
------------------------------------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY gates;
USE gates.MY_GATES.ALL;

--4-bit adder entity
ENTITY four_bit_adder IS
	PORT(a, b: IN std_logic_vector(3 DOWNTO 0);
	c_in: IN std_logic;
```

```
      sum: OUT std_logic_vector(3 DOWNTO 0);
      c_out: OUT std_logic);
END ENTITY four_bit_adder;


--ripple 4-bit adder
ARCHITECTURE ripple_four_bit_adder OF four_bit_adder IS
SIGNAL carry: std_logic_vector(2 DOWNTO 0);
BEGIN
      g1: ENTITY work.FA(simple) PORT MAP (a(0), b(0), c_in, sum(0),
carry(0));
      g2: ENTITY work.FA(simple) PORT MAP (a(1), b(1), carry(0), sum(1),
carry(1));
      g3: ENTITY work.FA(simple) PORT MAP (a(2), b(2), carry(1), sum(2),
carry(2));
      g4: ENTITY work.FA(simple) PORT MAP (a(3), b(3), carry(2), sum(3),
c_out);
END ARCHITECTURE ripple_four_bit_adder;


--carry-look-ahead 4-bit adder
ARCHITECTURE look_ahead_four_bit_adder OF four_bit_adder IS
SIGNAL carry: std_logic_vector(4 DOWNTO 0) := "00000";
SIGNAL p, g, n2, n1: std_logic_vector(3 DOWNTO 0) := "0000";

BEGIN
      carry(0) <= c_in;
      gen: FOR i IN 0 TO 3 GENERATE
            g1: xor2 PORT MAP (a(i), b(i), p(i));
            g2: and2 PORT MAP (a(i), b(i), g(i));
            g3: xor2 PORT MAP (p(i), carry(i), sum(i));
            g4: and2 PORT MAP (p(i), carry(i), n1(i));
            g5: or2 PORT MAP (g(i), n1(i), n2(i));
            carry(i+1) <= n2(i);
      END GENERATE gen;
      c_out <= carry(4);
END ARCHITECTURE look_ahead_four_bit_adder;
------------ combinational circuit added to build bcd circuit----------------

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY gates;
USE gates.MY_GATES.ALL;

entity combinational_circuit is
      port(a,b,c,d: in std_logic;
      e_out:out std_logic);
end entity combinational_circuit;

architecture simple of combinational_circuit is
signal and1_out,and2_out:std_logic;
begin
      g1: and2 PORT MAP (a,b,and1_out);
      g2: and2 PORT MAP (a,c,and2_out);
      g3: or3  PORT MAP (and1_out,and2_out,d,e_out);
end architecture simple;
```

```
----------------------bcd-adder  4-bit bcd adder entity----------------------


LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY gates;
USE gates.MY_GATES.ALL;

entity bcd_adder is
      port(a,b :in std_logic_vector(3 downto 0);
      c_in :in std_logic;
      sum: out std_logic_vector(3 downto 0);
      c_out :inout std_logic);
end entity bcd_adder;

architecture ripple_bcd_adder of bcd_adder is
signal sum1_signal: std_logic_vector(3 downto 0);
signal sum2_signal: std_logic_vector(3 downto 0) := (others => '0');
signal cout_signal: std_logic;
signal cin_signal: std_logic := '0';
signal egnored_signal: std_logic;
begin
      g1: ENTITY work.four_bit_adder(ripple_four_bit_adder) PORT MAP
(a,b,c_in,sum1_signal,cout_signal);
      g2: ENTITY work.combinational_circuit(simple) PORT MAP
(sum1_signal(3),sum1_signal(2),sum1_signal(1),cout_signal,c_out);
      sum2_signal(2) <= c_out;
      sum2_signal(1) <= c_out;
      g3: ENTITY work.four_bit_adder(ripple_four_bit_adder) PORT MAP
(sum1_signal,sum2_signal,cin_signal,sum,egnored_signal);
end architecture ripple_bcd_adder;

