



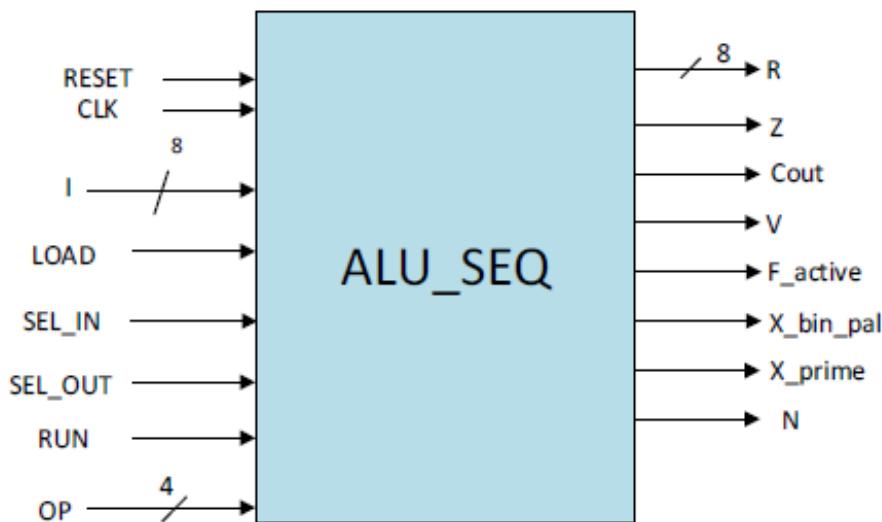
اعضای گروه:

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

عنوان پروژه:

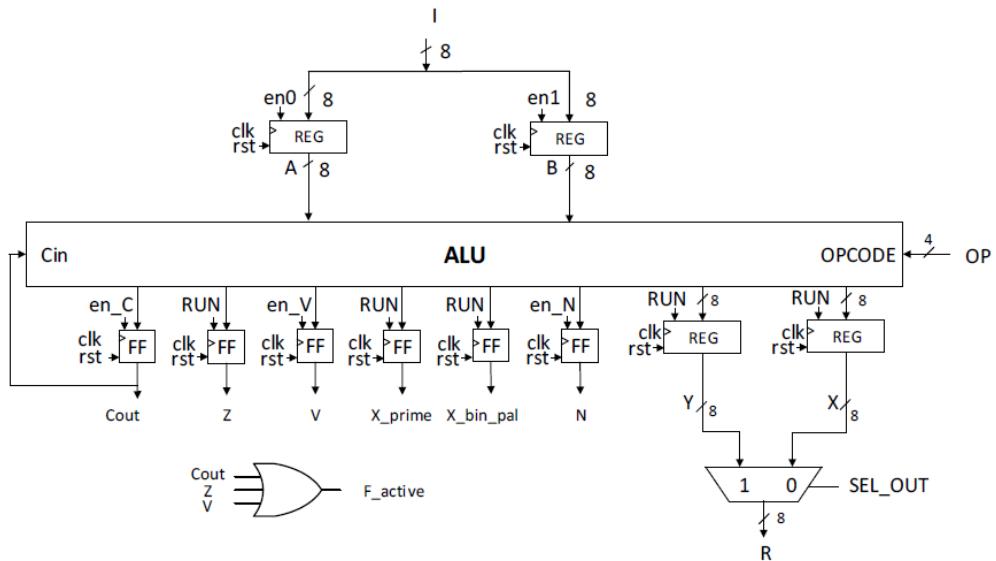
Task 2)

در Task1 کد زیرماژولهای ALU را نوشتیم و توضیح جامعی نیز درباره آنها دادیم. در Task2 قصد داریم کدها و ALU طراحی شدهی خود را توسعه دهیم. برای مثال همانطور که در متن پروژه امده است برای دریافت ورودی تنها از یک باس ۸ بیتی (A) استفاده می‌کنیم و بهوسیله یک پایه دیگر بنام SEL_IN انتخاب می‌کنیم که ورودی در کدام متغیر (A) یا (B) قرار داده شود. اینکار را برای خروجی نیز انجام می‌دهیم بطوری که برای خروجی یک باس ۸ بیتی (R) در نظر می‌گیریم و با ورودی SEL_OUT انتخاب می‌کنیم که خروجی در کدام متغیر (X) یا (Y) قرار بگیرد. همچنین برای ALU خود پایه‌های RESET , CLK, LOAD , RUN در نظر می‌گیریم.



شکل ۱: ALU_SEQ

که اگر دقیق‌تر بخواهیم به آن نگاه کنیم داریم:



شکل ۲: بلوک دیاگرام کلی ALU_SEQ

همانطور که از شکل پیداست پایه RUN مانند پایه X_{bin_pal} , Z , X_{prime} برای فلیپ‌فلابهای enable و دو رجیستر خروجی عمل می‌کند. همچنین فلیپ‌فلابهای دیگر ($Cout$, V , N) دارای پایه‌های enable (en_C , en_V , en_N) هستند.

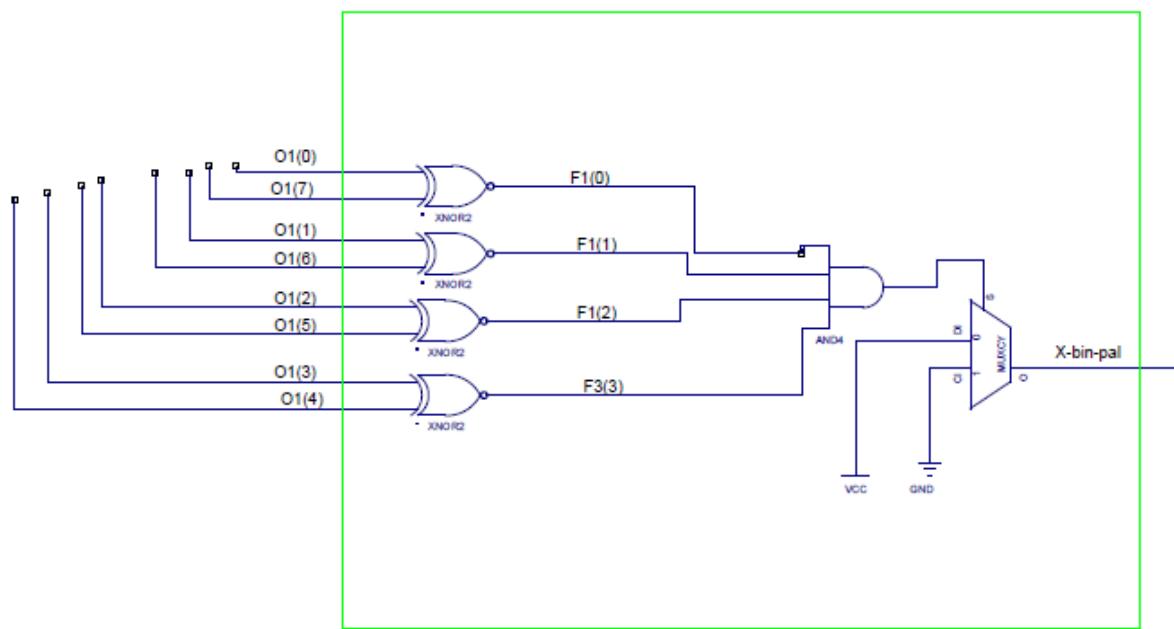
پایه LOAD در شکل و مشخصات پروژه مشخص نشده و ما آن را اینگونه فرض کردیم که زمانی که این پایه فعال باشد رجیسترها ورودی ما کار کنند و در غیر اینصورت غیرفعال باشند و ورودی را از ما نگیرند. پایه Rیست را اسنکرون قرار دادیم تا هر موقع که تمایل داشتیم ALU خود را Rیست کنیم. یک کلاک ورودی به آن دادیم تا طبق آن کار کند.

حال دوباره به بررسی کدها می‌پردازیم و تفاوت آن‌ها را توضیح می‌دهیم.

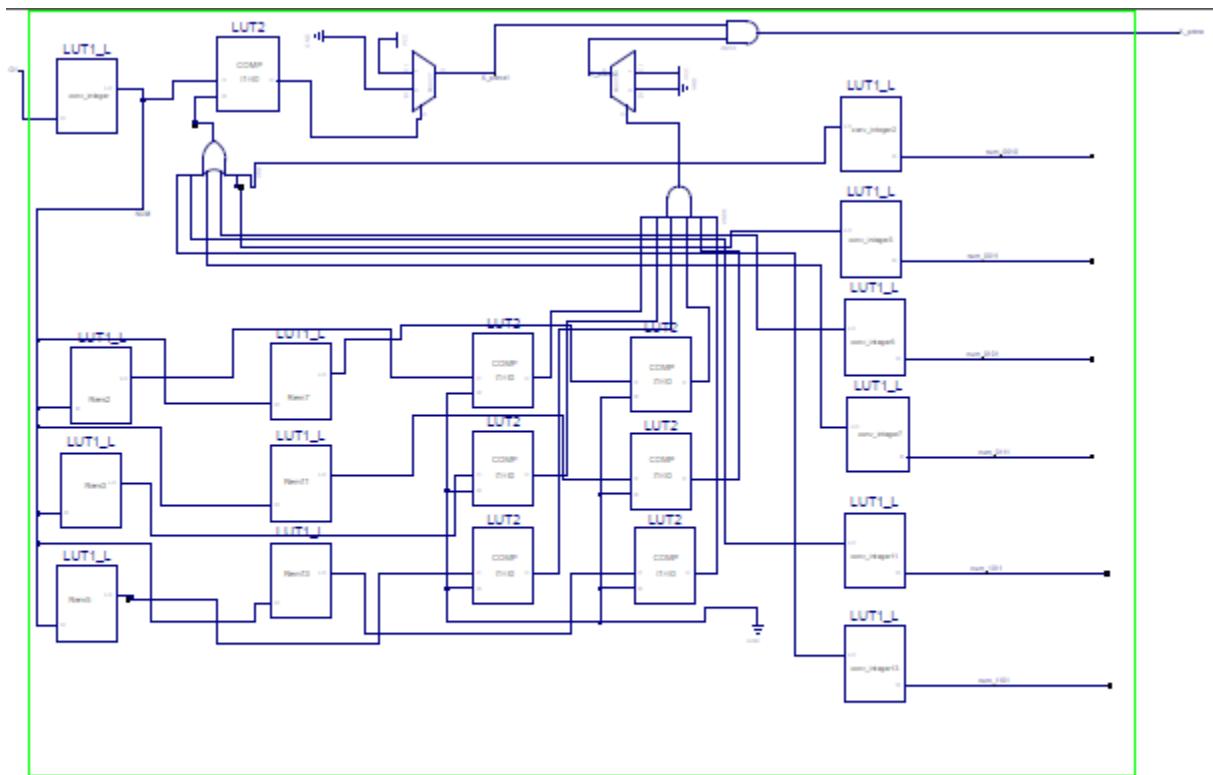
$:F_{active}$ و X_{prime} و X_{bin_pal}

چون این سه مورد جدا از ماهیت هر زیرماژول ثابت هستند و کار یکسانی انجام می‌دهند، تغییری نکرده‌اند.

Pallindromic check



شکل ۳: بلوک دیاگرام X_bin_pal



Prime Check

شکل ۴: بلوک دیاگرام X_{prime}

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

در این ۴ زیرماژول توابع V و $Cout$ استفاده نمی‌کنیم در نتیجه پایه آن‌ها را صفر قرار می‌دهیم و همچنین از خروجی دوم ($O2$) نیز استفاده‌ای نمی‌کنیم.

اما باید توجه داشت با اینکه پایه $enable$ صفر است ولی زمانی که یک کلک باید مقادیر متفاوتی به فلیپ فلاپ می‌آورد و این مطلوب ما نیست، چیزی که مد نظر ما است این است که مقدار قبلی تغییر نکند (no change).

بنابراین در هر زیرماژول، ابتدا کد مقادیر قبلی را در متغیرهایی (PrC , PrV , $PrRy$) می‌ریزیم و در انتهای کد آن‌ها را در $Cout$, V و $O2$ قرار می‌دهیم. از متغیر N استفاده نمی‌کنیم و خروجی آن \cdot است، پس پایه N را ۱ قرار می‌دهیم.

```
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);
           PrC : in STD_LOGIC;
           --Privious Value of Cout
           PrV : in STD_LOGIC;
           --Privious Value of V
           PrRy : in STD_LOGIC_VECTOR (7 downto 0);
           --Privious Value of Ry
           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;
           en_C : inout STD_LOGIC;
           en_V : inout STD_LOGIC;
           en_N : 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;
    O2 <= PrRy;
```

```

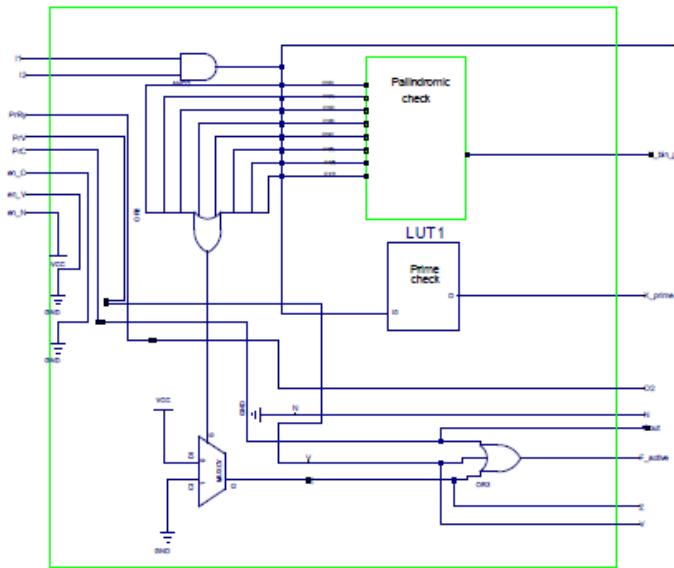
en_N <= '1';
en_V <= '0';
en_C <= '0';
N <= '0';
Cout <= PrC;
V <= PrV;
Z <= '1' when O1=x"00" else '0';
F_active <= Z or V or Cout;

-----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));

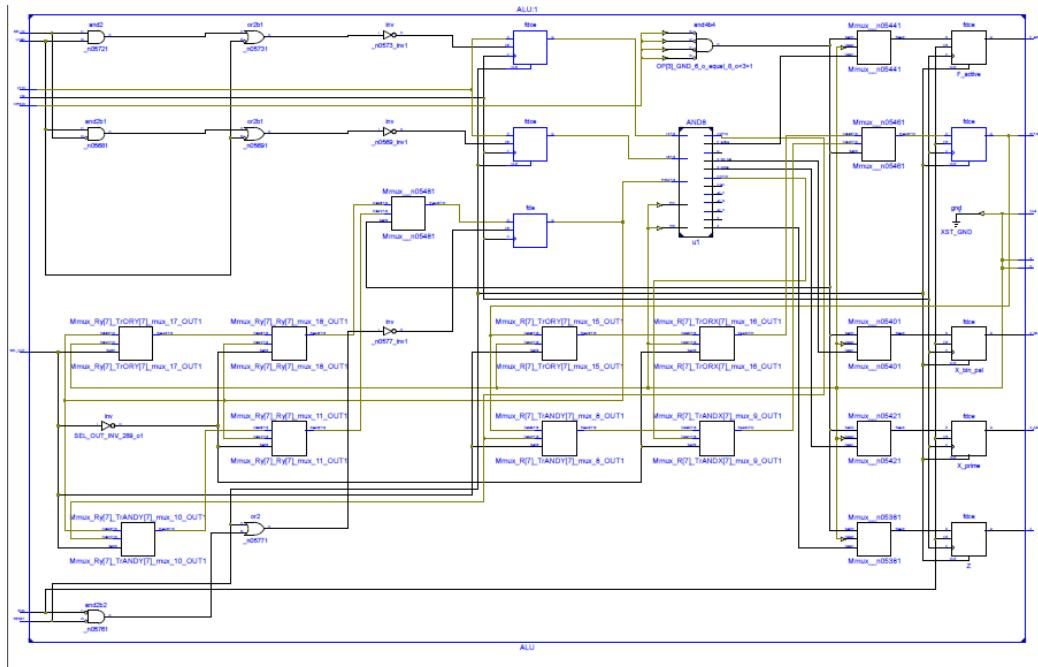
X_bin_pal <= '1' when (F = "1111") else
'0';
-----
-----Prime check-----
Process (O1)
    variable Num : integer;
begin
    Num := conv_integer(O1);
    if (Num = 0 or Num = 1) then
        X_prime <= '0';
    elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
        X_prime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

```

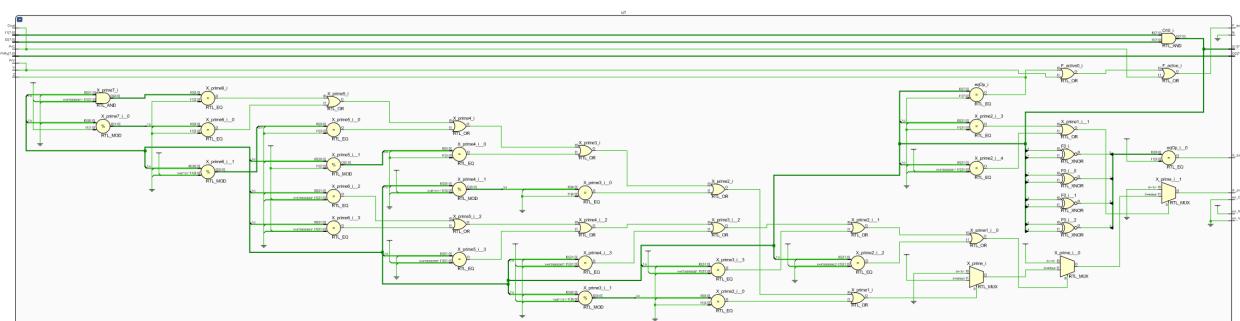
AND8



شکل ۵: بلوک دیاگرام مازول AND



شکل ۶: بلوک دیاگرام مازول AND به همراه فلایپ فلاپ‌های ورودی و خروجی



شکل ۷: بلوک دیاگرام مازول 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);
            PrC : in STD_LOGIC;
            --Privious Value of Cout
            PrV : in STD_LOGIC;
            --Privious Value of V
            PrRy : in STD_LOGIC_VECTOR (7 downto 0);
            --Privious Value of Ry
            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;
            en_C : inout STD_LOGIC;
            en_V : inout STD_LOGIC;
            en_N : 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;
    O2 <= PrRy;
    en_N <= '1';
    en_V <= '0';
    en_C <= '0';
    N <= '0';
    Cout <= PrC;
    V <= PrV;
    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));

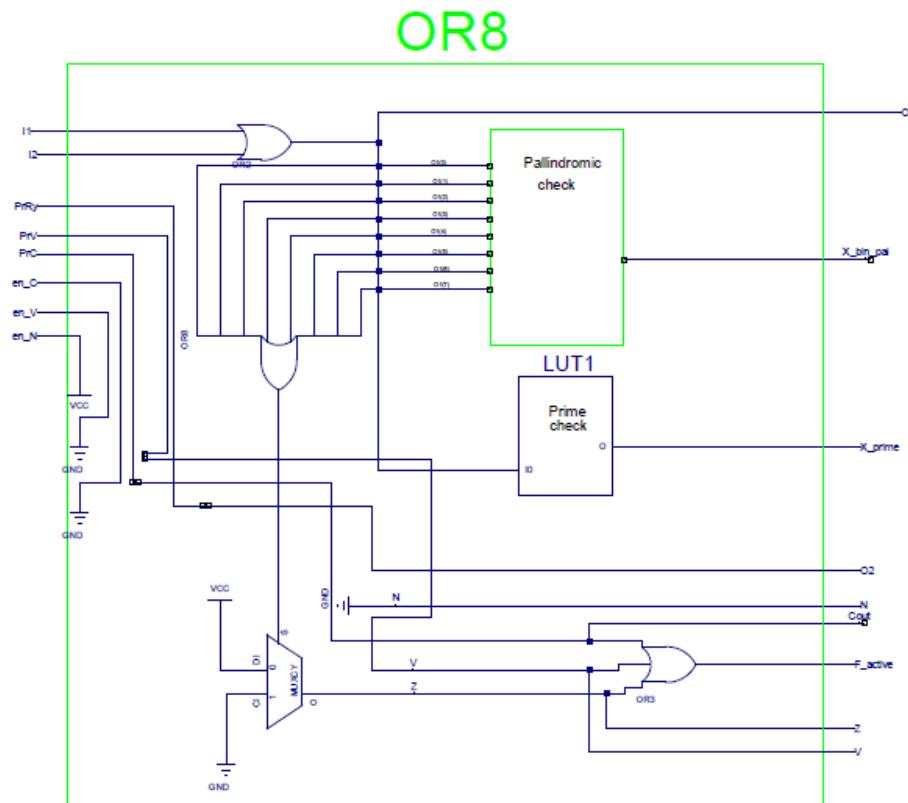
    X_bin_pal <= '1' when (F = "1111") else
                    '0';

```

```

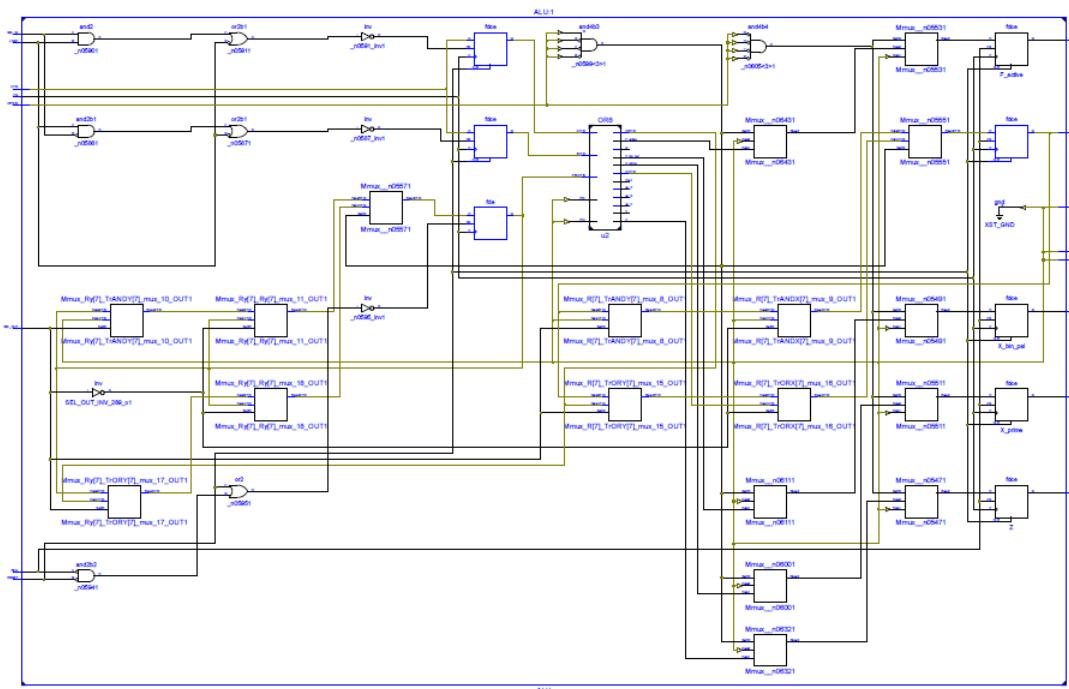
-----Prime check-----
Process (O1)
    variable Num : integer;
begin
    Num := conv_integer(O1);
    if (Num = 0 or Num = 1) then
        X_prime <= '0';
    elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
        X_prime <= '1';
    elsif (Num rem 2 = 0 or
           Num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

کد شماره‌ی ۲: مازول OR

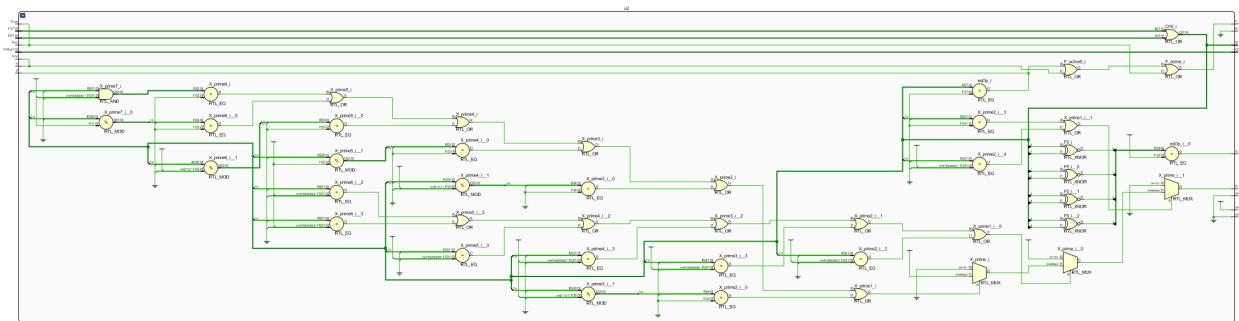


شکل ۱: بلوک دیاگرام مازول OR

پروژه درس مدار منطقی



شکل ۹: بلوک دیاگرام مازول OR به همراه فلیپ فلاب های ورودی و خروجی



شکل ۱۰: بلوک دیاگرام مازول OR خروجی نرم افزار 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);
           PrC : in STD_LOGIC;
           --Privious Value of Cout
           PrV : in STD_LOGIC;
           --Privious Value of V
           PrRy : in STD_LOGIC_VECTOR (7 downto 0);
           --Privious Value of Ry
           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;
        en_C : inout STD_LOGIC;
        en_V : inout STD_LOGIC;
        en_N : 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;
    O2 <= PrRy;
    en_N <= '1';
    en_V <= '0';
    en_C <= '0';
    N <= '0';
    Cout <= PrC;
    V <= PrV;
    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

