Amirkabir University of Technology
(Tehran Polytechnic)

پروژه درس مدار منطقی

ترم دوم ۰۰-۱۳۹۹

استاد درس: دکتر پورفرد

اعضای گروه:

سینا ربیعی	۹۸۲۳۰۳۵
مهدی رنجبر بافقی	۹۸۲۳۰۴۰
علیرضا فقیه علی آبادی	۹۸۲۳۰۷۱
ذاکر قلیچی	۹۸۲۳۰۷۳
بردیا سهامی	۹۹۲۳۵۰۳

عنوان پروژه:

Bonus)

در این قسمت کد بخش Task1 را به گونه‌ای تغییر می‌دهیم که خواسته‌های جدول زیر را برآورده کند.

OPCODE	OPERATION	FORMULA	Z	Cout	V	X_bin_pal	X_prime	N
1010	Variable Rotation Left	X = A <<< B	↕	–	–	↕	↕	0
1011	Variable Rotation Left with Carry	(Cout:X) = (Cin:A) <<< B	↕	↕	–	↕	↕	0
1100	Variable Logic Shift Right	X = A >> B	↕	↕	–	↕	↕	0
1101	Variable Arithmetic Shift Right with Rounding	X = (A >> B) + A(B-1)	↕	↕	–	↕	↕	↕
1110	Variable Logic Shift Left	X = A << B	↕	↕	↕	↕	↕	↕

OPCODE هایی که در جدول ذکر نشده، تفاوتی با Task1 ندارند. پس کافی است که این ۵ زیرماژول را تغییر دهیم:

OPCODE 1010 (Rotation Left):

مقادیر N و V و Cout صفرند و از خروجی دوم نیز استفاده نخواهیم کرد پس مقدار آن را صفر قرار می‌دهیم.
در این زیرماژول هر دو ورودی را دریافت می‌کنیم و ورودی دوم مشخص می‌کند که عمل Rotation چند بار روی ورودی اول انجام شود. از آن‌جایی که ورودی ها ۸ بیتی هستند، با هر ۸ بار چرخش ورودی اول به خودش می‌رسیم. به همین دلیل باقی‌مانده ورودی دوم را به ۸ محاسبه می‌کنیم و سپس به کمک یک if اگر این مقدار صفر بود، ورودی اول را بدون تغییر به خروجی می‌دهیم و در غیر این‌صورت به مقدار بدست آمده آن را می‌چرخانیم خروجی برای محاسبه Z همان O1 است.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity RotL is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
             I2 : in  STD_LOGIC_VECTOR (7 downto 0);
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
           O2 : out  STD_LOGIC_VECTOR (7 downto 0);
                N : out STD_LOGIC;
                Cout : inout STD_LOGIC;
                V : inout STD_LOGIC;
                Z : inout STD_LOGIC;
                X_bin_pal : out STD_LOGIC;
                X_prime : out STD_LOGIC;
                F_active : out STD_LOGIC);
    end RotL;

architecture Behavioral of RotL is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
     signal I2U: STD_LOGIC_VECTOR (7 downto 0); --Defined so
that the input I2 gets used and connected to other parts

begin

     I2U <= I2;
     Process (I2U, I1)
          variable I2Num : integer;
          variable B : integer;
          begin
               I2Num := conv_integer(I2U);
               B := (I2Num rem 8);
               if (B = 0) then
                    O1 <= I1;
               else
                    O1 <= I1(7-B downto 0) & I1(7 downto 8-B);
               end if;
     end Process;

     N <= '0';
     O2 <= x"00";
     Cout <= '0';
     V <= '0';
     Z <= '1' when O1=x"00" else '0';
     F_active <= Z or V or Cout;

---------------Palindromic check---------------
     F(0) <= (O1(0) xnor O1(7));
     F(1) <= (O1(1) xnor O1(6));
     F(2) <= (O1(2) xnor O1(5));
```

```
     F(3) <= (O1(3) xnor O1(4));

     X_bin_pal <= '1' when (F = "1111") else
                  '0';
-----------------------------------------------
------------------Prime check------------------
   Process (O1)
          variable Num : integer;
      begin
                Num := conv_integer(O1);
                if (Num = 0 or Num = 1) then
                    X_prime <= '0';
                elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
                    X_prime <= '1';
                elsif (Num rem 2 = 0 or
                           Num rem 3 = 0 or
                           Num rem 5 = 0 or
                           Num rem 7 = 0 or
                           Num rem 11 = 0 or
                           Num rem 13 = 0) then
                    X_prime <= '0';
                else
                    X_prime <= '1';
                end if;
   end Process;
-----------------------------------------------
end Behavioral;
```

کد شماره‌ی ۱: ماژول Variable Rotation left

OPCODE 1011 (Rotation Left with Carry):

مقادیر N و V صفرند و از خروجی دوم نیز استفاده نخواهیم کرد پس مقدار آن را صفر قرار می‌دهیم. در این زیرماژول هر دو ورودی را دریافت می‌کنیم و ورودی دوم مشخص می‌کند که عمل Rotation چند بار روی رشته بیت مورد نظر انجام شود. لازم به ذکر است که رشته بیت ورودی برای قرار گرفتن تحت Rotation حاصل اضافه کردن بیت Cin به سمت چپ ورودی اول است. حالا از آنجایی که رشته بیت ما ۹ بیتی است، با هر ۹ بار چرخش به خودش می‌رسیم. به همین دلیل باقی‌مانده ورودی دوم را به ۹ محاسبه می‌کنیم و سپس به کمک یک if اگر این مقدار صفر بود، ورودی اول را بدون تغییر به خروجی می‌دهیم و در غیر این‌صورت به مقدار بدست آمده آن را می‌چرخانیم و MSB آن را به Cout می‌دهیم و دیگر ۸ بیت را به O1. خروجی برای محاسبه Z همان Cout & O1 است.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity RotL_CARRY is
```

```
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
                I2 : in  STD_LOGIC_VECTOR (7 downto 0);
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
           O2 : out  STD_LOGIC_VECTOR (7 downto 0);
                N : out STD_LOGIC;
                Cin : in STD_LOGIC;
                Cout : inout STD_LOGIC;
                V : inout STD_LOGIC;
                Z : inout STD_LOGIC;
                X_bin_pal : out STD_LOGIC;
                X_prime : out STD_LOGIC;
                F_active : out STD_LOGIC);
    end RotL_CARRY;

architecture Behavioral of RotL_CARRY is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
     signal I2U: STD_LOGIC_VECTOR (7 downto 0); --Defined so
that the input I2 gets used and connected to other parts
     signal I1N: STD_LOGIC_VECTOR (8 downto 0); --The result of
Cin and I1 concatenation
     signal O1N: STD_LOGIC_VECTOR (8 downto 0); --The result of
Rotation of I1N

begin

     I1N <= Cin & I1;
     I2U <= I2;
     Process (I2U, I1N)
          variable I2Num : integer;
          variable B : integer;
          begin
               I2Num := conv_integer(I2U);
               B := (I2Num rem 9);
               if (B = 0) then
                    O1N <= I1N;
               else
                    O1N <= I1N(8-B downto 0) & I1N(8 downto 9-
B);
               end if;
     end Process;

     O1 <= O1N(7 downto 0);
     N <= '0';
     O2 <= x"00";
     Cout <= O1N(8);
     V <= '0';
     Z <= '1' when (Cout & O1 =o"000") else '0';
     F_active <= Z or V or Cout;

---------------Palindromic check---------------
     F(0) <= (O1(0) xnor O1(7));
```

```
     F(1) <= (O1(1) xnor O1(6));
     F(2) <= (O1(2) xnor O1(5));
     F(3) <= (O1(3) xnor O1(4));

     X_bin_pal <= '1' when (F = "1111") else
                  '0';
----------------------------------------------
------------------Prime check-----------------
   Process (O1)
          variable Num : integer;
      begin
                Num := conv_integer(O1);
                if (Num = 0 or Num = 1) then
                    X_prime <= '0';
                elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
                    X_prime <= '1';
                elsif (Num rem 2 = 0 or
                           Num rem 3 = 0 or
                           Num rem 5 = 0 or
                           Num rem 7 = 0 or
                           Num rem 11 = 0 or
                           Num rem 13 = 0) then
                    X_prime <= '0';
                else
                    X_prime <= '1';
                end if;
   end Process;
----------------------------------------------
end Behavioral;
```

