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:

### Task 1)

در قسمت‌های بعدی کدهای مربوط به هرکدام از زیرماژول‌ها که با ورودی OPCODE مشخص می‌شوند آورده شده.

| OPCODE | OPERATION | FORMULA | Z | Cout | V | X_bin_pal | X_prime | N |
|---|---|---|---|---|---|---|---|---|
| 0000 | AND | X = A AND B | ↕ | – | – | ↕ | ↕ | 0 |
| 0001 | OR | X = A OR B | ↕ | – | – | ↕ | ↕ | 0 |
| 0010 | XOR | X = A XOR B | ↕ | – | – | ↕ | ↕ | 0 |
| 0011 | XNOR | X = A XNOR B | ↕ | – | – | ↕ | ↕ | 0 |
| 0100 | Unsigned Addition | (Cout:X) = A + B | ↕ | ↕ | – | ↕ | ↕ | 0 |
| 0101 | Signed Addition | X = A + B | ↕ | – | ↕ | ↕ | ↕ | ↕ |
| 0110 | Unsigned Addition with Carry | (Cout:X) = A + B + Cin | ↕ | ↕ | – | ↕ | ↕ | 0 |
| 0111 | Signed Multiplication | (Y:X) = A * B | ↕ | – | – | ↕ | ↕ | ↕ |
| 1000 | Unsigned Multiplication | (Y:X) = A * B | ↕ | – | – | ↕ | ↕ | 0 |
| 1001 | Unsigned Subtraction | X = A - B | ↕ | ↕ | – | ↕ | ↕ | 0 |
| 1010 | Rotation Left | X = A <<< 1 | ↕ | – | – | ↕ | ↕ | 0 |
| 1011 | Rotation Left with Carry | (Cout:X) = (Cin:A) <<< 1 | ↕ | ↕ | – | ↕ | ↕ | 0 |
| 1100 | Logic Shift Right | X = A >> 1 | ↕ | ↕ | – | ↕ | ↕ | 0 |
| 1101 | Arithmetic Shift Right | X = A >> 1 | ↕ | ↕ | – | ↕ | ↕ | ↕ |
| 1110 | Logic Shift Left | X = A << 1 | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ |
| 1111 | BCD to Binary Conversion | (Y:X) = BCD2BIN(B:A) | ↕ | ↕ | – | ↕ | ↕ | – |

کد برخی خروجی‌ها برای تمامی زیر ماژول‌ها یکی می‌باشد پس بهتر است ابتدا به آن‌ها بپردازیم:

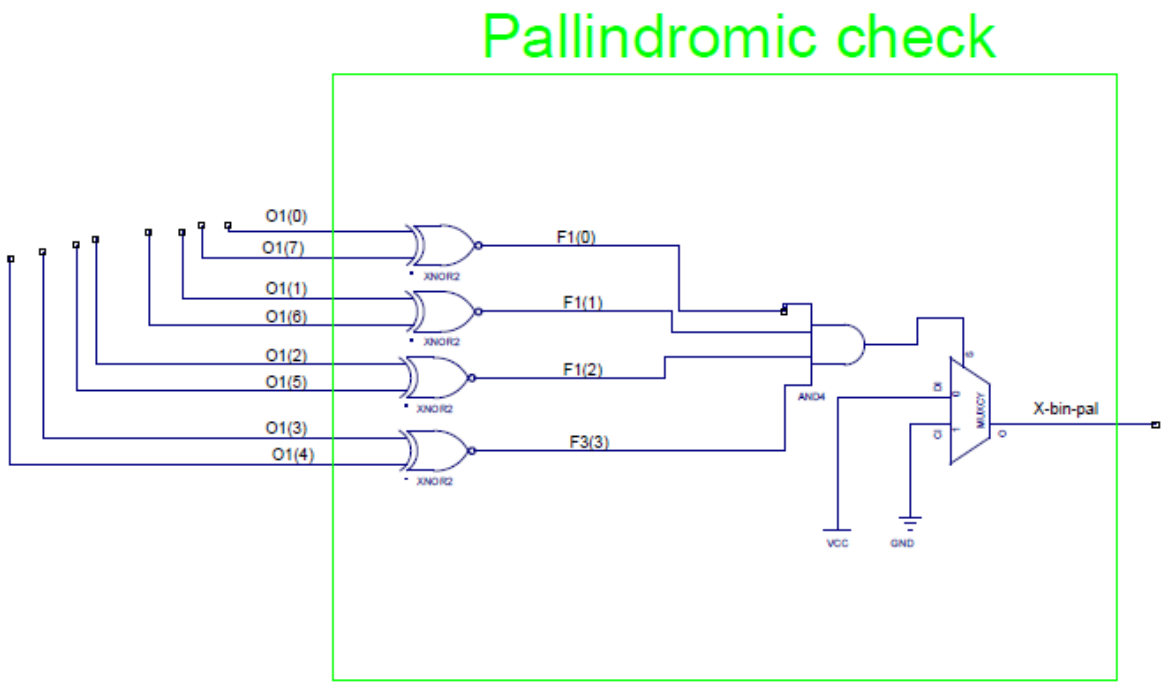
### X_bin_pal:

این قسمت چک می‌کند که آیا خروجی X به این صورت هست که اگر آن را برعکس کنیم با مقدار قبلی خود برابر باشد (خروجی ۱) یا خیر (خروجی ۰). کد آن به صورت زیر خواهد بود:

```
signal F: STD_LOGIC_VECTOR (3 downto 0);

---------------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';
-----------------------------------------------
```

*کد شماره‌ی ۱: چک کردن قرینه‌خوانی خروجی X*

در این زیرماژول‌ها، هر مولفه‌ی سیگنال F، حاصل XNOR دو مولفه خروجی ماژول می‌باشد. دلیل این کار این است که اگر این دو مولفه برابر باشند، حاصل XNOR آن‌ها برابر ۱ می‌شود و در غیر اینصورت برابر ۰ می‌شود. با چک کردن سیگنال F می‌توان گفت که اگر برابر ۱۱۱۱ باینری بود، خروجی ماژول متقارن است. به این ترتیب مقدار خروجی X_bin_pal مشخص می‌شود.




*شکل ۱: بلوک دیاگرام X_bin_pal*

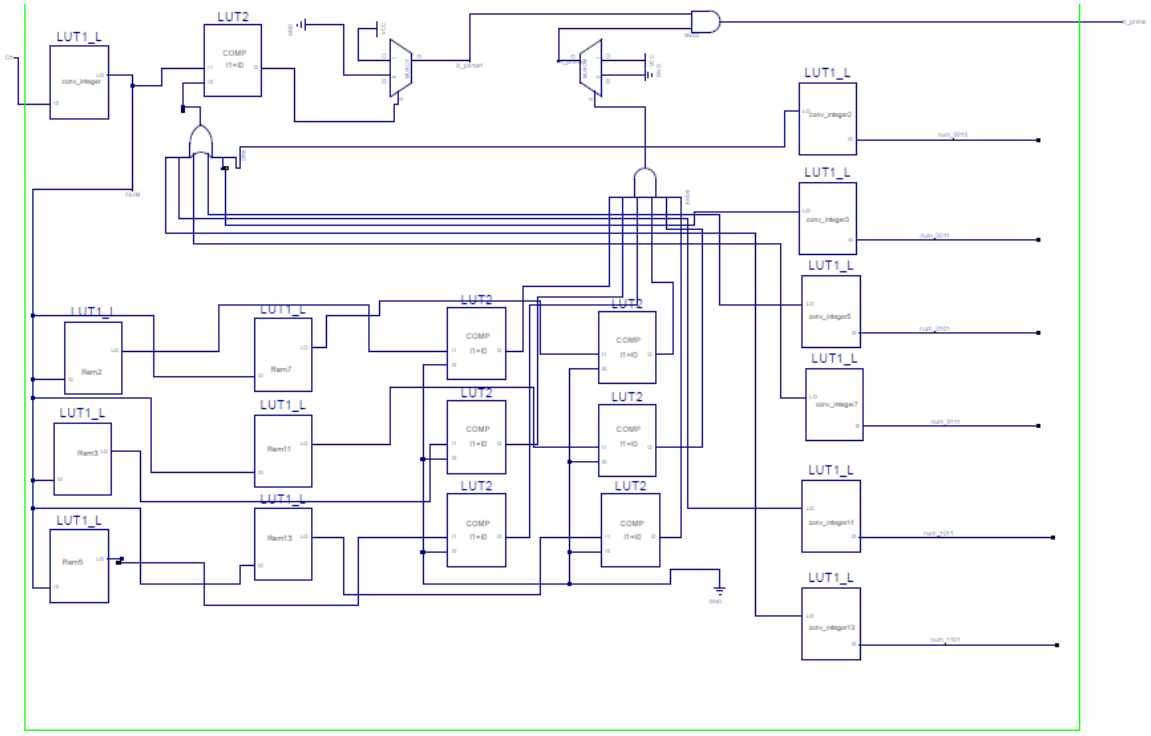
**X_prime:**

این کد چک می‌کند آیا خروجی X عددی مرکب است یا نه و درصورت اول بودن مقدار ۱ را به عنوان خورجی تحویل می‌دهد. کد آن به صورت زیر خواهد بود:

```
------------------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;
-----------------------------------------------
```

*کد شماره‌ی ۲: اول بودن خروجی X*

برای تعیین خروجی X_prime از تابع conv_integer استفاده می‌کنیم. به کمک این تابع، خروجی ماژول را از باینری به ده‌دهی (Decimal) تبدیل می‌کنیم. ابتدا بررسی می‌کنیم که عدد برابر ۰ یا ۱ است یا نه و اگر باشد اول نخواهد بود. عدد ما ۸ بیتی است یعنی حداکثر عدد ۲۵۵ در مبنای ۱۰ را نشان می‌دهد برای چک کردن اول بودن باید باقی مانده تقسیم خروجی بر اعداد اول تا ۱۳ را بررسی کنیم و بیشتر از ان کار بیهوده ای خواهد بود (با توجه به روابط ریاضی برای چک کردن اول بودن یک عدد کافی است آن را بر اعداد اول کوچک‌تر از جذر آن تقسیم کنیم). بنابراین بعد از آن‌که چک کردیم ۰ یا ۱ نباشد چک می‌کنیم اعداد اول ۲ تا ۱۳ نباشد بعد از آن می‌گوییم اگر باقی‌مانده آن به یکی از اعداد اول ۲ تا ۱۳ صفر باشد اول نخواهد بود.




شکل ۲: بلوک دیاگرام X_prime

### F_active:

این تابع طبق متن پروژه از OR کردن Z و Cout و V بدست می‌آید.

```
F_active <= Z or V or Cout;
```

کد شماره‌ی ۳: خروجی F_active

حال به بررسی زیرماژول‌ها می‌پردازیم.

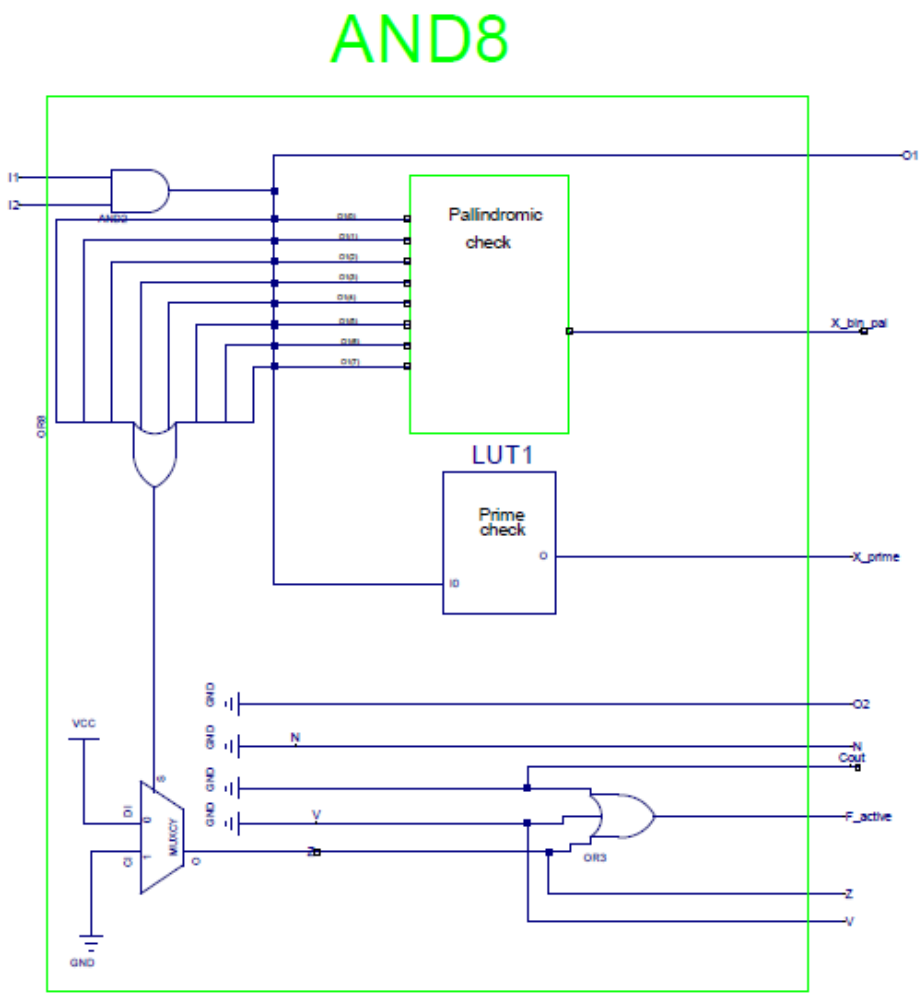
### OPCODE 0000 & 0001 & 0010 & 0011 (AND, OR, XOR, XNOR):

در ۴ زیرماژول اول، همانطور که در توضیحات پروژه ذکر شده پایه‌های Cout و V تغییری نمی‌کنند و در مدار ترکیبی Task 1 مقدار آن‌ها برابر ۰ است. و مقدار N نیز در این چهار OPCODE برابر صفر است.

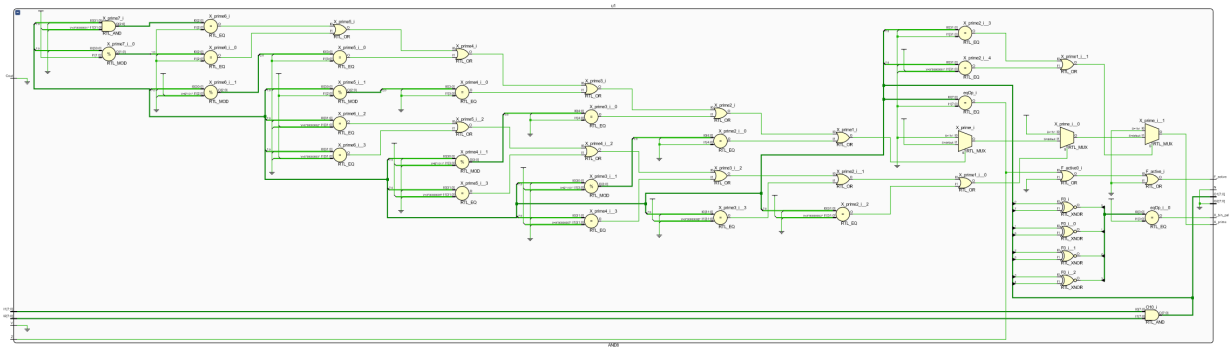
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity 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 AND8;

architecture Behavioral of AND8 is

     signal F: STD_LOGIC_VECTOR (3 downto 0);

begin
     O1 <= I1 and I2;
     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 ۴: ماژول AND




شکل ۳: بلوک دیاگرام ماژول AND



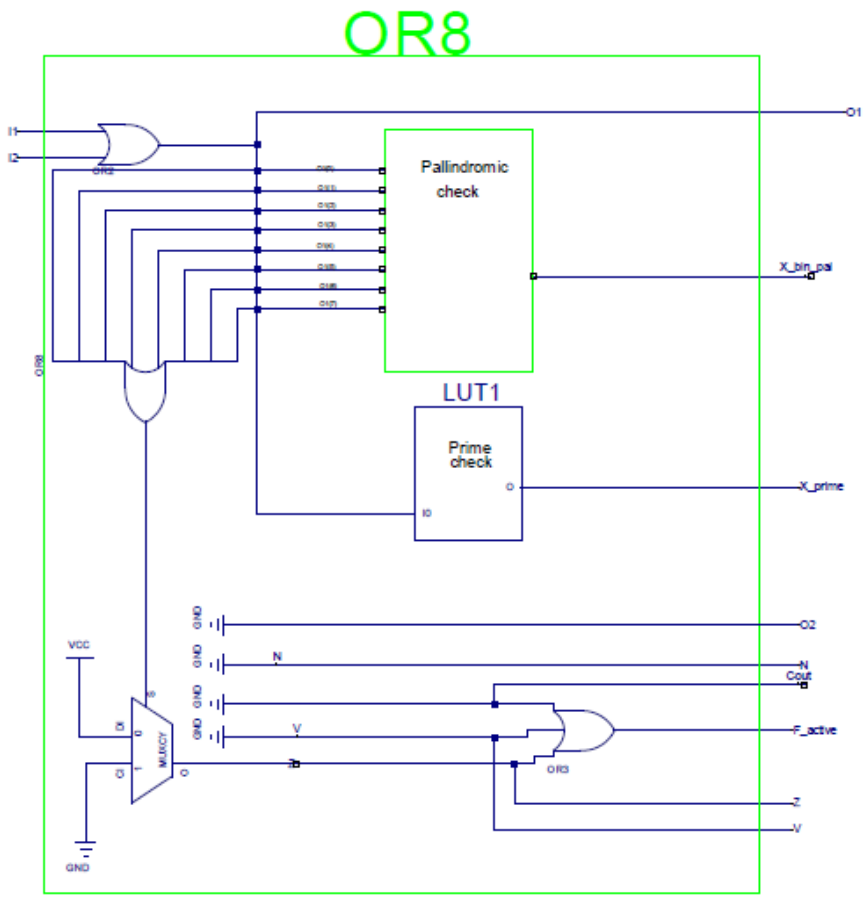
شکل ۴: بلوک دیاگرام ماژول AND (خروجی نرم‌افزار VIVADO)