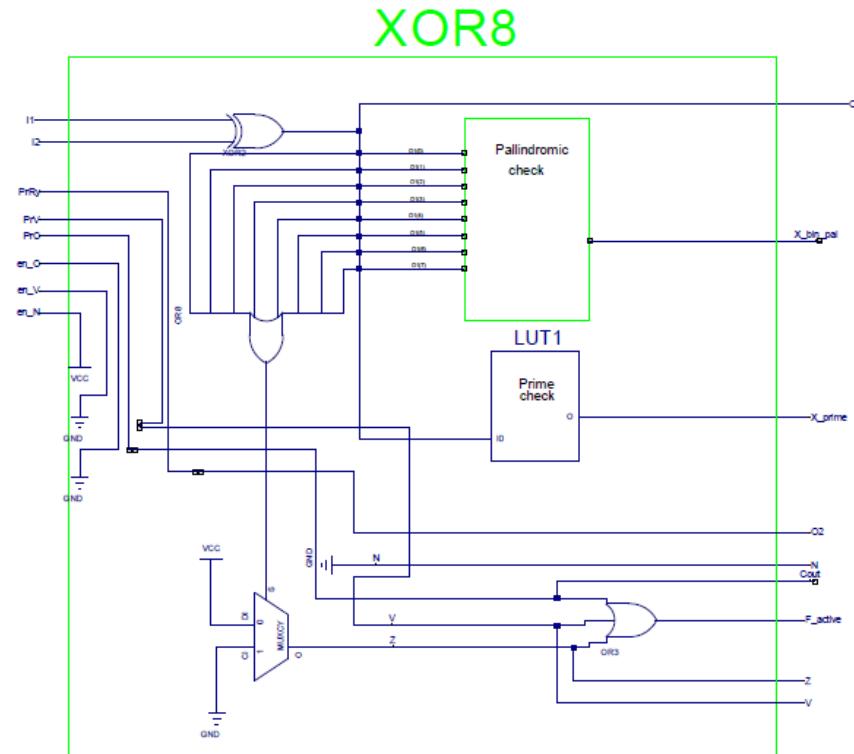
```

```

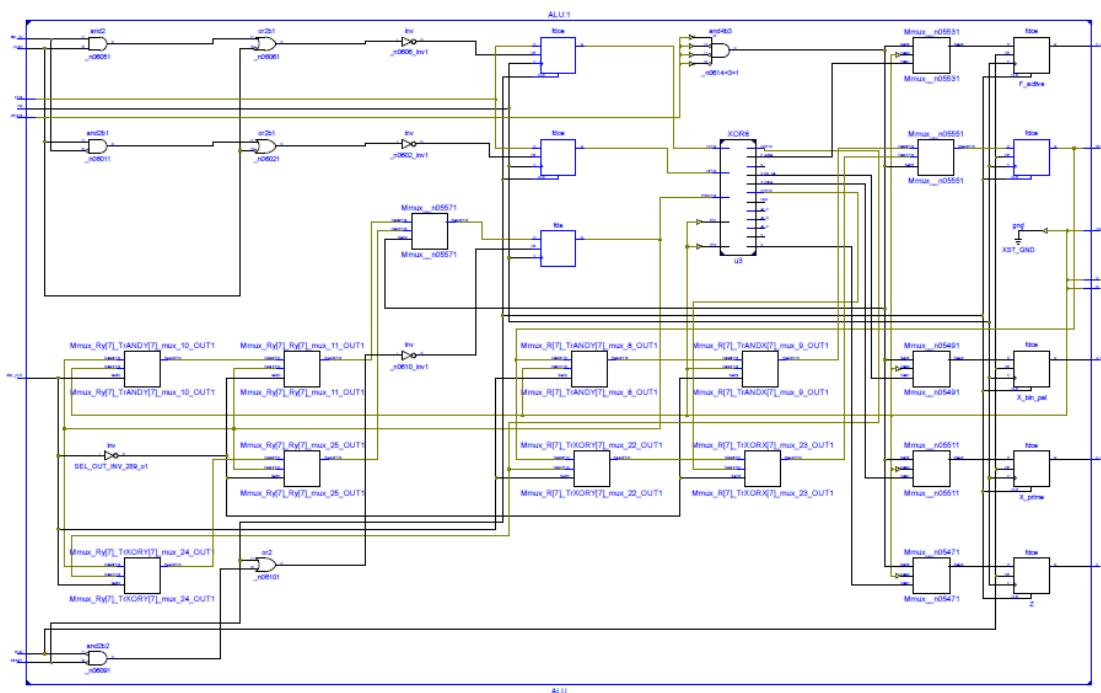
        X_prime <= '1';
    end if;
end Process;
-----
end Behavioral;

```

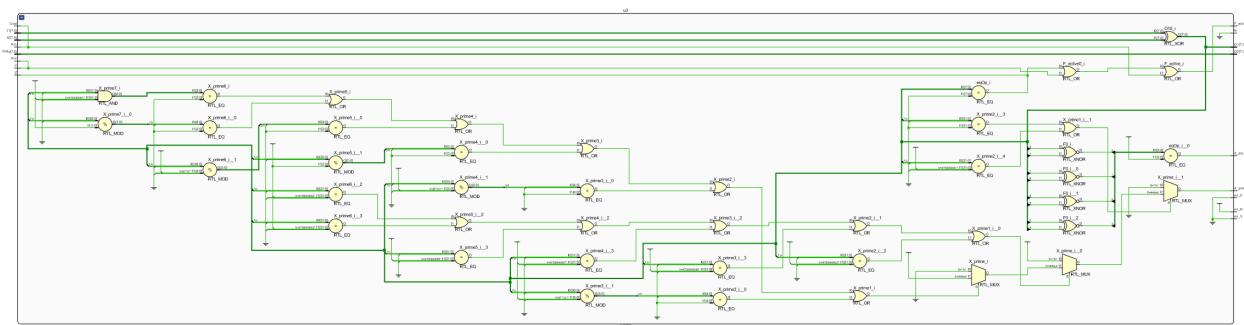
کد شماره‌ی ۳: مازول XOR



شکل ۱۱: بلوک دیاگرام مازول XOR



شکل ۱۲: بلوک دیاگرام مازول XOR به همراه فلیپ‌فلاپ‌های ورودی و خروجی



شکل ۱۳: بلوک دیاگرام ماژول XOR (خروجی نرم افزار 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);
           PrC : in STD_LOGIC;
           --Privious Value of Cout
           PrV : in STD_LOGIC;
           --Privious Value of V
           PrRy : in STD_LOGIC_VECTOR (7 downto 0);
           --Privious Value of Ry
           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;
           en_C : inout STD_LOGIC;
           en_V : inout STD_LOGIC;
           en_N : 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;
    O2 <= PrRy;
    en_N <= '1';
    en_V <= '0';
    en_C <= '0';
    N <= '0';
    Cout <= PrC;
    V <= PrV;

```

```

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));

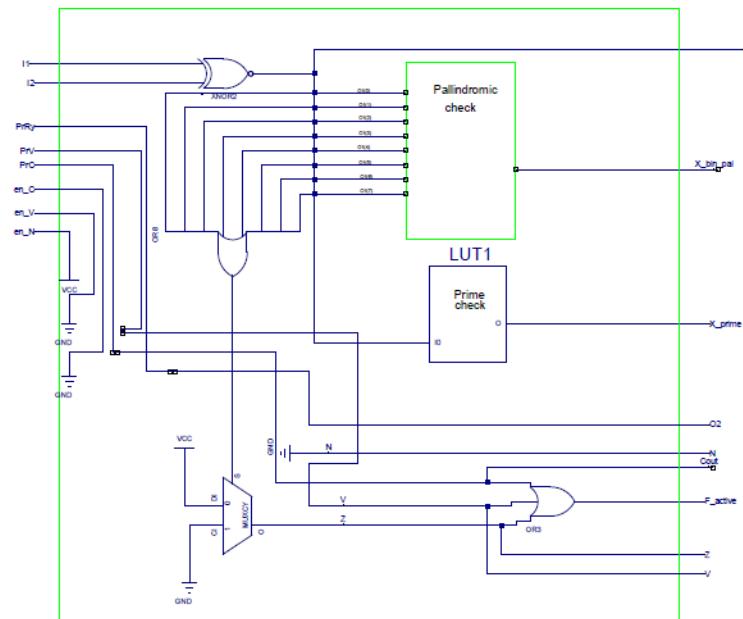
X_bin_pal <= '1' when (F = "1111") else
'0';

-----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;
-----  

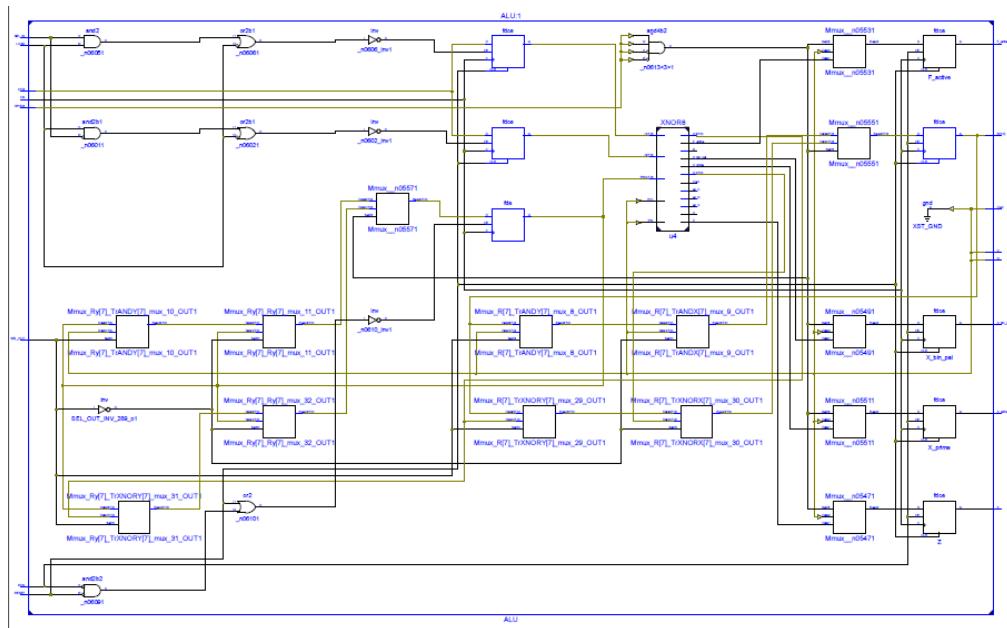
end Behavioral;

```

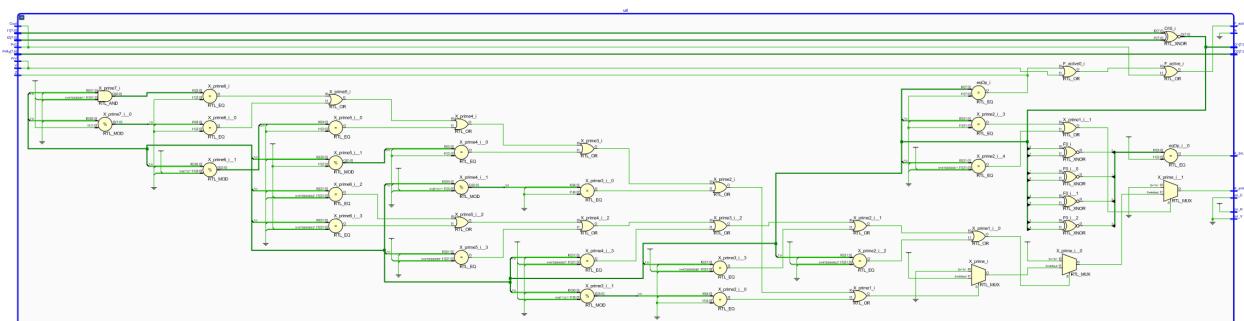
XNOR8



شکل ۱۴: بلوک دیاگرام مازول XNOR



شکل ۱۵: بلوک دیاگرام مازول XNOR به همراه فلیپ‌فلاب‌های ورودی و خروجی



شکل ۱۶: بلوک دیاگرام مازول XNOR (خروجی نرم‌افزار VIVADO)

OPCODE 0100 (Unsigned Addition):

در اینجا از متغیر V و خروجی $O2$ استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در PrV و $PrRy$ ذخیره کرده و در انتهای دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_V را صفر می‌کنیم. از متغیر N و $Cout$ استفاده می‌کنیم پس پایه N و en_N را ۱ قرار می‌دهیم. برای خروجی Z نیز صفر بودن حاصل جمع (S) را بررسی می‌کنیم.

```

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);
           PrV : in STD_LOGIC;
           --Previous Value of V
           PrRy : in STD_LOGIC_VECTOR (7 downto 0);
           --Previous Value of Ry
           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;
           en_C : inout STD_LOGIC;
           en_V : inout STD_LOGIC;
           en_N : 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 <= PrRy;
    en_N <= '1';
    en_V <= '0';
    en_C <= '1';
    N <= '0';
    Cout <= S(8);
    V <= PrV;
    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));

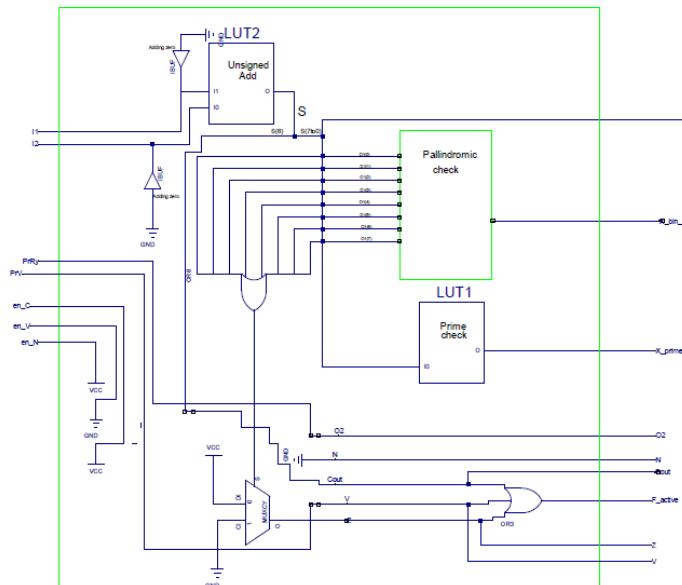
X_bin_pal <= '1' when (F = "1111") else
    '0';

-----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;
-----  

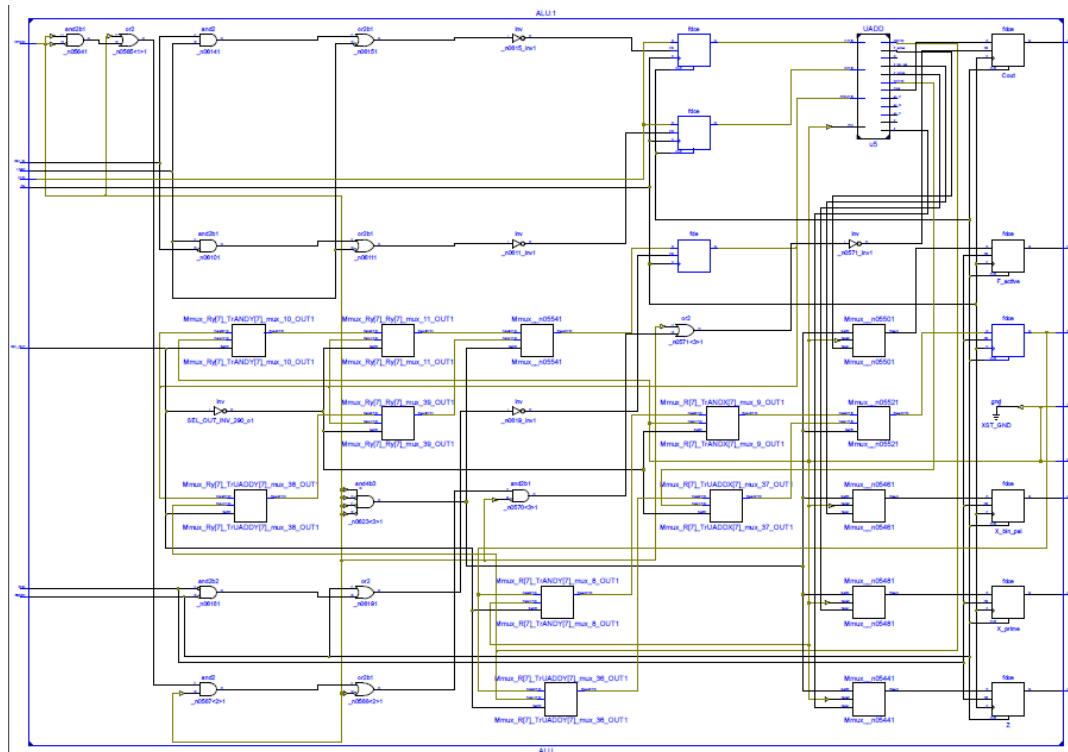
end Behavioral;

```

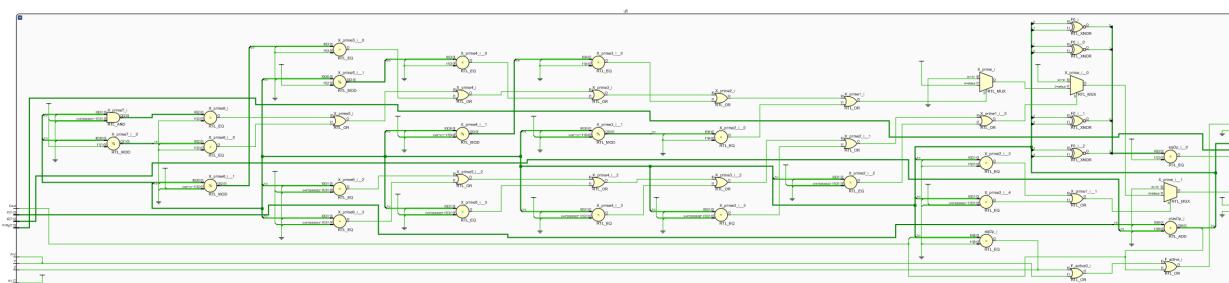
UADD



شکل ۱۷: بلوک دیاگرام مازول Unsigned Addition



شکل ۱۸: بلوک دیاگرام مازول Unsigned Addition به همراه فلیپفلابهای ورودی و خروجی



شکل ۱۹: بلوک دیاگرام مازول Unsigned Addition (خروجی نرمافزار VIVADO)

OPCODE 0101 (Signed Addition):

در اینجا از متغیر Cout و خروجی O2 استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در PrC و PrRy ذخیره کرده و در انتهای دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_C را صفر می‌کنیم. از متغیر N و V استفاده می‌کنیم پس پایه N و en_V را ۱ قرار می‌دهیم. برای خروجی Z نیز صفر بودن حاصل جمع (S) را بررسی می‌کنیم.

```

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);
           PrC : in STD_LOGIC;
           --Previous Value of Cout
           PrRy : in STD_LOGIC_VECTOR (7 downto 0);
           --Previous Value of Ry
           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;
           en_C : inout STD_LOGIC;
           en_V : inout STD_LOGIC;
           en_N : 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 <= PrRy;
    en_N <= '1';
    en_V <= '1';
    en_C <= '0';
    V <= '1' when (SS(8) /= SS(7)) else          --OverFlow
Detection
        '0';

```

```

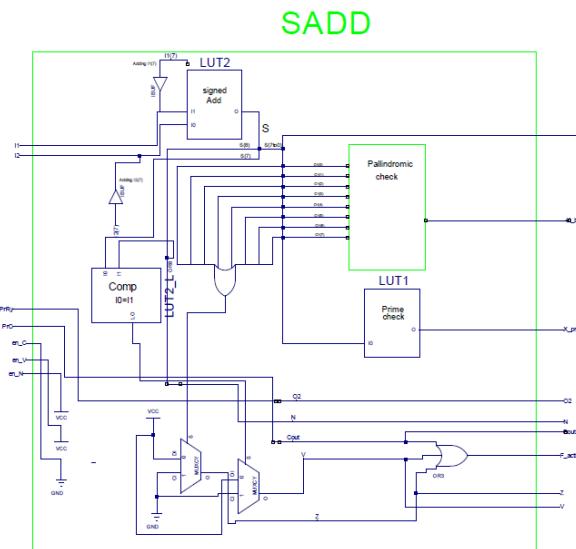
N <= S(8);
Cout <= PrC;
Z <= '1' when S=o"000" else '0';
F_active <= Z or V or Cout;

-----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));

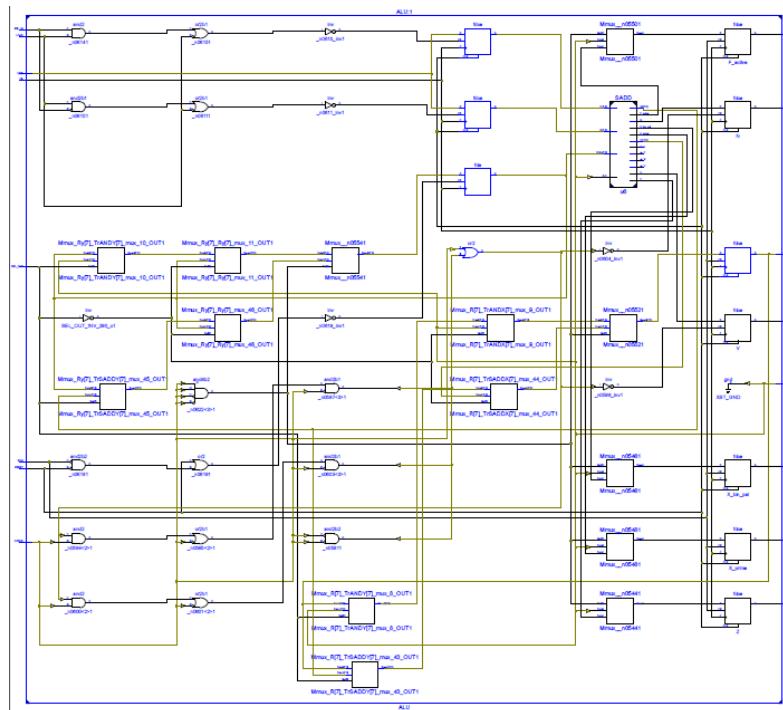
X_bin_pal <= '1' when (F = "1111") else
'0';

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

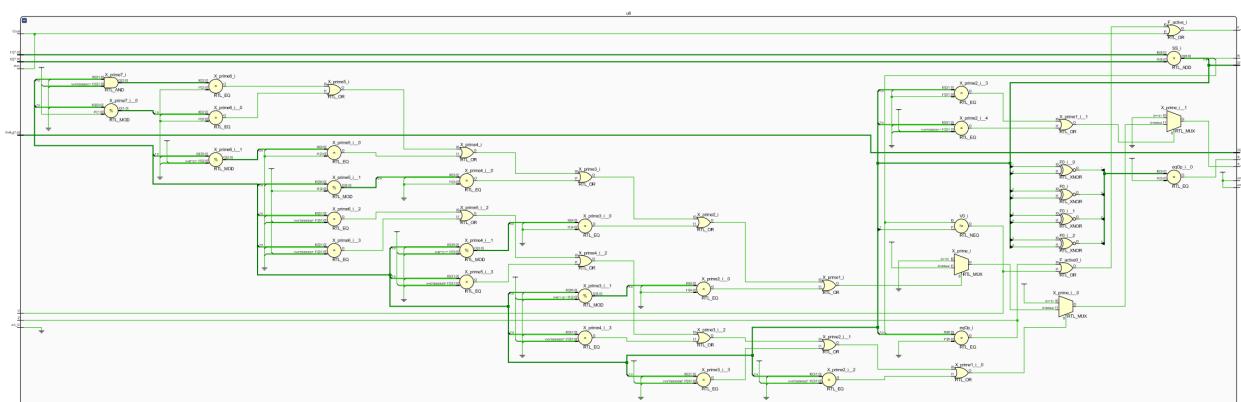
```
end Behavioral;
```



شکل ۲۰: بلوک دیاگرام مازول Signed Addition



شکل ۲۱: بلوک دیاگرام مازول Signed Addition به همراه فلیپ‌فلاپ‌های ورودی و خروجی



شکل ۲۲: بلوک دیاگرام مازول Signed Addition (خروجی نرم‌افزار VIVADO)

OPCODE 0110 (Unsigned Addition with Carry):

در اینجا از متغیر V و خروجی $O2$ استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در PrV و $PrRy$ ذخیره کرده و در انتهای دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_V را صفر می‌کنیم. از متغیر N و $Cout$ استفاده می‌کنیم پس پایه N و en_N را ۱ قرار می‌دهیم. برای خروجی Z نیز صفر بودن حاصل جمع (S) را بررسی می‌کنیم.