کد شماره‌ی ۲: ماژول *Variable Rotation left with Carry*

OPCODE 1100 (Logic Shift Right):

مقدار V و N صفر است و از خروجی دوم نیز استفاده نخواهیم کرد پس مقدار آن را صفر قرار می‌دهیم.

در این زیرماژول هر دو ورودی را دریافت می‌کنیم و ورودی دوم مشخص می‌کند که عمل شیفت به راست چند بار روی رشته بیت مورد نظر انجام شود. از آن‌جایی که ورودی ها ۸ بیتی هستند، اگر عمل شیفت بیش از ۸ بار انجام شود، خروجی کاملا صفر خواهد بود. همچنین چون Cout مقدار بیت خارج شده از رشته مورد نظر است، در این بعد از ۸ بار شیفت فقط صفر خارج می‌شود و Cout نیز ۰ خواهد بود. به همین ترتیب اگر دقیقا ۸ بار شیفت بدهیم، خروجی کامل ۰ و Cout برابر MSB رشته بیت ورودی است. برای مقادیر زیر ۸، از یک سیگنال صفر استفاده می‌کنیم، به این ترتیب که به تعداد دفعات شیفت به کمک این سیگنال به اول خروجی صفر اضافه می‌کنیم و باقی بیت‌ها را با رشته بیت ورودی از طرف MSB پر می‌کنیم. خروجی برای محاسبه Z همان Cout & O1 است.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity LSR is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
               I2 : in  STD_LOGIC_VECTOR (7 downto 0);
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
           O2 : out  STD_LOGIC_VECTOR (7 downto 0);
                 N : out STD_LOGIC;
                 Cout : inout STD_LOGIC;
                 V : inout STD_LOGIC;
                 Z : inout STD_LOGIC;
                 X_bin_pal : out STD_LOGIC;
                 X_prime : out STD_LOGIC;
                 F_active : out STD_LOGIC);
    end LSR;

architecture Behavioral of LSR is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
     signal I2U: STD_LOGIC_VECTOR (7 downto 0); --Defined so
that the input I2 gets used and connected to other parts

begin

     I2U <= I2;
     Process (I2U, I1)
          variable I2Num : integer;
          variable B: STD_LOGIC_VECTOR (7 downto 0);   --Zero
Signal
          begin
               B := x"00";
               I2Num := conv_integer(I2U);
               if (I2Num = 0) then
                    O1 <= I1;
                    Cout <= '0';
               elsif (I2Num = 8) then
                    O1 <= (others => '0');
                    Cout <= I1(7);
               elsif (I2Num > 8) then
                    O1 <= (others => '0');
                    Cout <= '0';
               else
                    O1 <= B(I2Num-1 downto 0) & I1(7 downto
I2Num);
                    Cout <= I1(I2Num-1);
               end if;
     end Process;

     N <= '0';
```

```
     O2 <= x"00";
     V <= '0';
     Z <= '1' when (Cout & O1 =o"000") else '0';
     F_active <= Z or V or Cout;

---------------Palindromic check---------------
     F(0) <= (O1(0) xnor O1(7));
     F(1) <= (O1(1) xnor O1(6));
     F(2) <= (O1(2) xnor O1(5));
     F(3) <= (O1(3) xnor O1(4));