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

entity OR8 is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           I2 : in  STD_LOGIC_VECTOR (7 downto 0);
```

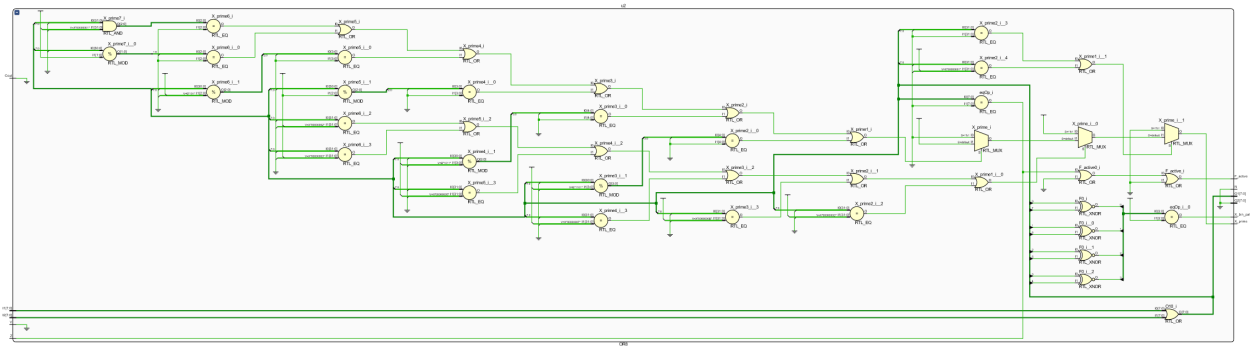
```
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
                 O2 : out  STD_LOGIC_VECTOR (7 downto 0);
                 N : out STD_LOGIC;
                 Cout : inout STD_LOGIC;
                 V : inout STD_LOGIC;
                 Z : inout STD_LOGIC;
                 X_bin_pal : out STD_LOGIC;
                 X_prime : out STD_LOGIC;
                 F_active : out STD_LOGIC);
    end OR8;

architecture Behavioral of OR8 is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
begin
     O1 <= I1 or I2;
     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 ۵: ماژول OR




شکل ۵: بلوک دیاگرام ماژول OR



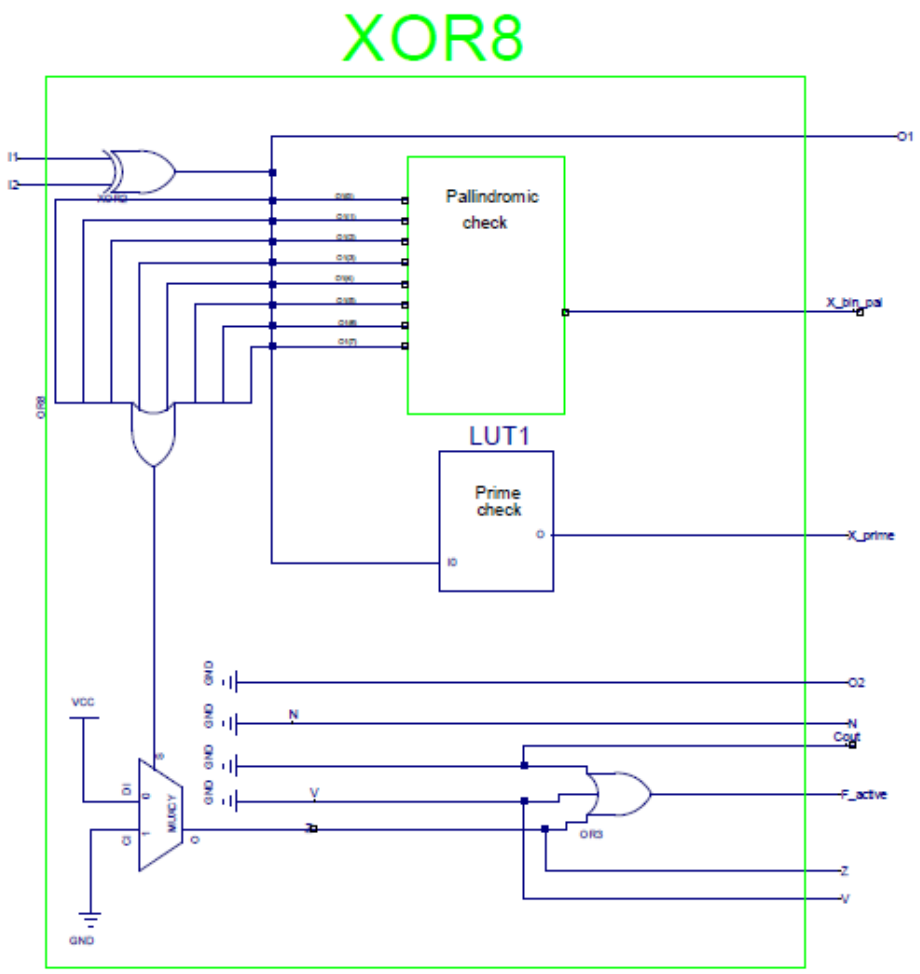
شکل ۶: بلوک دیاگرام ماژول OR (خروجی نرم‌افزار VIVADO)

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

entity XOR8 is
```

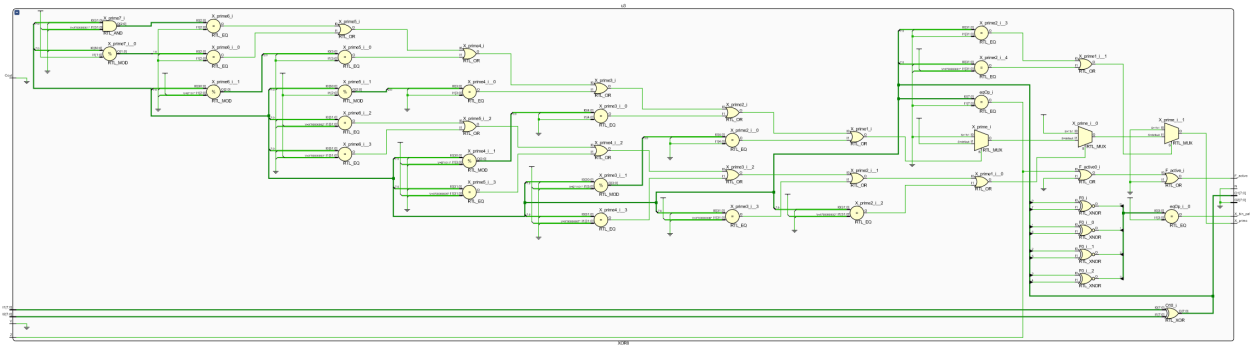
```
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           I2 : in  STD_LOGIC_VECTOR (7 downto 0);
           O1 : inout  STD_LOGIC_VECTOR (7 downto 0);
                O2 : out  STD_LOGIC_VECTOR (7 downto 0);
                N : out STD_LOGIC;
                Cout : inout STD_LOGIC;
                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 XOR8;

architecture Behavioral of XOR8 is

     signal F: STD_LOGIC_VECTOR (3 downto 0);

begin
     O1 <= I1 xor I2;
     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 ۶: ماژول XOR




شکل ۷: بلوک دیاگرام ماژول XOR



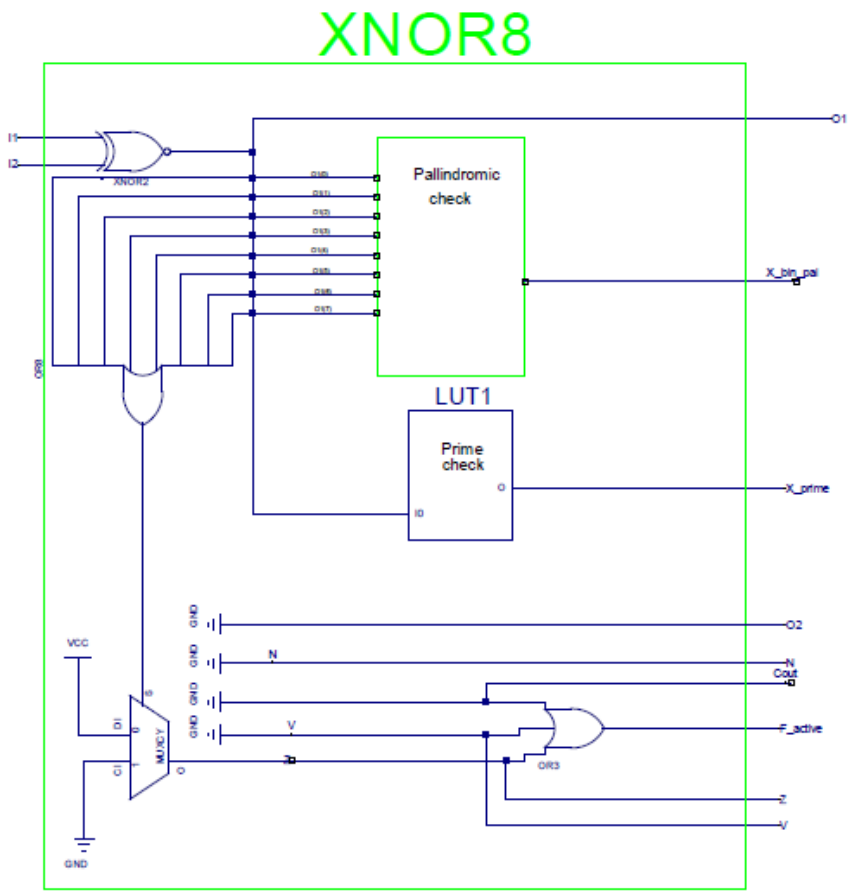
شکل ۸: بلوک دیاگرام ماژول XOR (خروجی نرم‌افزار VIVADO)