```

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;
           PrV : in STD_LOGIC;
--Privious Value of V
           PrRy : in STD_LOGIC_VECTOR (7 downto 0);
--Privious Value of Ry
           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;
           en_C : inout STD_LOGIC;
           en_V : inout STD_LOGIC;
           en_N : 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 <= PrRy;
    en_N <= '1';
    en_V <= '0';
    en_C <= '1';
    N <= '0';
    Cout <= S(8);
    V <= PrV;
    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';

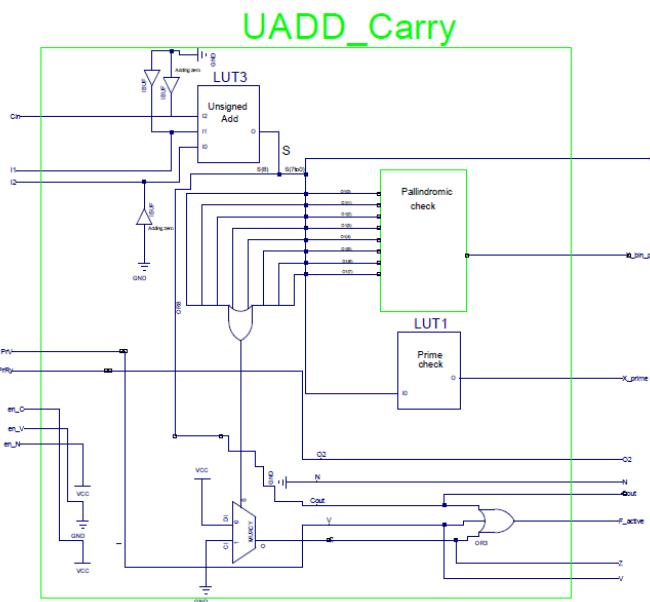
-----Prime check-----
Process (O1)
    variable Num : integer;
begin
    Num := conv_integer(O1);
    if (Num = 0 or Num = 1) then
        X_prime <= '0';
    elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
        X_prime <= '1';
    elsif (Num rem 2 = 0 or
           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

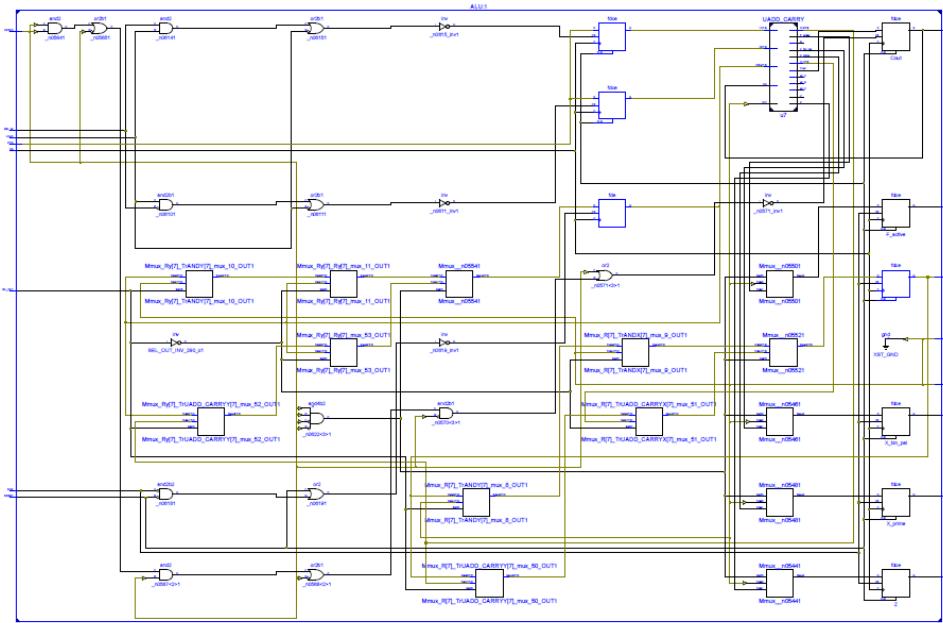
```

Unsigned Addition with Carry

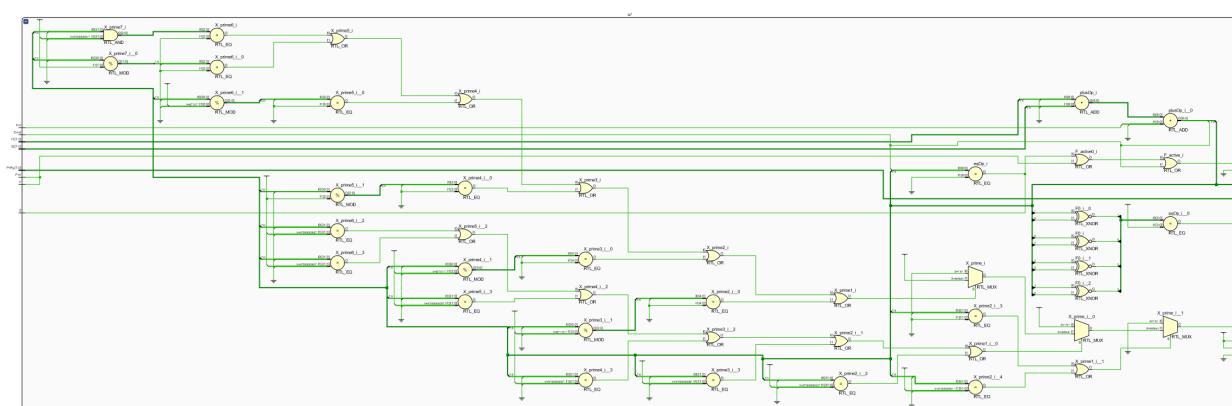
کد شماره‌ی ۷: مازنگل



شکل ۲۳: بلوک دیاگرام مازول Unsigned Addition with Carry



شکل ۲۴: بلوک دیاگرام مازول Unsigned Addition with Carry به همراه فلیپفلاب‌های ورودی و خروجی



شکل ۲۵: بلوک دیاگرام مازول Unsigned Addition with Carry (خروجی نرم‌افزار VIVADO)

OPCODE 0111 (Signed Multiplication):

در اینجا از متغیرهای Cout و PrV استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در C و PrV ذخیره کرده و در انتها دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_C و en_V را صفر می‌کنیم. از خروجی O2 استفاده می‌کنیم پس نیازی به تعریف PrRy نداریم. از متغیر N نیز استفاده می‌کنیم پس پایه en_N را ۱ قرار می‌دهیم. برای خروجی Z نیز صفر بودن حاصل ضرب (M) را بررسی می‌کنیم.

```

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);
            PrC : in STD_LOGIC;
            --Privious Value of Cout
            PrV : in STD_LOGIC;
            --Privious Value of V
            O1 : inout STD_LOGIC_VECTOR (7 downto 0);
            O2 : inout STD_LOGIC_VECTOR (7 downto 0);
            N : out STD_LOGIC;
            Cout : inout STD_LOGIC;
            V : inout STD_LOGIC;
            Z : inout STD_LOGIC;
            en_C : inout STD_LOGIC;
            en_V : inout STD_LOGIC;
            en_N : 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);
    en_N <= '1';
    en_V <= '0';
    en_C <= '0';
    N <= M(15);
    Cout <= PrC;
    V <= PrV;

```

```

Z <= '1' when M=x"0000" else '0';
F_active <= Z or V or Cout;

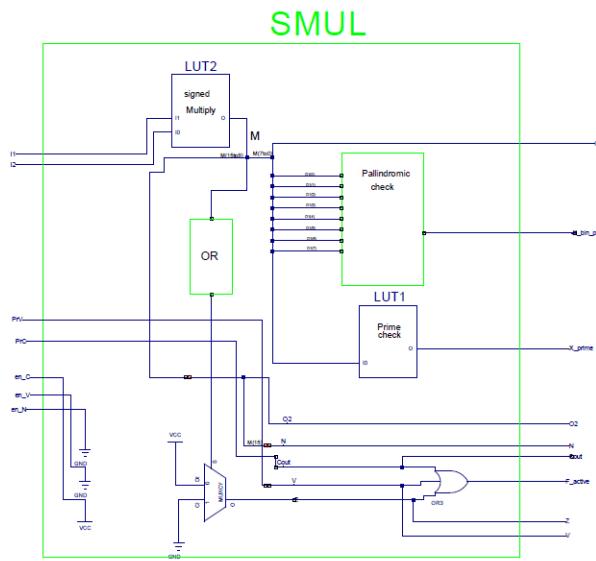
-----Palindromic check-----
F(0) <= (O1(0) xnor O1(7));
F(1) <= (O1(1) xnor O1(6));
F(2) <= (O1(2) xnor O1(5));
F(3) <= (O1(3) xnor O1(4));

X_bin_pal <= '1' when (F = "1111") else
'0';

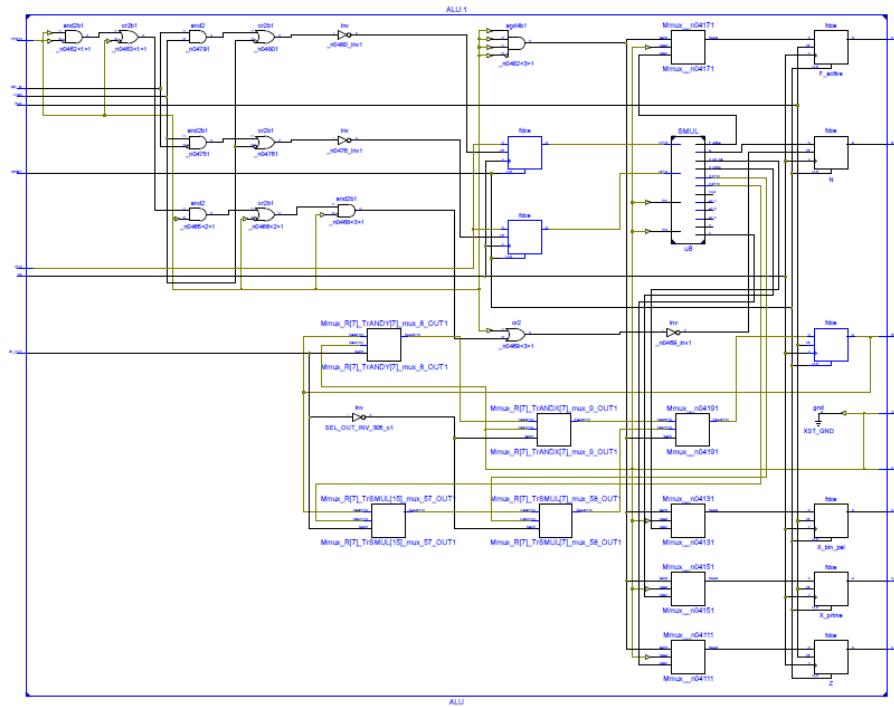
-----Prime check-----
Process (O1)
    variable Num : integer;
begin
    Num := conv_integer(O1);
    if (Num = 0 or Num = 1) then
        X_prime <= '0';
    elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
        X_prime <= '1';
    elsif (Num rem 2 = 0 or
           Num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

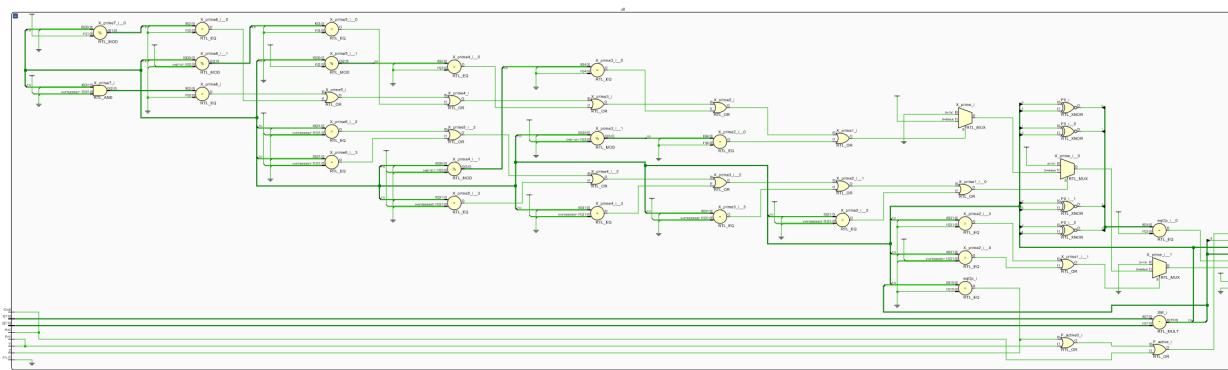
```



شکل ۲۶: بلوک دیاگرام مازول Signed multiplication



شکل ۲۷: بلوک دیاگرام مازول Signed multiplication به همراه فلیپ‌فلاب‌های ورودی و خروجی



شکل ۲۸: بلوک دیاگرام مازول Signed multiplication (خروجی نرم‌افزار VIVADO)

OPCODE 1000 (Unsigned Multiplication):

در اینجا از متغیرهای Cout و PrV استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در PrC و en_C ذخیره کرده و در انتها دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_V و en_C را صفر می‌کنیم. از خروجی O2 استفاده می‌کنیم پس نیازی به تعریف PrRy نداریم. از متغیر N نیز استفاده می‌کنیم (مقدار آن همیشه صفر است چون اعداد بدون علامت هستند) پس پایه en_N را ۱ قرار می‌دهیم. برای خروجی Z نیز صفر بودن حاصل ضرب (M) را بررسی می‌کنیم.

```

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);
           PrC : in STD_LOGIC;
           --Privious Value of Cout
           PrV : in STD_LOGIC;
           --Privious Value of V
           O1 : inout STD_LOGIC_VECTOR (7 downto 0);
           O2 : inout STD_LOGIC_VECTOR (7 downto 0);
           N : out STD_LOGIC;
           Cout : inout STD_LOGIC;
           V : inout STD_LOGIC;
           Z : inout STD_LOGIC;
           en_C : inout STD_LOGIC;
           en_V : inout STD_LOGIC;
           en_N : 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);
    en_N <= '1';
    en_V <= '0';
    en_C <= '0';
    N <= '0';
    Cout <= PrC;
    V <= PrV;

```

```

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));

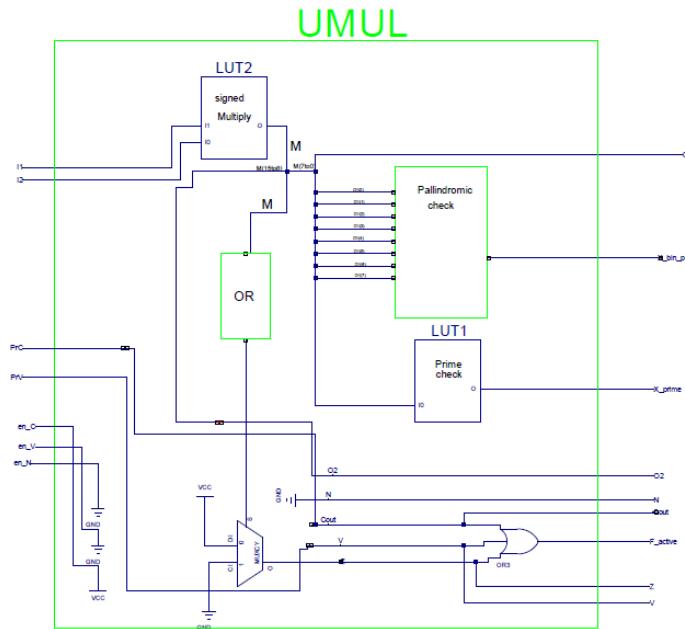
X_bin_pal <= '1' when (F = "1111") else
'0';

-----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;
-----  

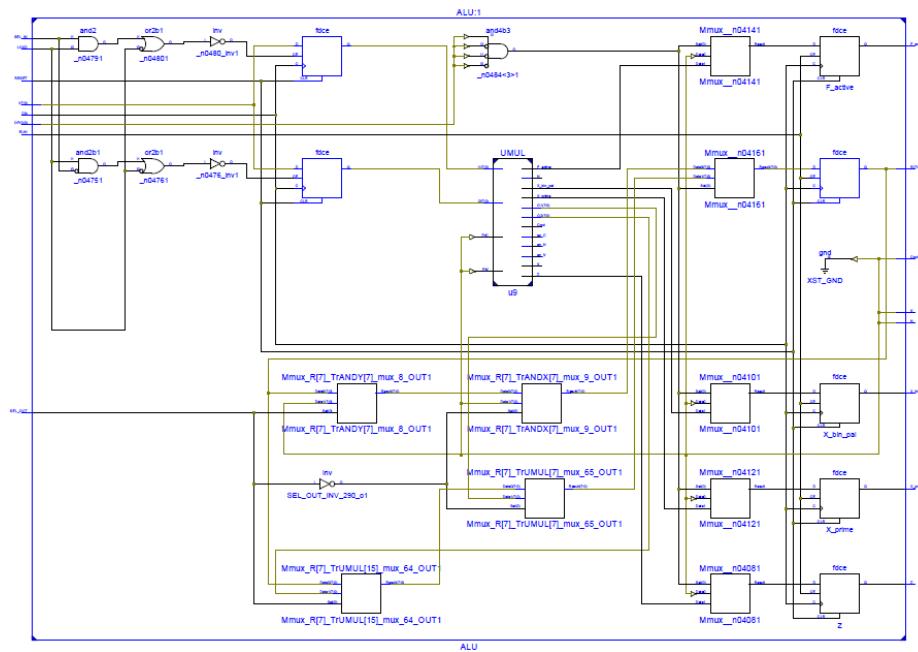
end Behavioral;

```

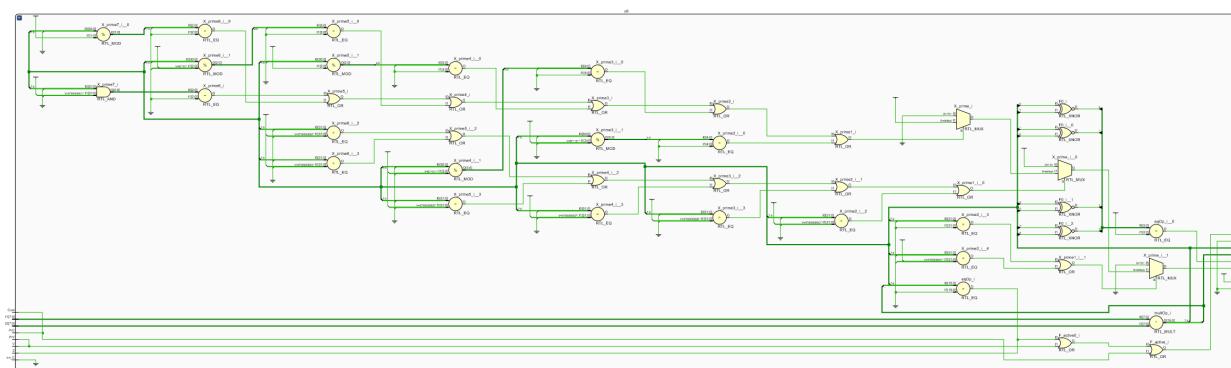
Unsigned multiplication ۹: مازوں کد شمارہ



شکل ۲۹: بلوک دیاگرام مازول Unsigned multiplication



شکل ۳۰: بلوک دیاگرام مازول Unsigned multiplication به همراه فلیپ فلاب های ورودی و خروجی



شکل ۳۱: بلوک دیاگرام مازول Unsigned multiplication (خروجی نرم افزار VIVADO)

OPCODE 1001 (Unsigned Subtraction):

در اینجا از متغیر V و خروجی $O2$ استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در PrV و $PrRy$ ذخیره کرده و در انتهای دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_V را صفر می‌کنیم. از متغیر N و $Cout$ استفاده می‌کنیم پس پایه N و en_N را ۱ قرار می‌دهیم و مقدار $Cout$ برای اعداد خارج از بازه اعداد ۸ بیتی بدون علامت یک می‌شود. برای خروجی Z نیز صفر بودن حاصل تفاضل (S) را بررسی می‌کنیم.

```

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);
           PrV : in STD_LOGIC;
           --Previous Value of V
           PrRy : in STD_LOGIC_VECTOR (7 downto 0);
           --Previous Value of Ry
           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;
           en_C : inout STD_LOGIC;
           en_V : inout STD_LOGIC;
           en_N : 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) + 1;
                Cout <= '0';
            else
                O1 <= not (S(7 downto 0));
                Cout <= '1';
            end if;
        end process;
    end;

```

```

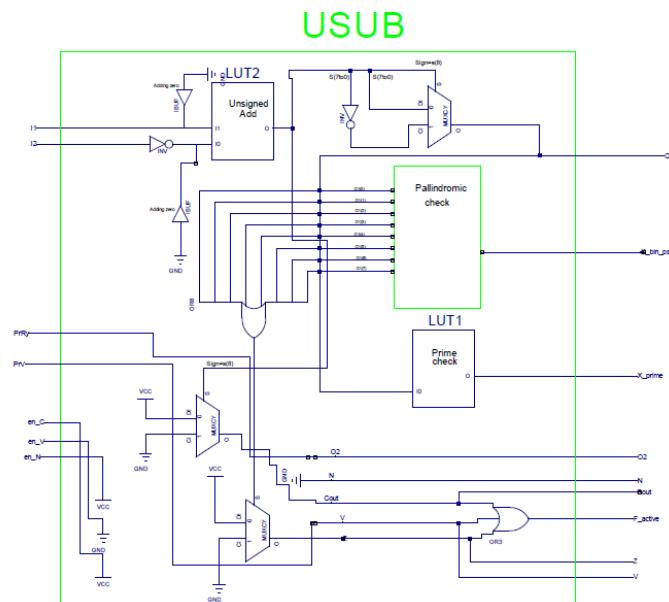
        end if;
    end Process;
O2 <= PrRy;
en_N <= '1';
en_V <= '0';
en_C <= '1';
N <= '0';
V <= PrV;
Z <= '1' when S=o"000" else '0';
F_active <= Z or V or Cout;

-----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));

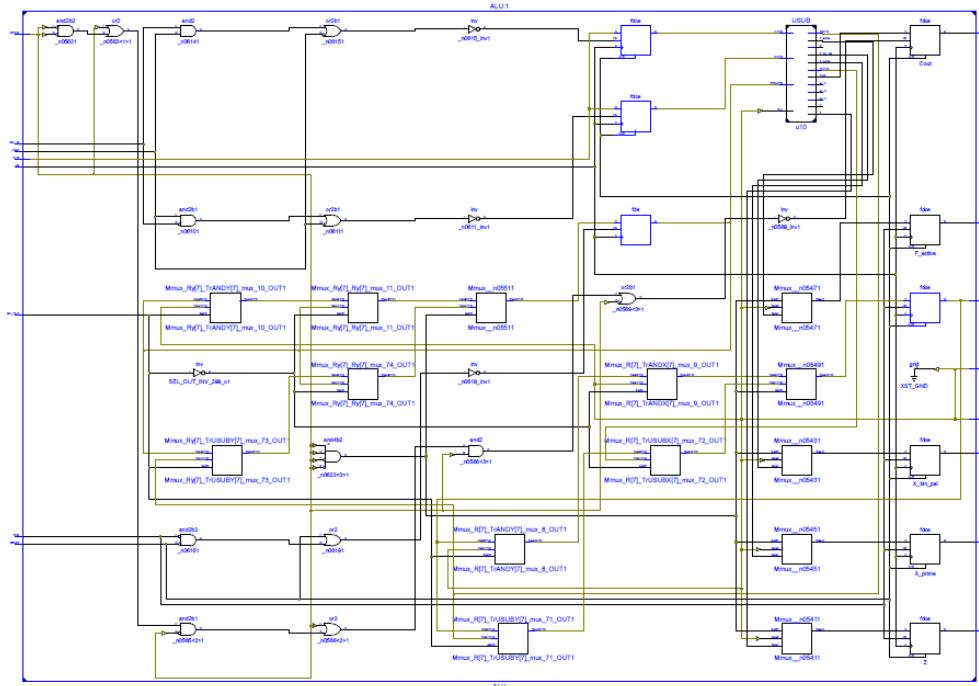
X_bin_pal <= '1' when (F = "1111") else
'0';

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

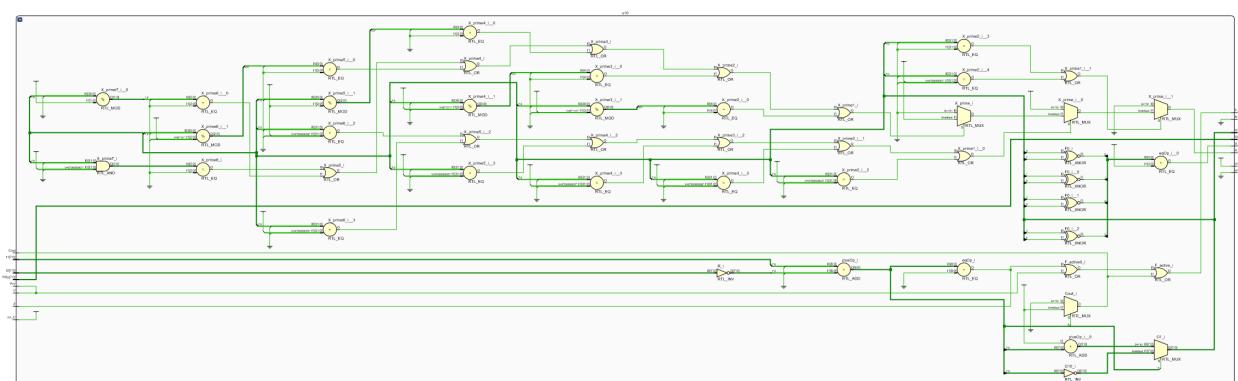
```
end Behavioral;
```



شکل ۳۲: بلوک دیاگرام مازول Unsigned subtraction



شکل ۳۳: بلوک دیاگرام مازول Unsigned subtraction به همراه فلیپ فلابهای ورودی و خروجی



شکل ۳۴: بلوک دیاگرام مازول Unsigned subtraction (خروجی نرم افزار VIVADO)

OPCODE 1010 (Rotation Left):

در اینجا از متغیر V و خروجی $Cout$ و $O2$ استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در PrC ، PrV و Z ذخیره کرده و در انتهای دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_C و en_V را صفر می‌کنیم. از متغیر N استفاده می‌کنیم پس پایه en_N را ۱ قرار می‌دهیم. برای خروجی Z نیز صفر بودن خروجی $O1$ را بررسی می‌کنیم.

```

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);
            PrC : in STD_LOGIC;
            --Previous Value of Cout
            PrV : in STD_LOGIC;
            --Previous Value of V
            PrRy : in STD_LOGIC_VECTOR (7 downto 0);
            --Previous Value of Ry
            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;
            en_C : inout STD_LOGIC;
            en_V : inout STD_LOGIC;
            en_N : 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);
    O2 <= PrRy;
    en_N <= '1';
    en_V <= '0';
    en_C <= '0';
    N <= '0';
    Cout <= PrC;
    V <= PrV;
    Z <= '1' when O1=x"00" else '0';
    F_active <= Z or V or Cout;