     X_bin_pal <= '1' when (F = "1111") else
                  '0';
-----------------------------------------------
------------------Prime check------------------
   Process (O1)
          variable Num : integer;
      begin
                Num := conv_integer(O1);
                if (Num = 0 or Num = 1) then
                    X_prime <= '0';
                elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
                    X_prime <= '1';
                elsif (Num rem 2 = 0 or
                           Num rem 3 = 0 or
                           Num rem 5 = 0 or
                           Num rem 7 = 0 or
                           Num rem 11 = 0 or
                           Num rem 13 = 0) then
                    X_prime <= '0';
                else
                    X_prime <= '1';
                end if;
   end Process;
-----------------------------------------------
end Behavioral;
```

کد شماره‌ی ۳: ماژول *Variable Logic shift right*

OPCODE 1101 (Arithmetic Shift Right with Rounding):

مقدار V صفر است و از خروجی دوم نیز استفاده نخواهیم کرد پس مقدار آن را صفر قرار می‌دهیم. با هر بار Arithmetic shift right، ورودی یک بار بر ۲ تقسیم می‌شود و بیت علامت در سمت چپ تکرار می‌شود. این موضوع برای اعداد فرد مشکل گرد کردن را بوجود می‌آورد. با بررسی چند حالت متوجه می‌شویم که برای اعداد مثبت فرد با هر بار شیفت به سمت ۰ گرد می‌شوند ولی اعداد منفی فرد در مبنای ۲ به سمت منفی بی‌نهایت گرد می‌شوند. بیتی که به عنوان بیت Rounding به مقدار شیفت داده شده اضافه می‌شود باعث می‌شود این گرد کردن برای اعداد منفی فرد نیز به سمت صفر انجام شود. برای این عمل جمع مشکل overflow داریم. برای اعداد مثبت این مشکل هیچ وقت رخ نمی‌دهد، زیرا حتی با یک واحد شیفت حداقل دو

صفر در سمت چپ خواهیم داشت (بیت علامت و تکرار آن به دلیل خاصیت Arithmetic Shift Right) که این اگر carry ورودی به این دو بیت ۱ باشد باز هم سرریز نخواهیم داشت. ولی برای خروجی‌های منفی با یک بار شیفت بیتی که به عنوان بیت Ronding اضافه می‌شود، بیت ۰ ورودی است. برای این که در این حالت overflow داشته باشیم، باید همه‌ی بیت ها یک باشند تا carry در خروجی حرکت و در نهایت از بیت هشتم خارج شود. در این حالت ورودی برابر ۱- در قرداد مکمل ۲ است. حالا که سرریز داریم اگر بیت MSB خروجی را در نظر نگیریم، خروجی صفر خواهد بود که همان حاصل تقسیم ۱- بر ۲ گرد شده به سمت ۰ می‌باشد. اگر دو واحد شیفت داشته باشیم، با نوشتن حالت‌های ممکن برای ورودی که باعث بوجود آمدن overflow شود، دو مقدار ۱- و ۲- خواهد بود. برای این حالت نیز اگر بیت سرریز را در نظر نگیریم، خروجی ۰ است و با حاصل تقسیم ۱- یا ۲- بر ۴ که به سمت ۰ گرد شده است همخوانی دارد. به همین ترتیب می‌توان برای دیگر اعداد منفی خروجی نیز اثبات کرد که بیت overflow باید حذف شود. بیت MSB خروجی همان بیت علامت است، بنابراین در N ریخته می‌شود. در پایان بیت خارج شده از سمت راست را در Cout قرار می‌دهیم. خروجی برای محاسبه Z همان Cout & O1 است.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ASR is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
                I2 : in  STD_LOGIC_VECTOR (7 downto 0);
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
           O2 : out  STD_LOGIC_VECTOR (7 downto 0);
                 N : out STD_LOGIC;
                 Cout : inout STD_LOGIC;
                 V : inout STD_LOGIC;
                 Z : inout STD_LOGIC;
                 X_bin_pal : out STD_LOGIC;
                 X_prime : out STD_LOGIC;
                 F_active : out STD_LOGIC);
    end ASR;

architecture Behavioral of ASR is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
     signal I2U: STD_LOGIC_VECTOR (7 downto 0); --Defined so
that the input I2 gets used and connected to other parts

begin

     I2U <= I2;
     Process (I2U, I1)
          variable I2Num : integer;
          variable RESULT: STD_LOGIC_VECTOR (8 downto 0);
```

```
          variable B: STD_LOGIC_VECTOR (7 downto 0);   --A
signal consisted of I1s sign bit
          begin
               I2Num := conv_integer(I2U);
               B := (others => I1(7));
               if (I2Num = 0) then
                    O1 <= I1;
                    Cout <= '0';
               elsif (I2Num = 8) then
                    RESULT := ('0' & B) + (I1(7));
                    O1 <= RESULT(7 downto 0);
                    Cout <= I1(7);
               elsif (I2Num > 8) then
                    O1 <= B;
                    Cout <= I1(7);
               else
                    RESULT := ('0' & B(I2Num-1 downto 0) & I1(7
downto I2Num)) + I1(I2Num-1);
                    O1 <= RESULT(7 downto 0);
                    Cout <= I1(I2Num-1);
               end if;
               --The Carry of the rounding operation must be
dicarded so we can have a correct result
     end Process;

     N <= O1(7);
     O2 <= x"00";
     V <= '0';
     Z <= '1' when (Cout & O1 =o"000") else '0';
     F_active <= Z or V or Cout;

---------------Palindromic check---------------
     F(0) <= (O1(0) xnor O1(7));
     F(1) <= (O1(1) xnor O1(6));
     F(2) <= (O1(2) xnor O1(5));
     F(3) <= (O1(3) xnor O1(4));

     X_bin_pal <= '1' when (F = "1111") else
                  '0';
-----------------------------------------------
------------------Prime check------------------
   Process (O1)
          variable Num : integer;
      begin
                Num := conv_integer(O1);
                if (Num = 0 or Num = 1) then
                    X_prime <= '0';
                elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
                    X_prime <= '1';
                elsif (Num rem 2 = 0 or
                           Num rem 3 = 0 or
```

```
                            Num rem 5 = 0 or
                            Num rem 7 = 0 or
                            Num rem 11 = 0 or
                            Num rem 13 = 0) then
                        X_prime <= '0';
                    else
                        X_prime <= '1';
                    end if;
    end Process;
-----------------------------------------------
end Behavioral;
```

کد شماره‌ی ۴: ماژول Variable Arithmetic shift right with Rounding

OPCODE 1110 (Logic Shift Left):

این زیرماژول مانند زیرماژول Logic shift right است با این تفاوت که این زیرماژول به سمت چپ شیفت می‌دهد. در توضیحات گزارش گفته شده که خروجی V در این کد دچار تغییر می‌شود ولی توضیحی در مورد نحوه مشخص کردن صفر یا یک بودن آن داده نشده. از طرفی چون در دیگر عملیات‌های شیفت V صفر است، در این مورد نیز آن را صفر می‌گیریم. از خروجی دوم نیز استفاده نخواهیم کرد پس مقدار آن را مانند N صفر قرار می‌دهیم.

در این زیرماژول هر دو ورودی را دریافت می‌کنیم و ورودی دوم مشخص می‌کند که عمل شیفت به چپ چند بار روی رشته بیت مورد نظر انجام شود. از آن‌جایی که ورودی ها ۸ بیتی هستند، اگر عمل شیفت بیش از ۸ بار انجام شود، خروجی کاملا صفر خواهد بود. همچنین چون Cout مقدار بیت خارج شده از رشته مورد نظر است، در این بعد از ۸ بار شیفت فقط صفر خارج می‌شود و Cout نیز ۰ خواهد بود. به همین ترتیب اگر دقیقا ۸ بار شیفت بدهیم، خروجی کامل ۰ و Cout برابر MSB رشته بیت ورودی است. برای مقادیر زیر ۸، از یک سیگنال صفر استفاده می‌کنیم، به این ترتیب که به تعداد دفعات شیفت به کمک این سیگنال به انتهای خروجی صفر اضافه می‌کنیم و باقی بیت‌ها را با رشته بیت ورودی از طرف LSB پر می‌کنیم. خروجی برای محاسبه Z همان Cout & O1 است.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity LSL is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
                I2 : in  STD_LOGIC_VECTOR (7 downto 0);
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
           O2 : out  STD_LOGIC_VECTOR (7 downto 0);
                N : out STD_LOGIC;
                Cout : inout STD_LOGIC;
                V : inout STD_LOGIC;
                Z : inout STD_LOGIC;
                X_bin_pal : out STD_LOGIC;
```

```
                  X_prime : out STD_LOGIC;
                  F_active : out STD_LOGIC);
    end LSL;

architecture Behavioral of LSL is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
     signal I2U: STD_LOGIC_VECTOR (7 downto 0); --Defined so
that the input I2 gets used and connected to other parts

begin

     I2U <= I2;
     Process (I2U, I1)
          variable I2Num : integer;
          variable B: STD_LOGIC_VECTOR (7 downto 0);   --Zero
Signal
          begin
               B := x"00";
               I2num := conv_integer(I2U);
               if (I2Num = 0) then
                    O1 <= I1;
                    Cout <= '0';
               elsif (I2Num = 8) then
                    O1 <= (others => '0');
                    Cout <= I1(0);
               elsif (I2Num > 8) then
                    O1 <= (others => '0');
                    Cout <= '0';
               else
                    O1 <= I1(7-I2Num downto 0) & B(I2Num-1
downto 0);
                    Cout <= I1(I2Num-1);
               end if;
     end Process;

     N <= O1(7);
     O2 <= x"00";
     V <= '0';
     Z <= '1' when (Cout & O1 =o"000") else '0';
     F_active <= Z or V or Cout;

---------------Palindromic check---------------
     F(0) <= (O1(0) xnor O1(7));
     F(1) <= (O1(1) xnor O1(6));
     F(2) <= (O1(2) xnor O1(5));
     F(3) <= (O1(3) xnor O1(4));

     X_bin_pal <= '1' when (F = "1111") else
                  '0';
-----------------------------------------------
------------------Prime check------------------
```

```
   Process (O1)
          variable Num : integer;
      begin
                Num := conv_integer(O1);
                if (Num = 2) then
                    X_prime <= '1';
                else
                    X_prime <= '0';
                end if;
   end Process;
-- The Shifted output is always Multiple of 2
-----------------------------------------------
end Behavioral;
```

کد شماره‌ی ۵: ماژول Variable Logic shift left

در مرحله آخر لازم به نوشتن یک Top Module داریم که تمامی component ها در آن map کنم و خروجی را با توجه به OPCODE و توسط یک if تعیین کنیم. تفاوت این کد با کد موجود در Task1 در این است که برای ۵ OPCODE مشخص شده در بخش Bonus، به کامپوننت‌ها ورودی I2 اضافه شده و در قسمت Port map نیز این ورودی به ورودی B واحد ALU وصل می‌شود.