architecture look_ahead_bcd_adder of bcd_adder is
signal sum1_signal: std_logic_vector(3 downto 0);
signal sum2_signal: std_logic_vector(3 downto 0) := (others => '0');
signal cout_signal: std_logic;
signal cin_signal: std_logic := '0';
signal egnored_signal: std_logic;
begin
      g1: ENTITY work.four_bit_adder(look_ahead_four_bit_adder) PORT MAP
(a,b,c_in,sum1_signal,cout_signal);
      g2: ENTITY work.combinational_circuit(simple) PORT MAP
(sum1_signal(3),sum1_signal(2),sum1_signal(1),cout_signal,c_out);
      sum2_signal(2) <= c_out;
      sum2_signal(1) <= c_out;
      g3: ENTITY work.four_bit_adder(look_ahead_four_bit_adder) PORT MAP
(sum1_signal,sum2_signal,cin_signal,sum,egnored_signal);
end architecture look_ahead_bcd_adder;

-------------------------verification file---------------------------------

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.NUMERIC_STD.ALL;
USE STD.TEXTIO.ALL;
```

```
use IEEE.std_logic_textio.all;
ENTITY verfication is
end entity verfication;
architecture simple of verfication is
signal expectedRes, actualRes: std_logic_vector(4 downto 0);
signal a: std_logic_vector(3 downto 0);
signal b: std_logic_vector(3 downto 0);
signal i,j: Integer := 0 ;

signal clock: std_logic := '0' ;
begin
      bcd: entity work.bcd_adder(ripple_bcd_adder) port map (a,
b,'0',actualRes(3 downto 0),actualRes(4));
            clock <= not clock after 120ns;
      process
      FILE outputFile : TEXT;
      VARIABLE file_status : FILE_OPEN_STATUS;
      VARIABLE buff : LINE;
      VAriable outt : INteger;
      begin
            FILE_OPEN(file_status, outputFile, "vereficationData.txt",
WRITE_MODE);
            for i in 0 to 9 loop
                  for j in 0 to 9 loop

                        a <= CONV_STD_LOGIC_VECTOR(i, 4);
                        b <= CONV_STD_LOGIC_VECTOR(j, 4);
                        if (I+J >9) then
                              expectedRes(4) <= '1';
                              expectedRes(3 downto 0) <=
CONV_STD_LOGIC_VECTOR(i+j+6,4);
                        else
                              expectedRes <= CONV_STD_LOGIC_VECTOR(i+j,5);
                        end if;

                        WRITE(buff, a);
                     WRITE(buff, " + ");
                     WRITE(buff, b);
                     WRITE(buff, " expected value =  ");
                     WRITE(buff, expectedRes);
                     WRITE(buff, " , actual value =  ");
                     WRITE(buff, actualRes);
                        if (actualRes = expectedRes) then
                              WRITE(buff, " MATCH ");
                        else
                              WRITE(buff, " MISMATCH ");
                        end if;
                     writeline(outputFile, buff);
                        assert (actualRes = expectedRes)
                        report ("mismatch")
                        severity WARNING;
                        if (i = 9 and j = 9) then
                              exit;
                        end if;
                        wait until rising_edge(clock);
                  end loop;
            end loop;
```

```
      end process;
end architecture simple;
------------------------------The Whole System-----------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY Whole_System IS
      PORT (clock: IN std_logic;
      a: IN std_logic_vector(3 DOWNTO 0);
      b: IN std_logic_vector(3 DOWNTO 0);
      res: OUT std_logic_vector(4 DOWNTO 0));
END ENTITY Whole_System;


ARCHITECTURE simple OF Whole_System IS
SIGNAL input_signal: std_logic_vector(7 DOWNTO 0);
SIGNAL output_signal: std_logic_vector(4 DOWNTO 0);
BEGIN
      inputregister: entity work.register8(simple) PORT
MAP(a,b,input_signal,clock);

      bcd_adder: ENTITY work.bcd_adder(look_ahead_bcd_adder) PORT MAP
(input_signal(3 downto 0), input_signal(7 downto 4), '0',output_signal(3
downto 0),output_signal(4));
      outputregister: ENTITY work.register5(simple) PORT MAP (output_signal,
res, clock);

END ARCHITECTURE simple ;
--------------------------------------------------------------------------
```