```

```

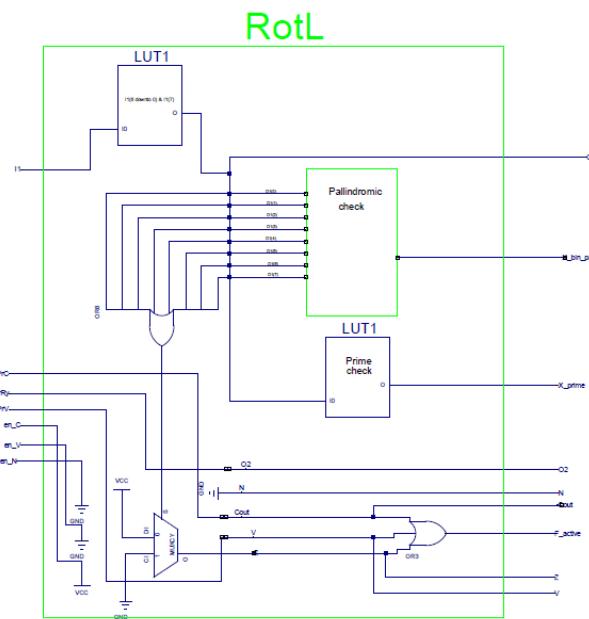
-----Palindromic check-----
F(0) <= (O1(0) xnor O1(7));
F(1) <= (O1(1) xnor O1(6));
F(2) <= (O1(2) xnor O1(5));
F(3) <= (O1(3) xnor O1(4));

X_bin_pal <= '1' when (F = "1111") else
    '0';

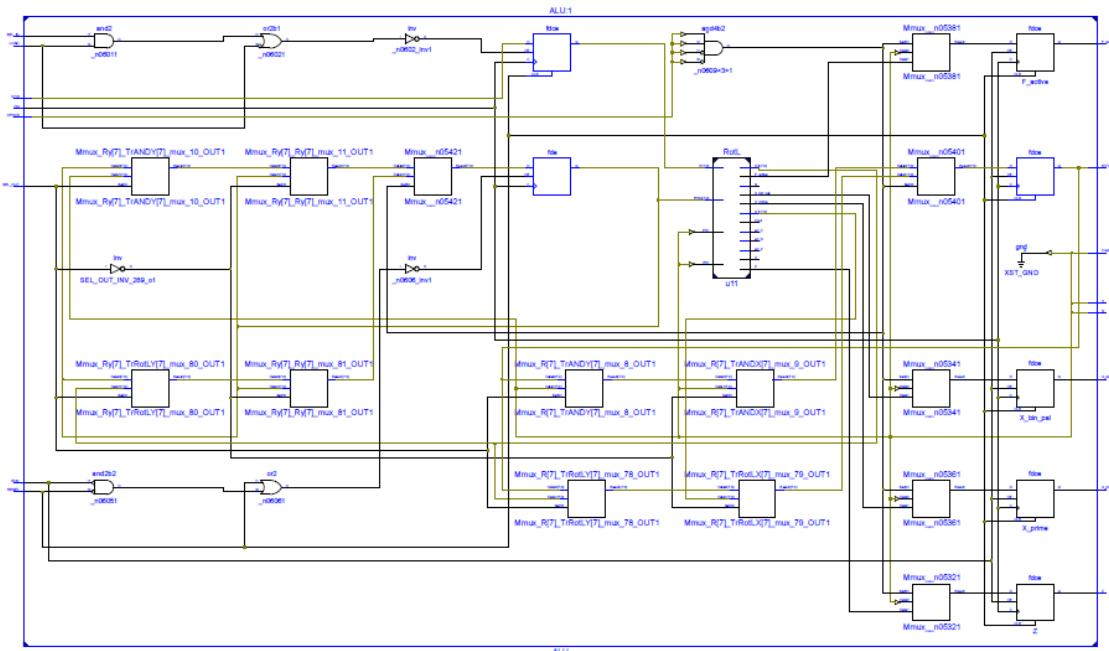
-----Prime check-----
Process (O1)
    variable Num : integer;
begin
    Num := conv_integer(O1);
    if (Num = 0 or Num = 1) then
        X_prime <= '0';
    elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
        X_prime <= '1';
    elsif (Num rem 2 = 0 or
           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

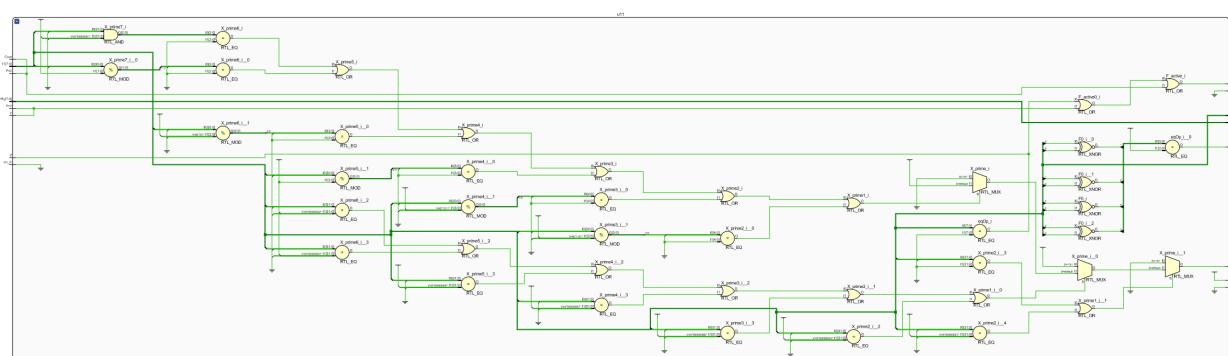
Rotation left ۱۱: مازوچ کد شماره‌ی



شکل ۳۴: بلوک دیاگرام مازول Rotation left



شکل ۳۵: بلوک دیاگرام مازول Rotation left به همراه فلیپ فلاب های ورودی و خروجی



شکل ۳۶: بلوک دیاگرام مازول Rotation left (خروجی نرم افزار VIVADO)

OPCODE 1011 (Rotation Left with Carry):

در اینجا از متغیر V و خروجی $O2$ استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در PrV و $PrRy$ ذخیره کرده و در انتهای دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_V را صفر می‌کنیم. از متغیر N و $Cout$ استفاده می‌کنیم پس پایه N و en_N را ۱ قرار می‌دهیم. برای خروجی Z نیز صفر بودن خروجی $O1$ را بررسی می‌کنیم.

```

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);
           PrV : in STD_LOGIC;
           --Previous Value of V
           PrRy : in STD_LOGIC_VECTOR (7 downto 0);
           --Previous Value of Ry
           O1 : inout STD_LOGIC_VECTOR (7 downto 0);
           O2 : out STD_LOGIC_VECTOR (7 downto 0);
           Cin : in STD_LOGIC;
           N : out STD_LOGIC;
           Cout : inout STD_LOGIC;
           V : inout STD_LOGIC;
           Z : inout STD_LOGIC;
           en_C : inout STD_LOGIC;
           en_V : inout STD_LOGIC;
           en_N : 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;
    O2 <= PrRy;
    en_N <= '1';
    en_V <= '0';
    en_C <= '1';
    N <= '0';
    Cout <= I1(7);
    V <= PrV;
    Z <= '1' when (Cout & O1 =o"000") else '0';
    F_active <= Z or V or Cout;
    -----
    -----Palindromic check-----

```

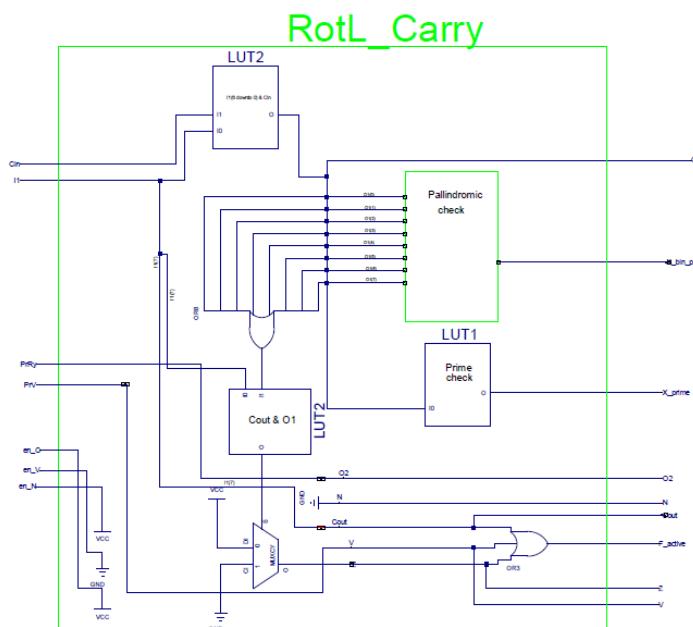
```

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));

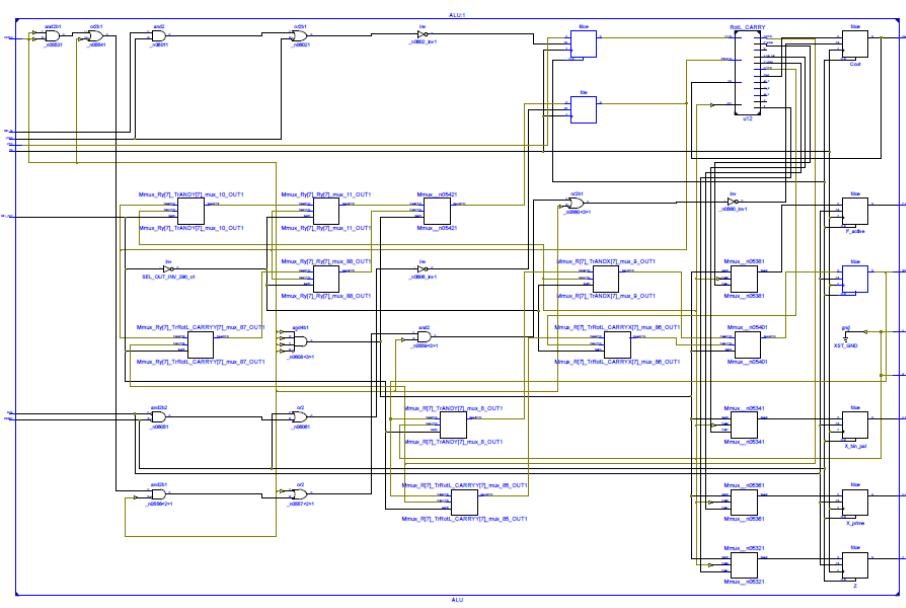
X_bin_pal <= '1' when (F = "1111") else
'0';
-----
-----Prime check-----
Process (O1)
    variable Num : integer;
begin
    Num := conv_integer(O1);
    if (Num = 0 or Num = 1) then
        X_prime <= '0';
    elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
        X_prime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

```

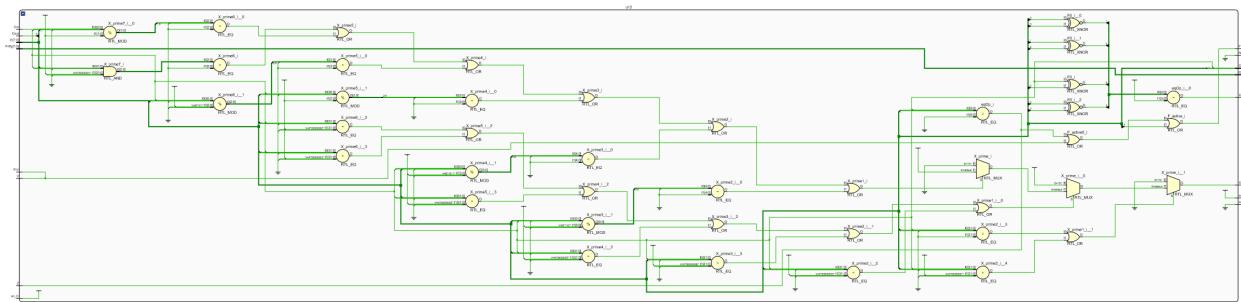
کد شماره‌ی ۱۲: مازوول Rotation left with Carry



شکل ۳۸: پلوک دیاگرام مازول



شکل ۳۹: بلوک دیاگرام مازول Rotation left with Carry به همراه فلیپ‌فلاپ‌های ورودی و خروجی



شکل ۴۰: بلوک دیاگرام مازول Rotation left with Carry (خروجی نرم‌افزار VIVADO)

OPCODE 1100 (Logic Shift Right):

در اینجا از متغیر V و خروجی $O2$ استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در PrV و $PrRy$ ذخیره کرده و در انتهای دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_V را صفر می‌کنیم. از متغیر N و $Cout$ نیز استفاده می‌کنیم (مقدار N همیشه صفر و $Cout$ برابر بیت خارج شده از ورودی در اثر شیفت) پس پایه en_N و en_C را ۱ قرار می‌دهیم. برای خروجی Z نیز صفر بودن خروجی $O1$ را بررسی می‌کنیم.

```

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);
           PrV : in STD_LOGIC;
           --Privious Value of V
           PrRy : in STD_LOGIC_VECTOR (7 downto 0);
           --Privious Value of Ry
           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;
        en_C : inout STD_LOGIC;
        en_V : inout STD_LOGIC;
        en_N : 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);
    O2 <= PrRy;
    en_N <= '1';
    en_V <= '0';
    en_C <= '1';
    N <= '0';
    Cout <= I1(0);
    V <= PrV;
    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

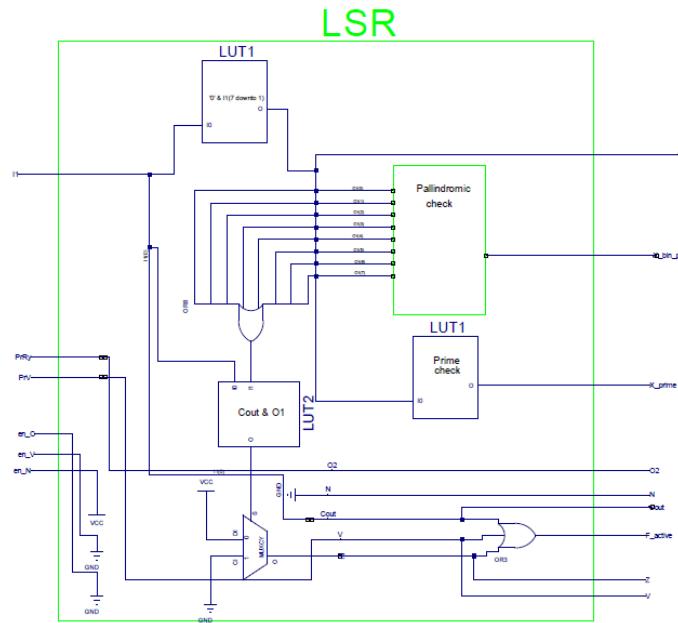
```

```

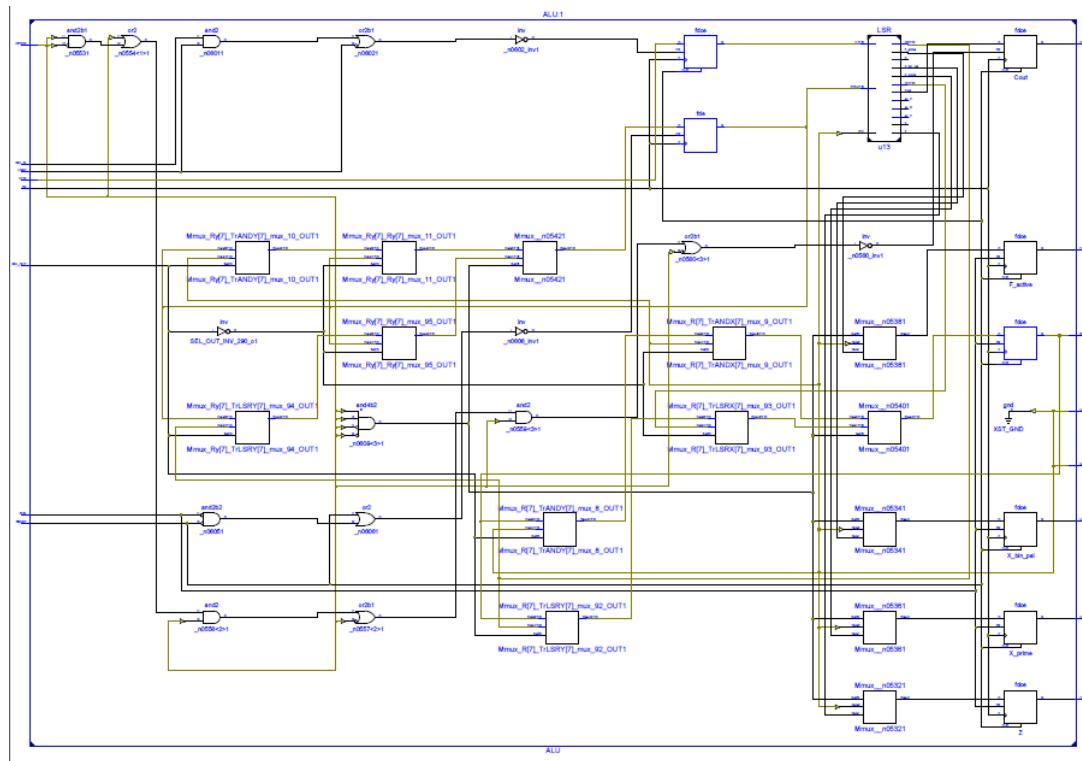
        Num rem 13 = 0) then
            X_prime <= '0';
        else
            X_prime <= '1';
        end if;
    end Process;
-----
end Behavioral;

```

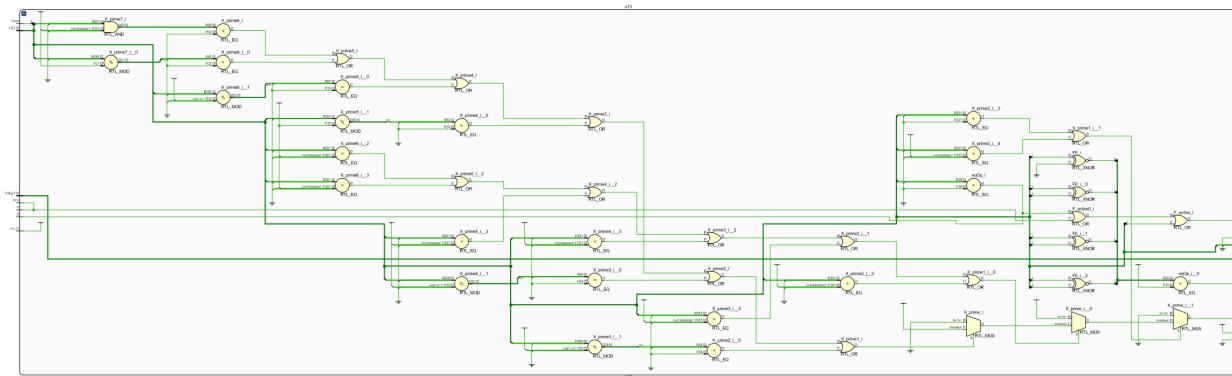
کد شماره‌ی ۱۳: مازول Logic shift right



شکل ۴۱: بلوک دیاگرام مازول Logic shift right



شکل ۴۲: بلوک دیاگرام مازول Logic shift right به همراه فلیپ‌فلاپ‌های ورودی و خروجی



شکل ۴۳: بلوک دیاگرام مازول VIVADO (خروجی نرم افزار Logic shift right)

OPCODE 1101 (Arithmetic Shift Right):

در اینجا از متغیر V و خروجی $O2$ استفاده نمی کنیم. پس مقدار قبلی آنها را در PrV و $PrRy$ ذخیره کرده و در انتهای دوباره در خود آنها می ریزیم و همچنین پایه en_V را صفر می کنیم. از متغیر N و $Cout$ استفاده می کنیم (مقدار N برابر با MSB خروجی $O1$ و $Cout$ برابر بیت خارج شده از ورودی در اثر شیفت) پس پایه $Cout \& O1$ نیز صفر بودن خروجی Z را بررسی می کنیم.

```

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);
           PrV : in STD_LOGIC;
           --Privious Value of V
           PrRy : in STD_LOGIC_VECTOR (7 downto 0);
           --Privious Value of Ry
           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;
           en_C : inout STD_LOGIC;
           en_V : inout STD_LOGIC;
           en_N : 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);
O2 <= PrRy;
en_N <= '1';
en_V <= '0';
en_C <= '1';
N <= I1(7);
Cout <= I1(0);
V <= PrV;
Z <= '1' when (Cout & O1 =o"000") else '0';
F_active <= Z or V or Cout;

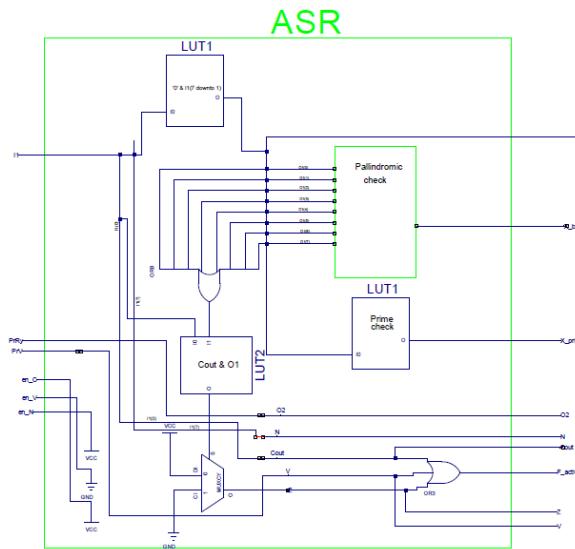
-----Palindromic check-----
F(0) <= (O1(0) xnor O1(7));
F(1) <= (O1(1) xnor O1(6));
F(2) <= (O1(2) xnor O1(5));
F(3) <= (O1(3) xnor O1(4));

X_bin_pal <= '1' when (F = "1111") else
'0';

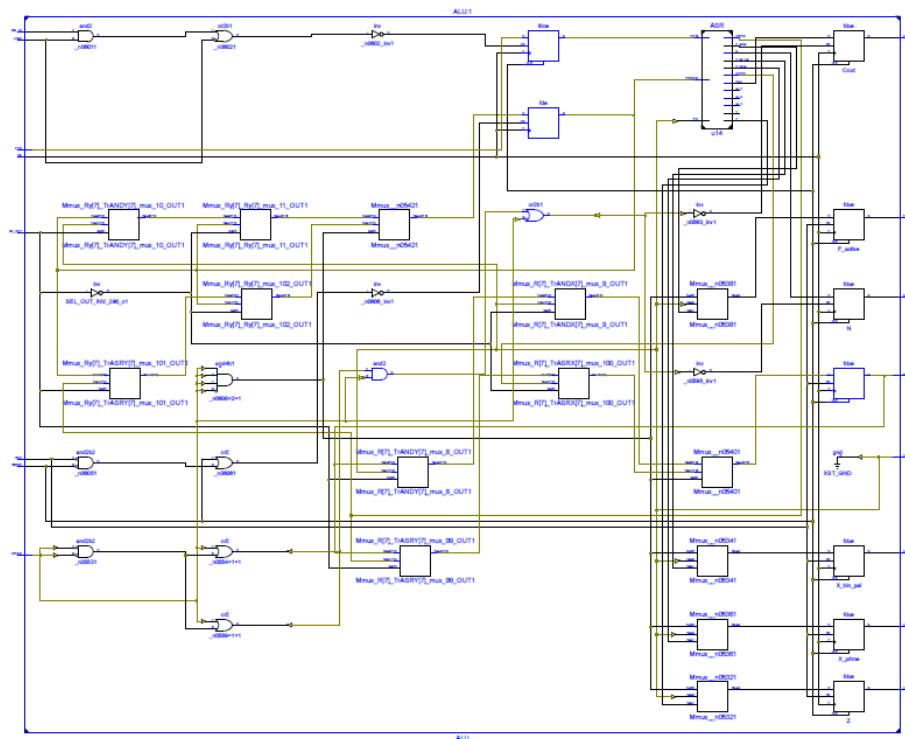
-----Prime check-----
Process (O1)
    variable Num : integer;
begin
    Num := conv_integer(O1);
    if (Num = 0 or Num = 1) then
        X_prime <= '0';
    elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
        X_prime <= '1';
    elsif (Num rem 2 = 0 or
           Num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

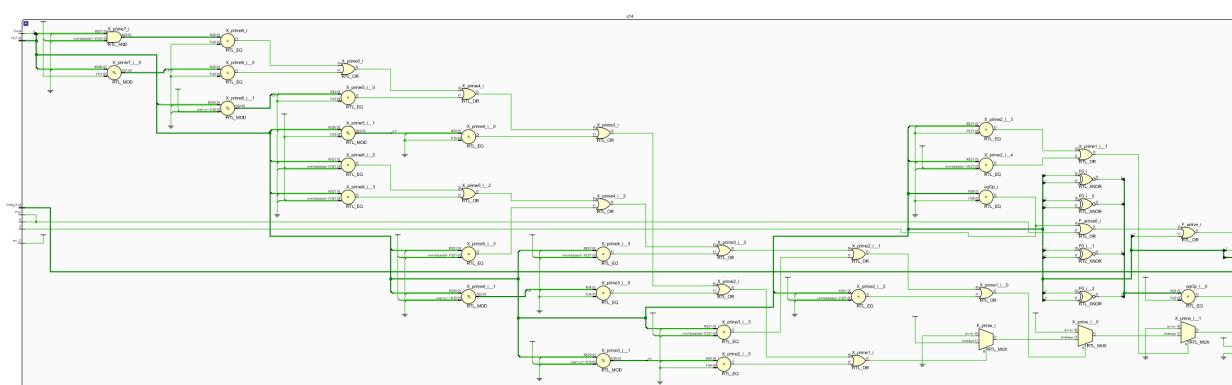
```



شکل ۴۴: بلوک دیاگرام مازول Arithmetic shift right



شکل ۴۵: بلوک دیاگرام مازول Arithmetic shift right به همراه فلیپ فلاب های ورودی و خروجی



شکل ۴۶: بلوک دیاگرام مازول Arithmetic shift right (خروجی نرم افزار VIVADO)

OPCODE 1110 (Logic Shift Left):

در اینجا از متغیر V و خروجی $O2$ استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در PrV و $PrRy$ ذخیره کرده و در انتهای دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_V را با توجه به فایل پروژه ۱ می‌کنیم ولی مقدار خروجی آن را PrV می‌گذاریم، چون این OPCODE تاثیری بر خروجی V ندارد. از متغیر N و $Cout$ استفاده می‌کنیم (مقدار N همیشه صفر و $Cout$ برابر بیت خارج شده از ورودی در اثر شیفت) پس پایه en_N و en_C را ۱ قرار می‌دهیم. برای خروجی Z نیز صفر بودن خروجی $Cout \& O1$ را بررسی می‌کنیم.