```
----------------------------------------------------------------------
-- Group Number: 10
-- Members: Sina Rabiee
--          Mohammad Mahdi Rnjbar Bafghi
--          Alireza Faghih Ali abadi
--          Zaker Ghelichi
--          Bardia Sahami
----------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ALU is
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
           B : in  STD_LOGIC_VECTOR (7 downto 0);
           Cin : in  STD_LOGIC;
           OPCODE : in  STD_LOGIC_VECTOR (3 downto 0);
           X : out  STD_LOGIC_VECTOR (7 downto 0);
           Y : out  STD_LOGIC_VECTOR (7 downto 0);
           Z : inout  STD_LOGIC;
           Cout : inout  STD_LOGIC;
           V : inout  STD_LOGIC;
           F_active : out  STD_LOGIC;
           X_bin_pal : out  STD_LOGIC;
           X_prime : out  STD_LOGIC;
           N : out  STD_LOGIC);
end ALU;
```

```
architecture Structral of ALU is

--------------------------Components---------------------------
     component AND8 is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           I2 : in  STD_LOGIC_VECTOR (7 downto 0);
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
           O2 : out  STD_LOGIC_VECTOR (7 downto 0);
           N : out STD_LOGIC;
           Cout : inout STD_LOGIC;
           V : inout STD_LOGIC;
           Z : inout STD_LOGIC;
           X_bin_pal : out STD_LOGIC;
           X_prime : out STD_LOGIC;
           F_active : out STD_LOGIC);
     end component;

     component OR8 is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           I2 : in  STD_LOGIC_VECTOR (7 downto 0);
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
           O2 : out  STD_LOGIC_VECTOR (7 downto 0);
           N : out STD_LOGIC;
           Cout : inout STD_LOGIC;
           V : inout STD_LOGIC;
           Z : inout STD_LOGIC;
           X_bin_pal : out STD_LOGIC;
           X_prime : out STD_LOGIC;
           F_active : out STD_LOGIC);
     end component;

     component XOR8 is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           I2 : in  STD_LOGIC_VECTOR (7 downto 0);
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
           O2 : out  STD_LOGIC_VECTOR (7 downto 0);
           N : out STD_LOGIC;
           Cout : inout STD_LOGIC;
           V : inout STD_LOGIC;
           Z : inout STD_LOGIC;
           X_bin_pal : out STD_LOGIC;
           X_prime : out STD_LOGIC;
           F_active : out STD_LOGIC);
     end component;

     component XNOR8 is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           I2 : in  STD_LOGIC_VECTOR (7 downto 0);
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
           O2 : out  STD_LOGIC_VECTOR (7 downto 0);
           N : out STD_LOGIC;
           Cout : inout STD_LOGIC;
```

```
        V : inout STD_LOGIC;
        Z : inout STD_LOGIC;
        X_bin_pal : out STD_LOGIC;
        X_prime : out STD_LOGIC;
        F_active : out STD_LOGIC);
 end component;

 component UADD is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
        I2 : in  STD_LOGIC_VECTOR (7 downto 0);
        O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
        O2 : out  STD_LOGIC_VECTOR (7 downto 0);
        N : out STD_LOGIC;
        Cout : inout STD_LOGIC;
        V : inout STD_LOGIC;
        Z : inout STD_LOGIC;
        X_bin_pal : out STD_LOGIC;
        X_prime : out STD_LOGIC;
        F_active : out STD_LOGIC);
 end component;

 component SADD is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
        I2 : in  STD_LOGIC_VECTOR (7 downto 0);
        O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
        O2 : out  STD_LOGIC_VECTOR (7 downto 0);
        N : out STD_LOGIC;
        Cout : inout STD_LOGIC;
        V : inout STD_LOGIC;
        Z : inout STD_LOGIC;
        X_bin_pal : out STD_LOGIC;
        X_prime : out STD_LOGIC;
        F_active : out STD_LOGIC);
 end component;

 component UADD_CARRY is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
        I2 : in  STD_LOGIC_VECTOR (7 downto 0);
        Cin : in  STD_LOGIC;
        O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
        O2 : out  STD_LOGIC_VECTOR (7 downto 0);
        N : out STD_LOGIC;
        Cout : inout STD_LOGIC;
        V : inout STD_LOGIC;
        Z : inout STD_LOGIC;
        X_bin_pal : out STD_LOGIC;
        X_prime : out STD_LOGIC;
        F_active : out STD_LOGIC);
 end component;

 component SMUL is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
```

```
        I2 : in  STD_LOGIC_VECTOR (7 downto 0);
        O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
        O2 : out  STD_LOGIC_VECTOR (7 downto 0);
        N : out STD_LOGIC;
        Cout : inout STD_LOGIC;
        V : inout STD_LOGIC;
        Z : inout STD_LOGIC;
        X_bin_pal : out STD_LOGIC;
        X_prime : out STD_LOGIC;
        F_active : out STD_LOGIC);
 end component;

 component UMUL is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
       I2 : in  STD_LOGIC_VECTOR (7 downto 0);
       O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
       O2 : out  STD_LOGIC_VECTOR (7 downto 0);
       N : out STD_LOGIC;
       Cout : inout STD_LOGIC;
       V : inout STD_LOGIC;
       Z : inout STD_LOGIC;
       X_bin_pal : out STD_LOGIC;
       X_prime : out STD_LOGIC;
       F_active : out STD_LOGIC);
 end component;

 component USUB is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
       I2 : in  STD_LOGIC_VECTOR (7 downto 0);
       O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
       O2 : out  STD_LOGIC_VECTOR (7 downto 0);
       N : out STD_LOGIC;
       Cout : inout STD_LOGIC;
       V : inout STD_LOGIC;
       Z : inout STD_LOGIC;
       X_bin_pal : out STD_LOGIC;
       X_prime : out STD_LOGIC;
       F_active : out STD_LOGIC);
 end component;

 component RotL is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
       I2 : in  STD_LOGIC_VECTOR (7 downto 0);
       O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
       O2 : out  STD_LOGIC_VECTOR (7 downto 0);
       N : out STD_LOGIC;
       Cout : inout STD_LOGIC;
       V : inout STD_LOGIC;
       Z : inout STD_LOGIC;
       X_bin_pal : out STD_LOGIC;
       X_prime : out STD_LOGIC;
       F_active : out STD_LOGIC);
```