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

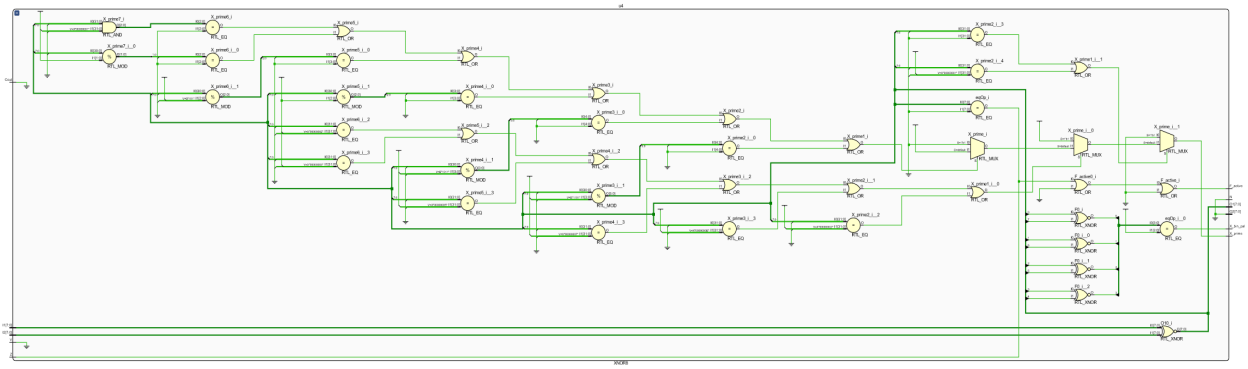
```
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity 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 XNOR8;

architecture Behavioral of XNOR8 is

     signal F: STD_LOGIC_VECTOR (3 downto 0);

begin
     O1 <= I1 xnor I2;
     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 ۷: ماژول XNOR




شکل ۹: بلوک دیاگرام ماژول XNOR



شکل ۱۰: بلوک دیاگرام ماژول XNOR (خروجی نرم‌افزار VIVADO)

در ۴ زیرماژول اول دستورات بسیار ساده‌اند و شامل دستورات ساده‌ای همچون AND و OR و XOR و XNOR هستند که در خروجی O1 زیرماژول ریخته می‌شود.

خروجی برای محاسبه Z همان O1 می‌باشد. چون خروجی ۸ بیتی است پس صفر را می‌توان به صورت x"00" (در مبنای HEX) نشان داد که همان 00000000 در مبنای ۲ است.

چون از خروجی دوم (O2) استفاده نمی‌کنیم مقدار ان را صفر (x"00") قرار می‌دهیم (طبق توضیحات پروژه در مدار ترکیبی Task1 باید تمامی خروجی‌ها به طور کامل تعیین شوند و اگر از برخی خروجی‌ها استفاده نمی‌شود، مقدار آن‌ها برابر صفر باشد).

**OPCODE 0100 (Unsigned Addition):**

مقدار N و V در اینجا نیز صفر است اما Cout ممکن است صفر نباشد. به کمک پکیج unsigned کار ما بسیار راحت می‌شود و تنها کافیست که دو ورودی را با هم جمع کنیم.

اما باید توجه کنیم که جمع دو عدد ۸ بیتی ممکن است ۹ بیت شود پس متغیر S (متغیر واسطه که بطور signal تعریف شده) را ۹ بیتی تعریف می‌کنیم.

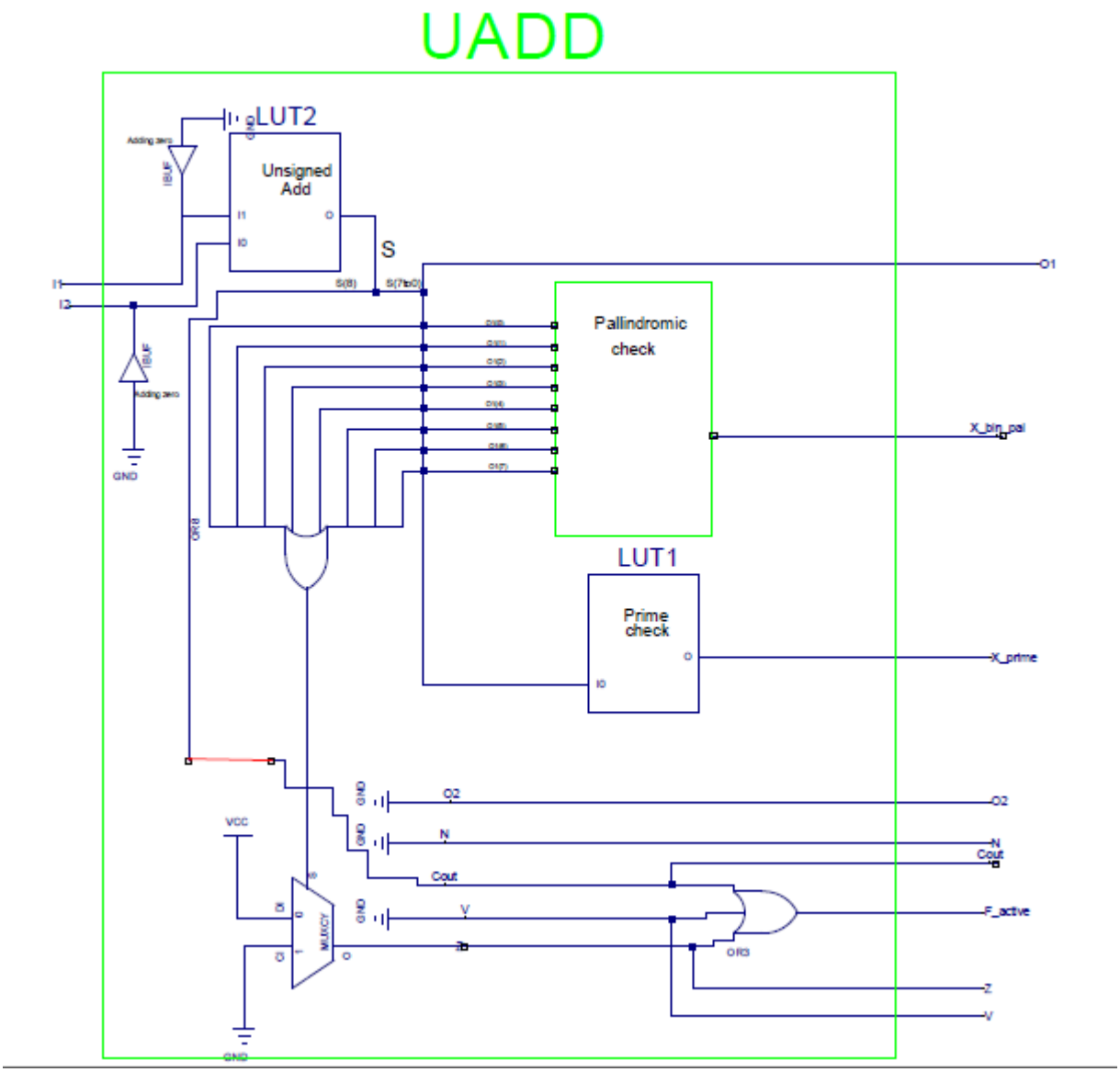
حال برای اینکه ورودی‌ها نیز ۹ بیتی شوند به کمک دستور & یک صفر('0') به ابتدای آن‌ها اضافه می‌کنیم (sign extension). ۸ بیت اول (از سمت چپ) S را به عنوان خروجی در O1 قرار می‌دهیم و بیت اخر S را به عنوان carry در Cout می‌ریزیم.

همانطور که پیداست متغیر خروجی برای محاسبه Z همان S می‌باشد. چون S، ۹ بیتی است عدد صفر را به صورت O"000" (مبنای Octa) نمایش می‌دهیم. چون از خروجی دوم (O2) استفاده نکردیم پس مقدار آن را صفر(X"00") قرار می‌دهیم.

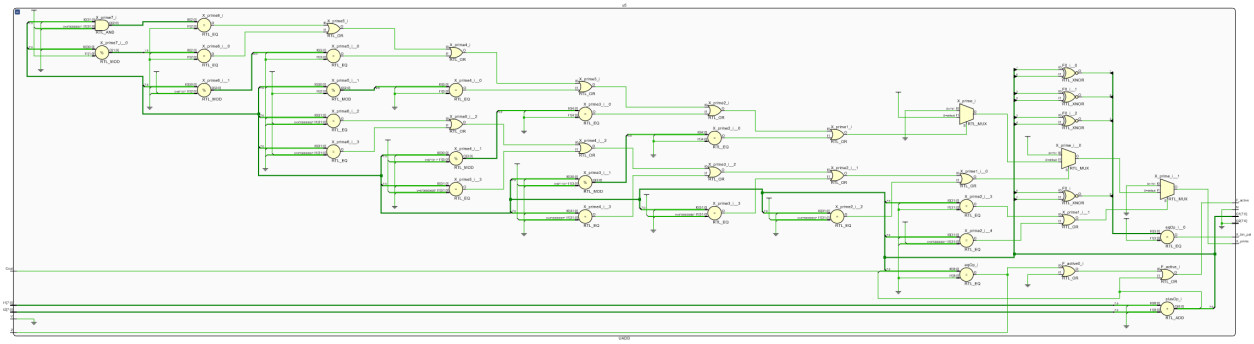
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity 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 UADD;
```

```
architecture Behavioral of UADD is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
     signal S: STD_LOGIC_VECTOR (8 downto 0);

begin
     S <= ('0' & I1) + ('0' & I2);
     O1 <= S(7 downto 0);
     O2 <= x"00";
     Cout <= S(8);
     N <= '0';
     V <= '0';
     Z <= '1' when S=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 ۸: ماژول Unsigned Addition




شکل ۱۱: بلوک دیاگرام ماژول Unsigned Addition



شکل ۱۲: بلوک دیاگرام ماژول Unsigned Addition (خروجی نرم‌افزار VIVADO)

## OPCODE 0101 (Signed Addition):

در اینجا نیز پکیج unsigned به کمک ما می‌آید. همچنین برای استفاده از توابعی مانند signed و conv_integer و ... باید از پکیج NUMERIC استفاده کرد.

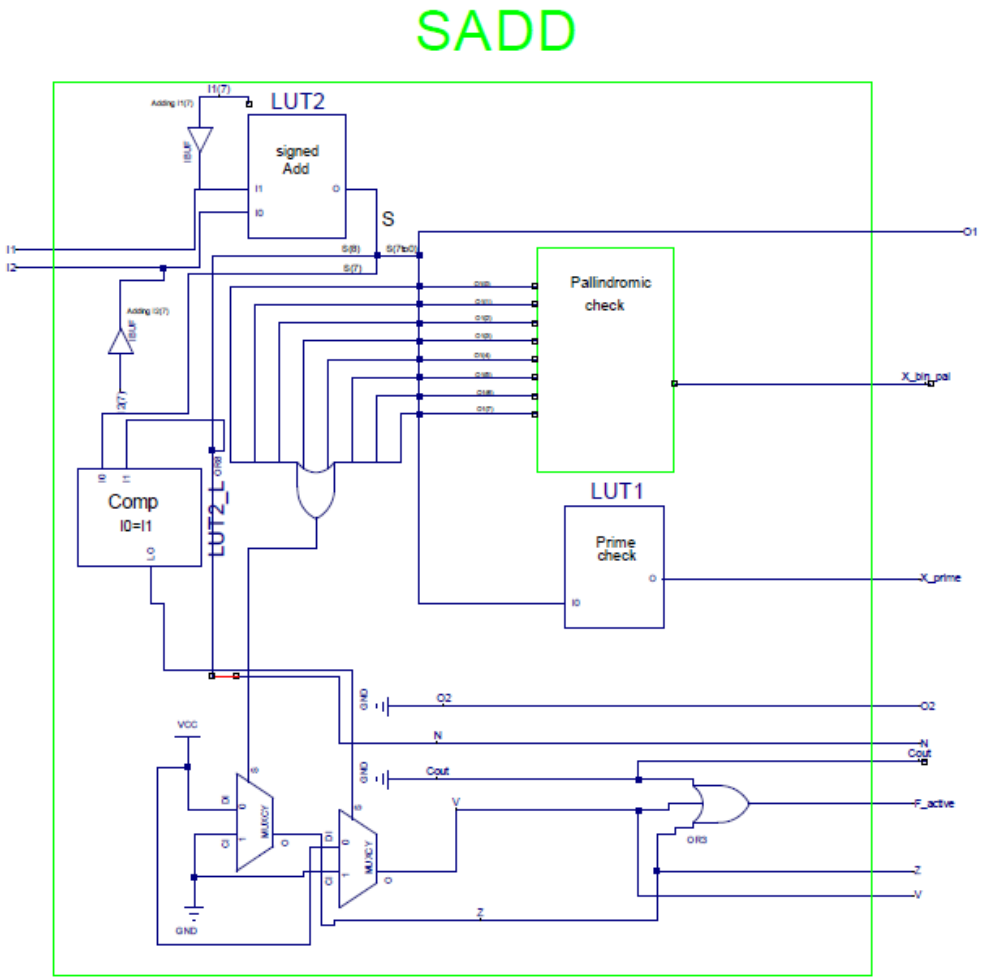
برای جمع مانند زیرماژول قبلی عمل می‌کنیم و با sign extension دو ورودی ۹ بیتی را با هم جمع می‌کنیم در متغیر SS که آن هم ۹ بیتی و علامت‌دار است می‌ریزیم (ورودی‌های اصلی را با تابع signed به عنوان باینری‌های علامت‌دار دریافت می‌کنیم). در اخر متغیر SS را بصورت STD LOGIC در S قرار می‌دهیم. ۸ بیت اول (از سمت چپ) S را بعنوان حاصل جمع در O1 می‌ریزیم.

برای تعیین V (overflow) به دو بیت ۸ و ۹ متغیر SS نگاه می‌کنیم اگر برابر نبودند V برابر ۱ می‌شود و اگر برابر بودند V برابر صفر می‌شود. مقدار N از روی علامت خروجی که بیت نهم S است مشخص می‌شود و مقدار Cout نیز صفر است. برای محاسبه Z خروجی همان S می‌باشد. از خروجی دوم استفاده نکردیم پس مقدار ان را صفر می‌گذاریم.

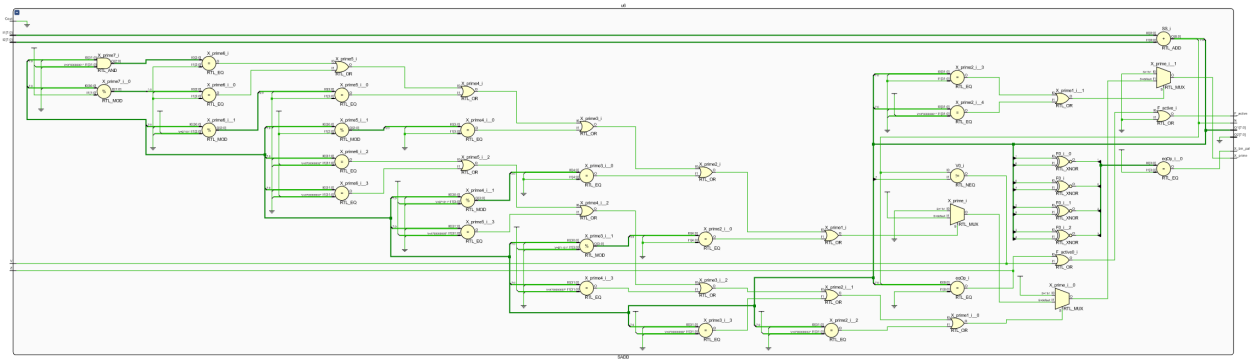
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity 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 SADD;

architecture Behavioral of SADD is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
     signal SS: SIGNED (8 downto 0);
     signal S: STD_LOGIC_VECTOR (8 downto 0);

begin
     SS <= SIGNED(I1(7) & I1) + SIGNED(I2(7) & I2);
     S <= STD_LOGIC_VECTOR(SS);
     O1 <= S(7 downto 0);
     O2 <= x"00";
     Cout <= '0';

     V <= '1' when (SS(8) /= SS(7)) else      --OverFlow
Detection
            '0';

     N <= S(8);
     Z <= '1' when S=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 ۹: ماژول Signed Addition




شکل ۱۳: بلوک دیاگرام ماژول Signed Addition



شکل ۱۴: بلوک دیاگرام ماژول Signed Addition (خروجی نرم‌افزار VIVADO)

## OPCODE 0110 (Unsigned Addition with Carry):

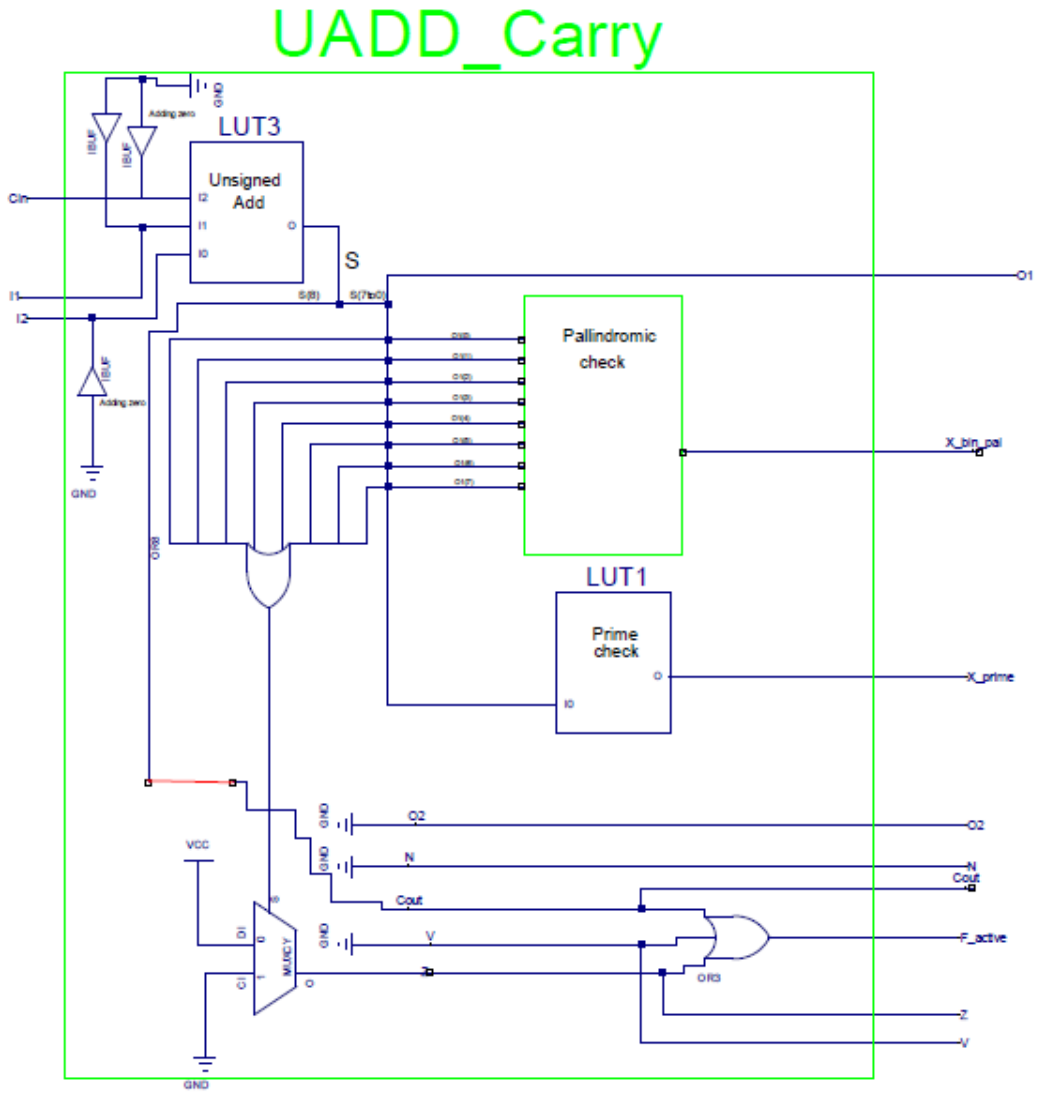
این زیرماژول شبیه زیرماژول unsigned addition است و تنها یک carry ورودی بیشتر دارد. اینجا نیز N و V صفر هستند. برای اینکه carry ورودی ۹ بیتی شود به کمک دستور & ، X”00” (00000000 در مبنای ۲) را به ابتدای Cin اضافه می‌کنیم. باقی دستورات مانند زیرماژول unsigned addition است.

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

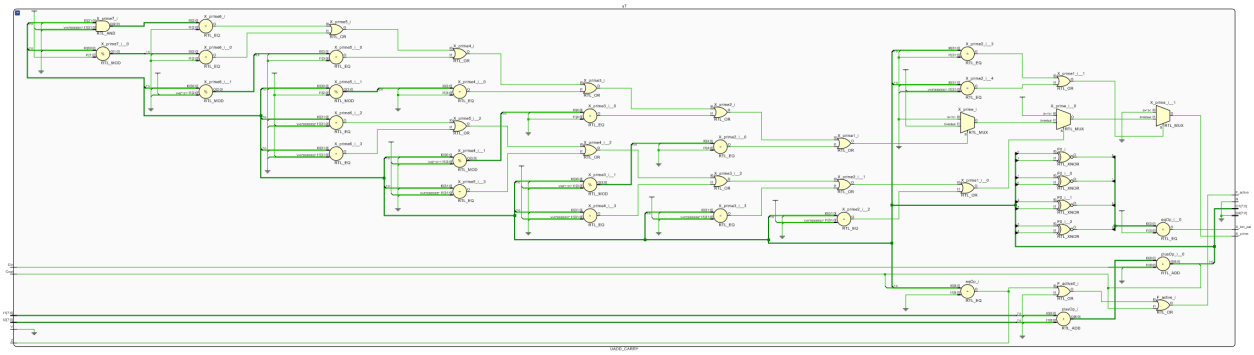
```
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity 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 UADD_CARRY;

architecture Behavioral of UADD_CARRY is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
     signal S: STD_LOGIC_VECTOR (8 downto 0);

begin
     S <= ('0' & I1) + ('0' & I2) + (x"00" & Cin);
     O1 <= S(7 downto 0);
     O2 <= x"00";
     Cout <= S(8);
     N <= '0';
     V <= '0';
     Z <= '1' when S=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 ۱۰: ماژول Unsigned Addition with Carry




شکل ۱۵: بلوک دیاگرام ماژول Unsigned Addition with Carry



شکل ۱۶: بلوک دیاگرام ماژول Unsigned Addition with Carry (خروجی نرم‌افزار VIVADO)

## OPCODE 0111 (Signed Multiplication):

در اینجا نیز Cout و V صفر هستند. با استفاده از پکیج unsigned به راحتی دو ورودی را در هم ضرب می‌کنیم. برای استفاده از توابعی مانند signed و conv_integer و ... باید از پکیج NUMERIC استفاده کرد.

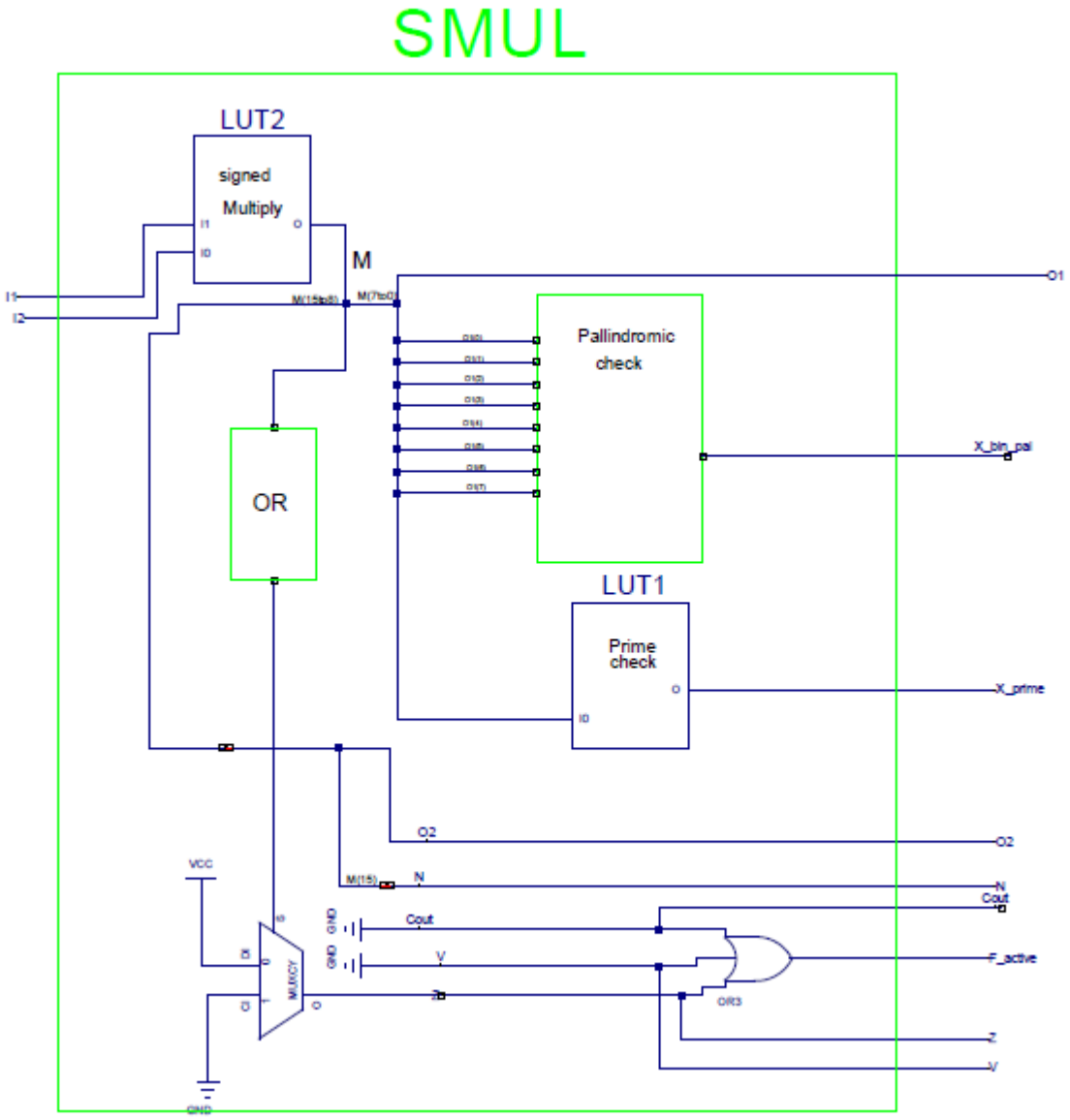
چون ضرب با علامت است پس باید یک متغیر واسطه (تعریف بصورت signal) بنام SM و از نوع علامت‌دار(signed) تعریف می‌کنیم. ۲ ورودی را با در نظر گرفتن اینکه با علامت هستند (به کمک تابع signed) در هم ضرب کرده و داخل SM می‌ریزیم. باید توجه داشت که ضرب ۲ عدد ۸ بیتی حداکثر میتواند ۱۶ بیتی باشد پس SM را ۱۶ بیتی تعریف کردیم.

حال SM را بصورت STD LOGIC در متغیر واسطه M که آن هم ۱۶ بیتی است می‌ریزیم. چون خروجی ها ۸ بیتی هستند برای نمایش خروجی که ۱۶ بیتی است باید از هر دو خروجی (O1 و O2) استفاده کرد. بطوری‌که ۸ بیت اول در O1 و ۸ بیت دوم در O2 ریخته می‌شود. متغیر خروجی برای محاسبه Z همان M می‌باشد. خروجی N علامت خروجی را نشان می‌دهد. علامت خروجی M(15) می‌باشد پس آن را در N قرار می‌دهیم.