```

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);
            PrV : in STD_LOGIC;
            --Previous Value of V
            PrRy : in STD_LOGIC_VECTOR (7 downto 0);
            --Previous Value of Ry
            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;
            en_C : inout STD_LOGIC;
            en_V : inout STD_LOGIC;
            en_N : 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';
    O2 <= PrRy;
    en_N <= '1';
    en_V <= '1'; --Wrong in the Description File
    en_C <= '1';
    N <= I1(6);
    Cout <= I1(7);
    V <= PrV;
    Z <= '1' when (Cout & O1 =o"000") else '0';
    F_active <= Z or V or Cout;

```

```

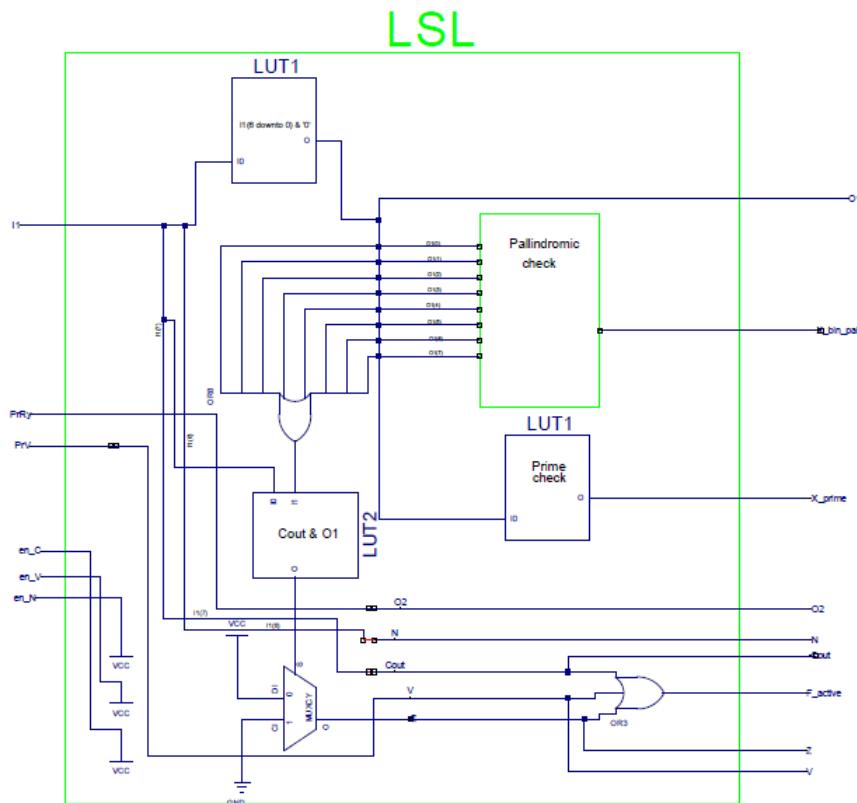
-----Palindromic check-----
F(0) <= (O1(0) xnor O1(7));
F(1) <= (O1(1) xnor O1(6));
F(2) <= (O1(2) xnor O1(5));
F(3) <= (O1(3) xnor O1(4));

X_bin_pal <= '1' when (F = "1111") else
'0';

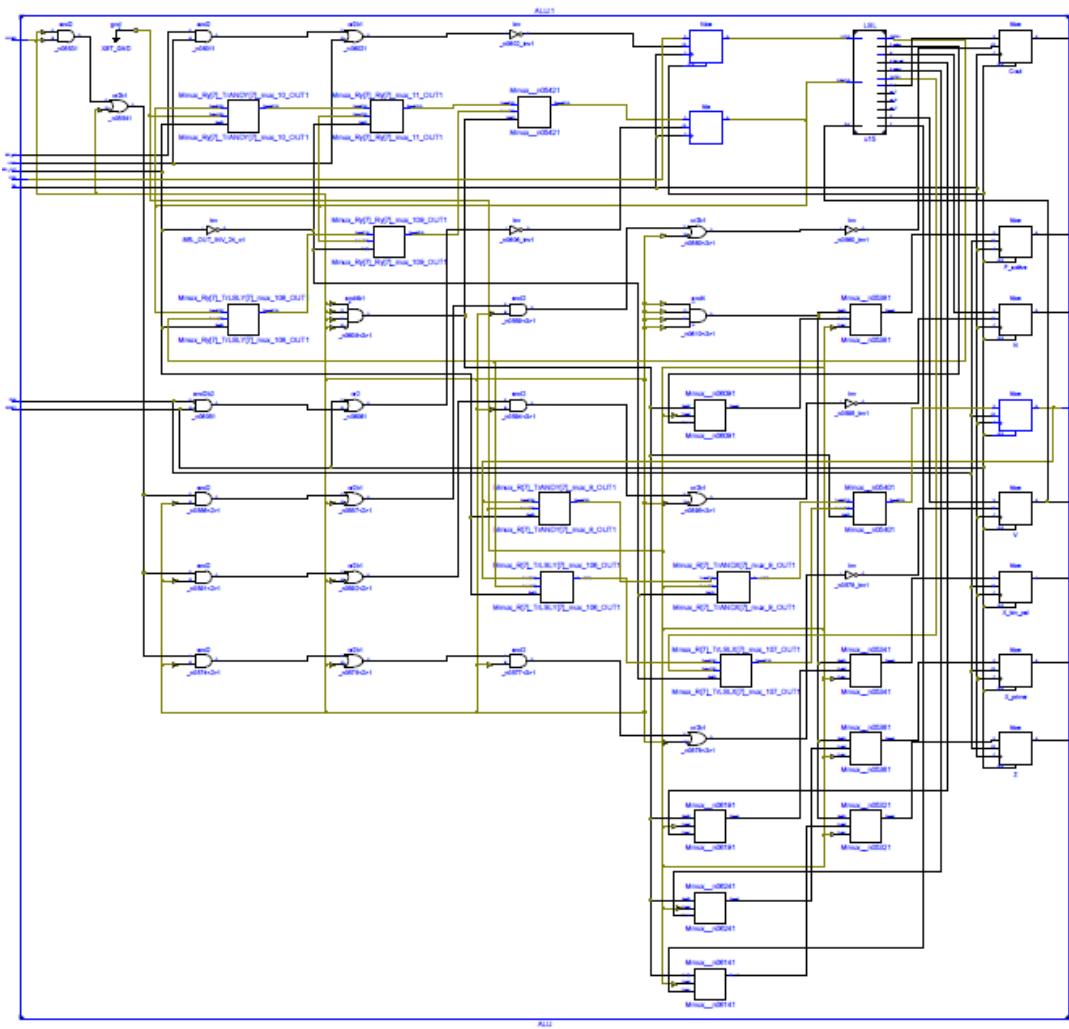
-----Prime check-----
Process (O1)
    variable Num : integer;
begin
    Num := conv_integer(O1);
    if (Num = 2) then
        X_prime <= '1';
    else
        X_prime <= '0';
    end if;
end Process;
-- The Shifted output is always Multiple of 2
-----
```

end Behavioral;

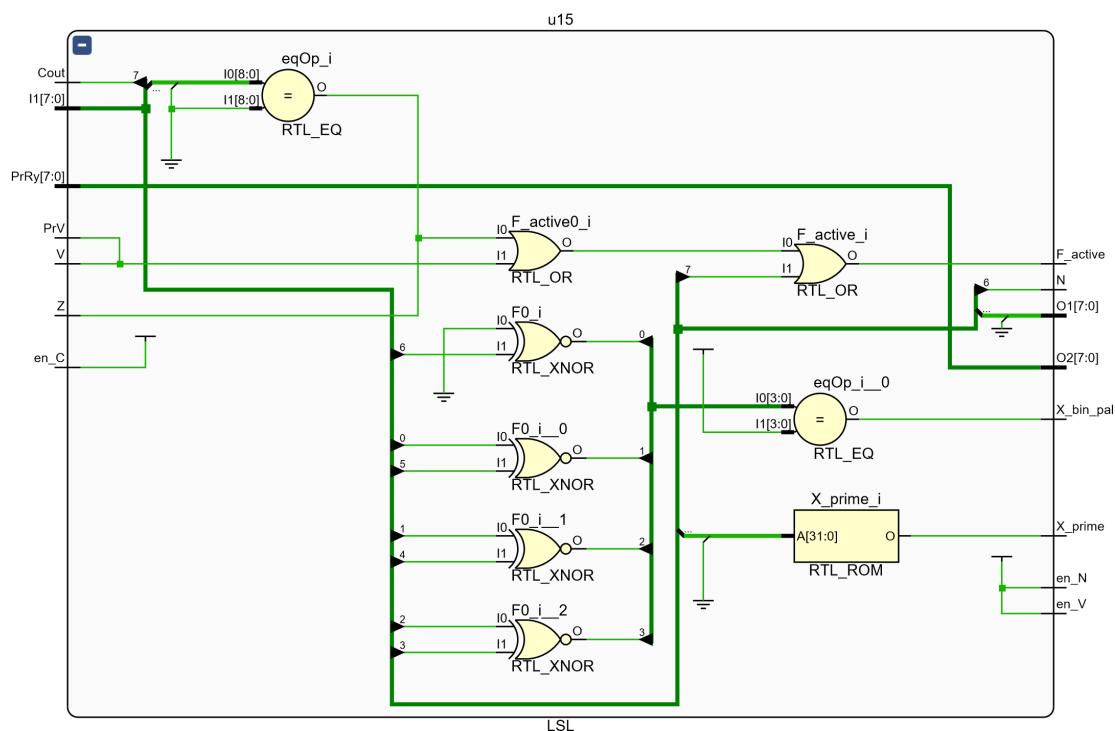
کد شماره ۱۵: مازول Logic shift left



شکل ۴۷: بلوک دیاگرام مازول Logic shift left



شکل ۴۱: بلوک دیاگرام مازول Logic shift left به همراه فلیپ فلاب های ورودی و خروجی



شکل ۴۹: بلوک دیاگرام مازول Logic shift left (خروجی نرم افزار VIVADO)

OPCODE 1111 (BCD to Binary Conversion):

در اینجا از متغیرهای V و N استفاده نمی‌کنیم. پس مقدار قبلی آن‌ها را در PrC و PrN ذخیره کرده و در انتها دوباره در خود آن‌ها می‌ریزیم و همچنین پایه en_N و en_V را صفر می‌کنیم. از خروجی $O2$ استفاده می‌کنیم پس نیازی به تعریف $PrRy$ نداریم. از خروجی $Cout$ نیز استفاده می‌کنیم (مقدار آن برابر خروجی دوم است) پس پایه en_C را ۱ قرار می‌دهیم. برای خروجی Z نیز صفر بودن حاصل تبدیل (S) را بررسی می‌کنیم.

```

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);
           PrN : in STD_LOGIC;
           --Previous Value of V
           PrV : in STD_LOGIC;
           --Previous Value of V
           O1 : inout STD_LOGIC_VECTOR (7 downto 0);
           O2 : inout STD_LOGIC_VECTOR (7 downto 0);
           N : out STD_LOGIC;
           Cout : inout STD_LOGIC;
           V : inout STD_LOGIC;
           Z : inout STD_LOGIC;
           en_C : inout STD_LOGIC;
           en_V : inout STD_LOGIC;
           en_N : 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);
en_N <= '0';
en_V <= '0';
en_C <= '1';
N <= PrN;
Cout <= S(15);
V <= PrV;
Z <= '1' when S=x"0000" else '0';
F_active <= Z or V or Cout;

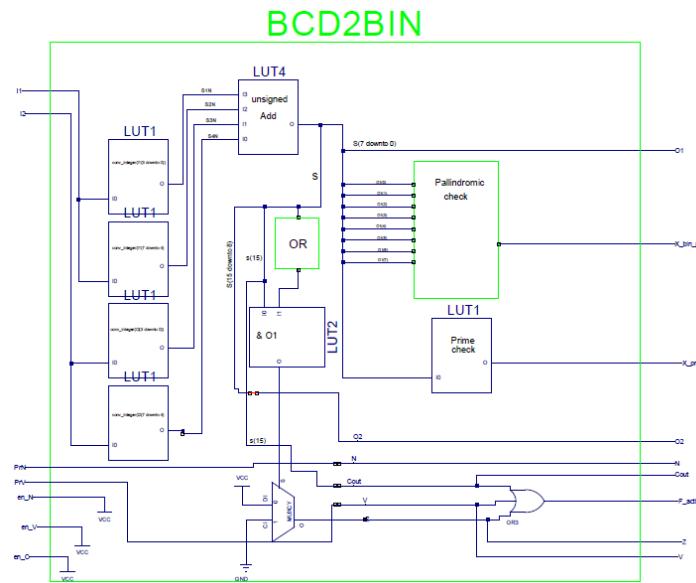
-----Palindromic check-----
F(0) <= (O1(0) xnor O1(7));
F(1) <= (O1(1) xnor O1(6));
F(2) <= (O1(2) xnor O1(5));
F(3) <= (O1(3) xnor O1(4));

X_bin_pal <= '1' when (F = "1111") else
    '0';

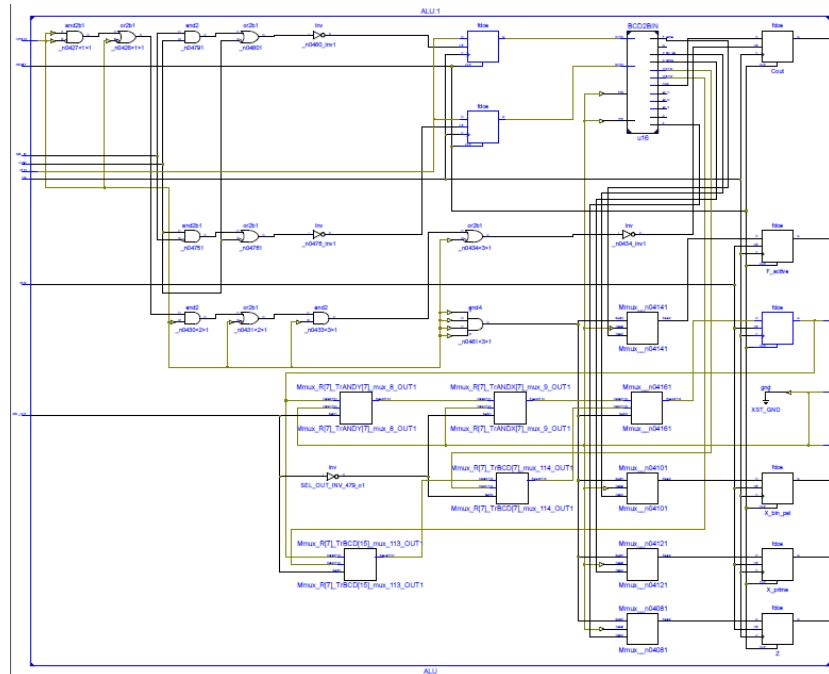
-----Prime check-----
Process (O1)
    variable Num : integer;
begin
    Num := conv_integer(O1);
    if (Num = 0 or Num = 1) then
        X_prime <= '0';
    elsif (Num = 2 or Num = 3 or Num = 5 or Num = 7
or Num = 11 or Num = 13) then
        X_prime <= '1';
    elsif (Num rem 2 = 0 or
           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

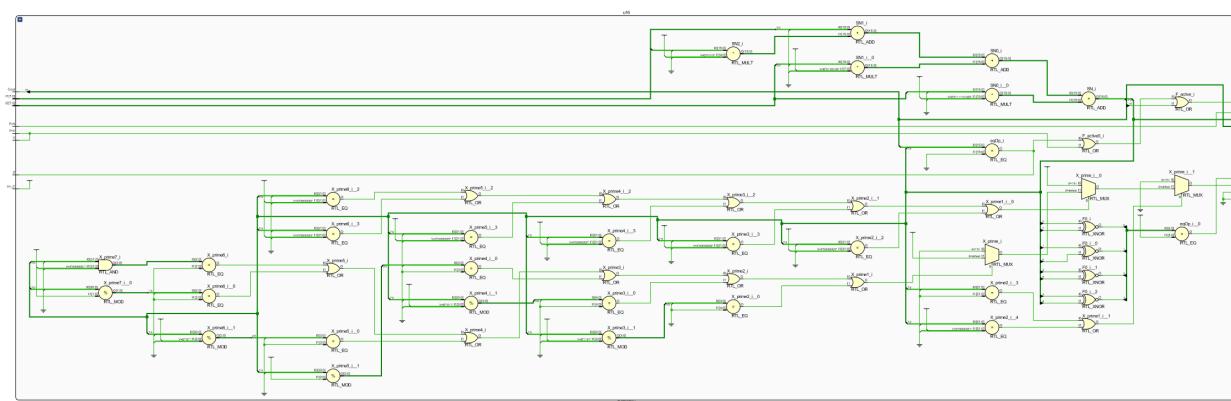
```



شکل ۵: بلوک دیاگرام مازول BCD to Binary



شکل ۶: بلوک دیاگرام مازول BCD to Binary به همراه فلیپ فلاب های ورودی و خروجی



شکل ۷: بلوک دیاگرام مازول BCD to Binary (خروجی نرم افزار 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. با توجه به حجم بالای بخش های آنها را کنار می گذاریم و فقط بدنہ اصلی کد را بررسی می کنیم.

Process ابتدایی کد مربوط به دریافت ورودی می باشد که با توجه به این که ۱ باشد و پایه **IN SEL_IN** **LOAD** ۱ باشد و پایه **IN**

چه مقداری باشد ورودی را در **IA** یا **IB** قرار می دهد و اگر مقدار **RESET** ۱ باشد هر دو ورودی را صفر می کند.

Process دوم مربوط به دریافت **OPCODE** و زیرمأژول ها می باشد و اگر مقدار **RESET** ۱ باشد تمامی متغیرهای خروجی صفر خواهند شد. در هر زیرمأژول با توجه به اینکه کدام پایه های **enable** یک هستند خروجی ها به خروجی فلیپ فلاپ ها (خروچی کلی) منتقل می شود.

بر اساس مقدار **SEL_OUT** مشخص می شود که کدام خروجی (خروچی اول **X** یا خروجی دوم **Y**) در خروجی اصلی (**R**) ریخته شود.

```

begin

    Process (SEL_IN, I, Clk, RESET)
        begin
            if (RESET = '1') then
                IA <= x"00";
                IB <= x"00";
            elsif rising_edge(Clk) then
                if (Load = '1') then
                    if (SEL_IN = '0') then
                        IA <= I;
                    elsif (SEL_IN = '1') then
                        IB <= I;
                    end if;
                end if;
            end if;
        end Process;

    Process (OP, Clk, RESET, 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,
                           TrLSLY, TrBCD, TrN, TrCout, TrV, TrZ,
TrBinPal, TrPrime, TrActive, Tren_C, Tren_V, Tren_N)
        begin
            if (RESET = '1') then
                R <= (OTHERS => '0');
                Cout <= '0';
                Z <= '0';
                V <= '0';
                N <= '0';

```

```

        X_prime <= '0';
        X_bin_pal <= '0';
        F_active <= '0';
    elsif rising_edge(Clk) then
        if (OP = "0000") then
--AND Operation
            if (Tren_C(0) = '1') then
                Cout <= TrCout(0);
            end if;
            if (Tren_V(0) = '1') then
                V <= TrV(0);
            end if;
            if (Tren_N(0) = '1') then
                N <= TrN(0);
            end if;
            if (RUN = '1') then
                Z <= TrZ(0);
                X_bin_pal <= TrBinPal(0);
                X_prime <= TrPrime(0);
                F_active <= TrActive(0);
                if (SEL_OUT = '0') then
                    R <= TrANDX;
                elsif (SEL_OUT = '1') then
                    R <= TrANDY;
                    Ry <= TrANDY;
                end if;
            end if;

            elsif (OP = "0001") then
--OR Operation
                if (Tren_C(1) = '1') then
                    Cout <= TrCout(1);
                end if;
                if (Tren_V(1) = '1') then
                    V <= TrV(1);
                end if;
                if (Tren_N(1) = '1') then
                    N <= TrN(1);
                end if;
                if (RUN = '1') then
                    Z <= TrZ(1);
                    X_bin_pal <= TrBinPal(1);
                    X_prime <= TrPrime(1);
                    F_active <= TrActive(1);
                    if (SEL_OUT = '0') then
                        R <= TrORX;
                    elsif (SEL_OUT = '1') then
                        R <= TrORY;
                        Ry <= TrORY;
                    end if;
                end if;

```

```

        elsif (OP = "0010") then
--XOR Operation
        if (Tren_C(2) = '1') then
            Cout <= TrCout(2);
        end if;
        if (Tren_V(2) = '1') then
            V <= TrV(2);
        end if;
        if (Tren_N(2) = '1') then
            N <= TrN(2);
        end if;
        if (RUN = '1') then
            Z <= TrZ(2);
            X_bin_pal <= TrBinPal(2);
            X_prime <= TrPrime(2);
            F_active <= TrActive(2);
            if (SEL_OUT = '0') then
                R <= TrXORX;
            elsif (SEL_OUT = '1') then
                R <= TrXORY;
                Ry <= TrXORY;
            end if;
        end if;

        elsif (OP = "0011") then
--XNOR Operation
        if (Tren_C(3) = '1') then
            Cout <= TrCout(3);
        end if;
        if (Tren_V(3) = '1') then
            V <= TrV(3);
        end if;
        if (Tren_N(3) = '1') then
            N <= TrN(3);
        end if;
        if (RUN = '1') then
            Z <= TrZ(3);
            X_bin_pal <= TrBinPal(3);
            X_prime <= TrPrime(3);
            F_active <= TrActive(3);
            if (SEL_OUT = '0') then
                R <= TrXNORX;
            elsif (SEL_OUT = '1') then
                R <= TrXNORY;
                Ry <= TrXNORY;
            end if;
        end if;

        elsif (OP = "0100") then
--Unsigned Addition Operation
        if (Tren_C(4) = '1') then
            Cout <= TrCout(4);

```

```

        end if;
        if (Tren_V(4) = '1') then
            V <= TrV(4);
        end if;
        if (Tren_N(4) = '1') then
            N <= TrN(4);
        end if;
        if (RUN = '1') then
            Z <= TrZ(4);
            X_bin_pal <= TrBinPal(4);
            X_prime <= TrPrime(4);
            F_active <= TrActive(4);
            if (SEL_OUT = '0') then
                R <= TrUADDX;
            elsif (SEL_OUT = '1') then
                R <= TrUADDY;
                Ry <= TrUADDY;
            end if;
        end if;

        elsif (OP = "0101") then
--Signed Addition Operation
            if (Tren_C(5) = '1') then
                Cout <= TrCout(5);
            end if;
            if (Tren_V(5) = '1') then
                V <= TrV(5);
            end if;
            if (Tren_N(5) = '1') then
                N <= TrN(5);
            end if;
            if (RUN = '1') then
                Z <= TrZ(5);
                X_bin_pal <= TrBinPal(5);
                X_prime <= TrPrime(5);
                F_active <= TrActive(5);
                if (SEL_OUT = '0') then
                    R <= TrSADDX;
                elsif (SEL_OUT = '1') then
                    R <= TrSADDY;
                    Ry <= TrSADDY;
                end if;
            end if;

            elsif (OP = "0110") then
--Unsigned Addition with Cin Operation
                if (Tren_C(6) = '1') then
                    Cout <= TrCout(6);
                end if;
                if (Tren_V(6) = '1') then
                    V <= TrV(6);
                end if;

```

```

        if (Tren_N(6) = '1') then
            N <= TrN(6);
        end if;
        if (RUN = '1') then
            Z <= TrZ(6);
            X_bin_pal <= TrBinPal(6);
            X_prime <= TrPrime(6);
            F_active <= TrActive(6);
            if (SEL_OUT = '0') then
                R <= TrUADD_CARRYX;
            elsif (SEL_OUT = '1') then
                R <= TrUADD_CARRYY;
                Ry <= TrUADD_CARRYY;
            end if;
        end if;

        elsif (OP = "0111") then
--Signed Multiplication
        if (Tren_C(7) = '1') then
            Cout <= TrCout(7);
        end if;
        if (Tren_V(7) = '1') then
            V <= TrV(7);
        end if;
        if (Tren_N(7) = '1') then
            N <= TrN(7);
        end if;
        if (RUN = '1') then
            Z <= TrZ(7);
            X_bin_pal <= TrBinPal(7);
            X_prime <= TrPrime(7);
            F_active <= TrActive(7);
            if (SEL_OUT = '0') then
                R <= TrSMUL(7 downto 0);
            elsif (SEL_OUT = '1') then
                R <= TrSMUL(15 downto 8);
                Ry <= TrSMUL(15 downto 8);
            end if;
        end if;

        elsif (OP = "1000") then
--UnSigned Multiplication
        if (Tren_C(8) = '1') then
            Cout <= TrCout(8);
        end if;
        if (Tren_V(8) = '1') then
            V <= TrV(8);
        end if;
        if (Tren_N(8) = '1') then
            N <= TrN(8);
        end if;
        if (RUN = '1') then
    
```

```

        Z <= TrZ(8);
        X_bin_pal <= TrBinPal(8);
        X_prime <= TrPrime(8);
        F_active <= TrActive(8);
        if (SEL_OUT = '0') then
            R <= TrUMUL(7 downto 0);
        elsif (SEL_OUT = '1') then
            R <= TrUMUL(15 downto 8);
            Ry <= TrUMUL(15 downto 8);
        end if;
    end if;

    elsif (OP = "1001") then
--Unsigned Subtraction
        if (Tren_C(9) = '1') then
            Cout <= TrCout(9);
        end if;
        if (Tren_V(9) = '1') then
            V <= TrV(9);
        end if;
        if (Tren_N(9) = '1') then
            N <= TrN(9);
        end if;
        if (RUN = '1') then
            Z <= TrZ(9);
            X_bin_pal <= TrBinPal(9);
            X_prime <= TrPrime(9);
            F_active <= TrActive(9);
            if (SEL_OUT = '0') then
                R <= TrUSUBX;
            elsif (SEL_OUT = '1') then
                R <= TrUSUBY;
                Ry <= TrUSUBY;
            end if;
        end if;

    elsif (OP = "1010") then
--Rotation Left
        if (Tren_C(10) = '1') then
            Cout <= TrCout(10);
        end if;
        if (Tren_V(10) = '1') then
            V <= TrV(10);
        end if;
        if (Tren_N(10) = '1') then
            N <= TrN(10);
        end if;
        if (RUN = '1') then
            Z <= TrZ(10);
            X_bin_pal <= TrBinPal(10);
            X_prime <= TrPrime(10);
            F_active <= TrActive(10);
        end if;
    end if;

```

```

        if (SEL_OUT = '0') then
            R <= TrRotLX;
        elsif (SEL_OUT = '1') then
            R <= TrRotLY;
            Ry <= TrRotLY;
        end if;
    end if;

    elsif (OP = "1011") then
--Rotation Left with Carry
        if (Tren_C(11) = '1') then
            Cout <= TrCout(11);
        end if;
        if (Tren_V(11) = '1') then
            V <= TrV(11);
        end if;
        if (Tren_N(11) = '1') then
            N <= TrN(11);
        end if;
        if (RUN = '1') then
            Z <= TrZ(11);
            X_bin_pal <= TrBinPal(11);
            X_prime <= TrPrime(11);
            F_active <= TrActive(11);
            if (SEL_OUT = '0') then
                R <= TrRotL_CARRYX;
            elsif (SEL_OUT = '1') then
                R <= TrRotL_CARRYY;
                Ry <= TrRotL_CARRYY;
            end if;
        end if;

        elsif (OP = "1100") then
--Logic Shift Right
        if (Tren_C(12) = '1') then
            Cout <= TrCout(12);
        end if;
        if (Tren_V(12) = '1') then
            V <= TrV(12);
        end if;
        if (Tren_N(12) = '1') then
            N <= TrN(12);
        end if;
        if (RUN = '1') then
            Z <= TrZ(12);
            X_bin_pal <= TrBinPal(12);
            X_prime <= TrPrime(12);
            F_active <= TrActive(12);
            if (SEL_OUT = '0') then
                R <= TrLSRX;
            elsif (SEL_OUT = '1') then
                R <= TrLSRY;
            end if;
        end if;
    end if;

```