```
 end component;

 component RotL_CARRY is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
       I2 : in  STD_LOGIC_VECTOR (7 downto 0);
       O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
       O2 : out  STD_LOGIC_VECTOR (7 downto 0);
       N : out STD_LOGIC;
       Cin : in STD_LOGIC;
       Cout : inout STD_LOGIC;
       V : inout STD_LOGIC;
       Z : inout STD_LOGIC;
       X_bin_pal : out STD_LOGIC;
       X_prime : out STD_LOGIC;
       F_active : out STD_LOGIC);
 end component;

 component LSR is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
       I2 : in  STD_LOGIC_VECTOR (7 downto 0);
       O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
       O2 : out  STD_LOGIC_VECTOR (7 downto 0);
       N : out STD_LOGIC;
       Cout : inout STD_LOGIC;
       V : inout STD_LOGIC;
       Z : inout STD_LOGIC;
       X_bin_pal : out STD_LOGIC;
       X_prime : out STD_LOGIC;
       F_active : out STD_LOGIC);
 end component;

 component ASR is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
       I2 : in  STD_LOGIC_VECTOR (7 downto 0);
       O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
       O2 : out  STD_LOGIC_VECTOR (7 downto 0);
       N : out STD_LOGIC;
       Cout : inout STD_LOGIC;
       V : inout STD_LOGIC;
       Z : inout STD_LOGIC;
       X_bin_pal : out STD_LOGIC;
       X_prime : out STD_LOGIC;
       F_active : out STD_LOGIC);
 end component;

 component LSL is
Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
       I2 : in  STD_LOGIC_VECTOR (7 downto 0);
       O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
       O2 : out  STD_LOGIC_VECTOR (7 downto 0);
       N : out STD_LOGIC;
       Cout : inout STD_LOGIC;
```

```
           V : inout STD_LOGIC;
           Z : inout STD_LOGIC;
           X_bin_pal : out STD_LOGIC;
           X_prime : out STD_LOGIC;
           F_active : out STD_LOGIC);
    end component;

    component BCD2BIN is
   Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
          I2 : in  STD_LOGIC_VECTOR (7 downto 0);
          O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
          O2 : out  STD_LOGIC_VECTOR (7 downto 0);
          N : out STD_LOGIC;
          Cout : inout STD_LOGIC;
          V : inout STD_LOGIC;
          Z : inout STD_LOGIC;
          X_bin_pal : out STD_LOGIC;
          X_prime : out STD_LOGIC;
          F_active : out STD_LOGIC);
    end component;
------------------------------------------------------------
---------------------------Signals--------------------------
    signal TrANDX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrANDY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrORX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrORY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrXORX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrXORY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrXNORX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrXNORY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrUADDX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrUADDY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrSADDX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrSADDY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrUADD_CARRYX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrUADD_CARRYY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrSMUL : STD_LOGIC_VECTOR (15 downto 0);

    signal TrUMUL : STD_LOGIC_VECTOR (15 downto 0);

    signal TrUSUBX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrUSUBY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrRotLX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrRotLY : STD_LOGIC_VECTOR (7 downto 0);
```

```
    signal TrRotL_CARRYX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrRotL_CARRYY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrLSRX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrLSRY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrASRX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrASRY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrLSLX : STD_LOGIC_VECTOR (7 downto 0);
    signal TrLSLY : STD_LOGIC_VECTOR (7 downto 0);

    signal TrBCD : STD_LOGIC_VECTOR (15 downto 0);

    signal TrN : STD_LOGIC_VECTOR (15 downto 0);
    signal TrCout : STD_LOGIC_VECTOR (15 downto 0);
    signal TrV : STD_LOGIC_VECTOR (15 downto 0);
    signal TrZ : STD_LOGIC_VECTOR (15 downto 0);
    signal TrBinPal : STD_LOGIC_VECTOR (15 downto 0);
    signal TrPrime : STD_LOGIC_VECTOR (15 downto 0);
    signal TrActive : STD_LOGIC_VECTOR (15 downto 0);
-----------------------------------------------------------------

begin
    u1 : AND8 PORT MAP (I1 => A, I2 => B, O1 => TrANDX, O2 =>
TrANDY, N => TrN(0), Cout => TrCout(0),
                                      V => TrV(0), Z => TrZ(0),
X_bin_pal => TrBinPal(0), X_prime => TrPrime(0), F_active =>
TrActive(0));

    u2 : OR8 port map (I1 => A, I2 => B, O1 => TrORX, O2 =>
TrORY, N => TrN(1), Cout => TrCout(1),
                                     V => TrV(1), Z => TrZ(1),
X_bin_pal => TrBinPal(1), X_prime => TrPrime(1), F_active =>
TrActive(1));

    u3 : XOR8 port map (I1 => A, I2 => B, O1 => TrXORX, O2 =>
TrXORY, N => TrN(2), Cout => TrCout(2),
                                      V => TrV(2), Z => TrZ(2),
X_bin_pal => TrBinPal(2), X_prime => TrPrime(2), F_active =>
TrActive(2));

    u4 : XNOR8 port map (I1 => A, I2 => B, O1 => TrXNORX, O2
=> TrXNORY, N => TrN(3), Cout => TrCout(3),
                                         V => TrV(3), Z =>
TrZ(3), X_bin_pal => TrBinPal(3), X_prime => TrPrime(3),
F_active => TrActive(3));

    u5 : UADD port map (I1 => A, I2 => B, O1 => TrUADDX, O2 =>
TrUADDY, N => TrN(4), Cout => TrCout(4),
```

```
                                          V => TrV(4), Z =>
TrZ(4), X_bin_pal => TrBinPal(4), X_prime => TrPrime(4),
F_active => TrActive(4));


     u6 : SADD port map (I1 => A, I2 => B, O1 => TrSADDX, O2 =>
TrSADDY, N => TrN(5), Cout => TrCout(5),
                                          V => TrV(5), Z =>
TrZ(5), X_bin_pal => TrBinPal(5), X_prime => TrPrime(5),
F_active => TrActive(5));

     u7 : UADD_CARRY port map (I1 => A, I2 => B, Cin => Cin, O1
=> TrUADD_CARRYX, O2 => TrUADD_CARRYY, N => TrN(6), Cout =>
TrCout(6),
                                          V => TrV(6), Z =>
TrZ(6), X_bin_pal => TrBinPal(6), X_prime => TrPrime(6),
F_active => TrActive(6));

     u8 : SMUL port map (I1 => A, I2 => B, O1 => TrSMUL(7
downto 0), O2 => TrSMUL(15 downto 8), N => TrN(7), Cout =>
TrCout(7),
                                          V => TrV(7), Z =>
TrZ(7), X_bin_pal => TrBinPal(7), X_prime => TrPrime(7),
F_active => TrActive(7));

     u9 : UMUL port map (I1 => A, I2 => B, O1 => TrUMUL(7
downto 0), O2 => TrUMUL(15 downto 8), N => TrN(8), Cout =>
TrCout(8),
                                          V => TrV(8), Z =>
TrZ(8), X_bin_pal => TrBinPal(8), X_prime => TrPrime(8),
F_active => TrActive(8));

     u10 : USUB port map (I1 => A, I2 => B, O1 => TrUSUBX, O2
=> TrUSUBY, N => TrN(9), Cout => TrCout(9),
                                          V => TrV(9), Z =>
TrZ(9), X_bin_pal => TrBinPal(9), X_prime => TrPrime(9),
F_active => TrActive(9));


     u11 : RotL port map (I1 => A, I2 => B, O1 => TrRotLX, O2
=> TrRotLY, N => TrN(10), Cout => TrCout(10),
                                          V => TrV(10), Z =>
TrZ(10), X_bin_pal => TrBinPal(10), X_prime => TrPrime(10),
F_active => TrActive(10));

     u12 : RotL_CARRY port map (I1 => A, I2 => B, Cin => Cin,
O1 => TrRotL_CARRYX, O2 => TrRotL_CARRYY, N => TrN(11), Cout =>
TrCout(11),
                                          V => TrV(11), Z =>
TrZ(11), X_bin_pal => TrBinPal(11), X_prime => TrPrime(11),
F_active => TrActive(11));
```