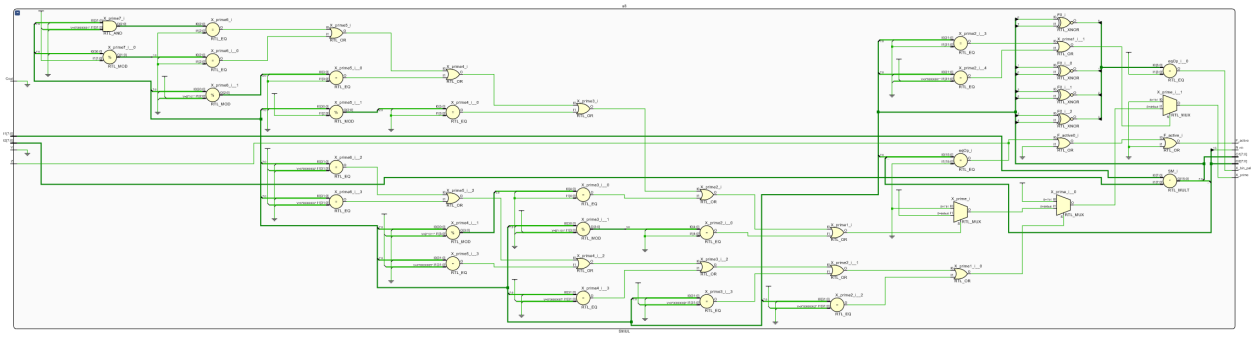
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity 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 SMUL;

architecture Behavioral of SMUL is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
     signal SM: SIGNED (15 downto 0);
     signal M: STD_LOGIC_VECTOR (15 downto 0);

begin
     SM <= SIGNED(I1) * SIGNED(I2);
     M <= STD_LOGIC_VECTOR(SM);
     O1 <= M(7 downto 0);
     O2 <= M(15 downto 8);
     Cout <= '0';
     V <= '0';
     N <= M(15);
     Z <= '1' when M=x"0000" 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 ۱۱: ماژول Signed multiplication




شکل ۱۷: بلوک دیاگرام ماژول Signed multiplication



شکل ۱۸: بلوک دیاگرام ماژول Signed multiplication (خروجی نرم‌افزار VIVADO)

**OPCODE 1000 (Unsigned Multiplication):**

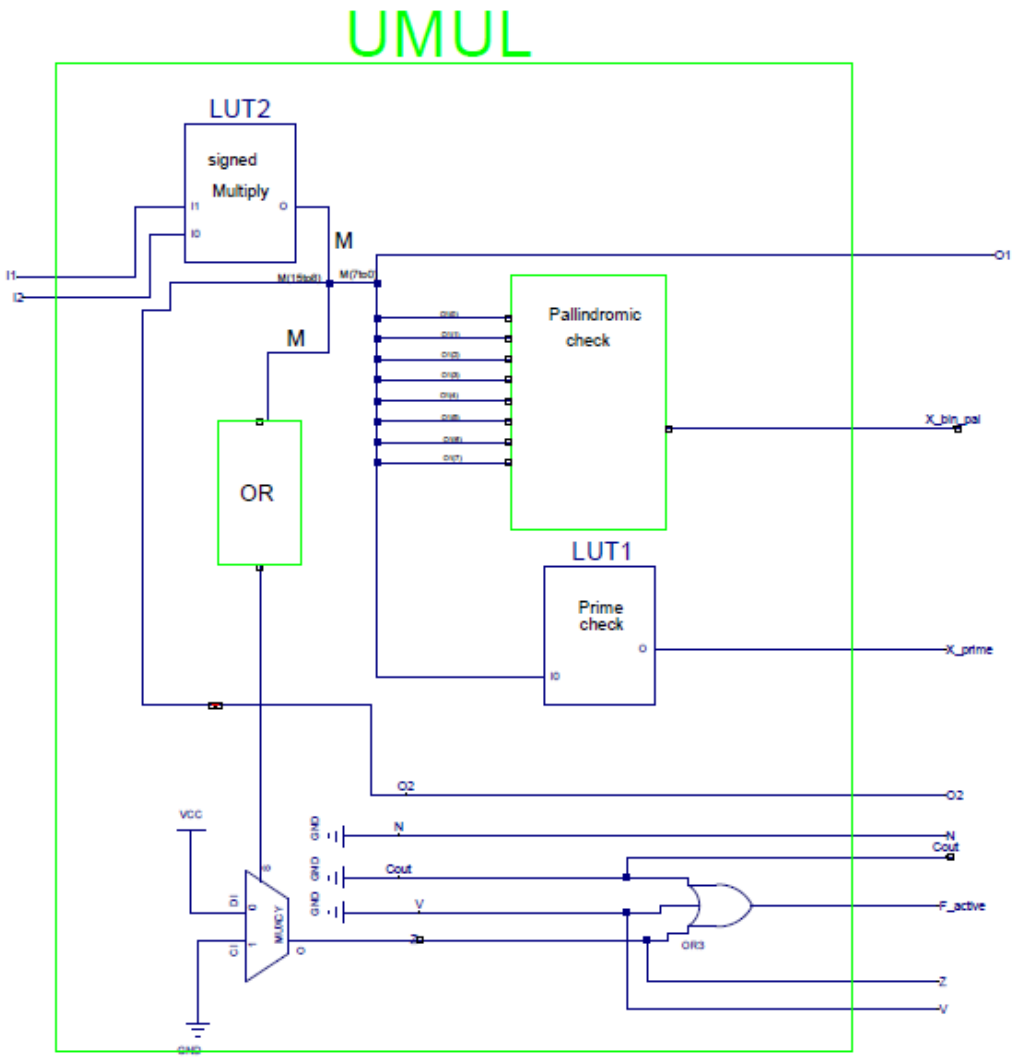
در اینجا Cout و V صفر هستند و چون ضرب بدون علامت است پس مقدار N نیز صفر می‌باشد. با استفاده از پکیج unsigned به راحتی دو ورودی را در هم ضرب می‌کنیم.

باز هم باید توجه کنیم که ضرب ۲ عدد ۸ بیتی حداکثر می‌تواند ۱۶ بیت باشد پس باید متغیر M (متغیر واسطه که بصورت signal تعریف شده) را ۱۶ بیتی تعریف کنیم. برای نمایش این ۱۶ بیت از هر دو خروجی O1 و O2 استفاده می‌کنیم بطوریکه ۸ بیت اول را در O1 و ۸ بیت دوم را در O2 قرار می‌دهیم. متغیر خروجی در اینجا برای بررسی مقدار Z همان M است. عدد 0000000000000000 در مبنای ۲ را بصورت "X"0000 در مبنای HEX نمایش می‌دهیم.

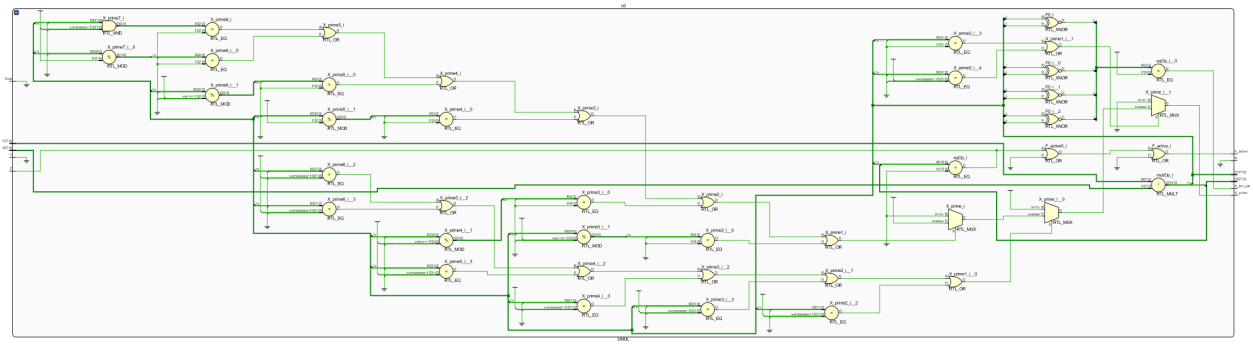
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity 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 UMUL;

architecture Behavioral of UMUL is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
     signal M: STD_LOGIC_VECTOR (15 downto 0);

begin
     M <= I1 * I2;
     O1 <= M(7 downto 0);
     O2 <= M(15 downto 8);
     Cout <= '0';
     V <= '0';
     N <= '0';
     Z <= '1' when M=x"0000" 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 ۱۲: ماژول Unsigned multiplication




شکل ۱۹: بلوک دیاگرام ماژول Signed multiplication



شکل ۲۰: بلوک دیاگرام ماژول Signed multiplication (خروجی نرم‌افزار VIVADO)

**OPCODE 1001 (Unsigned Subtraction):**

با استفاده از پکیج unsigned می‌توان به راحتی از عملیات جمع و ضرب و ... استفاده کرد. چون عمل تفریق بدون‌علامت است پس N صفر است (توجه داریم که اگر حاصل منفی باشد، طبق تعریف پروژه خارج از بازه می‌باشد و این موضوع روی Cout تاثیر می‌گذارد). بر اساس جدول پروژه V نیز صفر است.

برای تفریق مانند تئوری عمل می‌کنیم. ابتدا مکمل ۱ عددی که قرار است کم شود را با تاگل کردن آن (گیت not) بدست می‌آوریم و سپس آن را با عددی که قرار است از آن کم شود جمع می‌کنیم. چون جمع ۸ بیتی ممکن است ۹ بیت شود پس S را بصورت ۹ بیتی تعریف می‌کنیم. بیت آخر را در متغیر sign قرار می‌دهیم. بر اساس sign جواب نهایی را تشخیص می‌دهیم که آیا همین مقدار بدست امده است یا مکمل آن.

برای استفاده از دستورات رفتاری مثل if و case و ... باید از process استفاده کرد. شرط ما براساس sign است پس داخل پرانتز process (sensitivity list) تنها sign را قرار می‌دهیم.

حال اگر sign ۱ باشد پاسخ ما همان جواب بدست آمده است (به جز بیت نهم) و Cout برابر صفر است ولی اگر ۱ نباشد پاسخ ما مکمل جواب بدست امده است و Cout برابر ۱ می‌شود (حاصل منفی و خارج از بازه است). باید توجه داشت که sign شامل پاسخ ما نیست بلکه به ما کمک می‌کند که پاسخ درست را بدست آوریم.

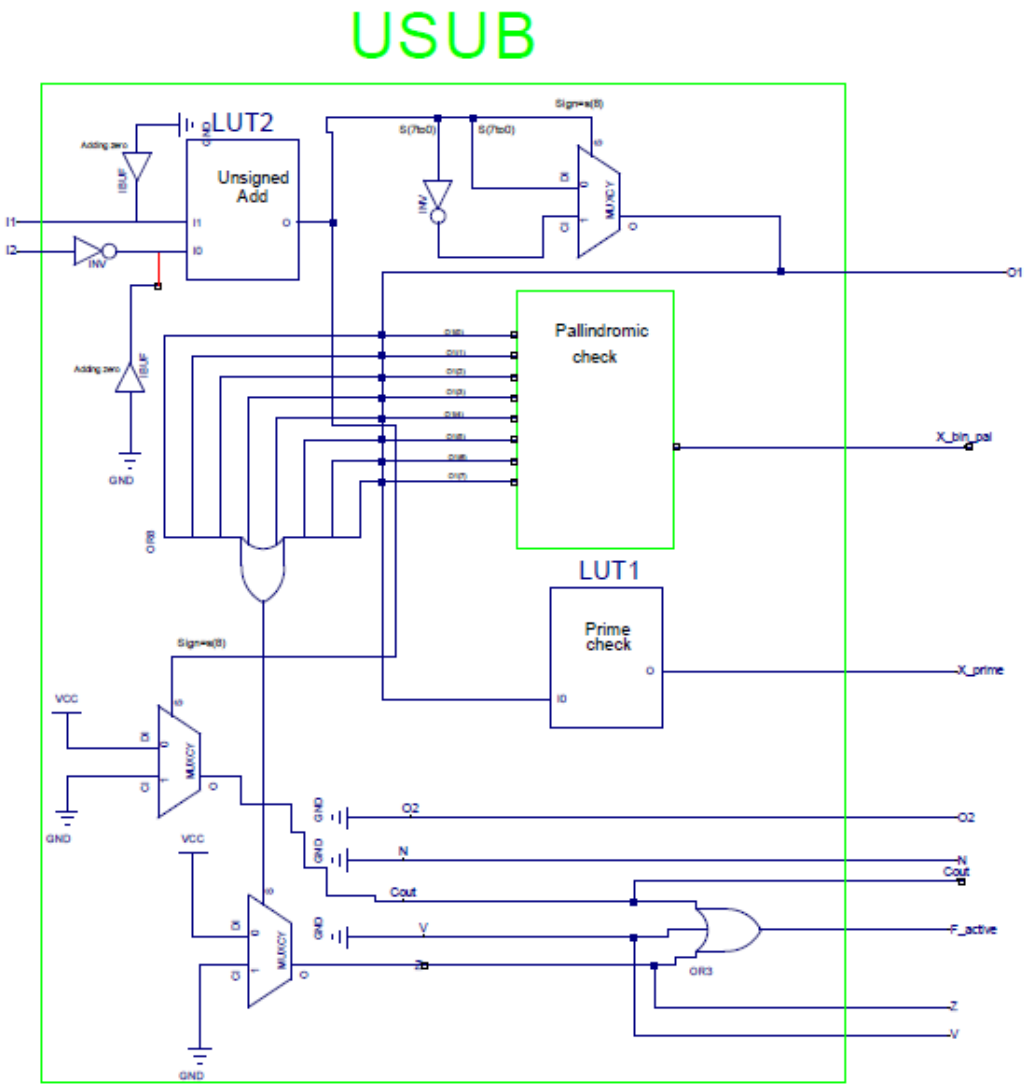
از خروجی دوم (O2) استفاده نکردیم پس آن را صفر (X"00") قرار می‌دهیم و خروجی برای محاسبه Z همان S می‌باشد.

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

entity 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 USUB;

architecture Behavioral of USUB is

     signal F: STD_LOGIC_VECTOR (3 downto 0);
```

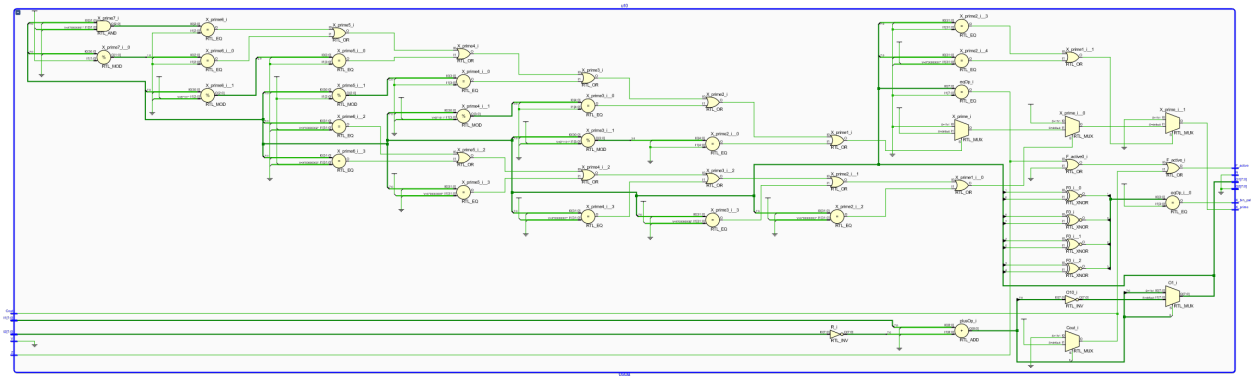
```
     signal S: STD_LOGIC_VECTOR (8 downto 0);
     signal Sign: STD_LOGIC;

begin
     S <= ('0' & I1) + ('0' & not(I2));
     Sign <= S(8);
     Process (Sign, S)
          begin
               if (Sign = '1') then
                    O1 <= S(7 downto 0);
                    Cout <= '0';
               else
                    O1 <= not (S(7 downto 0));
                    Cout <= '1';
               end if;
          end Process;
     N <= '0';
     O2 <= x"00";
     V <= '0';
     Z <= '1' when S=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 ۱۳: ماژول Unsigned subtraction




شکل ۲۱: بلوک دیاگرام ماژول Unsigned subtraction



شکل ۲۲: بلوک دیاگرام ماژول Unsigned subtraction (خروجی نرم‌افزار VIVADO)

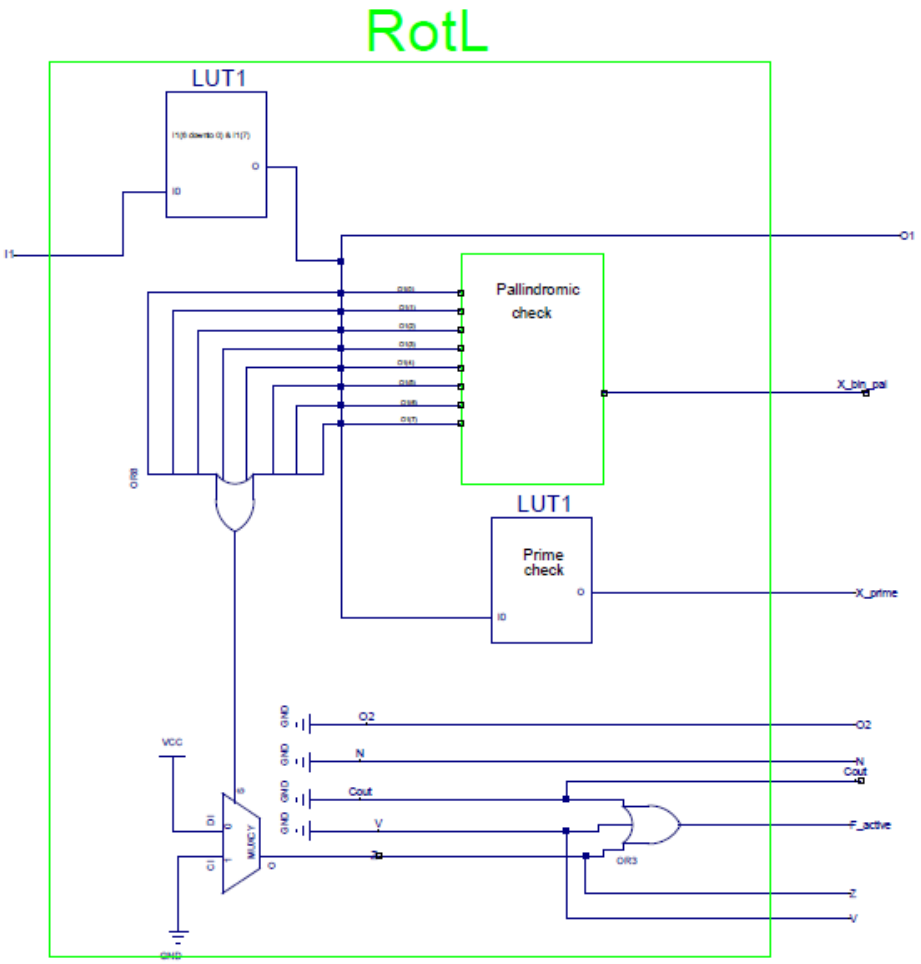
**OPCODE 1010 (Rotation Left):**

مقادیر N و V و Cout صفرند و از خروجی دوم نیز استفاده نخواهیم کرد پس مقدار آن را صفر قرار می‌دهیم. عملکرد این زیرماژول این‌گونه است که با ارزش‌ترین بیت ورودی را برداشته و آن را در سمت راست کم ارزش‌ترین بیت قرار می‌دهد. برای اینکار MSB ورودی را به کمک دستور & در ابتدای ۷ بیت اول ورودی قرار می‌دهیم. خروجی برای محاسبه Z همان O1 است.