```

                    Ry <= TrLSRY;
                end if;
            end if;

            elsif (OP = "1101") then
--Arithmatic Shift Right
                if (Tren_C(13) = '1') then
                    Cout <= TrCout(13);
                end if;
                if (Tren_V(13) = '1') then
                    V <= TrV(13);
                end if;
                if (Tren_N(13) = '1') then
                    N <= TrN(13);
                end if;
                if (RUN = '1') then
                    Z <= TrZ(13);
                    X_bin_pal <= TrBinPal(13);
                    X_prime <= TrPrime(13);
                    F_active <= TrActive(13);
                    if (SEL_OUT = '0') then
                        R <= TrASRX;
                    elsif (SEL_OUT = '1') then
                        R <= TrASRY;
                        Ry <= TrASRY;
                    end if;
                end if;

            elsif (OP = "1110") then
--Logic Shift Left
                if (Tren_C(14) = '1') then
                    Cout <= TrCout(14);
                end if;
                if (Tren_V(14) = '1') then
                    V <= TrV(14);
                end if;
                if (Tren_N(14) = '1') then
                    N <= TrN(14);
                end if;
                if (RUN = '1') then
                    Z <= TrZ(14);
                    X_bin_pal <= TrBinPal(14);
                    X_prime <= TrPrime(14);
                    F_active <= TrActive(14);
                    if (SEL_OUT = '0') then
                        R <= TrLSLX;
                    elsif (SEL_OUT = '1') then
                        R <= TrLSLY;
                        Ry <= TrLSLY;
                    end if;
                end if;