```
     u13 : LSR port map (I1 => A, I2 => B, O1 => TrLSRX, O2 =>
TrLSRY, N => TrN(12), Cout => TrCout(12),
                                        V => TrV(12), Z =>
TrZ(12), X_bin_pal => TrBinPal(12), X_prime => TrPrime(12),
F_active => TrActive(12));

     u14 : ASR port map (I1 => A, I2 => B, O1 => TrASRX, O2 =>
TrASRY, N => TrN(13), Cout => TrCout(13),
                                        V => TrV(13), Z =>
TrZ(13), X_bin_pal => TrBinPal(13), X_prime => TrPrime(13),
F_active => TrActive(13));

     u15 : LSL port map (I1 => A, I2 => B, O1 => TrLSLX, O2 =>
TrLSLY, N => TrN(14), Cout => TrCout(14),
                                        V => TrV(14), Z =>
TrZ(14), X_bin_pal => TrBinPal(14), X_prime => TrPrime(14),
F_active => TrActive(14));

     u16 : BCD2BIN port map (I1 => A, I2 => B, O1 => TrBCD(7
downto 0), O2 => TrBCD(15 downto 8), N => TrN(15), Cout =>
TrCout(15),
                                        V => TrV(15), Z =>
TrZ(15), X_bin_pal => TrBinPal(15), X_prime => TrPrime(15),
F_active => TrActive(15));


     Process (OPCODE, TrANDX, TrANDY, TrORX, TrORY, TrXORX,
TrXORY, TrXNORX, TrXNORY, TrUADDX, TrUADDY, TrSADDX, TrSADDY,
TrUADD_CARRYX, TrUADD_CARRYY,
                    TrSMUL, TrUMUL, TrUSUBX, TrUSUBY, TrRotLX,
TrRotLY, TrRotL_CARRYX, TrRotL_CARRYY, TrLSRX, TrLSRY, TrASRX,
TrASRY, TrLSLX, TrLSLX,
                    TrLSLY, TrBCD, TrN, TrCout, TrV, TrZ,
TrBinPal, TrPrime, TrActive)
     begin
          if (OPCODE = "0000") then     --AND Operation
               X <= TrANDX;
               Y <= TrANDY;
               N <= TrN(0);
               Cout <= TrCout(0);
               V <= TrV(0);
               Z <= TrZ(0);
               X_bin_pal <= TrBinPal(0);
               X_prime <= TrPrime(0);
               F_active <= TrActive(0);
          elsif (OPCODE = "0001") then  --OR Operation
               X <= TrORX;
               Y <= TrORY;
               N <= TrN(1);
               Cout <= TrCout(1);
               V <= TrV(1);
```

```
                Z <= TrZ(1);
                X_bin_pal <= TrBinPal(1);
                X_prime <= TrPrime(1);
                F_active <= TrActive(1);
           elsif (OPCODE = "0010") then  --XOR Operation
                X <= TrXORX;
                Y <= TrXORY;
                N <= TrN(2);
                Cout <= TrCout(2);
                V <= TrV(2);
                Z <= TrZ(2);
                X_bin_pal <= TrBinPal(2);
                X_prime <= TrPrime(2);
                F_active <= TrActive(2);
           elsif (OPCODE = "0011") then  --XNOR Operation
                X <= TrXNORX;
                Y <= TrXNORY;
                N <= TrN(3);
                Cout <= TrCout(3);
                V <= TrV(3);
                Z <= TrZ(3);
                X_bin_pal <= TrBinPal(3);
                X_prime <= TrPrime(3);
                F_active <= TrActive(3);
           elsif (OPCODE = "0100") then  --Unsigned Addition
Operation
                X <= TrUADDX;
                Y <= TrUADDY;
                N <= TrN(4);
                Cout <= TrCout(4);
                V <= TrV(4);
                Z <= TrZ(4);
                X_bin_pal <= TrBinPal(4);
                X_prime <= TrPrime(4);
                F_active <= TrActive(4);
           elsif (OPCODE = "0101") then  --Signed Addition
Operation
                X <= TrSADDX;
                Y <= TrSADDY;
                N <= TrN(5);
                Cout <= TrCout(5);
                V <= TrV(5);
                Z <= TrZ(5);
                X_bin_pal <= TrBinPal(5);
                X_prime <= TrPrime(5);
                F_active <= TrActive(5);
           elsif (OPCODE = "0110") then  --Usigned Addition with
Cin Operation
                X <= TrUADD_CARRYX;
                Y <= TrUADD_CARRYY;
                N <= TrN(6);
                Cout <= TrCout(6);
```

```
                V <= TrV(6);
                Z <= TrZ(6);
                X_bin_pal <= TrBinPal(6);
                X_prime <= TrPrime(6);
                F_active <= TrActive(6);
           elsif (OPCODE = "0111") then  --Signed Multiplication
                X <= TrSMUL(7 downto 0);
                Y <= TrSMUL(15 downto 8);
                N <= TrN(7);
                Cout <= TrCout(7);
                V <= TrV(7);
                Z <= TrZ(7);
                X_bin_pal <= TrBinPal(7);
                X_prime <= TrPrime(7);
                F_active <= TrActive(7);
           elsif (OPCODE = "1000") then  --Usigned
Multiplication
                X <= TrUMUL(7 downto 0);
                Y <= TrUMUL(15 downto 8);
                N <= TrN(8);
                Cout <= TrCout(8);
                V <= TrV(8);
                Z <= TrZ(8);
                X_bin_pal <= TrBinPal(8);
                X_prime <= TrPrime(8);
                F_active <= TrActive(8);
           elsif (OPCODE = "1001") then  --Signed Subtraction
                X <= TrUSUBX;
                Y <= TrUSUBY;
                N <= TrN(9);
                Cout <= TrCout(9);
                V <= TrV(9);
                Z <= TrZ(9);
                X_bin_pal <= TrBinPal(9);
                X_prime <= TrPrime(9);
                F_active <= TrActive(9);
           elsif (OPCODE = "1010") then  --Rotation Left
                X <= TrRotLX;
                Y <= TrRotLY;
                N <= TrN(10);
                Cout <= TrCout(10);
                V <= TrV(10);
                Z <= TrZ(10);
                X_bin_pal <= TrBinPal(10);
                X_prime <= TrPrime(10);
                F_active <= TrActive(10);
           elsif (OPCODE = "1011") then  --Rotation Left with
Carry
                X <= TrRotL_CARRYX;
                Y <= TrRotL_CARRYY;
                N <= TrN(11);
                Cout <= TrCout(11);
```