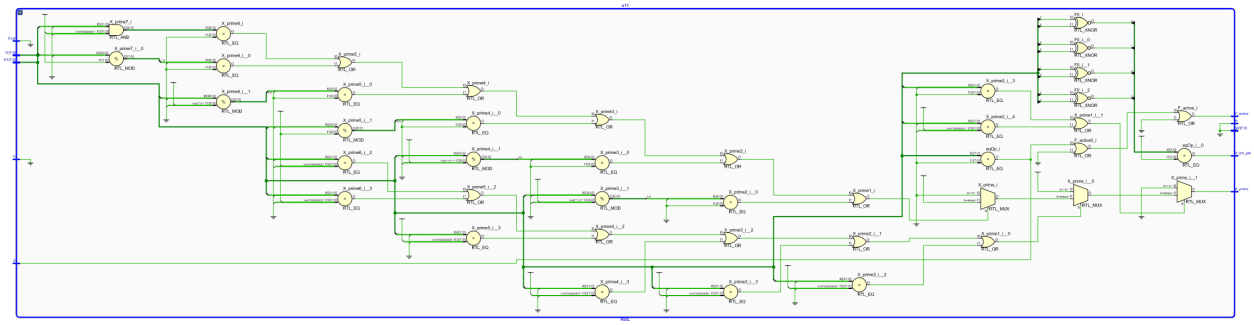
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity RotL is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           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);

begin
     O1 <= I1(6 downto 0) & I1(7);
     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 ۱۴: ماژول Rotation left




شکل ۲۳: بلوک دیاگرام ماژول Rotation left



شکل ۲۴: بلوک دیاگرام ماژول Rotation left (خروجی نرم‌افزار VIVADO)

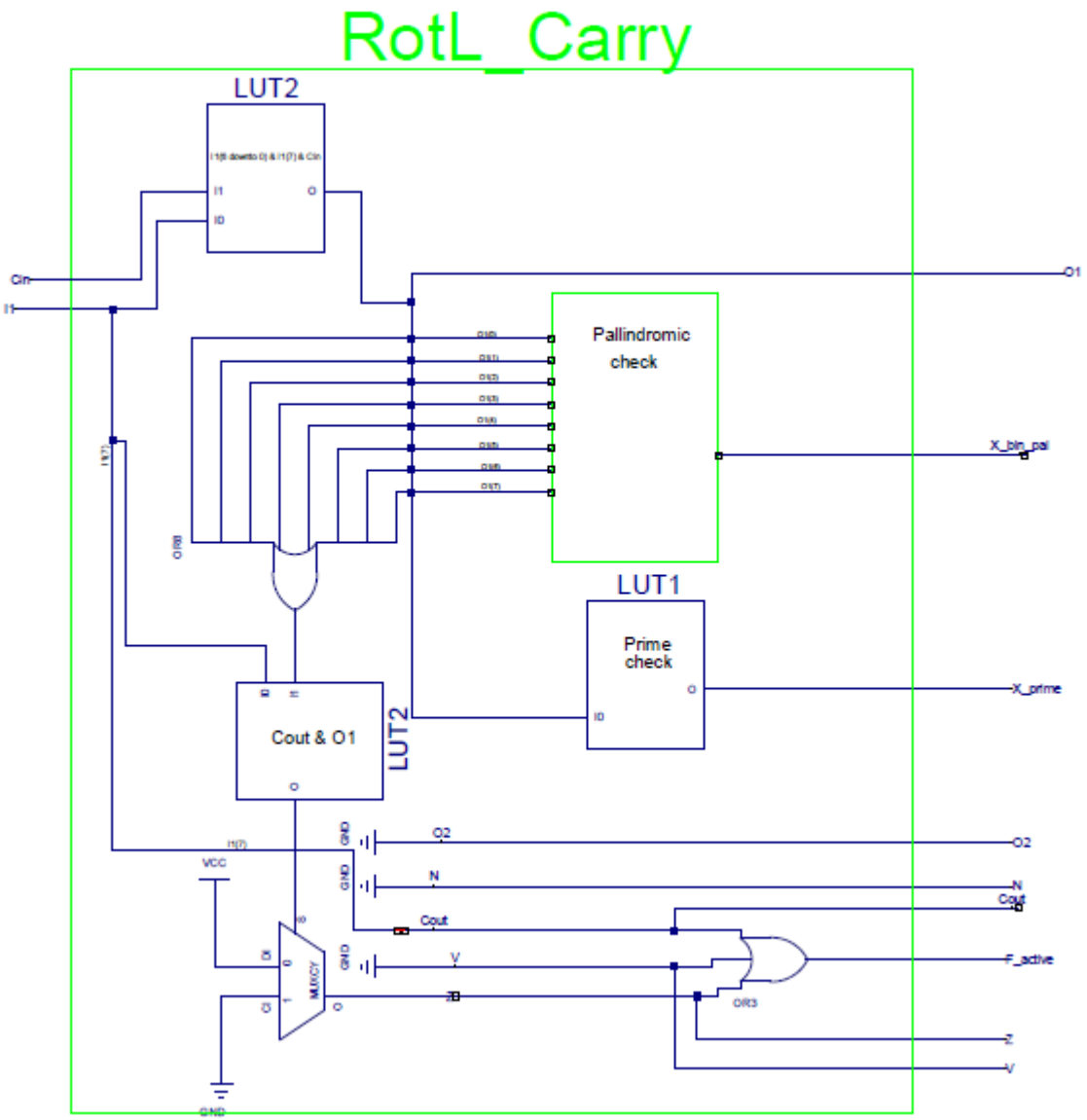
## OPCODE 1011 (Rotation Left with Carry):

مقادیر N و V صفرند و از خروجی دوم نیز استفاده نخواهیم کرد پس مقدار آن را صفر قرار می‌دهیم. این زیرماژول نیز مانند زیرماژول قبلی است با این تفاوت که دارای carry (Cin) می‌باشد. در این صورت MSB ورودی، Cin می‌شود. درنتیجه Cin به کمک دستور & به ابتدای ۷ بیت اول ورودی وصل می‌شود و بیت آخر ورودی به جایگاه Cout منتقل می‌شود. متغیر خروجی برای محاسبه Z، O1 & Cout است.

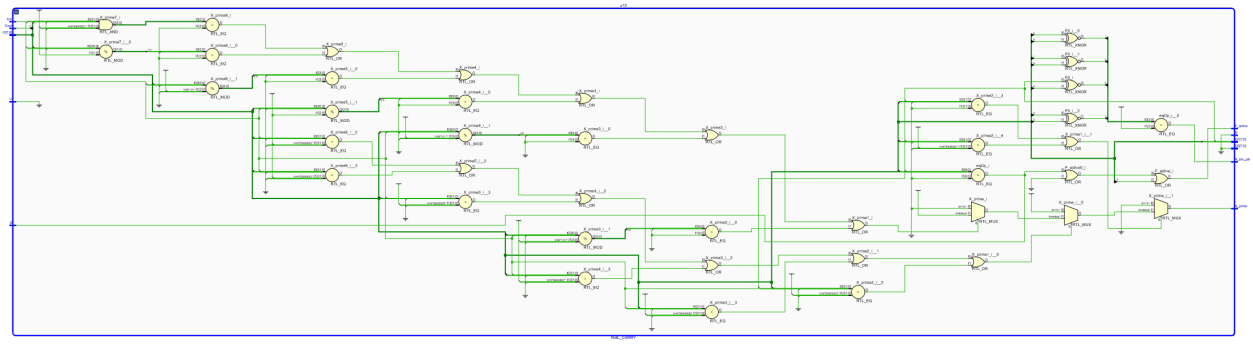
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity RotL_CARRY is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           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);

begin
     O1 <= I1(6 downto 0) & Cin;
     N <= '0';
     O2 <= x"00";
     Cout <= I1(7);
     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 ۱۵: ماژول Rotation left with Carry




شکل ۲۵: بلوک دیاگرام ماژول Rotation left with Carry



شکل ۲۶: بلوک دیاگرام ماژول Rotation left with Carry (خروجی نرم‌افزار VIVADO)

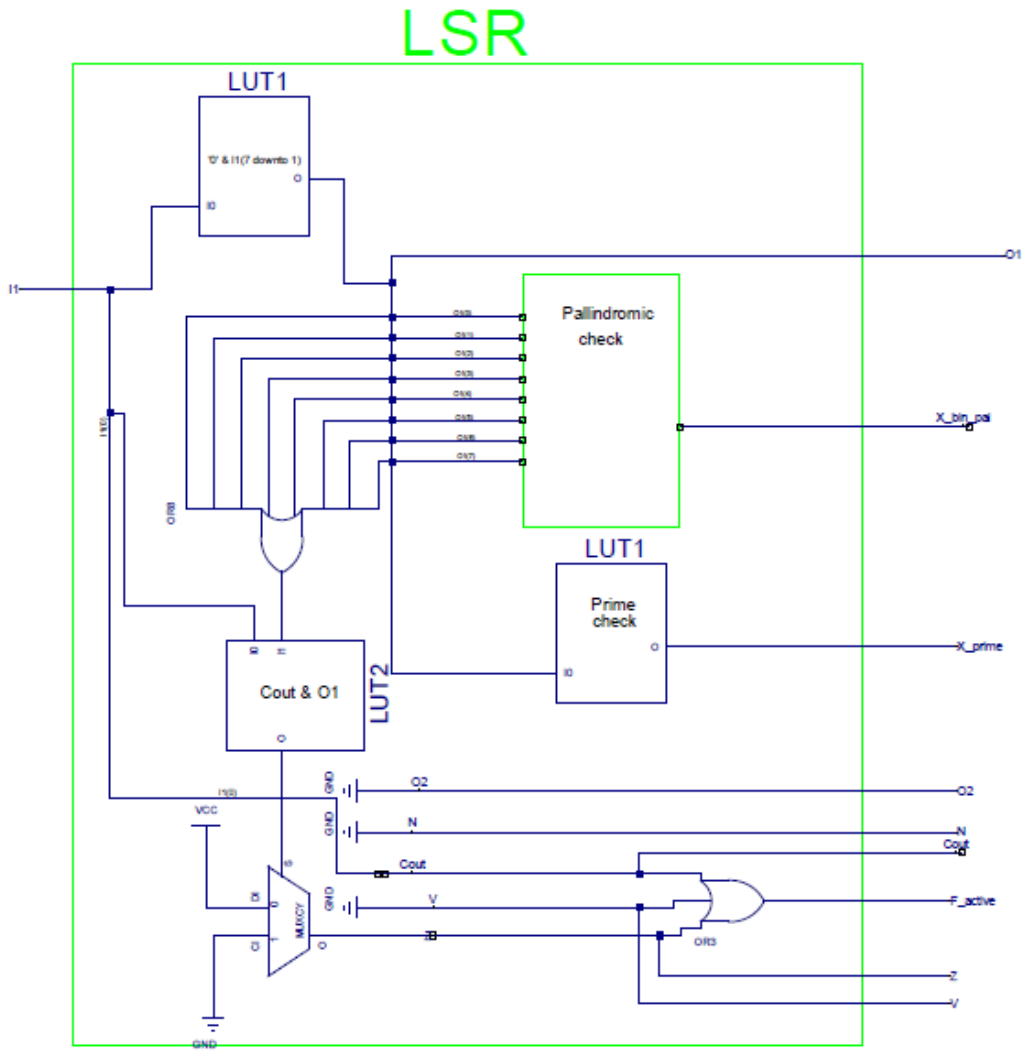
**OPCODE 1100 (Logic Shift Right):**

مقدار V صفر است و از خروجی دوم نیز استفاده نخواهیم کرد پس مقدار آن را صفر قرار می‌دهیم. عملکرد این زیرماژول این گونه است که ورودی را به اندازه یک بیت به سمت راست شیفت می‌دهد. در اینصورت با هربار شیفت از سمت چپ به رشته بیت ما صفر وارد می‌شود. برای اینکار به کمک دستور & ، مقدار '0' را به انتهای ۷ بیت اخر ورودی وصل می‌کنیم. در انتها بیت خارج شده از سمت راست را در Cout قرار می‌دهیم. اخرین بیت (از سمت راست)که بیت علامت است صفر می‌باشد درنتیجه مقدار N نیز صفر می‌شود. خروجی برای محاسبه Z همان Cout & O1 می‌باشد.

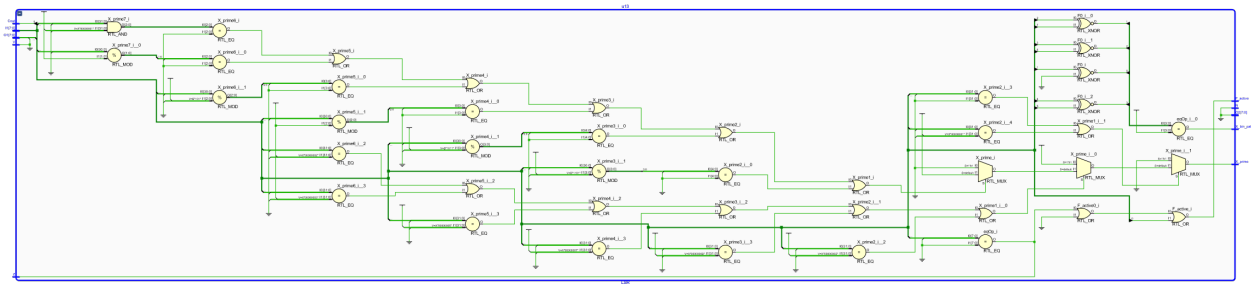
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity LSR is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           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);

begin
     O1 <= '0' & I1(7 downto 1);
     N <= '0';
     O2 <= x"00";
     Cout <= I1(0);
     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 ۱۶: ماژول Logic shift right




شکل ۲۷: بلوک دیاگرام ماژول Logic shift right



شکل ۲۸: بلوک دیاگرام ماژول Logic shift right (خروجی نرم‌افزار VIVADO)

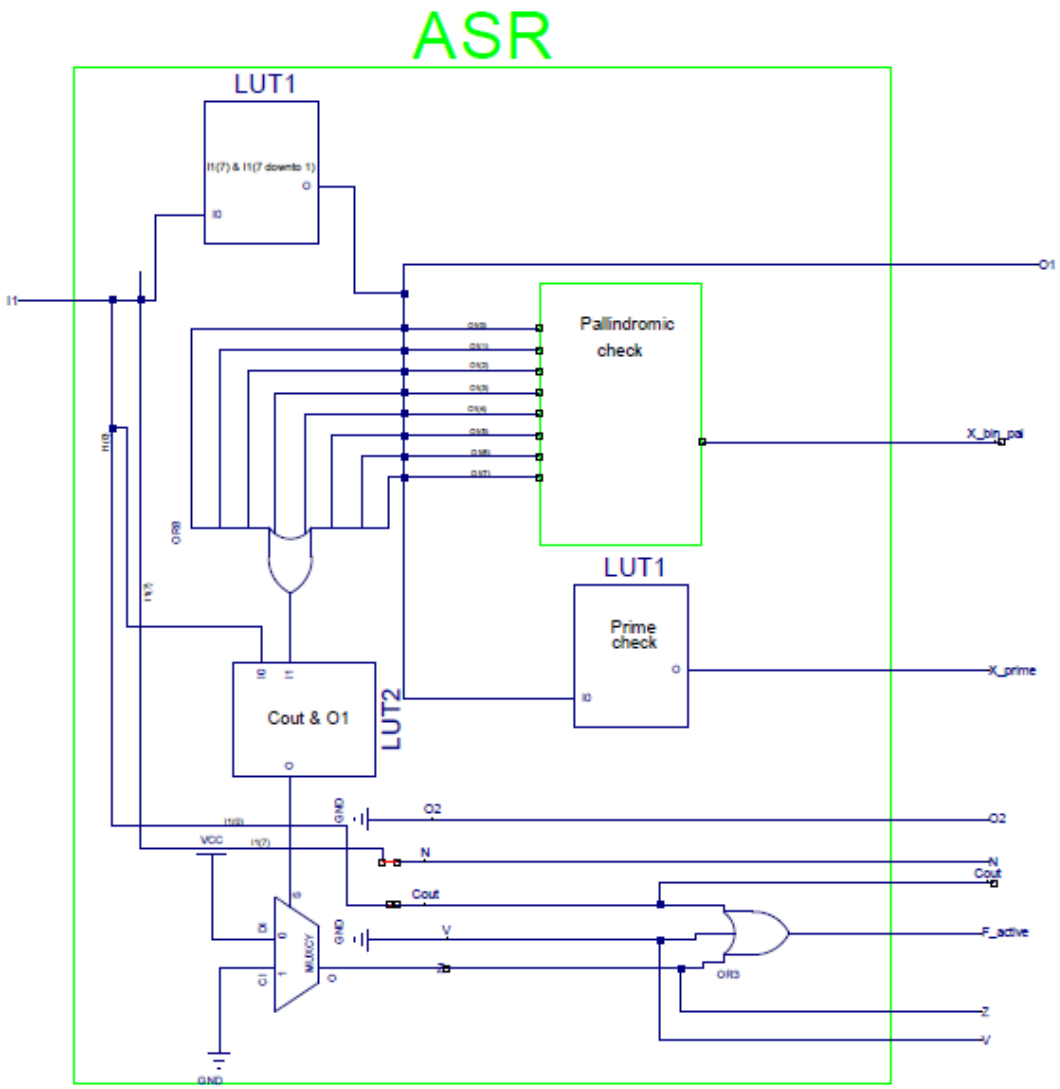
**OPCODE 1101 (Arithmetic Shift Right):**

مقدار V صفر است و از خروجی دوم نیز استفاده نخواهیم کرد پس مقدار آن را صفر قرار می‌دهیم.
Arithmetic shift right مانند Logic shift right است با این تفاوت که به جای اینکه با هر بار شیفت مقدار صفر از سمت چپ به ورودی تزریق کند بیت اخر (sign bit) را دوباره به سمت چپ تزریق می‌کند. برای اینکار کافیست به کمک دستور & بیت آخر را به انتهای ۷ بیت آخر متصل کرد. بیت تزریق شده از چپ به عنوان بیت علامت شناخته می‌شود 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.