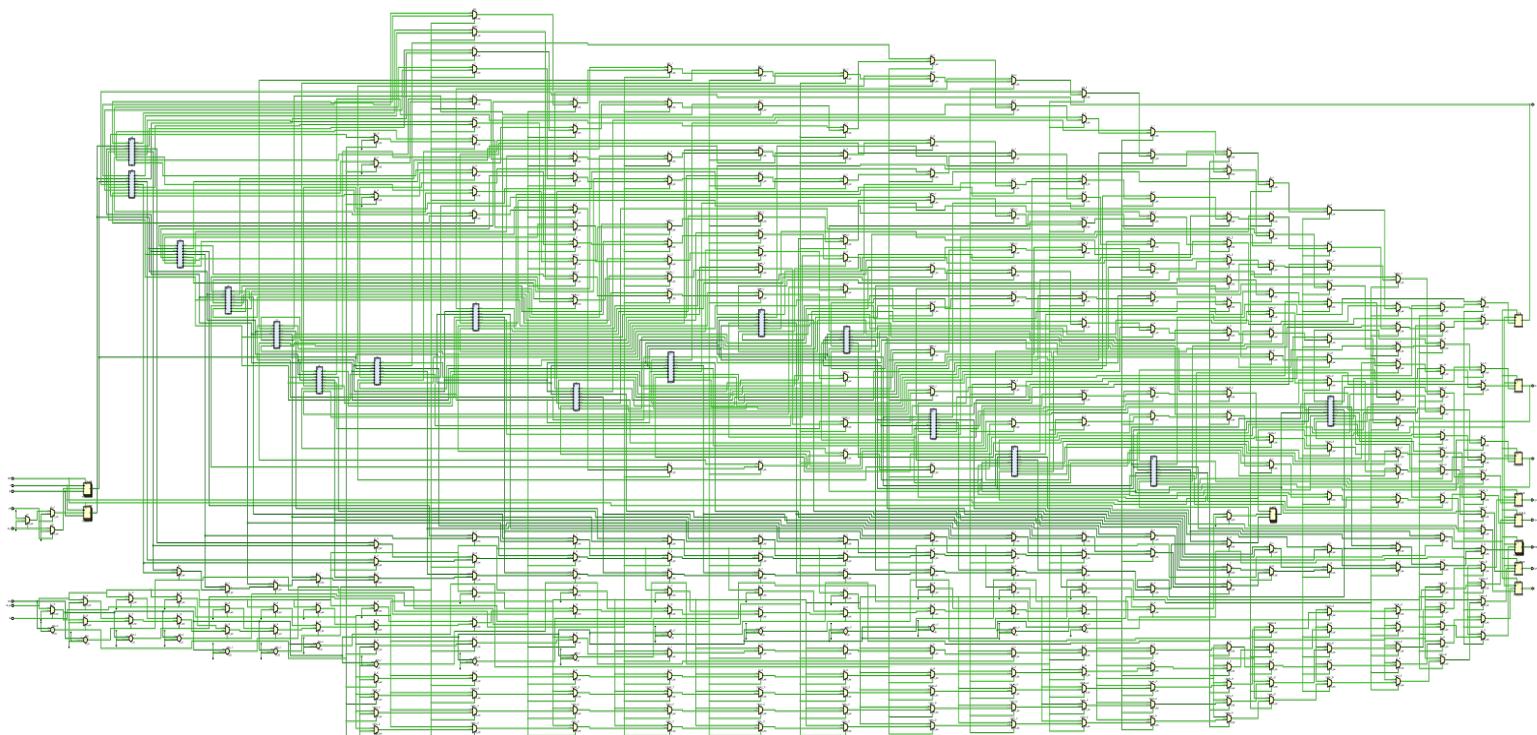
```

```

        elsif (OP = "1111") then
--BCD to Binary Conversion
            if (Tren_C(15) = '1') then
                Cout <= TrCout(15);
            end if;
            if (Tren_V(15) = '1') then
                V <= TrV(15);
            end if;
            if (Tren_N(15) = '1') then
                N <= TrN(15);
            end if;
            if (RUN = '1') then
                Z <= TrZ(15);
                X_bin_pal <= TrBinPal(15);
                X_prime <= TrPrime(15);
                F_active <= TrActive(15);
                if (SEL_OUT = '0') then
                    R <= TrBCD(7 downto 0);
                elsif (SEL_OUT = '1') then
                    R <= TrBCD(15 downto 8);
                    Ry <= TrBCD(15 downto 8);
                end if;
            end if;
        end if;
    end Process;

```

کد شماره‌ی ۱۷ ALU_SEQ Top Module



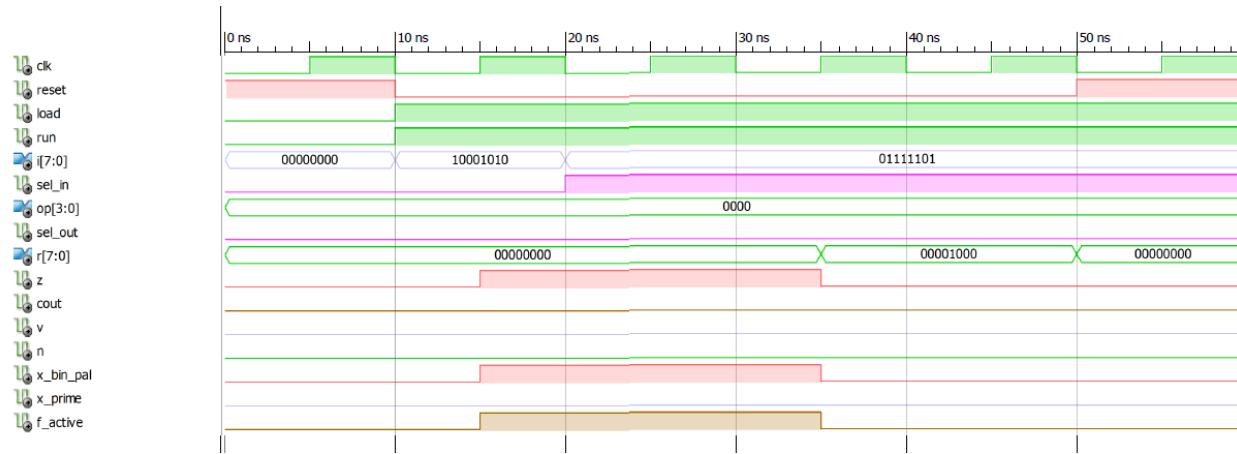
شکل ۵۳: بلوک دیاگرام Task2

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

حال به سراغ test bench می‌رویم.

کد آن در فایل‌های پروژه موجود می‌باشد و برای کاهش حجم گزارش کار از آوردن آن خودداری می‌کنیم.

AND:



مقدار OPCODE برابر است با ۰۰۰۰ که زیرماژول AND می‌باشد. در ۰ تا ۱۰ نانو ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند. در ۱۰ تا ۲۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی اول (10001010) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z, X_bin_pal, X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

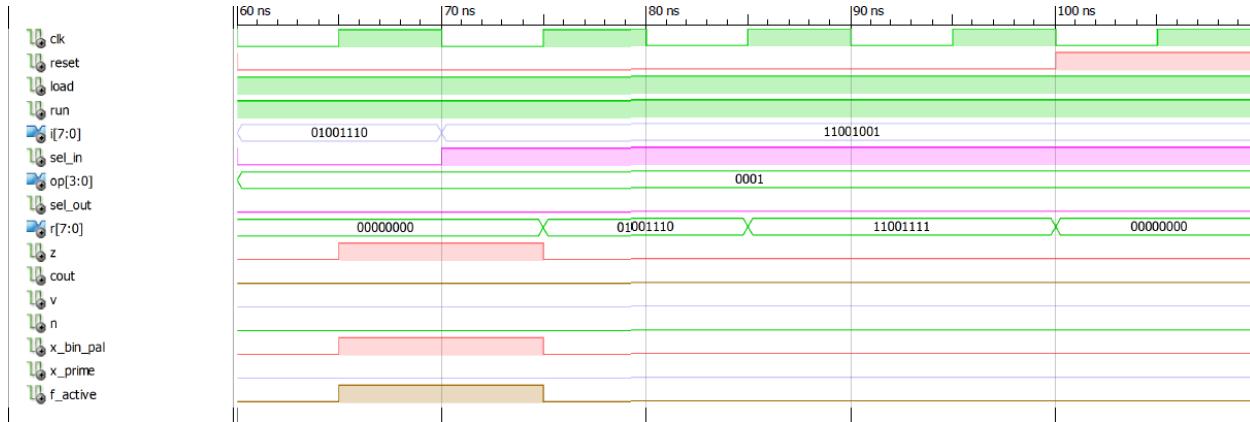
در این بازه خروجی صفر است زیرا صفر با صفر AND می‌شود پس Z مقدار ۱ می‌گیرد و در نتیجه F_active ۱ می‌شود. خروجی (00000000) را برعکس کنیم با خودش برابر می‌شود پس X_bin_pal ۱ برابر ۱ می‌شود. صفر اول نیست پس X_prime صفر می‌شود.

در ۲۰ تا ۳۰ نانو ثانیه پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN یک است پس ورودی دوم (01111101) در IB قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_prime, Z, X_bin_pal و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. در این بازه چون هنوز ورودی دوم وارد نشده ورودی اول را با صفر AND می‌کند و خروجی صفر می‌دهد پس Z مقدار ۱ می‌گیرد و در نتیجه F_active ۱ نیز ۱ می‌شود. خروجی (00000000) را برعکس کنیم با خودش برابر می‌شود پس X_bin_pal ۱ برابر ۱ می‌شود. صفر اول نیست پس X_prime صفر می‌شود.

در ۳۰ تا ۵۰ نانو ثانیه پایه RUN یک است پس فلیپ فلاپ‌های Z, X_bin_pal, X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند و خروجی اصلی (00001000) به ما داده می‌شود. پایه SEL_OUT صفر است پس خروجی اول (X) در R قرار خواهد گرفت. خروجی در مبنای ۱۰ برابر ۸ است که اول نیست. اگر خروجی را برعکس کنیم

با خودش برابر نمی‌شود پس X_{bin_pal} برابر \cdot می‌شود. مقادیر V , Z و $Cout$ همگی صفر است F_{active} صفر است. در ۵۰ تا ۶۰ نانو ثانیه ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

OR:



مقدار OPCODE برابر است با ۱۰۰۰ که زیرمازوی OR می‌باشد. در ۶۰ تا ۷۰ نانو ثانیه ریست \cdot است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی اول (01001110) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_{bin_pal} , X_{prime} , Z و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

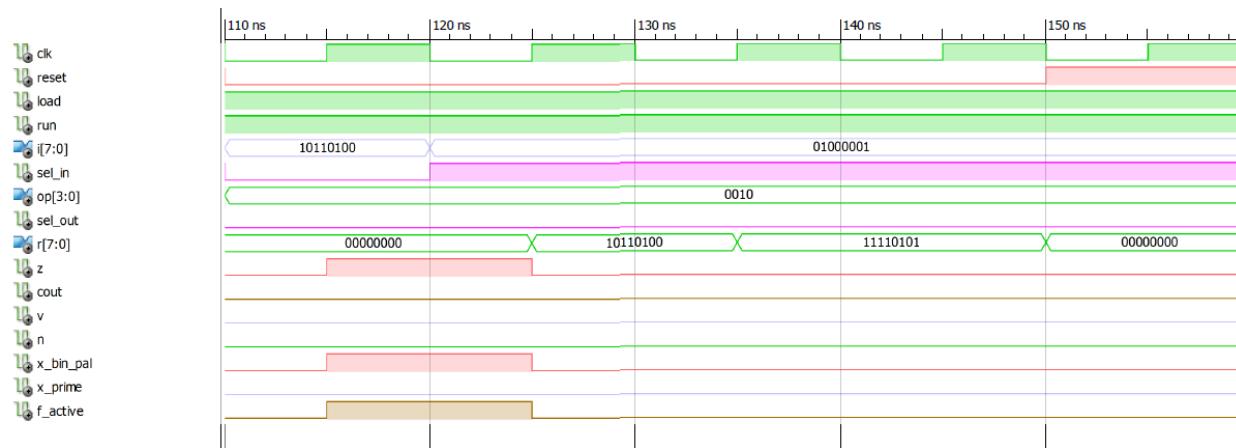
در این بازه خروجی صفر است زیرا صفر با صفر OR می‌شود پس Z مقدار ۱ می‌گیرد و در نتیجه F_{active} نیز ۱ می‌شود. خروجی (00000000) را بر عکس کنیم با خودش برابر می‌شود پس X_{bin_pal} برابر ۱ می‌شود. صفر اول نیست پس X_{prime} صفر می‌شود.

در ۷۰ تا ۸۰ نانو ثانیه پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN یک است پس ورودی دوم (11001001) در IB قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_{prime} , Z , X_{bin_pal} و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. در این بازه چون هنوز ورودی دوم وارد نشده ورودی اول را با صفر OR می‌کند و خروجی همان ورودی اول را می‌دهد پس Z مقدار \cdot می‌گیرد و در نتیجه F_{active} نیز \cdot می‌شود. خروجی (01001110) را بر عکس کنیم با خودش برابر نمی‌شود پس X_{bin_pal} برابر \cdot می‌شود. خروجی در مبنای ۱۰ برابر ۷۸ است که اول نیست پس X_{prime} صفر می‌شود.

در ۸۰ تا ۹۰ نانو ثانیه هنگامی که کلاک ۱ شود خروجی اصلی (11001111) به ما داده می‌شود. پایه SEL_OUT صفر است پس خروجی اول (X) در R قرار خواهد گرفت. خروجی در مبنای ۱۰ برابر ۲۰۷ است که اول نیست. اگر خروجی را بر عکس کنیم با خودش برابر نمی‌شود پس X_{bin_pal} برابر \cdot می‌شود. مقادیر V ,

Z و $Cout$ همگی صفر است پس F_{active} صفر است. در ۱۰۰ تا ۱۱۰ نانو ثانیه SEL_IN ریست ۱ است در نتیجه تمامی متغیرهای خروجی ۰ اند.

XOR:



مقدار OPCODE برابر است با ۱۰۰ که زیرماژول XOR می‌باشد. در ۱۰۰ تا ۱۱۰ نانو ثانیه SEL_IN ریست ۰ است. پایه $load$ یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی اول (10110100) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z , X_bin_pal , X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

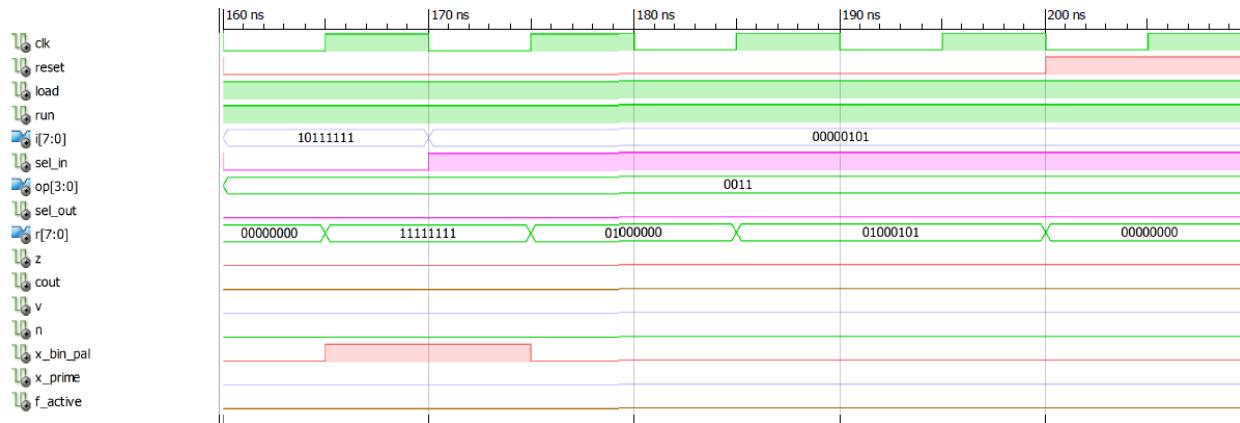
در این بازه خروجی صفر است زیرا صفر با صفر XOR می‌شود پس Z مقدار ۱ می‌گیرد و در نتیجه F_{active} ۱ می‌شود. خروجی (00000000) را برعکس کنیم با خودش برابر می‌شود پس X_bin_pal برابر ۱ می‌شود. صفر اول نیست پس X_prime صفر می‌شود.

در ۱۲۰ تا ۱۳۰ نانو ثانیه ریست ۰ است. پایه $load$ یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN یک است پس ورودی دوم (01000001) در IB قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z , X_bin_pal , X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. در اینجا چون هنوز ورودی دوم وارد نشده، ورودی اول را با صفر XOR می‌کند و در خروجی نشان می‌دهد که همان ورودی اول است. ورودی اول در مبنای ۱۰ برابر ۱۸۰ است که عددی اول نیست ($X_prime=0$) و اگر آن را برعکس کنیم با خودش برابر نمی‌شود($X_bin_pal=0$). مقدار Z صفر است پس F_{active} نیز صفر است.

در ۱۳۰ تا ۱۴۰ نانو ثانیه هنگامی که کلاک ۱ شود خروجی اصلی (11110101) به ما داده می‌شود. پایه SEL_OUT صفر است پس خروجی اول (X) در R قرار خواهد گرفت. خروجی در مبنای ۱۰ برابر ۲۴۵ است که اول نیست. اگر خروجی را برعکس کنیم با خودش برابر نمی‌شود پس X_bin_pal برابر ۰ می‌شود. مقادیر \forall

Z و $Cout$ همگی صفر است پس F_{active} صفر است. در ۱۵۰ تا ۱۶۰ نانو ثانیه Th ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

XNOR:



مقدار OPCODE برابر است با ۱۱ که زیرمازوی XNOR می‌باشد. در ۱۶۰ تا ۱۷۰ نانو ثانیه ریست ۰ است. پایه $load$ یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی اول (10111111) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_bin_pal , X_prime , Z , X_bin_pal , X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

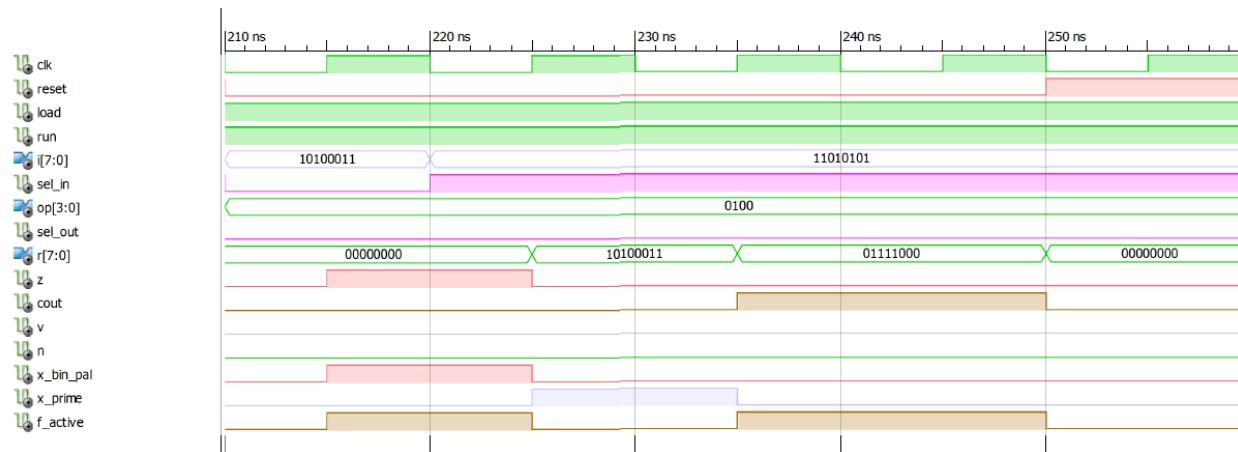
در این بازه خروج در نیمه دوم حاصل XNOR صفر ورودی اول و صفر ورودی دوم است که برابر ۱ می‌شود پس Z مقدار ۰ می‌گیرد و در نتیجه F_{active} نیز ۰ می‌شود. خروجی (11111111) را برعکس کنیم با خودش برابر می‌شود پس X_bin_pal برابر ۱ می‌شود. ۲۵۵ اول نیست پس X_prime صفر می‌شود.

در ۱۷۰ تا ۱۸۰ نانو ثانیه ریست ۰ است. پایه $load$ یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN یک است پس ورودی دوم (00000101) در IB قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z , X_bin_pal , X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. در اینجا چون هنوز ورودی دوم وارد نشده، ورودی اول را با صفر XNOR می‌کند و در خروجی نشان می‌دهد که برابر نات ورودی اول (01000000) است. نات ورودی اول در مبنای ۱۰ برابر ۶۴ است که عددی اول نیست ($X_{prime}=0$) و اگر آن را برعکس کنیم با خودش برابر نمی‌شود($X_{bin_pal}=0$). مقدار Z صفر است پس F_{active} نیز صفر است.

در ۱۸۰ تا ۱۹۰ نانو ثانیه هنگامی که کلک ۱ شود خروجی اصلی (01000101) به ما داده می‌شود. پایه SEL_OUT صفر است پس خروجی اول (X) در R قرار خواهد گرفت. خروجی در مبنای ۱۰ برابر ۶۹ است که اول نیست. اگر خروجی را برعکس کنیم با خودش برابر نمی‌شود پس X_bin_pal برابر ۰ می‌شود. مقادیر \forall

Z و $Cout$ همگی صفر است پس F_{active} صفر است. در ۲۰۰ تا ۲۱۰ نانو ثانیه run ریست ۱ است در نتیجه تمامی متغیرهای خروجی ۰ اند.

Unsigned Addition:



مقدار OPCODE برابر است با ۱۰۰ که زیرماژول UADD می‌باشد. در ۲۰ تا ۲۲۰ نانو ثانیه run ریست ۰ است. پایه $load$ یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی اول (10100011) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_bin_pal , X_prime , Z , X_bin_pal و X_prime رجیسترها را خروجی فعال هستند. در نتیجه هنگامی که کلک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه خروجی صفر است زیرا صفر را با صفر جمع می‌کند (مقادیر رجیسترها بعد از ریست) پس Z مقدار ۱ می‌گیرد و در نتیجه F_{active} نیز ۱ می‌شود. خروجی (00000000) را بر عکس کنیم با خودش برابر می‌شود پس X_bin_pal برابر ۱ می‌شود. صفر اول نیست پس X_prime صفر می‌شود.

در ۲۲۰ تا ۲۳۰ نانو ثانیه ریست ۰ است. پایه $load$ یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN یک است پس ورودی دوم (11010101) در IB قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z , X_bin_pal , X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. در اینجا چون هنوز ورودی دوم وارد نشده، ورودی اول را با صفر جمع می‌کند و در خروجی نشان می‌دهد که همان ورودی اول است. ورودی اول در مبنای ۱۰ برابر ۱۶۳ است که عددی اول است ($X_prime=1$) و اگر آن را بر عکس کنیم با خودش برابر نمی‌شود ($X_bin_pal=0$). مقدار Z صفر است پس F_{active} نیز صفر است.

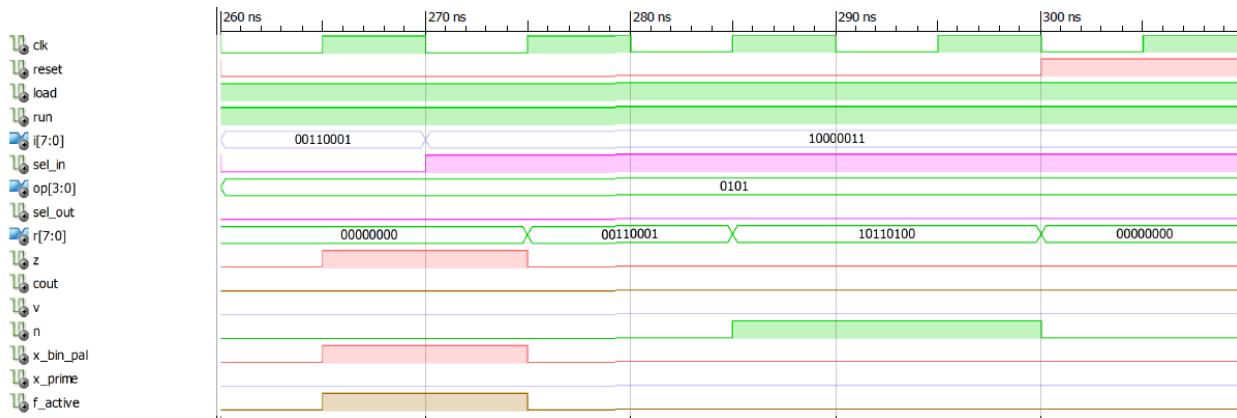
در ۲۳۰ تا ۲۵۰ نانو ثانیه هنگامی که کلک ۱ شود خروجی اصلی (01111000) به ما داده می‌شود. پایه SEL_OUT صفر است پس خروجی اول (X) در R قرار خواهد گرفت. در مبنای ۱۰ داریم:

$$a + b = 163 + 213 = 376 = 256 + 120$$

خروجی در مبنای ۱۰ برابر ۱۲۰ است که اول نیست. اگر خروجی را برعکس کنیم با خودش برابر نمی‌شود پس X_{bin_pal} برابر ۰ می‌شود.

مقادیر Z و V صفر هستند ولی Cout یک می‌شود (چون خروجی بزرگ‌تر از ۲۵۵ است) پس F_active یک است. در ۲۵۰ تا ۲۶۰ نانو ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

Signed Addition:



مقدار OPCODE برابر است با ۱۱۰ که زیرماژول SADD می‌باشد. در ۲۶۰ تا ۲۷۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی اول (00110001) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_prime, Z, X_bin_pal و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه خروجی صفر است زیرا صفر را با صفر جمع می‌کند (مقادیر رجیسترها بعد از ریست) پس Z مقدار ۱ می‌گیرد و در نتیجه F_active نیز ۱ می‌شود. خروجی (00000000) را برعکس کنیم با خودش برابر می‌شود پس X_{bin_pal} برابر ۱ می‌شود. صفر اول نیست پس X_{prime} صفر می‌شود.

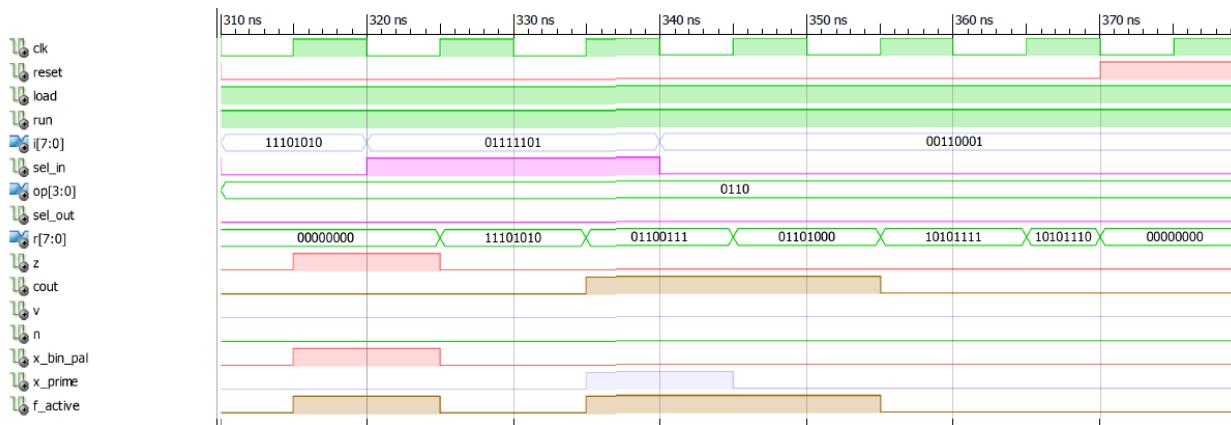
در ۲۷۰ تا ۲۸۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN یک است پس ورودی دوم (10000011) در IB قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z, X_bin_pal, X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. در اینجا چون هنوز ورودی دوم وارد نشده، ورودی اول را با صفر جمع می‌کند و در خروجی نشان می‌دهد که همان ورودی اول است. ورودی اول در مبنای ۱۰ برابر ۴۹ است که عددی اول نیست ($X_{prime}=0$) و اگر آن را برعکس کنیم با خودش برابر نمی‌شود (X_{bin_pal}=0). مقدار Z صفر است پس F_active نیز صفر است.

در ۲۸۰ تا ۳۰۰ نانو ثانیه هنگامی که کلاک ۱ شود خروجی اصلی (10110100) به ما داده می‌شود. پایه SEL_OUT صفر است پس خروجی اول (X) در R قرار خواهد گرفت. در مبنای ۱۰ داریم:

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

خروجی X در حالت بدون علامت در مبنای ۱۰ برابر ۱۸۰ است که اول نیست. اگر خروجی را برعکس کنیم با خودش برابر نمی‌شود پس X_{bin_pal} برابر ۰ می‌شود. مقادیر Z، Cout و V صفر هستند پس F_{active} یک است. مقدار N برابر ۱ می‌باشد (چون حاصل منفی است). در ۳۰۰ تا ۳۱۰ نانو ثانیه ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

Unsigned Addition with Carry:



مقدار OPCODE برابر است با ۱۱۰ که زیرماژول UADD_CARRY می‌باشد. در ۳۱۰ تا ۳۲۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی اول X_{bin_pal} (۱۱۱۰۱۰۱۰) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_{prime} و X_{bin_pal} رجیسترهای خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه خروجی صفر است زیرا صفر را با صفر جمع می‌کند (مقادیر رجیسترها بعد از ریست) پس Z مقدار ۱ می‌گیرد و در نتیجه F_{active} نیز ۱ می‌شود. خروجی (۰۰۰۰۰۰۰۰) را برعکس کنیم با خودش برابر می‌شود پس X_{bin_pal} برابر ۱ می‌شود. صفر اول نیست پس X_{prime} صفر می‌شود.

در ۳۲۰ تا ۳۳۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN یک است پس ورودی دوم (۰۱۱۱۱۱۰۱) در IB قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z ، X_{bin_pal} ، X_{prime} و رجیسترهای خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. در اینجا چون هنوز ورودی دوم وارد نشده، ورودی اول را با صفر جمع می‌کند و در خروجی نشان می‌دهد که همان ورودی اول است. ورودی اول در مبنای ۱۰ برابر ۲۳۴ است که عددی اول نیست ($X_{prime}=0$) و اگر آن را برعکس کنیم با خودش برابر نمی‌شود. مقدار Z صفر است پس F_{active} نیز صفر است.

در ۳۴۰ تا ۳۴۰ نانو ثانیه هنگامی که کلاک ۱ شود خروجی اصلی (01111000) به ما داده می‌شود. پایه SEL_OUT صفر است پس خروجی اول (X) در R قرار خواهد گرفت. در مبنای ۱۰ داریم:

$$a + b + Cin = 234 + 125 + 0 = 359 = 256 + 103$$

خروجی X در مبنای ۱۰ برابر ۱۰۳ است که اول است. اگر خروجی را برعکس کنیم با خودش برابر نمی‌شود پس X_bin_pal برابر ۰ می‌شود. مقادیر Z و V صفر هستند ولی Cout یک می‌شود (چون خروجی بزرگتر از ۲۵۵ است) پس F_active یک است.

در ۳۴۰ تا ۳۵۰ نانو ثانیه پایه load یک است پس ورودی دریافت می‌کنیم. پایه IN_SEL صفر است پس ورودی اول (00110001) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلابپهای X_prime، Z، X_bin_pal و رجیسترهاي خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلابپهای و رجیسترها داده ها را منتقل می‌کنند.

در این بازه خروجی همان خروجی قبلی به اضافه Cin می‌باشد (Cin این مرحله همان Cout مرحله قبل است)، زیرا با این که ورودی وارد ALU شده ولی برای دریافت خروجی مطلوب باید یک کلاک دیگر منتظر بمانیم در مبنای ۱۰:

$$X(Previous\ State) + Cin = 359 + 1 = 360 = 256 + 104$$

خروجی X در مبنای ۱۰ برابر ۱۰۴ است که اول نیست. اگر خروجی را برعکس کنیم با خودش برابر نمی‌شود پس X_bin_pal برابر ۰ می‌شود. مقادیر Z و V صفر هستند ولی Cout یک می‌شود (چون خروجی بزرگتر از ۲۵۵ است) پس F_active یک است.

در ۳۵۰ تا ۳۶۰ نانو ثانیه با رسیدن کلاک خروجی ورودی IA قدیم و IB قدیم و همچنین Cin برابر یک (کری خروجی مرحله قبل) محاسبه می‌شود. در مبنای ۱۰:

$$a + b + Cin = 49 + 125 + 1 = 175$$

خروجی X در مبنای ۱۰ برابر ۱۷۵ است که اول نیست. اگر خروجی را برعکس کنیم با خودش برابر نمی‌شود پس X_bin_pal برابر ۰ می‌شود. مقادیر Z و V صفر هستند و Cout نیز صفر می‌شود (چون خروجی کوچکتر از ۲۵۵ است) پس F_active صفر است.

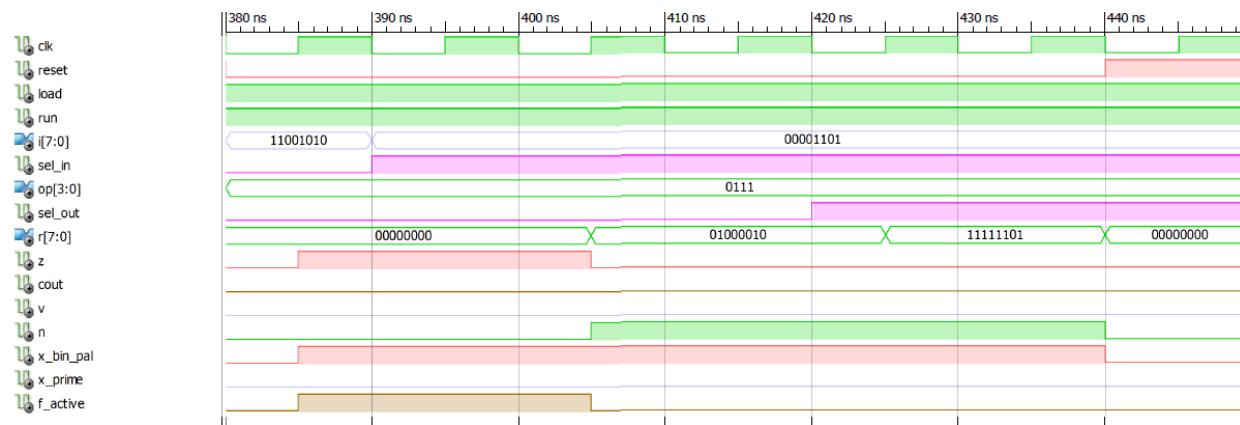
در ۳۶۰ تا ۳۷۰ نانو ثانیه با رسیدن کلاک خروجی ورودی IA قدیم و IB قدیم و همچنین Cin برابر صفر (کری خروجی مرحله قبل) محاسبه می‌شود. در مبنای ۱۰:

$$a + b + Cin = 49 + 125 + 0 = 174$$

خروجی X در مبنای ۱۰ برابر ۱۷۴ است که اول نیست. اگر خروجی را برعکس کنیم با خودش برابر نمی‌شود پس X_bin_pal برابر ۰ می‌شود. مقادیر Z و V صفر هستند و Cout نیز صفر می‌شود (چون خروجی

کوچکتر از ۲۵۵ است) پس F_{active} صفر است. در ۳۷۰ تا ۳۸۰ نانو ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

Signed Multiplication:



مقدار OPCODE برابر است با ۱۱۱ که زیرماژول SMUL می‌باشد. در ۳۸۰ تا ۳۹۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی اول (11001010) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_bin_pal، X_prime، Z، و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه خروجی صفر است زیرا صفر را در صفر (خروجی رجیسترها بعد از ریست) ضرب می‌کند پس Z مقدار ۱ می‌گیرد و در نتیجه F_{active} نیز ۱ می‌شود. خروجی (00000000) را برعکس کنیم با خودش برابر می‌شود پس X_{bin_pal} برابر ۱ می‌شود. صفر اول نیست پس X_{prime} صفر می‌شود.

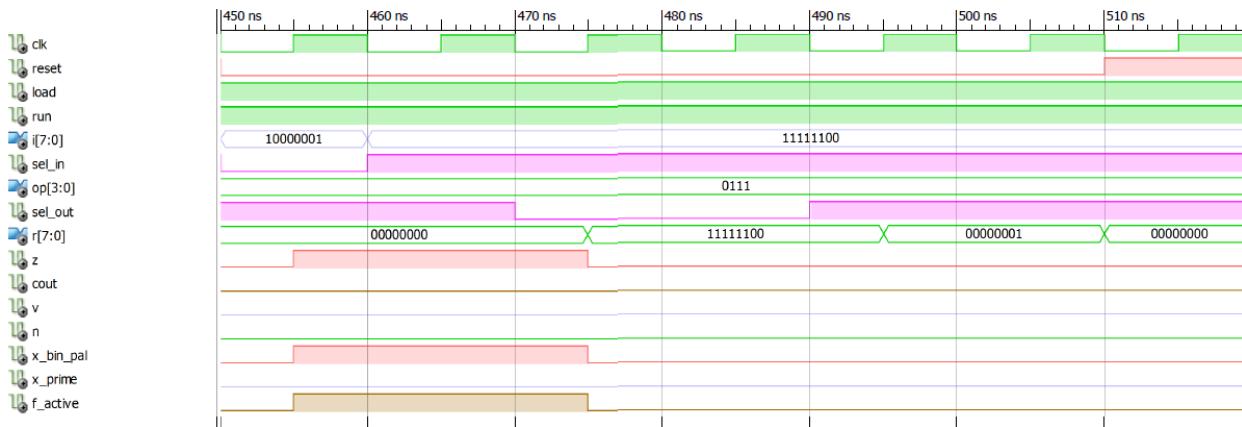