```
                V <= TrV(11);
                Z <= TrZ(11);
                X_bin_pal <= TrBinPal(11);
                X_prime <= TrPrime(11);
                F_active <= TrActive(11);
           elsif (OPCODE = "1100") then  --Logic Shift Right
                X <= TrLSRX;
                Y <= TrLSRY;
                N <= TrN(12);
                Cout <= TrCout(12);
                V <= TrV(12);
                Z <= TrZ(12);
                X_bin_pal <= TrBinPal(12);
                X_prime <= TrPrime(12);
                F_active <= TrActive(12);
           elsif (OPCODE = "1101") then  --Arithmatic Shift
Right
                X <= TrASRX;
                Y <= TrASRY;
                N <= TrN(13);
                Cout <= TrCout(13);
                V <= TrV(13);
                Z <= TrZ(13);
                X_bin_pal <= TrBinPal(13);
                X_prime <= TrPrime(13);
                F_active <= TrActive(13);
           elsif (OPCODE = "1110") then  --Logic Shift Left
                X <= TrLSLX;
                Y <= TrLSLY;
                N <= TrN(14);
                Cout <= TrCout(14);
                V <= TrV(14);
                Z <= TrZ(14);
                X_bin_pal <= TrBinPal(14);
                X_prime <= TrPrime(14);
                F_active <= TrActive(14);
           elsif (OPCODE = "1111") then  --BCD to Binary
Conversion
                X <= TrBCD(7 downto 0);
                Y <= TrBCD(15 downto 8);
                N <= TrN(15);
                Cout <= TrCout(15);
                V <= TrV(15);
                Z <= TrZ(15);
                X_bin_pal <= TrBinPal(15);
                X_prime <= TrPrime(15);
                F_active <= TrActive(15);
           end if;
     end Process;

end Structral;
```

کد شماره‌ی ۶: ALU Top Module

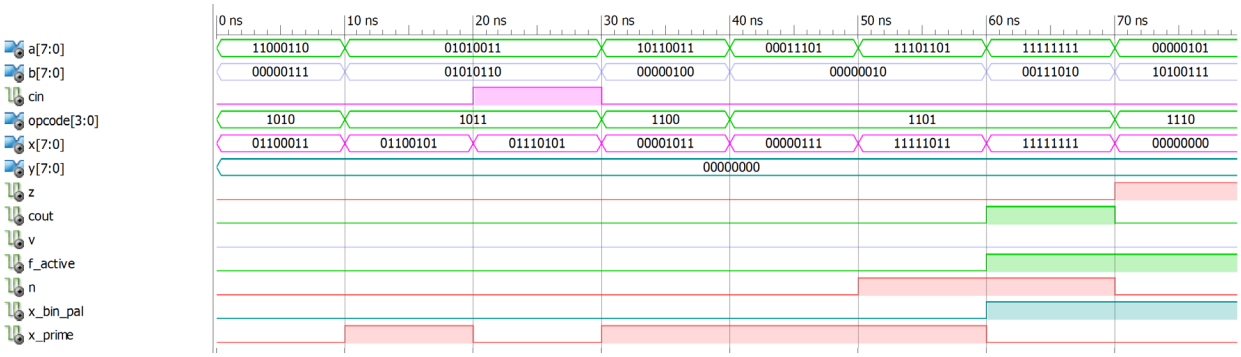
در بخش بعدی به testbench کدهای نوشته شده می‌پردازیم. در این تست فقط ۵ کد تغییر داده شده را تست می‌کنیم. همچنین به کلاک نیازی نداریم پس آن را حذف می‌کنیم.

```
--------------------------------------------------------------------
-- Group Number: 10
-- Members: Sina Rabiee
--             Mohammad Mahdi Rnjbar Bafghi
--                  Alireza Faghih Ali abadi
--                  Zaker Ghelichi
--                  Bardia Sahami
--------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

ENTITY Tb_Bonus IS
END Tb_Bonus;

ARCHITECTURE behavior OF Tb_Bonus IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT ALU
    PORT(
         A : IN  std_logic_vector(7 downto 0);
         B : IN  std_logic_vector(7 downto 0);
         Cin : IN  std_logic;
         OPCODE : IN  std_logic_vector(3 downto 0);
         X : OUT  std_logic_vector(7 downto 0);
         Y : OUT  std_logic_vector(7 downto 0);
         Z : INOUT  std_logic;
         Cout : INOUT  std_logic;
         V : INOUT  std_logic;
         F_active : OUT  std_logic;
         X_bin_pal : OUT  std_logic;
         X_prime : OUT  std_logic;
         N : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal A : std_logic_vector(7 downto 0) := (others => '0');
   signal B : std_logic_vector(7 downto 0) := (others => '0');
   signal Cin : std_logic := '0';
   signal OPCODE : std_logic_vector(3 downto 0) := (others =>
'0');

     --BiDirs
```

```
   signal Z : std_logic;
   signal Cout : std_logic;
   signal V : std_logic;

     --Outputs
   signal X : std_logic_vector(7 downto 0);
   signal Y : std_logic_vector(7 downto 0);
   signal F_active : std_logic;
   signal X_bin_pal : std_logic;
   signal X_prime : std_logic;
   signal N : std_logic;
   -- No clocks detected in port list. Replace <clock> below
with
   -- appropriate port name
     --This Curcuit is completely Combinational and doesn't
need a clock

BEGIN

     -- Instantiate the Unit Under Test (UUT)
   uut: ALU PORT MAP (
          A => A,
          B => B,
          Cin => Cin,
          OPCODE => OPCODE,
          X => X,
          Y => Y,
          Z => Z,
          Cout => Cout,
          V => V,
          F_active => F_active,
          X_bin_pal => X_bin_pal,
          X_prime => X_prime,
          N => N
        );

   -- Stimulus process
   stim_proc: process
   begin
          OPCODE <= "1010";   --Rotation Left
          Cin <= '0';
          A <= "11000110";
          B <= "00000111";
      wait for 10 ns;

          OPCODE <= "1011";   --Rotation Left with Cin
Operation
          Cin <= '0';
          A <= "01010011";
          B <= "01010110";
      wait for 10 ns;
          Cin <= '1';
```

```
        A <= "01010011";
        B <= "01010110";
    wait for 10 ns;

        OPCODE <= "1100";        --Logic Shift Right
        Cin <= '0';
        A <= "10110011";
        B <= "00000100";
    wait for 10 ns;

        OPCODE <= "1101";        --Arithmatic Shift Right
        Cin <= '0';
        A <= "00011101";
        B <= "00000010";
    wait for 10 ns;
        Cin <= '0';
        A <= "11101101";
        B <= "00000010";
    wait for 10 ns;
        A <= "11111111";
        B <= "00111010";
    wait for 10 ns;

        OPCODE <= "1110";        --Logic Shift Left
        Cin <= '0';
        A <= "00000101";
        B <= "10100111";

    wait;
  end process;