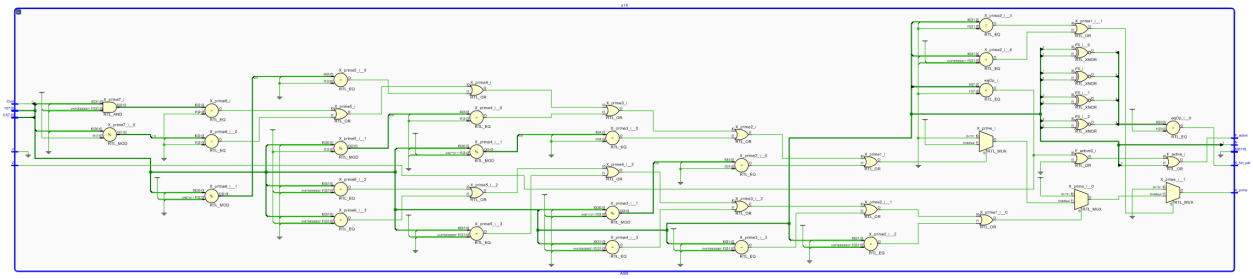
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ASR is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           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);

begin
     O1 <= I1(7) & I1(7 downto 1);
     N <= I1(7);
     O2 <= x"00";
     Cout <= I1(0);
     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 ۱۷: ماژول Arithmetic shift right




شکل ۲۹: بلوک دیاگرام ماژول Arithmetic shift right



شکل ۳۰: بلوک دیاگرام ماژول Arithmetic shift right (خروجی نرم‌افزار VIVADO)

**OPCODE 1110 (Logic Shift Left):**

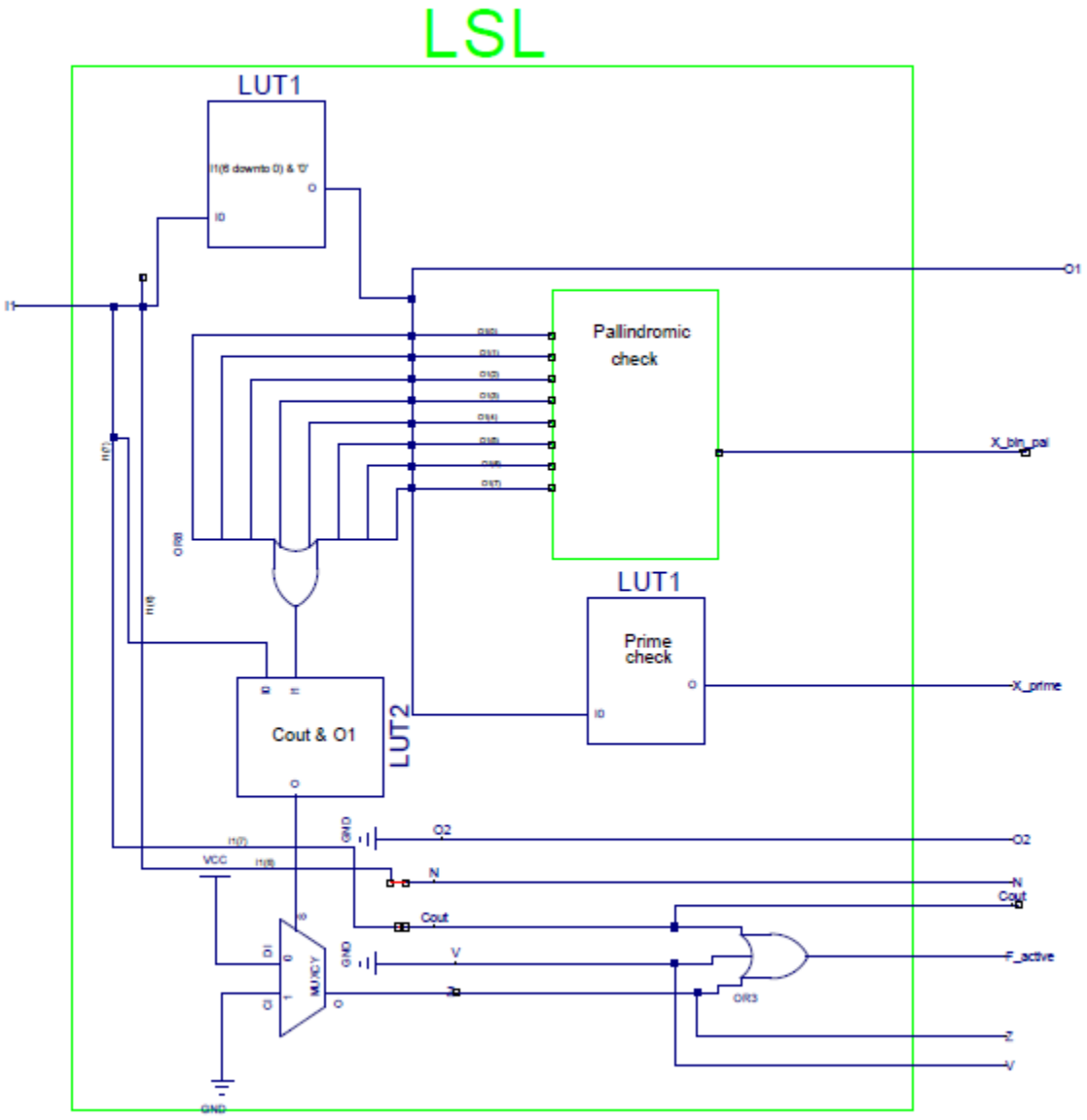
این زیرماژول مانند زیرماژول Logic shift right است با این تفاوت که این زیرماژول به سمت چپ شیفت می‌دهد. در توضیحات گزارش گفته شده که خروجی V در این کد دچار تغییر می‌شود ولی توضیحی در مورد نحوه مشخص کردن صفر یا یک بودن آن داده نشده. از طرفی چون در دیگر عملیات‌های شیفت V صفر است، در این مورد نیز آن را صفر می‌گیریم. از خروجی دوم نیز استفاده نخواهیم کرد پس مقدار آن را صفر قرار می‌دهیم. خروجی این قسمت به کمک دستور & از اضافه کردن مقدار '0' را به ۷ بیت اول متصل بدست می‌آید. بیت خارج شده از سمت چپ را در Cout قرار می‌دهیم. بیت اخر (از سمت راست) بیت علامت است پس مقدار آن در N ریخته می‌شود. خروجی برای محاسبه Z همان Cout & O1 است.

برای چک کردن اول بودن خروجی این ماژول از روند ساده‌تری استفاده می‌کنیم. با توجه به عملیات شیفت به چپ، LSB خروجی همیشه صفر می باشد و این نشان‌دهنده این است که خروجی در مبنای ۱۰ همواره ضریبی است ۲ می‌باشد. به همین دلیل خروجی اول نیست مگر این که فقط بیت دوم آن ۱ باشد و عدد در مبنای ۱۰ برابر ۲ باشد.

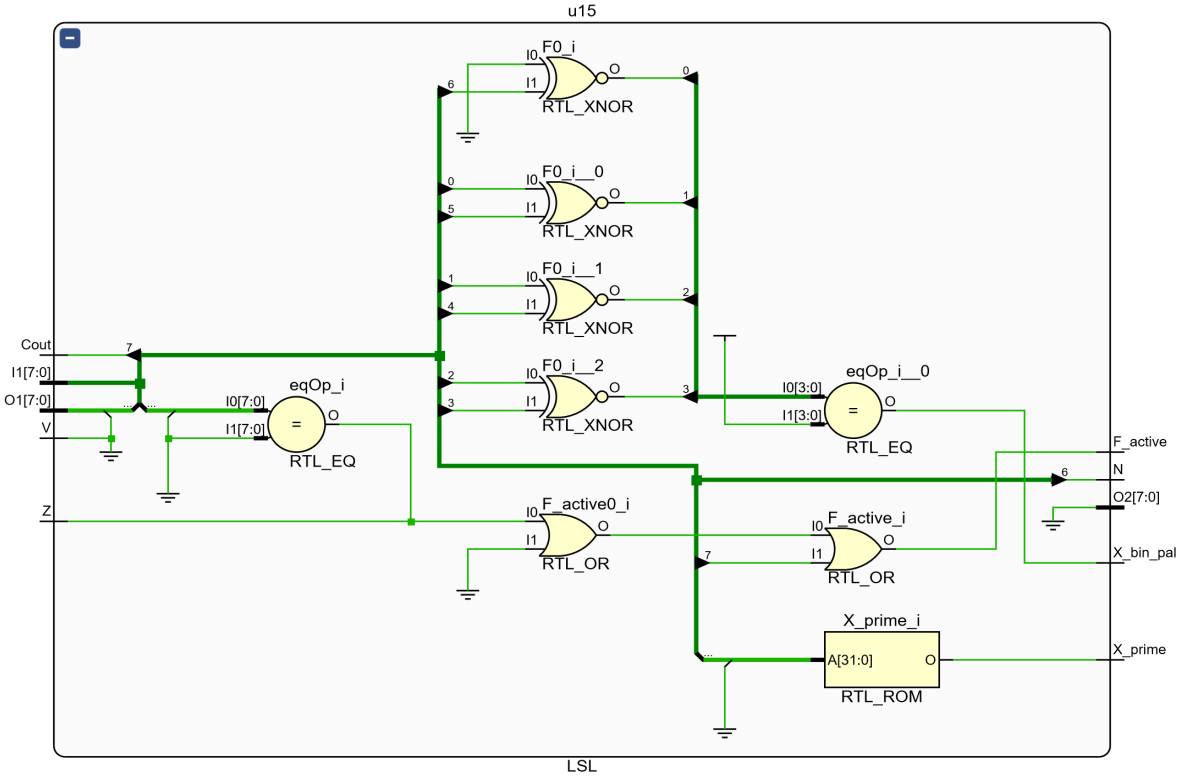
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity LSL is
    Port ( I1 : in  STD_LOGIC_VECTOR (7 downto 0);
           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);

begin
     O1 <= I1(6 downto 0) & '0';
     N <= I1(6);
     O2 <= x"00";
     Cout <= I1(7);
     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 ۱۸: ماژول Logic shift left




شکل ۳۱: بلوک دیاگرام ماژول *Logic shift left*




*شکل ۳۲: بلوک دیاگرام ماژول Logic shift left (خروجی نرم‌افزار VIVADO)*

## OPCODE 1111 (BCD to Binary Conversion):

مقادیر N و V صفرند. همانطور که می‌دانیم در کد BCD هر ۴ بیت معادل یک عدد در مبنای ۱۰ می‌باشند. با ۲ ورودی I1 و I2 ما حداکثر ۱۶ بیت می‌توانیم دریافت کنیم که معادل ۴ رقم در مبنای ۱۰ است. پس ۴ عدد متغیر از نوع integer (SN1 ,SN2 ,SN3 ,SN4) تعریف می‌کنیم و ۴ بیت ۴ بیت به ترتیب از ورودی‌ها جدا کرده و بصورت integer در داخل این ۴ متغیر می‌ریزیم. برای مقدار نهایی یک متغیر integer (SN) تعریف می‌کنیم. مقادیر آن ۴ متغیر را در ارزش جایگاه‌های آن‌ها ضرب می‌کنیم و در نهایت آن‌ها را با هم جمع می‌کنیم و در داخل SN می‌ریزیم.

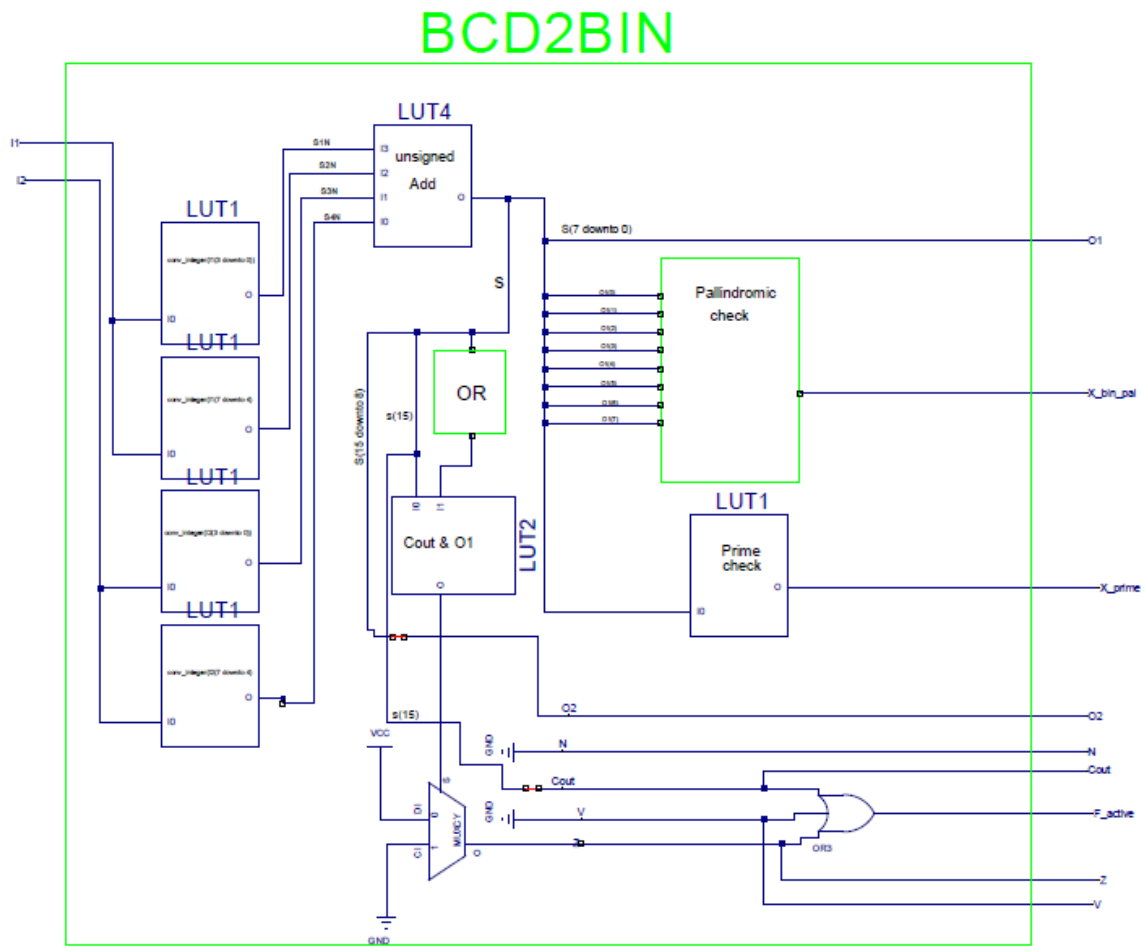
در نهایت SN را بصورت STD LOGIC در متغیر S قرار می‌دهیم. برای نمایش در خروجی باید از هر دو خروجی (O1 و O2) استفاده کرد. خروجی برای محاسبه Z همان S می‌باشد. برای استفاده از توابعی مانند signed و conv_integer و ... باید از پکیج NUMERIC استفاده کرد.