در ۳۹۰ تا ۴۰۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN یک است پس ورودی دوم (۱۱۰۱۰۰۰۰) در IB قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z، X_bin_pal، X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. در اینجا چون هنوز ورودی دوم وارد نشده، ورودی اول را در صفر ضرب می‌کند و در خروجی نشان می‌دهد که صفر است. عددی اول نیست (X_prime=0) و اگر آن را برعکس کنیم با خودش برابر می‌شود ($X_{bin_pal}=1$). مقدار Z یک است پس F_{active} نیز یک است.

در ۴۰۰ تا ۴۲۰ نانو ثانیه هنگامی که کلاک ۱ شود خروجی اصلی به ما داده می‌شود که باید از هر دو خروجی استفاده کرد. پایه SEL_OUT صفر است پس خروجی اول (X) که شامل ۸ بیت اول از سمت راست حاصل ضرب می‌باشد در R قرار خواهد گرفت. در ۴۲۰ تا ۴۳۰ نانو ثانیه هنگامی که کلاک ۱ شود پایه SEL_OUT یک است پس خروجی دوم (Y) که شامل ۸ بیت دوم از سمت راست حاصل ضرب می‌باشد در R قرار خواهد گرفت. در مبنای ۱۰ داریم:

$$a \times b = -54 \times 13 = -702$$

$$= -(001010111110)_2 \xrightarrow{2s Complement (to 16Bits)} 1111110101000010$$

خروجی X در مبنای ۱۰ برابر ۶۶ است که اول نیست ($X_{prime}=0$) و اگر آن را برعکس کنیم با خودش برابر می‌شود پس X_{bin_pal} برابر ۱ می‌شود. برای بازه ۴۳۰ تا ۴۰۰ مقادیر V، Cout و Z همگی صفر است پس F_{active} نیز صفر می‌شود. خروجی مقداری منفی است به همین دلیل N ۱ می‌شود. در ۴۴۰ تا ۴۵۰ نانو ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.



مقدار OPCODE برابر است با ۱۱۱ که زیرماژول SMUL می‌باشد. در ۴۵۰ تا ۴۶۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی اول (10000001) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_{bin_pal} , X_{prime} , Z, X، Cout و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه هر دو خروجی صفر است زیرا صفر را در صفر (خروجی رجیسترها بعد از ریست) ضرب می‌کند پس Z مقدار ۱ می‌گیرد و در نتیجه F_{active} نیز ۱ می‌شود. خروجی (00000000) را برعکس کنیم با خودش برابر می‌شود پس X_{bin_pal} برابر ۱ می‌شود. صفر اول نیست پس X_{prime} صفر می‌شود.

در ۴۶۰ تا ۴۷۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN یک است پس ورودی دوم (11111100) در IB قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z, X_{bin_pal} , X_{prime} و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. در اینجا چون هنوز ورودی دوم وارد نشده، ورودی اول را در صفر ضرب می‌کند و در خروجی نشان می‌دهد که صفر است. عددی اول نیست ($X_{prime}=0$) و اگر آن را برعکس کنیم با خودش برابر می‌شود ($X_{bin_pal}=1$). مقدار Z یک است پس F_{active} نیز یک است.

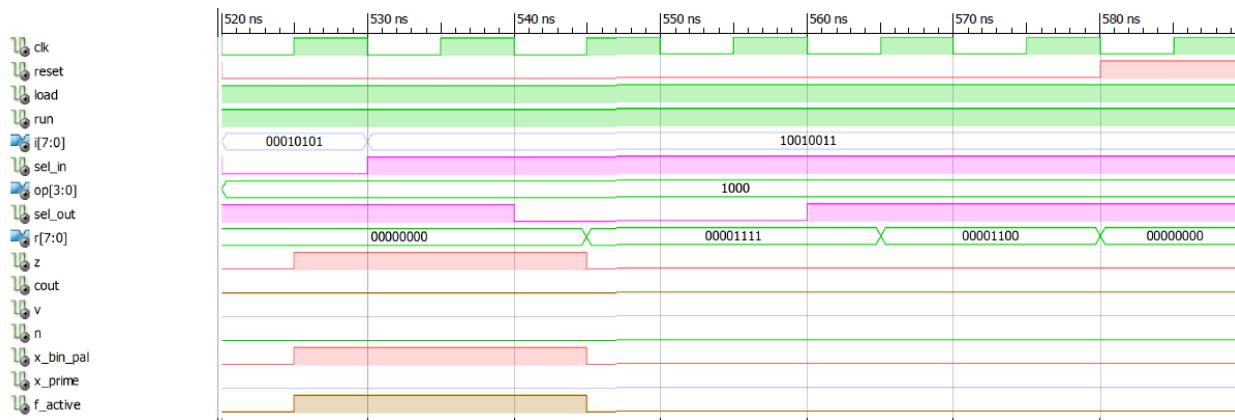
در ۴۷۰ تا ۴۹۰ نانو ثانیه هنگامی که کلک ۱ شود خروجی اصلی به ما داده می‌شود که باید از هر دو خروجی استفاده کرد. پایه SEL_OUT صفر است پس خروجی اول (X) که شامل ۸ بیت اول از سمت راست حاصل ضرب می‌باشد در R قرار خواهد گرفت. در ۴۹۰ تا ۵۱۰ نانو ثانیه هنگامی که کلک ۱ شود پایه SEL_OUT یک است پس خروجی دوم (Y) که شامل ۸ بیت دوم از سمت راست حاصل ضرب می‌باشد در R قرار خواهد گرفت. در مبنای ۱۰ داریم:

$$a \times b = -127 \times -4 = 508$$

$$= (00011111100)_2 \xrightarrow{2s Complement (to 16Bits)} 000000011111100$$

خروجی X در مبنای ۱۰ برابر ۲۵۲ است که اول نیست (X_prime=0) و اگر آن را برعکس کنیم با خودش برابر نمی‌شود پس X_bin_pal برابر ۰ می‌شود. برای بازه ۴۷۰ تا ۵۱۰ مقادیر V, Cout و Z همگی صفر است پس F_active نیز صفر می‌شود. خروجی مقداری مثبت است به همین دلیل N صفر می‌شود. در ۵۲۰ تا ۵۴۰ نانو ثانیه ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

Unsigned Multiplication:



مقدار OPCODE برابر است با ۱۰۰۰ که زیرماژول UMUL می‌باشد. در ۵۲۰ تا ۵۳۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی اول (00010101) در ۴۹۰ قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_bin_pal, X_prime, X, cout و Z رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه خروجی صفر است زیرا صفر را در صفر ضرب می‌کند (مقادیر رجیسترها بعد از ریست) پس Z مقدار ۱ می‌گیرد و در نتیجه F_active نیز ۱ می‌شود. خروجی (00000000) را برعکس کنیم با خودش برابر می‌شود پس X_bin_pal برابر ۱ می‌شود. صفر اول نیست پس X_prime صفر می‌شود.

در ۵۳۰ تا ۵۴۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN یک است پس ورودی دوم (10010011) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های

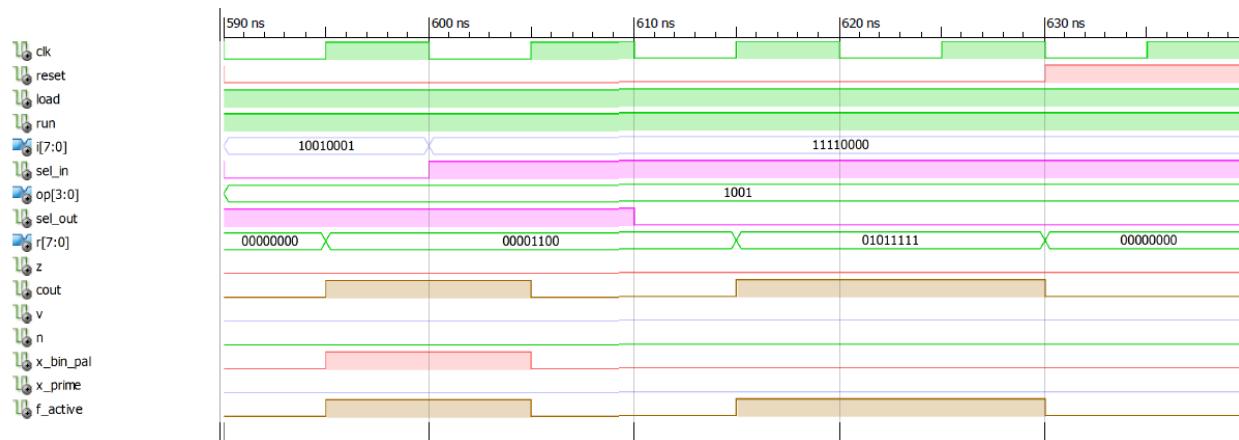
$X_{\text{bin_pal}}$, X_{prime} , Z و رجیسترهای خروجی فعال هستند. در نتیجه هنگامی که کلک ۱ شود (لبه بالارونده) فلیپ فلابپها و رجیسترها داده ها را منتقل می کنند. در اینجا چون هنوز ورودی دوم وارد نشده، ورودی اول را در صفر ضرب می کند و در خروجی نشان می دهد که صفر است. عددی اول نیست ($X_{\text{prime}}=0$) و اگر آن را برعکس کنیم با خودش برابر می شود ($X_{\text{bin_pal}}=1$). مقدار Z یک است پس F_{active} نیز یک است.

در ۵۶۰ تا ۵۴۰ نانو ثانیه هنگامی که کلک ۱ شود خروجی اصلی به ما داده می شود که باید از هر دو خروجی استفاده کرد. پایه SEL_OUT صفر است پس خروجی اول (X) که شامل ۸ بیت اول از سمت راست حاصل ضرب می باشد در R قرار خواهد گرفت. در ۵۶۰ تا ۵۷۰ نانو ثانیه هنگامی که کلک ۱ شود پایه SEL_OUT یک است پس خروجی دوم (Y) که شامل ۸ بیت دوم از سمت راست حاصل ضرب می باشد در R قرار خواهد گرفت. در مبنای ۱۰ داریم:

$$a * b = 21 * 147 = 3087 = (0000110000001111)_2$$

خروجی X در مبنای ۱۰ برابر ۱۵ است که اول نیست ($X_{\text{prime}}=0$) و اگر آن را برعکس کنیم با خودش برابر نمی شود پس $X_{\text{bin_pal}}$ برابر ۰ می شود. برای بازه ۵۷۰ تا ۵۹۰ مقدار V , $Cout$ و Z همگی صفر است پس F_{active} نیز صفر می شود. در ۵۸۰ تا ۵۹۰ نانو ثانیه ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

Unsigned Subtraction:



مقدار OPCODE برابر است با ۱۰۰۱ که زیرمازوی USUB می باشد. در ۵۰۰ تا ۶۰۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می کنیم. پایه SEL_IN صفر است پس ورودی اول (10010001) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلابپهای $X_{\text{bin_pal}}$, X_{prime} , Z و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلک ۱ شود (لبه بالارونده) فلیپ فلابپها و رجیسترها داده ها را منتقل می کنند.

در ۶۰۰ تا ۶۱۰ نانو ثانیه ریست ۰ است. پایه **load** یک است پس ورودی دریافت می‌کنیم. پایه **SEL_IN** یک است پس ورودی دوم (۱۱۱۱۰۰۰۰) در **IA** قرار خواهد گرفت. پایه **RUN** یک است پس فلیپ فلاپ‌های **Z**, **X_bin_pal**, **X_prime** و رجیسترهای خروجی فعال هستند.

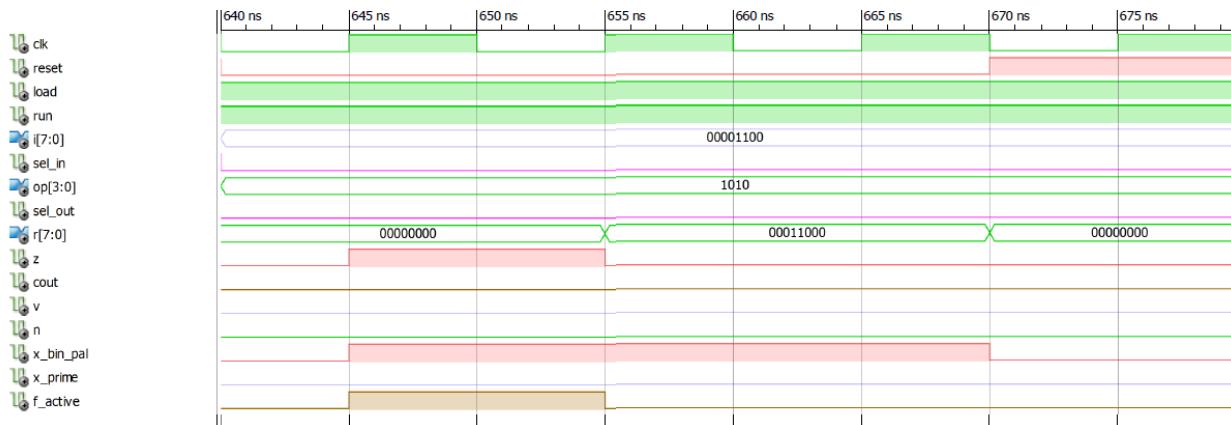
در ۶۱۰ تا ۶۳۰ نانو ثانیه هنگامی که کلاک ۱ شود خروجی اصلی (۰۱۰۱۱۱۱۱) به ما داده می‌شود. پایه **SEL_OUT** صفر است پس خروجی اول (**X**) در **R** قرار خواهد گرفت. در مبنای ۱۰ داریم:

$$a + b = 145 - 240 = -95 = -(01011111)_2$$

چون مقدار بدست آمده منفی است، صفر می‌شود. خروجی در مبنای ۱۰ برابر ۹۵ است که اول نیست. اگر خروجی را برعکس کنیم با خودش برابر نمی‌شود پس **X_bin_pal** برابر ۰ می‌شود.

مقادیر **Z** و **V** صفر هستند ولی **Cout** یک می‌شود (چون خروجی کوچکتر از ۰ است) پس **F_active** یک است. در ۶۴۰ تا ۶۴۰ نانو ثانیه ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

Rotation Left:



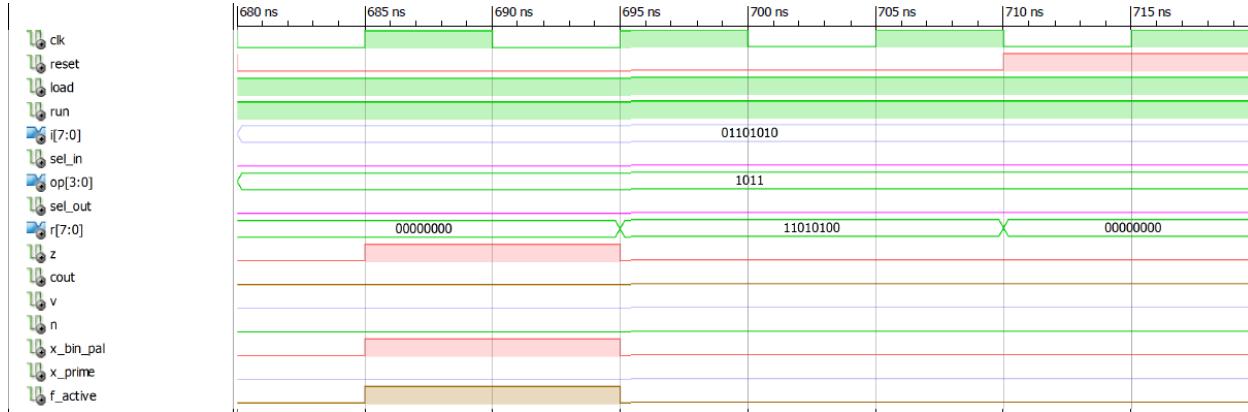
مقدار **OPCODE** برابر است با ۱۰۱ که زیرماژول **RotL** می‌باشد. در ۶۴۰ تا ۶۴۰ نانو ثانیه ریست ۰ است. پایه **load** یک است پس ورودی دریافت می‌کنیم. پایه **SEL_IN** صفر است پس ورودی (۰۰۰۰۱۱۰۰) در **IA** قرار خواهد گرفت. پایه **RUN** یک است پس فلیپ فلاپ‌های **Z**, **X_bin_pal**, **X_prime** و رجیسترهای خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه خروجی صفر است زیرا صفر (۰۰۰۰۰۰۰۰) **rotate** می‌شود، پس **Z** مقدار ۱ می‌گیرد و در نتیجه **X_bin_pal** نیز ۱ می‌شود. خروجی (۰۰۰۰۰۰۰۰) را برعکس کنیم با خودش برابر می‌شود پس **f_active** ۱ می‌شود. صفر اول نیست پس **X_prime** صفر می‌شود.

در ۶۵۰ تا ۶۷۰ نانو ثانیه ریست ۰ است پایه **RUN** یک است پس فلیپ فلاپ‌های **Z**, **X_prime** و رجیسترهای خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. خروجی اصلی (۰۰۰۱۱۰۰) به ما داده می‌شود. پایه **SEL_OUT** صفر

است پس خروجی اول (X) در R قرار خواهد گرفت. خروجی در مبنای ۱۰ برابر ۲۴ است که عددی اول نیست ($X_{prime}=0$) و اگر آن را برعکس کنیم با خودش برابر می‌شود ($X_{bin_pal}=1$). مقدار Z صفر است پس F_{active} نیز صفر است. در ۶۷۰ تا ۶۸۰ نانو ثانیه ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

Rotation Left with Carry:

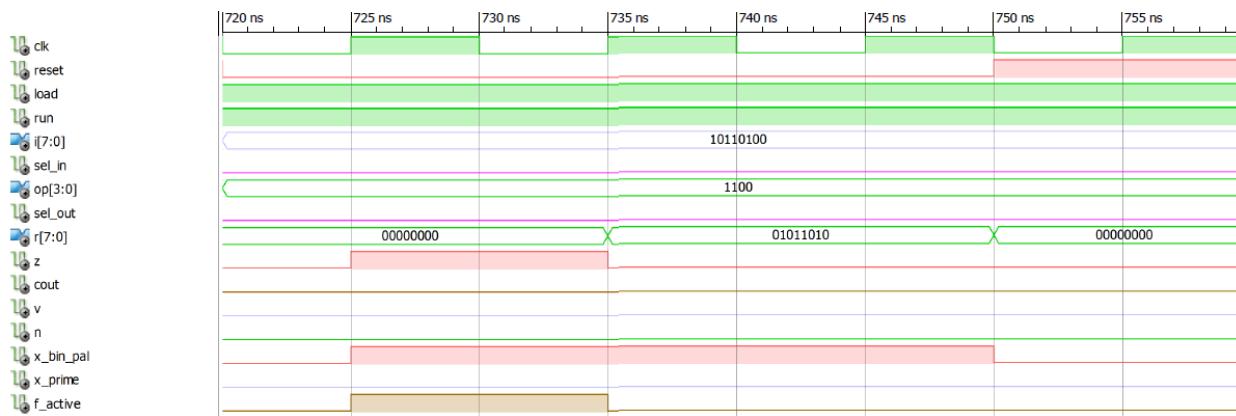


مقدار OPCODE برابر است با ۱۰۱۱ که زیرماژول RotL_CARRY می‌باشد. در ۶۸۰ تا ۶۹۰ نانو ثانیه ریست ۰ است. پایه **load** یک است پس ورودی دریافت می‌کنیم. پایه **SEL_IN** صفر است پس ورودی (01101010) در IA قرار خواهد گرفت. پایه **RUN** یک است پس فلیپ فلاپ‌های X_{bin_pal} , X_{prime} , Z و f_{active} رجیسترها را منقال می‌کند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منقال می‌کنند.

در این بازه خروجی صفر است زیرا صفر موجود در رجیستر rotate (00000000) می‌شود، پس Z مقدار ۱ می‌گیرد و در نتیجه F_{active} نیز ۱ می‌شود. خروجی (00000000) را برعکس کنیم با خودش برابر می‌شود پس X_{bin_pal} برابر ۱ می‌شود. صفر اول نیست پس X_{prime} صفر می‌شود.

در ۶۹۰ تا ۷۱۰ نانو ثانیه ریست ۰ است پایه **RUN** یک است پس فلیپ فلاپ‌های X_{bin_pal} , X_{prime} , Z و رجیسترها را منقال می‌کند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منقال می‌کنند. خروجی اصلی (11010100) به ما داده می‌شود. پایه **SEL_OUT** صفر است پس خروجی اول (X) در R قرار خواهد گرفت. خروجی در مبنای ۱۰ برابر ۲۱۲ است که عددی اول نیست ($X_{prime}=0$) و اگر آن را برعکس کنیم با خودش برابر نمی‌شود ($X_{bin_pal}=0$). مقدار Z صفر است پس F_{active} نیز صفر است. در ۷۱۰ تا ۷۲۰ نانو ثانیه ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

Logic Shift Right:

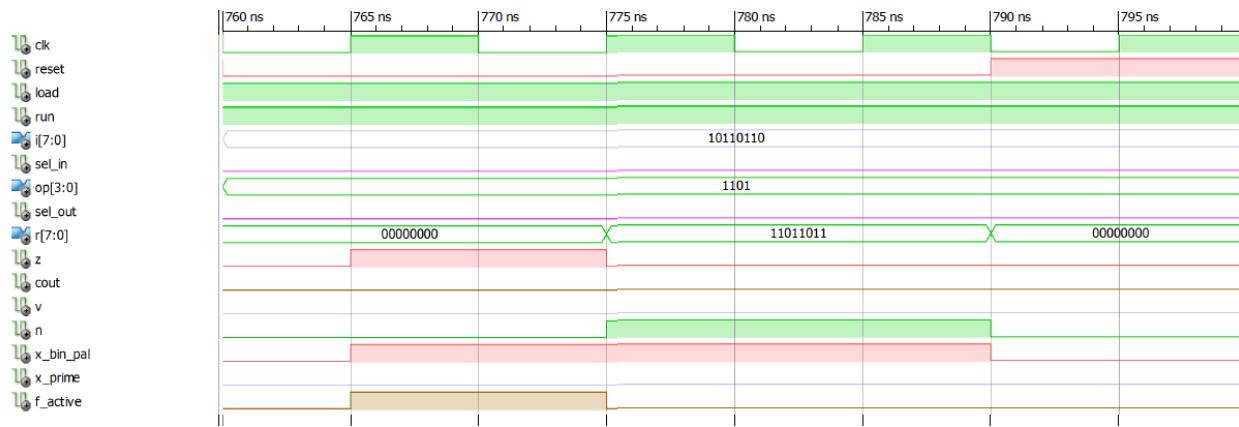


مقدار OPCODE برابر است با ۱۱۰۰ که زیرماژول LSR می‌باشد. در ۷۲۰ تا ۷۳۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی (10110100) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های X_bin_pal, X_prime, Z و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه خروجی صفر است زیرا صفر (00000000) دچار شیفت به سمت راست می‌شود. پس Z مقدار ۱ می‌گیرد و در نتیجه F_active نیز ۱ می‌شود. خروجی (00000000) را بر عکس کنیم با خودش برابر می‌شود پس X_bin_pal برابر ۱ می‌شود. صفر اول نیست پس X_prime صفر می‌شود.

در ۷۳۰ تا ۷۴۰ نانو ثانیه ریست ۰ است پایه RUN یک است پس فلیپ فلاپ‌های X_prime, Z و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. خروجی اصلی (01011010) به ما داده می‌شود. پایه SEL_OUT صفر است پس خروجی اول (X) در R قرار خواهد گرفت. خروجی در مبنای ۱۰ برابر ۹۰ است که عددی اول نیست و اگر آن را بر عکس کنیم با خودش برابر نمی‌شود ($X_{bin_pal}=0$). مقدار Z صفر است پس X_prime=0) صفر هستند.

Arithmetic Shift Right:

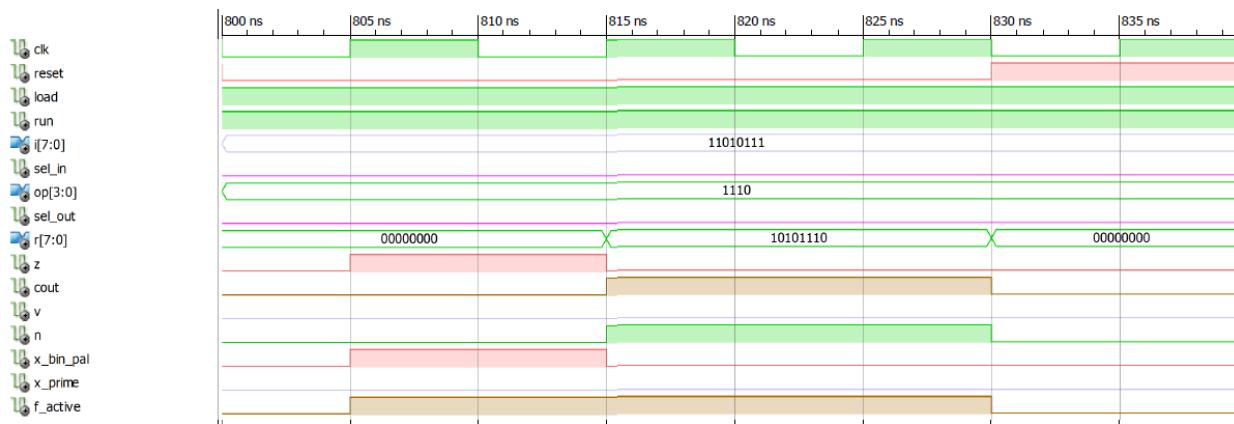


مقدار OPCODE برابر است با ۱۱۰۱ که زیرماژول ASR می‌باشد. در ۷۶۰ تا ۷۷۰ نانو ثانیه ریست · است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی (۱۰۱۱۰۱۱۰) در IA خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z, X_bin_pal, X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه خروجی صفر است زیرا صفر (۰۰۰۰۰۰۰۰) دچار شیفت به سمت راست می‌شود. پس Z مقدار ۱ می‌گیرد و در نتیجه F_active نیز ۱ می‌شود. خروجی (۰۰۰۰۰۰۰۰) را بر عکس کنیم با خودش برابر می‌شود پس X_bin_pal برابر ۱ می‌شود. صفر اول نیست پس X_prime صفر می‌شود.

در ۷۷۰ تا ۷۸۰ نانو ثانیه ریست · است پایه RUN یک است پس فلیپ فلاپ‌های Z, X_prime و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. خروجی اصلی (۱۱۰۱۱۰۱۱) به ما داده می‌شود. پایه SEL_OUT صفر است پس خروجی اول (X) در R قرار خواهد گرفت. خروجی در مبنای ۱۰ برابر ۲۱۹ است که عددی اول نیست ($X_{prime}=0$) و اگر آن را بر عکس کنیم با خودش برابر نمی‌شود ($X_{bin_pal}=0$). بیت خارج شده از سمت چپ که مقدار آن · است در Cout قرار می‌گیرد، پس F_active نیز صفر است. بیت آخر از سمت راست که مقدار آن ۱ است به عنوان بیت علامت در N قرار می‌گیرد. در ۷۹۰ تا ۸۰۰ نانو ثانیه ریست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

Logic Shift Left:

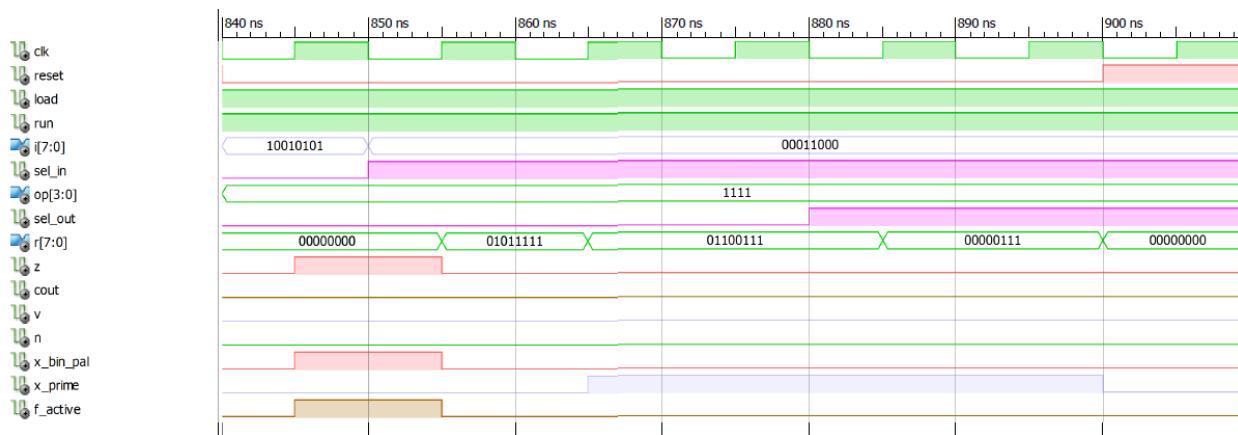


مقدار OPCODE برابر است با ۱۱۱۰ که زیرماژول LSL می‌باشد. در ۸۰۰ تا ۸۱۰ نانو ثانیه ریست ۰ است. پایه load یک است پس ورودی دریافت می‌کنیم. پایه SEL_IN صفر است پس ورودی (11010111) در IA قرار خواهد گرفت. پایه RUN یک است پس فلیپ فلاپ‌های Z, X_bin_pal, X_prime و رجیسترهای خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه خروجی صفر است زیرا صفر (00000000) دچار شیفت به سمت چپ می‌شود. پس Z مقدار ۱ می‌گیرد و در نتیجه F_active نیز ۱ می‌شود. خروجی (00000000) را بر عکس کنیم با خودش برابر می‌شود پس X_bin_pal ۱ می‌شود. صفر اول نیست پس X_prime صفر می‌شود.

در ۸۱۰ تا ۸۲۰ نانو ثانیه ریست ۰ است پایه RUN یک است پس فلیپ فلاپ‌های Z, X_prime و رجیسترهای خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. خروجی اصلی (10101110) به ما داده می‌شود. پایه SEL_OUT صفر است پس خروجی اول (X) در R قرار خواهد گرفت. خروجی در مبنای ۱۰ برابر ۱۷۴ است که عددی اول نیست (X_prime=0) و اگر آن را بر عکس کنیم با خودش برابر نمی‌شود (X_bin_pal=0). بیت خارج شده از سمت چپ که مقدار آن ۱ است در Cout قرار می‌گیرد، پس F_active نیز یک است. بیت آخر از سمت راست که مقدار آن ۱ است به عنوان بیت علامت در N قرار می‌گیرد. در ۸۳۰ تا ۸۴۰ نانو ثانیه ۱ نیست ۱ است در نتیجه تمامی متغیرهای خروجی صفر هستند.

BCD to Binary Conersion:



مقدار OPCODE برابر است با ۱۱۱۱ که زیرماژول BCD2BIN می‌باشد. در ۸۴۰ تا ۸۵۰ نانو ثانیه ریست است. پایه **load** یک است پس ورودی دریافت می‌کنیم. پایه **SEL_IN** صفر است پس ورودی اول **X_bin_pal** (۱۰۰۱۰۱۰۱) در IA قرار خواهد گرفت. پایه **RUN** یک است پس فلیپ فلاپ‌های **X_prime** و **Z** و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند.

در این بازه خروجی صفر است زیرا صفر در مبنای ۱۰ صفر است پس **Z** مقدار ۱ می‌گیرد و در نتیجه **X_bin_pal** نیز ۱ می‌شود. خروجی (۰۰۰۰۰۰۰۰) را برعکس کنیم با خودش برابر می‌شود پس **f_active** برابر ۱ می‌شود. صفر اول نیست پس **X_prime** صفر می‌شود.

در ۸۵۰ تا ۸۶۰ نانو ثانیه ریست است. پایه **load** یک است پس ورودی دریافت می‌کنیم. پایه **SEL_IN** یک است پس ورودی دوم (۰۰۰۱۱۰۰۰) در IB قرار خواهد گرفت. پایه **RUN** یک است پس فلیپ فلاپ‌های **Z**, **X_bin_pal**, **X_prime** و رجیسترها خروجی فعال هستند. در نتیجه هنگامی که کلاک ۱ شود (لبه بالارونده) فلیپ فلاپ‌ها و رجیسترها داده‌ها را منتقل می‌کنند. در اینجا چون هنوز ورودی دوم وارد نشده، ورودی اول را باینری تبدیل می‌کند و در خروجی نشان می‌دهد.

$$a \rightarrow \begin{cases} 0101 = (5)_{10} \\ 1001 = (9)_{10} \end{cases} \quad NUM = 5 + 9 * 10 = 95 = 01011111$$

این خروجی عددی اول نیست (**X_prime=0**) و اگر آن را برعکس کنیم با خودش برابر نمی‌شود (**X_bin_pal=0**). مقدار **Z** صفر است پس **f_active** نیز صفر است.

در ۸۶۰ تا ۸۷۰ نانو ثانیه هنگامی که کلاک ۱ شود خروجی اصلی به ما داده می‌شود که باید از هر دو خروجی استفاده کرد. پایه **SEL_OUT** صفر است پس خروجی اول (**X**) که شامل ۸ بیت اول از سمت راست خروجی بدست امده می‌باشد که در R قرار خواهد گرفت. در ۸۷۰ تا ۸۹۰ نانو ثانیه هنگامی که کلاک ۱ شود پایه

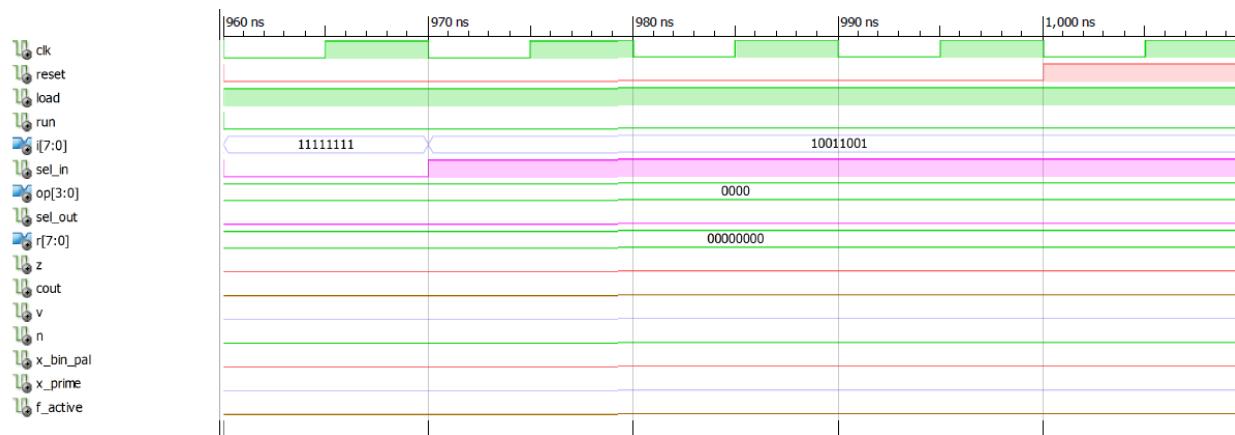
SEL_OUT یک است پس خروجی دوم (γ) که شامل ۸ بیت دوم از سمت راست خروجی بددست امده می باشد که در R قرار خواهد گرفت. در مبنای ۱۰ داریم:

$$a \rightarrow \begin{cases} 0101 = (5)_{10} \\ 1001 = (9)_{10} \end{cases} \quad b \rightarrow \begin{cases} 0001 = (1)_{10} \\ 1000 = (8)_{10} \end{cases}$$

$$NUM = 5 + 9 * 10 + 8 * 100 + 1 * 1000 = 1895 = (0000011101100111)_2$$

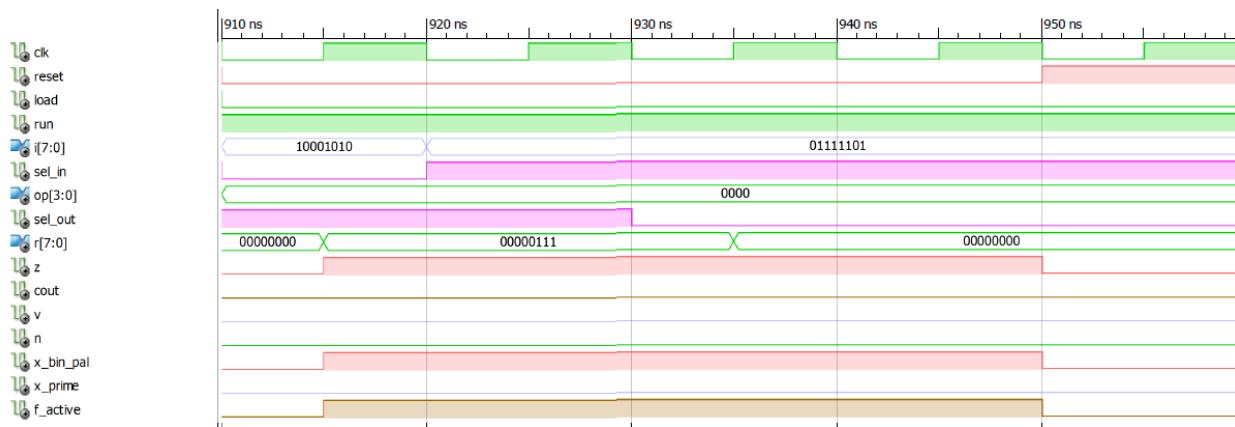
خروجی X در مبنای ۱۰ برابر 10^3 است که اول است ($X_{\text{prime}}=1$) و اگر آن را برعکس کنیم با خودش برابر نمی شود پس $X_{\text{bin_pal}}$ برابر ۰ می شود. برای بازه ۸۶۰ تا ۸۹۰ مقادیر V, Cout, Z همگی صفر است پس F_active صفر است. در نتیجه تمامی متغیرهای خروجی صفر هستند.

Run=0:



هنگامی که RUN صفر باشد فلیپ فلاپ های $X_{\text{bin_pal}}$, X_{prime} , Z و رجیسترها خروجی غیرفعال خواهند بود و چون قبل از آن دستگاه ریست شده بود مقادیر صفر نشان می دهند.

LOAD=0:



هنگامی که LOAD صفر باشد ما ورودی دریافت نخواهیم کرد و خروجی حاصل AND مقادیر اولیه IA و IB که صفر هستند می باشد. SEL_OUT یک است پس خروجی دوم (γ) را نشان می دهد که آن هم مقدار قبلی مازول

قبلی را دارد که اگر مقادیر را چک کنیم به درستی آن پی خواهیم برد (ماژول آخری که تاثیری بر خروجی ۷ داشته ماژول **BCD2Binary** است).