END;
```

کد شماره‌ی ۷: Testbench



در ۱۰ نانو ثانیه اول ورودی اول (a=11000110) و ورودی دوم (b=0000111) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۰۱۰ می‌باشد (RotL). در این OPCODE به اندازه باقی‌مانده b بر ۸ ، ورودی a به سمت چپ چرخانده می‌شود. در مبنای ۱۰ داریم:

$$b \bmod 8 = 7 \bmod 8 = 7$$

اگر عملیات را بصورت دستی انجام دهیم مشاهده می‌کنیم که در نهایت خروجی x=01100011 بدست می‌آید.

خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. مقدار V نیز طبق پیش‌بینی برابر ۰ شده است. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد. همچنین Cout نداریم پس مقدار آن صفر است.

خروجی را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۹۹ می‌باشد که عددی اول نیست پس x_prime مقدار ۰ را می‌گیرد. خروجی بی‌علامت است پس n مقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که آن صفر می‌شود.

در ۱۰ نانو ثانیه دوم ورودی اول (a=01010011) و ورودی دوم (b=01010110) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۰۱۱ می‌باشد (RotL_CARRY). در این OPCODE عمل چرخش به چپ روی رشته بیت حاصل از قرار دادن Cin در سمت چپ ورودی a به اندازه باقی‌مانده b بر ۸ دفعه انجام می‌شود. در اینجا چون carry ورودی صفر است، رشته‌ای که مورد چرخش قرار می‌گیرد Cin & x=001010011 است. ورودی دوم در مبنای ۱۰ برابر ۸۶ و باقی‌مانده آن بر ۸ برابر ۶ است. در نهایت خروجی x=01100101 بدست می‌آید و مقدار ۰ که از رشته بیرون افتاده در Cout قرار می‌گیرد.

خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. مقدار V نیز طبق پیش‌بینی برابر ۰ شده است. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد.

اگر خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۱۰۱ می‌باشد که عددی اول است پس x_prime مقدار ۱ را می‌گیرد. خروجی بی‌علامت است پس n مقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که در اینجا همگی آن‌ها صفرند پس مقدار خود آن نیز صفر می‌شود.

در ۱۰ نانو ثانیه سوم ورودی‌ها و OPCODE تغییری نکرده، و فقط Cin یک شده. پس رشته‌ای که مورد ۶ مرحله چرخش به چپ قرار می‌گیرد Cin & x=101010011 است. در نهایت خروجی x=01110101 بدست می‌آید و مقدار ۰ که از رشته بیرون افتاده در Cout قرار می‌گیرد.

خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. مقدار V نیز طبق پیش‌بینی برابر ۰ شده است. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد.

اگر خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۱۱۷ می‌باشد که عددی اول نیست پس x_prime مقدار ۰ را می‌گیرد. خروجی بی‌علامت است پس n مقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که در اینجا همگی آن‌ها صفرند پس مقدار خود آن نیز صفر می‌شود.

در ۱۰ نانو ثانیه چهارم ورودی اول (a=10110011) و ورودی دوم (b=00000100) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۱۰۰ می‌باشد (LSR). در این OPCODE رشته بیت a به اندازه b بیت به راست شیفت داده می‌شود و به ازای هر شیفت یک صفر از سمت چپ به a تزریق می‌شود. بیت آخر خارج شده از سمت راست در Cout قرار می‌گیرد. مشاهده می‌کنیم که در نهایت خروجی x=00001011 بدست می‌آید و Cout = 0.

خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. مقدار V نیز طبق پیش‌بینی برابر ۰ شده است. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد.

اگر خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۱۱ می‌باشد که عددی اول است پس x_prime مقدار ۱ را می‌گیرد. خروجی بی‌علامت است پس n مقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که صفر می‌شود.

در ۱۰ نانو ثانیه پنجم ورودی اول (a=00011101) و ورودی دوم (b=00000010) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۱۰۱ می‌باشد (ASR). در این OPCODE رشته بیت a به اندازه b بیت به راست شیفت داده می‌شود و به ازای هر شیفت بیت اخر از سمت راست در Cout قرار می‌گیرد و بیت علامت رشته ورودی b+۱ تکرار می‌شود. در انتها حاصل شیفت با بیت bام ورودی جمع می‌شود. b در مبنای ۱۰ برابر ۲ می‌باشد. می‌بینیم که خروجی شیفت برابر 00000111 می‌باشد و اگر بیت دوم ورودی را به آن اضافه کنیم (صفر) خروجی x=00000111 بدست می‌آید و Cout = 0.

خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. مقدار V نیز طبق پیش‌بینی برابر ۰ شده است. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد.

اگر خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۷ می‌باشد که عددی اول است پس x_prime مقدار ۱ را می‌گیرد. خروجی مثبت است پس n مقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که مقدار آن صفر می‌شود.

در ۱۰ نانو ثانیه ششم ورودی اول (a=11101101) و ورودی دوم (b=00000010) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۱۰۱ می‌باشد (ASR). در این OPCODE رشته بیت a به اندازه b بیت به راست شیفت داده می‌شود و به ازای هر شیفت بیت اخر از سمت راست در Cout قرار می‌گیرد و بیت علامت رشته ورودی b+۱ تکرار می‌شود. در انتها حاصل شیفت با بیت bام ورودی جمع می‌شود. b در مبنای ۱۰ برابر ۲ می‌باشد. می‌بینیم که خروجی شیفت برابر 11111011 می‌باشد و اگر بیت دوم ورودی را به آن اضافه کنیم (صفر) خروجی x=11111011 بدست می‌آید و Cout = 0. مشاهده می‌شود که ورودی در قرارداد مکمل ۲ برابر ۱۹- است و خروجی شیفت برابر ۵- است.

خروجی برابر "x"00 نیست پس z برابر ۰ می‌شود. مقدار V نیز طبق پیش‌بینی برابر ۰ شده است. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد.

اگر خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۲۵۱ می‌باشد که عددی اول است پس x_prime مقدار ۱ را می‌گیرد. خروجی منفی است پس n مقدار ۱ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که مقدار آن یک می‌شود.

در ۱۰ نانو ثانیه هفتم ورودی اول (a=11111111) و ورودی دوم (b=00111010) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۱۰۱ می‌باشد (ASR). در این OPCODE رشته بیت a به اندازه b بیت به راست شیفت داده می‌شود و به ازای هر شیفت بیت اخر از سمت راست در Cout قرار می‌گیرد و بیت علامت رشته ورودی b+۱ تکرار می‌شود. در انتها حاصل شیفت با بیت bام ورودی جمع می‌شود. b در مبنای ۱۰ برابر ۵۸ می‌باشد. می‌بینیم که خروجی شیفت برابر 11111111 می‌باشد و چون خروجی بیت ۵۸ام ندارد ، عمل Rounding انجام نمی‌شود و خروجی x=11111111 بدست می‌آید و Cout = 1.

خروجی برابر "x"00 نیست پس z برابر ۰ می‌شود. مقدار V نیز طبق پیش‌بینی برابر ۰ شده است. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد.

اگر خروجی X را اگر برعکس کنیم با خودش برابر می‌شود پس x_bin_pal یک می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۲۵۵ می‌باشد که عددی اول نیست پس x_prime مقدار ۰ را می‌گیرد. خروجی منفی است پس n مقدار ۱ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که مقدار آن یک می‌شود.

در ۱۰ نانو ثانیه هشتم ورودی اول (a=00000101) و ورودی دوم (b=10100111) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۱۱۰ می‌باشد (LSL). در این OPCODE رشته بیت a به اندازه b واحد به چپ شیفت داده می‌شود و به ازای هر شیفت یک صفر از سمت راست به a تزریق می‌شود. بیت خارج شده از سمت چپ در Cout قرار می‌گیرد. مشاهده می‌کنیم که چون مقدار b در مبنای ۱۰ بزرگ‌تر از ۸ است ُ خروجی کاملا صفر می‌باشد و Cout = 0.

خروجی برابر "x"00 است پس z برابر ۱ می‌شود. مقدار V نیز طبق پیش‌بینی برابر ۰ شده است. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد.

اگر خروجی X را اگر برعکس کنیم با خودش برابر می‌شود پس x_bin_pal یک می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۰ می‌باشد که عددی اول نیست پس x_prime مقدار ۰ را می‌گیرد. خروجی بی‌علامت است پس nمقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید پس مقدار آن یک می‌شود.