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

entity 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);
```

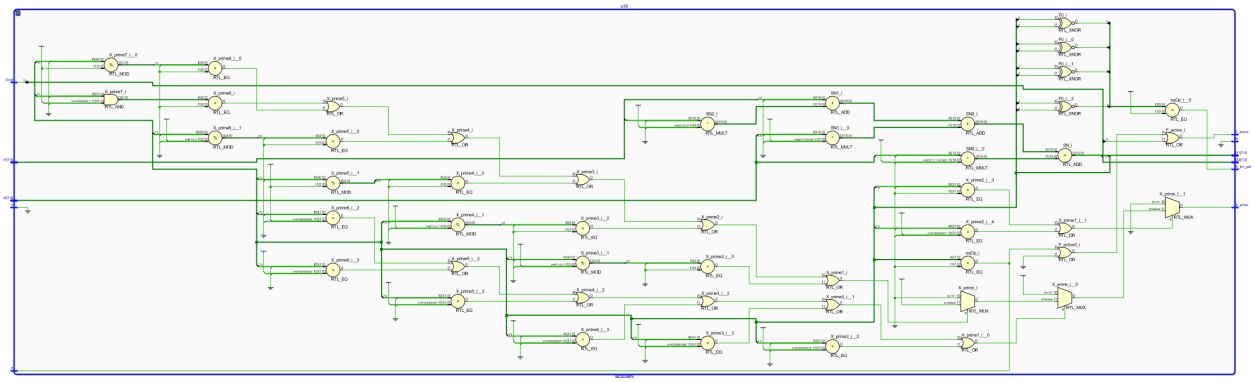
```
                  N : out STD_LOGIC;
                  Cout : inout STD_LOGIC;
                  V : inout STD_LOGIC;
                  Z : inout STD_LOGIC;
                  X_bin_pal : out STD_LOGIC;
                  X_prime : out STD_LOGIC;
                  F_active : out STD_LOGIC);
    end BCD2BIN;

architecture Behavioral of BCD2BIN is

     signal S1N: integer :=0;
     signal S2N: integer :=0;
     signal S3N: integer :=0;
     signal S4N: integer :=0;
     signal S: STD_LOGIC_VECTOR (15 downto 0);
     signal SN: integer :=0;

     signal F: STD_LOGIC_VECTOR (3 downto 0);

begin
     S1N <= conv_integer(I1(3 downto 0));
     S2N <= conv_integer(I1(7 downto 4));
     S3N <= conv_integer(I2(3 downto 0));
     S4N <= conv_integer(I2(7 downto 4));

     SN <= S1N + (S2N * 10) + (S3N * 100) + (S4N * 1000);
     S <= STD_LOGIC_VECTOR(to_unsigned(SN,16));
     O1 <= S(7 downto 0);
     O2 <= S(15 downto 8);
     Cout <= S(15);
     N <= '0';
     V <= '0';
     Z <= '1' when S=x"0000" 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 ۱۹: ماژول BCD to Binary




شکل ۳۳: بلوک دیاگرام ماژول BCD to Binary



*شکل ۳۴: بلوک دیاگرام ماژول BCD to Binary (خروجی نرم‌افزار VIVADO)*

در مرحله آخر لازم به نوشتن یک Top Module داریم که تمامی component ها را در آن map کنیم و خروجی را با توجه به OPCODE و توسط یک if تعیین کنیم.

```
----------------------------------------------------------------------
-- 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);
        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);
        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);
           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);
           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);
           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, 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, 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, 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, O1 => TrASRX, O2 => TrASRY, N
=> TrN(13), Cout => TrCout(13),
```

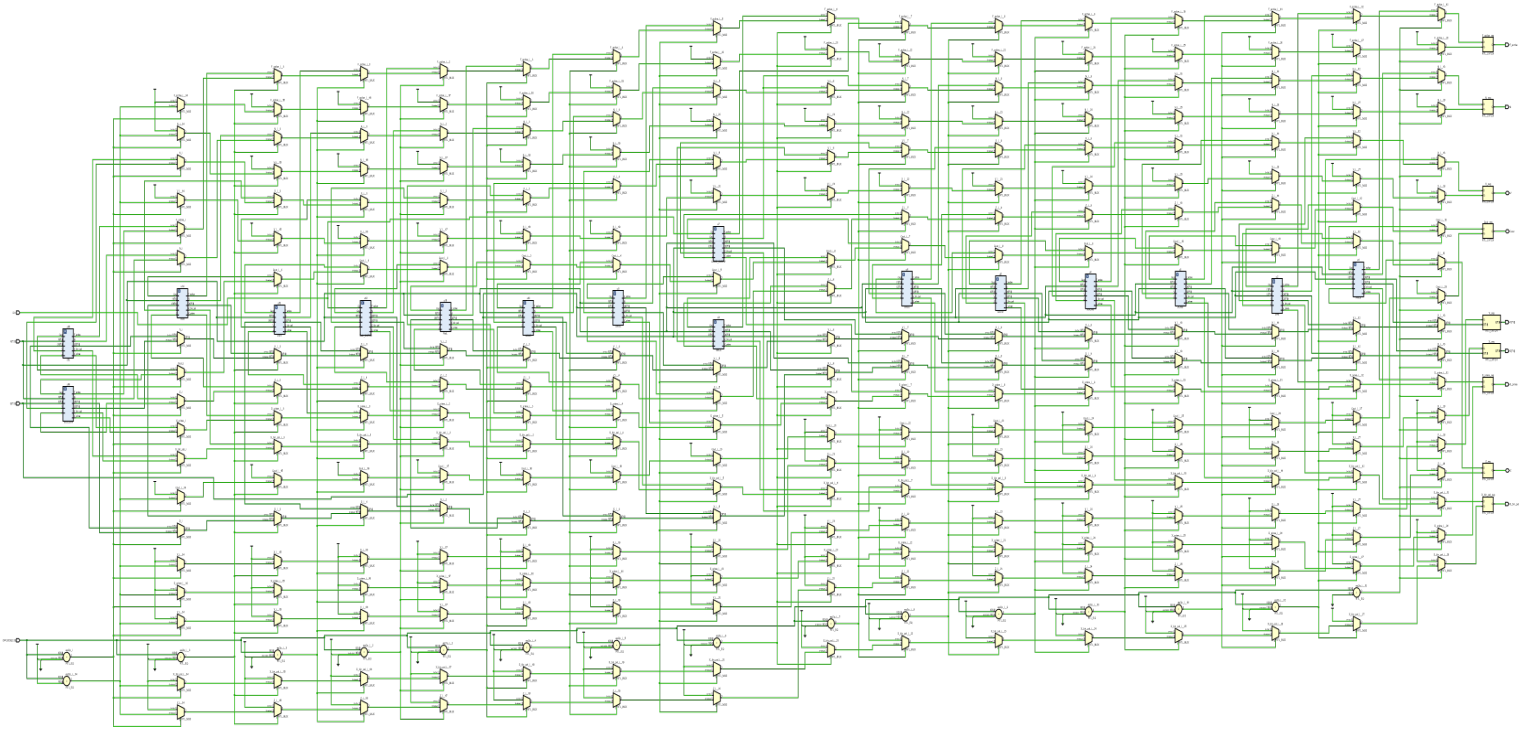
```
                                        V => TrV(13), Z =>
TrZ(13), X_bin_pal => TrBinPal(13), X_prime => TrPrime(13),
F_active => TrActive(13));

     u15 : LSL port map (I1 => A, 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 ۲۰: ALU Top Module



شکل ۳۵: بلوک دیاگرام Task1

تمامی دیاگرام‌ها با کیفیت بالاتر در فایل‌های PDF و شماتیک در فایل پیوست قرار دارد.

در بخش بعدی به testbench کدهای نوشته شده می‌پردازیم.

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

ENTITY Tb_Task1 IS
END Tb_Task1;

ARCHITECTURE behavior OF Tb_Task1 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

   signal CLK : std_logic;
   constant CLK_period : time := 1000 ns;
     --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
      );

   -- Clock process definitions
   CLK_process :process
   begin
        CLK <= '1';
        wait for CLK_period/2;
        CLK <= '1';
        wait for CLK_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin

        OPCODE <= "0000";   --AND Operation
        Cin <= '0';
        A <= "10101011";
        B <= "01010111";
     wait for 100 ns;

        OPCODE <= "0001";   --OR Operation
        Cin <= '0';
        A <= "10101101";
        B <= "11111010";
     wait for 100 ns;

        OPCODE <= "0010";   --XOR Operation
        Cin <= '0';
        A <= "11101001";
        B <= "00101011";
     wait for 100 ns;

        OPCODE <= "0011";   --XNOR Operation
        Cin <= '0';
        A <= "00110011";
        B <= "00110011";
     wait for 100 ns;

        OPCODE <= "0100";   --Unsigned Addition Operation
        Cin <= '0';
        A <= "11110001";
        B <= "10001010";
     wait for 100 ns;

        OPCODE <= "0101";   --Signed Addition Operation
        Cin <= '0';
        A <= "00110001";
        B <= "10000011";
     wait for 100 ns;
```

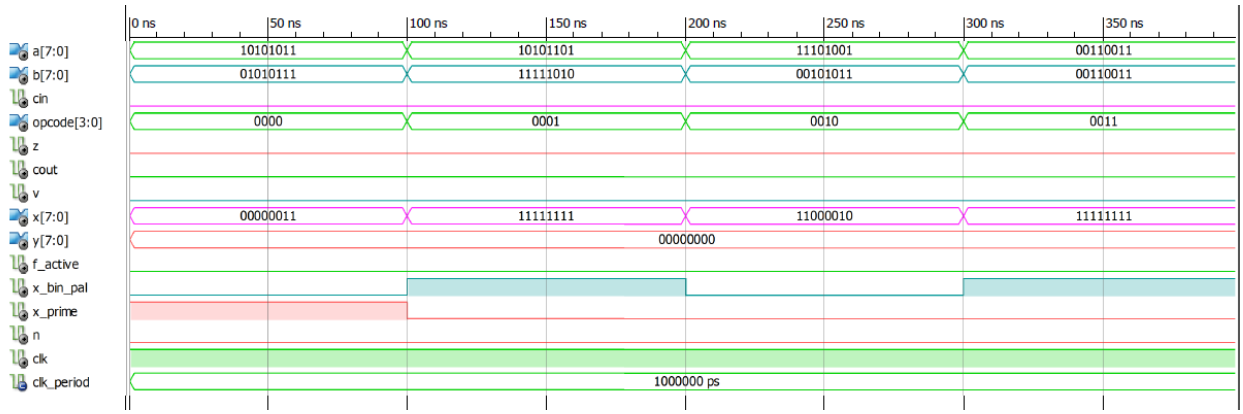
```
          Cin <= '0';
          A <= "10110001";
          B <= "10101100";
      wait for 100 ns;

          OPCODE <= "0110";   --Usigned Addition with Cin
Operation
          Cin <= '1';
          A <= "10101000";
          B <= "01110001";
      wait for 100 ns;
          Cin <= '0';
          A <= "10101000";
          B <= "01110001";
      wait for 100 ns;

          OPCODE <= "0111";   --Signed Multiplication
          Cin <= '0';
          A <= "11100100";
          B <= "00011010";
      wait for 100 ns;
          Cin <= '0';
          A <= "10001100";
          B <= "11101001";
      wait for 100 ns;
          Cin <= '0';
          A <= "00001000";
          B <= "01110001";
      wait for 100 ns;
          Cin <= '0';
          A <= "01010001";
          B <= "10001011";
      wait for 100 ns;

          OPCODE <= "1000";   --Unsigned Multiplication
          Cin <= '0';
          A <= "10101000";
          B <= "01010100";
      wait for 100 ns;

          OPCODE <= "1001";   --Unsigned Subtraction
          Cin <= '0';
          A <= "00011000";
          B <= "01110010";
      wait for 100 ns;

          OPCODE <= "1010";   --Rotation Left
          Cin <= '0';
          A <= "11110110";
          B <= "11111111";
      wait for 100 ns;
```

```
          OPCODE <= "1011";   --Rotation Left with Cin
Operation
          Cin <= '0';
          A <= "01010011";
          B <= "00000000";
      wait for 100 ns;
          Cin <= '1';
          A <= "01010011";
          B <= "00000000";
      wait for 100 ns;

          OPCODE <= "1100";   --Logic Shift Right
          Cin <= '0';
          A <= "10110011";
          B <= "11111111";
      wait for 100 ns;

          OPCODE <= "1101";   --Arithmatic Shift Right
          Cin <= '0';
          A <= "00011101";
          B <= "00000000";
      wait for 100 ns;

          OPCODE <= "1110";   --Logic Shift Left
          Cin <= '0';
          A <= "00000001";
          B <= "11111111";
      wait for 100 ns;

          OPCODE <= "1111";   --BCD to Binary Conversion
          Cin <= '0';
          A <= "10011001";
          B <= "00100110";
      wait for 100 ns;

      wait;
   end process;

END;
```

کد شماره‌ی ۲۱: Testbench

حال به بررسی نتایج testbench می‌پردازیم.

**Logical (AND, OR, XOR, XNOR):**




در ۱۰۰ نانو ثانیه اول ورودی اول (a=101010011) و ورودی دوم (b=01010111) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۰۰۰۰ می‌باشد (AND). در این OPCODE بیت‌های ورودی و خروجی که جایگاه یکسانی دارند با هم AND می‌شوند و در داخل خروجی (x) قرار می‌گیرند. در نهایت خروجی x=00000011 بدست می‌آید.

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

خروجی را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۳ 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 می‌اید که در این‌جا همگی صفرند پس مقدار خود آن نیز صفر می‌شود.

در ۱۰۰ نانو ثانیه دوم ورودی اول (a=10101101) و ورودی دوم (b=11111010) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۰۰۰۱ می‌باشد (OR). در این OPCODE بیت‌های ورودی و خروجی که جایگاه یکسانی دارند با هم OR می‌شوند و در داخل خروجی (x) قرار می‌گیرند. در نهایت خروجی x=11111111 بدست می‌آید.

خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. همانطور که گفته شده بود چون کری خروجی نداریم مقدار cout برابر ۰ است. 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.

خروجی 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 بی‌علامت است پس n مقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌اید که در اینجا همگی صفرند پس مقدار خود آن نیز صفر می‌شود.

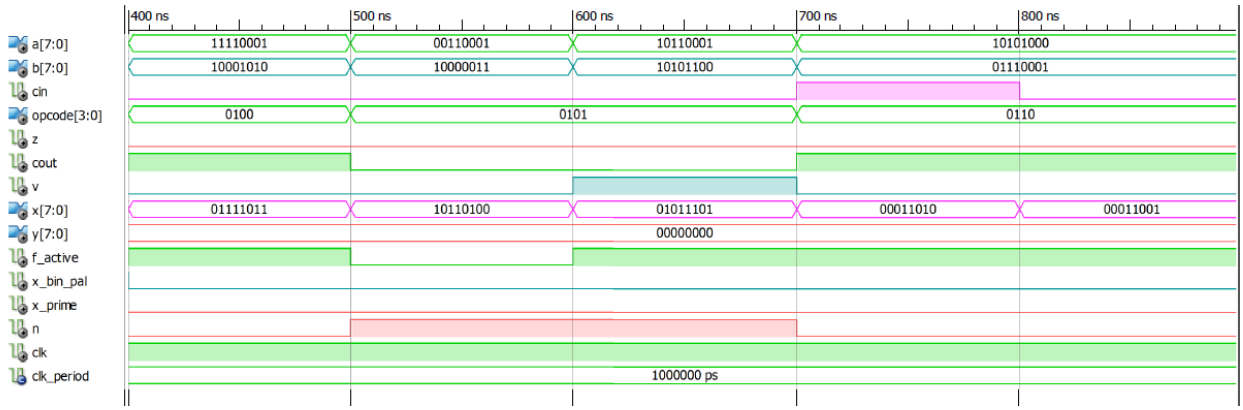
در ۱۰۰ نانو ثانیه سوم ورودی اول (a=11101001) و ورودی دوم (b=00101011) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۰۰۱۰ می‌باشد (XOR). در این OPCODE بیت‌های ورودی و خروجی که جایگاه یکسانی دارند با هم XOR می‌شوند و در داخل خروجی (x) قرار می‌گیرند. در نهایت خروجی x=11000010 بدست می‌آید.
خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. همانطور که گفته شده بود چون کری خروجی نداریم مقدار cout برابر ۰ است. 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.

خروجی را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۱۹۴ 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 می‌اید که در اینجا همگی صفرند پس مقدار خود آن نیز صفر می‌شود.

در ۱۰۰ نانو ثانیه چهارم ورودی اول (a=00110011) و ورودی دوم (b=00110011) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۰۰۱۱ می‌باشد (XNOR). در این OPCODE بیت‌های ورودی و خروجی که جایگاه یکسانی دارند با هم XNOR می‌شوند و در داخل خروجی (x) قرار می‌گیرند. در نهایت خروجی x=11111111 بدست می‌آید.
خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. همانطور که گفته شده بود چون کری خروجی نداریم مقدار cout برابر ۰ است. 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.

خروجی 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 بی‌علامت است پس n مقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌اید که در اینجا همگی آن‌ها صفر اند پس مقدار خود آن نیز صفر می‌شود.

**Additions (UADD, SADD, UADD_CARRY):**




در ۱۰۰ نانو ثانیه پنجم ورودی اول (a=11110001) و ورودی دوم (b=10001010) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۰۱۰۰ می‌باشد (UADD). در این OPCODE بیت‌های ورودی از LSB شروع به جمع شدن می‌کنند و جلو می‌روند و اگر جمع آن‌ها بیشتر از ۱ شد یک carry به جمع بیت‌های بعد از خود می‌دهند. در مبنای ۱۰ داریم:

$$a + b = 241 + 138 = 379 = 256 + 123$$

اگر جمع را بصورت دستی انجام دهیم مشاهده می‌کنیم که در نهایت خروجی x=01111011 بدست می‌آید و Carry خروجی خواهیم داشت (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.

خروجی را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۱۲۳ 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 Cout یک است پس مقدار آن نیز یک می‌شود.

در ۱۰۰ نانو ثانیه ششم ورودی اول (a=00110001) و ورودی دوم (b=10000011) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۰۱۰۱ م می‌باشد (SADD). در این OPCODE بیت‌های ورودی‌های باعلامت از LSB شروع به جمع شدن می‌کنند و جلو می‌روند و اگر جمع آن‌ها بیشتر از ۱ شد یک carry به جمع بیت‌های بعد از خود می‌دهند. در مبنای ۱۰ داریم (دقت داریم که اعداد در قرارداد مکمل دو هستند و b منفی است):

$$a + b = 49 + (-125) = -76 = -(01001100)_2 \xrightarrow{2s\ Complement} 10110100$$

مشاهده می‌کنیم که خروجی testbench با نتایج محاسبات دستی هم‌خوانی دارد.

خروجی برابر "x"00 نیست پس z برابر ۰ می‌شود. همانطور که گفته شده بود چون کری خروجی نداریم مقدار cout برابر ۰ است. مقدار V نیز صفر می‌باشد چون overflow نداریم و خروجی در بازه اعداد ۸ بیتی با علامت قرار دارد. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد.

خروجی را اگر برعکس کنیم با خودش برابر می‌شود پس x_bin_pal یک می‌باشد. خروجی X در حالت بدون‌علامت و در مبنای ۱۰ برابر عدد ۱۸۰ 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 در اینجا همگی آن‌ها صفر اند پس مقدار خود آن نیز صفر می‌شود.

در ۱۰۰ نانو ثانیه هفتم ورودی اول (a=10110001) و ورودی دوم (b=10101100) داده شده است و OPCODE داده شده برابر ۰۱۰۱ م می‌باشد (SADD). در این OPCODE بیت‌های ورودی‌های باعلامت از LSB شروع به جمع شدن می‌کنند و جلو می‌روند و اگر جمع آن‌ها بیشتر از ۱ شد یک carry به جمع بیت‌های بعد از خود می‌دهند. در مبنای ۱۰ داریم (دقت داریم که اعداد در قرارداد مکمل دو هستند و b منفی است):

$$a + b = (-79) + (-84) = -168$$
$$= -(10101000)_2 \xrightarrow{2s\ Complement\ with\ OV} 101011000$$

مشاهده می‌کنیم که اگر بیت علامت مقدار محاسبه شده را در نظر نگیریم (overflow) خروجی testbench با نتایج محاسبات دستی هم‌خوانی دارد.

خروجی برابر "x"00 نیست پس z برابر ۰ می‌شود. همانطور که گفته شده بود چون کری خروجی نداریم مقدار cout برابر ۰ است. مقدار V یک می‌باشد چون overflow داریم و خروجی از ۱۲۸- کوچک‌تر است. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد.

خروجی را اگر برعکس کنیم با خودش برابر می‌شود پس x_bin_pal یک می‌باشد. خروجی X در حالت بدون‌علامت و در مبنای ۱۰ برابر عدد ۱۸۰ می‌باشد که عددی اول نیست پس x_prime مقدار ۰ را می‌گیرد. خروجی منفی است پس n مقدار ۱ می‌گیرد (برای بررسی n کل عدد با در نظر گرفتن overflow بررسی می‌شود). F_active نیز از or کردن z و Cout و V بدست می‌آید که یک می‌شود.

در ۱۰۰ نانو ثانیه هشتم ورودی اول (a=10101000) و ورودی دوم (b=01110001) داده شده است و cin برابر ۱ است و OPCODE داده شده برابر ۰۱۱۰ می‌باشد (UADD_CARRY). در این OPCODE بیت‌های ورودی از LSB شروع به جمع شدن می‌کنند و جلو می‌روند و اگر جمع آن‌ها بیشتر از ۱ شد یک carry به جمع بیت‌های بعد از خود می‌دهند اما در اینجا carry ورودی (Cin) صفر نیست بلکه ۱ است پس باید به جمع بیت های LSB ۱ را اضافه کرد. در مبنای ۱۰ داریم:

$$a + b + Cin = 168 + 113 + 1 = 282 = 256 + 26$$

اگر جمع را بصورت دستی انجام دهیم مشاهده می‌کنیم که در نهایت خروجی x=00011010 بدست می‌آید و Carry خروجی خواهیم داشت (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.

خروجی را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۲۶ 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 Cout یک است پس مقدار آن نیز یک می‌شود.

در ۱۰۰ نانو ثانیه نهم ورودی اول (a=10101000) و ورودی دوم (b=01110001) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۰۱۱۰ می‌باشد (UADD_CARRY). در این OPCODE بیت‌های ورودی از LSB شروع به جمع شدن می‌کنند و جلو می‌روند و اگر جمع آن‌ها بیشتر از ۱ شد یک carry به جمع بیت‌های بعد از خود می‌دهند اما در اینجا carry ورودی (Cin) صفر است. در مبنای ۱۰ داریم:

$$a + b + Cin = 168 + 113 + 0 = 281 = 256 + 25$$

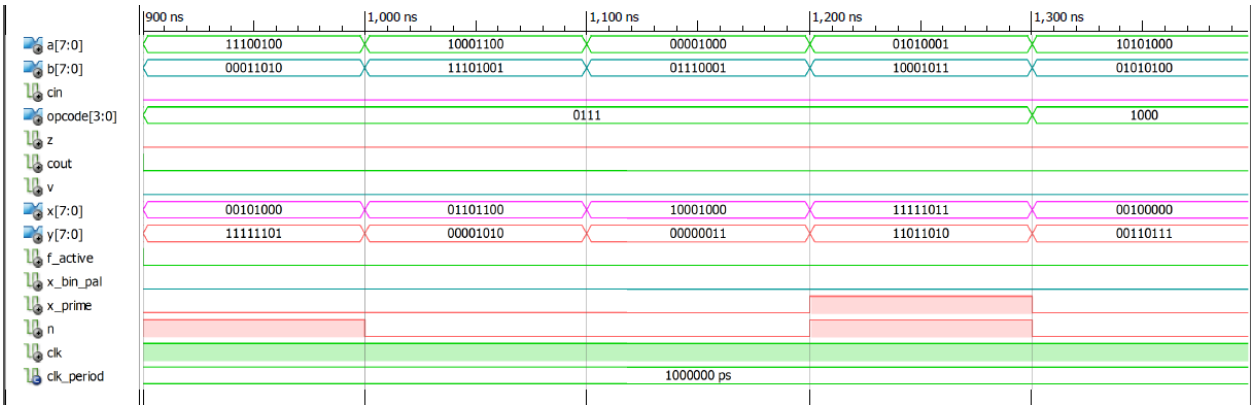
اگر جمع را بصورت دستی انجام دهیم مشاهده می‌کنیم که در نهایت خروجی x=00011001 بدست می‌آید و Carry خروجی خواهیم داشت (Cout = 1).

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.

خروجی را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۲۵ 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 که در اینجا Cout یک است پس مقدار آن نیز یک می‌شود.

تفاوت این دو بازه در مقدار ورودی Cin بود که مشاهده کردیم در حالتی که Cin یک بود، مقدار خروجی ۱ واحد بیشتر است.

## Multiplications (SMUL, UMUL):




در ۱۰۰ نانو ثانیه دهم ورودی اول (a=11100100) و ورودی دوم (b=00011010) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۰۱۱۱ می‌باشد (SMUL). در این OPCODE ورودی‌ها با فرض علامت دار بودن در قرارداد مکمل دو در هم ضرب می‌شوند. در مبنای ۱۰ داریم:

$$a \times b = -28 \times 26 = -728$$
$$= -(001011011000)_2 \xrightarrow{2s\ Complement\ (to\ 16Bits)} 1111110100101000$$

مشاهده می‌کنیم که خروجی testbench با نتایج محاسبات دستی هم‌خوانی دارد.

خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. همانطور که گفته شده بود چون کری خروجی نداریم مقدار cout برابر ۰ است. مقدار V نیز صفر می‌باشد.

خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی X در حالت بدون‌علامت و در مبنای ۱۰ برابر عدد ۴۰ 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 در اینجا همگی آن‌ها صفر اند پس مقدار خود آن نیز صفر می‌شود.

در ۱۰۰ نانو ثانیه یازدهم ورودی اول (a=10001100) و ورودی دوم (b=11101001) داده شده است و cinبرابر ۰ است و OPCODE داده شده برابر ۰۱۱۱ می‌باشد (SMUL). در این OPCODE ورودی‌ها با فرض علامت دار بودن در قرارداد مکمل دو در هم ضرب می‌شوند. در مبنای ۱۰ داریم:

$$a \times b = -23 \times -116 = 2668$$
$$= (101001101100)_2 \xrightarrow{2s\ Complement\ (to\ 16Bits)} 0000101001101100$$

مشاهده می‌کنیم که خروجی testbench با نتایج محاسبات دستی هم‌خوانی دارد.

خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. همانطور که گفته شده بود چون کری خروجی نداریم مقدار cout برابر ۰ است. مقدار V نیز صفر می‌باشد.

خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی X در حالت بدون‌علامت و در مبنای ۱۰ برابر عدد ۱۰۸ می‌باشد که عددی اول نیست پس x_prime مقدار ۰ را می‌گیرد. خروجی مثبت است پس n مقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که در اینجا همگی آن‌ها صفر اند پس مقدار خود آن نیز صفر می‌شود.

در ۱۰۰ نانو ثانیه دوازدهم ورودی اول (a=00001000) و ورودی دوم (b=01110001) داده شده است و cinبرابر ۰ است و OPCODE داده شده برابر ۰۱۱۱ می‌باشد (SMUL). در این OPCODE ورودی‌ها با فرض علامت دار بودن در قرارداد مکمل دو در هم ضرب می‌شوند. در مبنای ۱۰ داریم:

$$a \times b = 8 \times 113 = 904$$
$$= (001110001000)_2 \xrightarrow{2s\ Complement\ (to\ 16Bits)} 0000001110001000$$

مشاهده می‌کنیم که خروجی testbench با نتایج محاسبات دستی هم‌خوانی دارد.

خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. همانطور که گفته شده بود چون کری خروجی نداریم مقدار cout برابر ۰ است. مقدار V نیز صفر می‌باشد.

خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی X در حالت بدون‌علامت و در مبنای ۱۰ برابر عدد ۱۳۶ می‌باشد که عددی اول نیست پس x_prime مقدار ۰ را می‌گیرد. خروجی مثبت است پس n مقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که در اینجا همگی آن‌ها صفر اند پس مقدار خود آن نیز صفر می‌شود.

در ۱۰۰ نانو ثانیه سیزدهم ورودی اول (a=01010001) و ورودی دوم (b=10001011) داده شده است و cinبرابر ۰ است و OPCODE داده شده برابر ۰۱۱۱ می‌باشد (SMUL). در این OPCODE ورودی‌ها با فرض علامت دار بودن در قرارداد مکمل دو در هم ضرب می‌شوند. در مبنای ۱۰ داریم:

$$a \times b = 81 \times -117 = -9477$$
$$= -(0010010100000101)_2 \xrightarrow{2s\ Complement} 1101101011111011$$

مشاهده می‌کنیم که خروجی testbench با نتایج محاسبات دستی هم‌خوانی دارد.

خروجی برابر x"00" نیست پس z برابر ۰ می‌شود. همانطور که گفته شده بود چون کری خروجی نداریم مقدار cout برابر ۰ است. مقدار V نیز صفر می‌باشد.

خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی X در حالت بدون‌علامت و 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 در اینجا همگی آن‌ها صفر اند پس مقدار خود آن نیز صفر می‌شود.

در ۱۰۰ نانو ثانیه چهاردهم ورودی اول (a=10101000) و ورودی دوم (b=01010100) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۰۰۰ می‌باشد (UMUL). در این OPCODE دو ورودی در هم ضرب بدون علامت می‌شوند. در مبنای ۱۰ داریم:

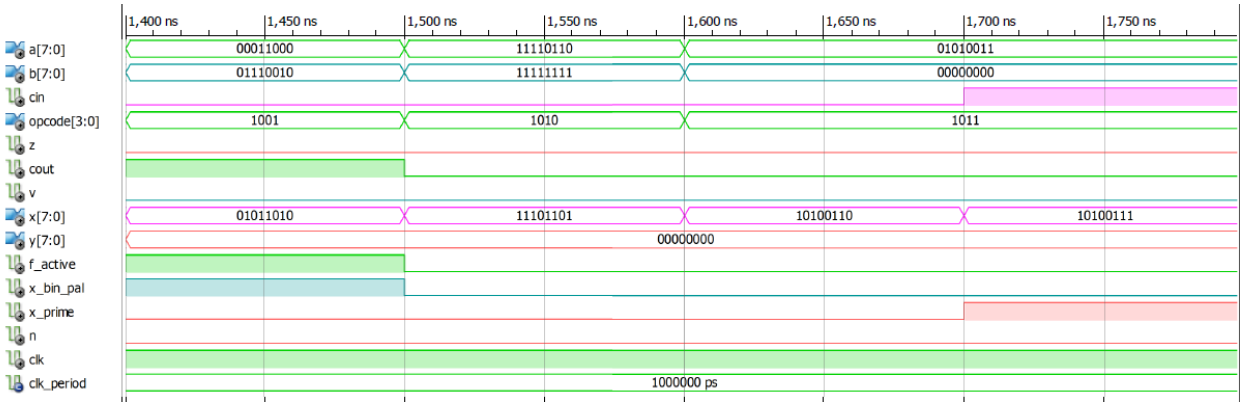
$$a \times b = 168 \times 84 = 14112 = (11011100\ 100000)_2$$

اگر ضرب را بصورت دستی انجام دهیم مشاهده می‌کنیم که در نهایت خروجی , x=00100000 Y=00110111 (X:Y) 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.

اگر خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی X در مبنای ۱۰ برابر عدد ۳۲ 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.

**Subtraction (USUB), Rotations (RotL, RotL_CARRY):**




در ۱۰۰ نانو ثانیه پانزدهم ورودی اول (a=00011000) و ورودی دوم (b=01110010) داده شده است و cinبرابر ۰ است و OPCODE داده شده برابر ۱۰۰۱ می‌باشد (USUB). در این OPCODE ورودی b از ورودی a که هردو بی‌علامت هستند کم می‌شود. در مبنای ۱۰ داریم:

$$a - b = 24 - 114 = -90 = -(01011010)_2$$

اگر عملیات را بصورت دستی انجام دهیم مشاهده می‌کنیم که در نهایت خروجی x=01011010 بدست می‌آید. در محاسبه دستی باید توجه کنیم چون مقدار b از a بیشتر است پس برای رسیدن به پاسخ درست باید از جواب اخر جمع a و مکمل b ، مکمل گرفت و همچنین باید بدانیم که جواب ما منفی این مکمل بدست آمده است.

خروجی برابر "x"00 نیست پس z برابر ۰ می‌شود. مقدار V نیز طبق پیش‌بینی برابر ۰ شده است. چون مقدار به دست آمده عددی منفی و خارج از بازه ۸ بیتی بدون‌علامت می‌باشد، Cout برابر ۱ می‌شود. چون از خروجی دوم استفاده نکرده بودیم مقدار آن را صفر کرده بودیم که در نمودار نیز همین مقدار را نشان می‌دهد.

خروجی را اگر برعکس کنیم با خودش برابر می‌شود پس x_bin_pal یک می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۹۰ 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 Cout یک است پس مقدار آن نیز یک می‌شود.

در ۱۰۰ نانو ثانیه شانزدهم ورودی اول (a=11110110) و ورودی دوم (b=11111111) داده شده است و cinبرابر ۰ است و OPCODE داده شده برابر ۱۰۱۰ می‌باشد (RotL). در این OPCODE راست‌ترین بیت ورودی a به سمت چپ چپ‌ترین بیت a منتقل می‌شود. در نهایت خروجی x=1110110 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.

اگر خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی X در مبنای ۱۰ برابر عدد ۱۱۸ 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 ۱۰۰ نانو ثانیه هفدهم ورودی اول (a=01010011) و ورودی دوم (b=00000000) 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 انجام می‌شود. در اینجا چون carry ورودی صفر است، رشته‌ای که مورد چرخش قرار می‌گیرد Cin & x=001010011 است. در نهایت خروجی x=10100110 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.

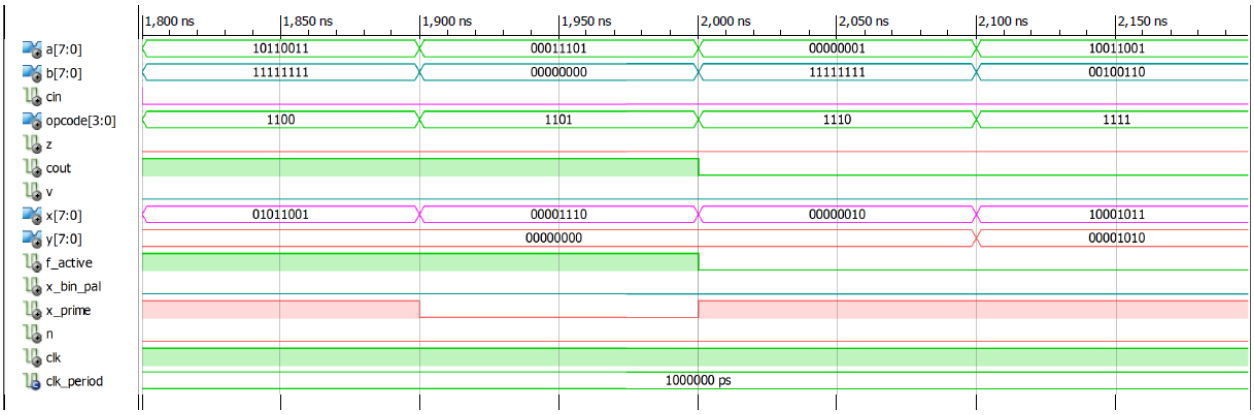
اگر خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۱۶۶ می‌باشد که عددی اول نیست پس x_prime مقدار ۰ را می‌گیرد. خروجی بی‌علامت است پس n مقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که در اینجا همگی آن‌ها صفرند پس مقدار خود آن نیز صفر می‌شود.

در ۱۰۰ نانو ثانیه هجدهم ورودی اول (a=01010011) و ورودی دوم (b=00000000) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۰۱۱ می‌باشد (RotL_CARRY). در این بازه تمامی مشخصات مانند ۱۰۰ نانو ثانیه پیشین است و فقط مقدار Cin به یک تغییر کرده. پس رشته‌ای که مورد چرخش قرار می‌گیرد Cin & x=101010011 است. در نهایت خروجی x=10100111 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 ۱۶۷ 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.

**Shifts (LSR, ASR, LSL), BCD to Binary:**




در ۱۰۰ نانو ثانیه نوزدهم ورودی اول (a=10110011) و ورودی دوم (b=11111111) داده شده است و cinبرابر ۰ است و OPCODE داده شده برابر ۱۱۰۰ می‌باشد (LSR). در این OPCODE رشته بیت a به اندازه یک بیت به راست شیفت داده می‌شود و به ازای هر شیفت یک صفر از سمت چپ به a تزریق می‌شود. بیت خارج شده از سمت راست در Cout قرار می‌گیرد. مشاهده می‌کنیم که در نهایت خروجی x=01011001 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 نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۸۹ 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 Cout یک است پس مقدار آن نیز یک می‌شود.

در ۱۰۰ نانو ثانیه بیستم ورودی اول (a=00011101) و ورودی دوم (b=00000000) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۱۰۱ می‌باشد (ASR). در این OPCODE رشته بیت a به اندازه یک 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 به سمت چپ a تزریق می‌شود. مشاهده می‌کنیم که در نهایت خروجی x=00001110 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 نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی در مبنای ۱۰ برابر عدد ۱۴ 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 Cout یک است پس مقدار آن نیز یک می‌شود.

در ۱۰۰ نانو ثانیه بیست و یکم ورودی اول (a=00000001) و ورودی دوم (b=11111111) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۱۱۰ می‌باشد (LSL). در این OPCODE رشته بیت a به اندازه یک بیت 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 در نهایت خروجی x=00000010 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 ۲ می‌باشد که عددی اول است پس x_prime مقدار ۱ را می‌گیرد. خروجی بی‌علامت است پس nمقدار ۰ می‌گیرد. F_active نیز از or کردن z و Cout و V بدست می‌آید که در اینجا Cout صفر است پس مقدار آن نیز صفر می‌شود.

در ۱۰۰ نانو ثانیه بیست و دوم ورودی اول (a=10011001) و ورودی دوم (b=00100110) داده شده است و cin برابر ۰ است و OPCODE داده شده برابر ۱۱۱۱ می‌باشد (BCD to Binary). در این OPCODE ۴ بیت ۴ بیت ورودی‌ها را جدا می‌کنیم و معادل آن‌ها در مبنای ۱۰ را پیدا می‌کنیم سپس هر کدام را در ارزش خود ضرب کرده سپس با هم جمع می‌کنیم. در آخر جواب بدست آمده را به مبنای ۲ برده و در خروجی نمایش می‌دهیم.

$$a \rightarrow \begin{cases} 1001 = (9)_{10} \\ 1001 = (9)_{10} \end{cases} \quad b \rightarrow \begin{cases} 0010 = (2)_{10} \\ 0110 = (6)_{10} \end{cases}$$

$$NUM = 9 + 9 * 10 + 6 * 100 + 2 * 1000 = 2699 = (101010001011)_2$$

مشاهده می‌کنیم که در نهایت خروجی Y=00001010, X=10001011 (X:Y) 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.

اگر خروجی X را اگر برعکس کنیم با خودش برابر نمی‌شود پس x_bin_pal صفر می‌باشد. خروجی X در مبنای ۱۰ برابر عدد ۱۳۹ 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.