**دانشکده مهندسی برق دانشگاه صنعتی امیرکبیر**

# درس مدار منطقی

## پروژه چند مرحله ای alu

نام : محمد حسین کاشانکی – علی امیری راد

نام استاد: دکتر شریعتمدار مرتضوی

ماژول مذکور باید شامل سه ورودی زیر باشد. :۱ ورودی اول ۳۲ بیتی. ( A( ۲ ورودی دوم ۳۲ بیتی. ( B( ۳ سیگنال کنترلی 3 بیتی ALUControl)

```
a             : in STD_LOGIC_VECTOR (32-1 downto 0);
alucontroler : in STD_LOGIC_VECTOR (3-1 downto 0);
b             : in STD_LOGIC_VECTOR (32-1 downto 0);
```

ماژول مذکور باید شامل خروجی های زیر باشد. :۱ خروجی ۳۲ بیتی برای نتیجه. ( Y( ۲ خروجی یک بیتی بیت نقلی-( out Carry ( C)

```
c          : out STD_LOGIC;
y          : out STD_LOGIC_VECTOR (32-1 downto 0));
```

۳ خروجی یک بیتی برای نشان دادن صفر بودن خروجی. (Z (۴ خروجی یک بیتی بیت سرریز. (O )-( Overflow(۵ خروجی یک بیتی نشان دهنده منفی بودن نتیجه. ( N(۶ فلگ یک بیتی که نشان دهنده آماده بودن خروجی استReady(.

```
z          : out STD_LOGIC;
o          : out STD_LOGIC;
n          : out STD_LOGIC;
rdy        : out STD_LOGIC;
```

هر کدام از خواسته ها را در سیگنالی تعریف میکنیم

```
end component;
signal add_op : std_logic_vector(33-1 downto 0) :=( others=>'0');
signal sub_op : std_logic_vector(33-1 downto 0) :=( others=>'0');
signal mult_op : std_logic_vector(32-1 downto 0) :=( others=>'0');
signal slt_op : std_logic_vector(32-1 downto 0) :=( others=>'0');
signal And_Op : std_logic_vector(32-1 downto 0) :=( others=>'0');
signal or_op : std_logic_vector(32-1 downto 0) :=( others=>'0');
signal xor_op : std_logic_vector(32-1 downto 0) :=( others=>'0');
signal out_put : std_logic_vector(32-1 downto 0) :=( others=>'0');
signal start : std_logic := '0' ;
signal rdy_m : std_logic := '0'  ;
signal i : integer := 0  ;
signal a2 : signed (32-1 downto 0);
signal b2 : signed (32-1 downto 0);
```

برای استفاده از ماژول ضرب کننده به این صورت میزنیم

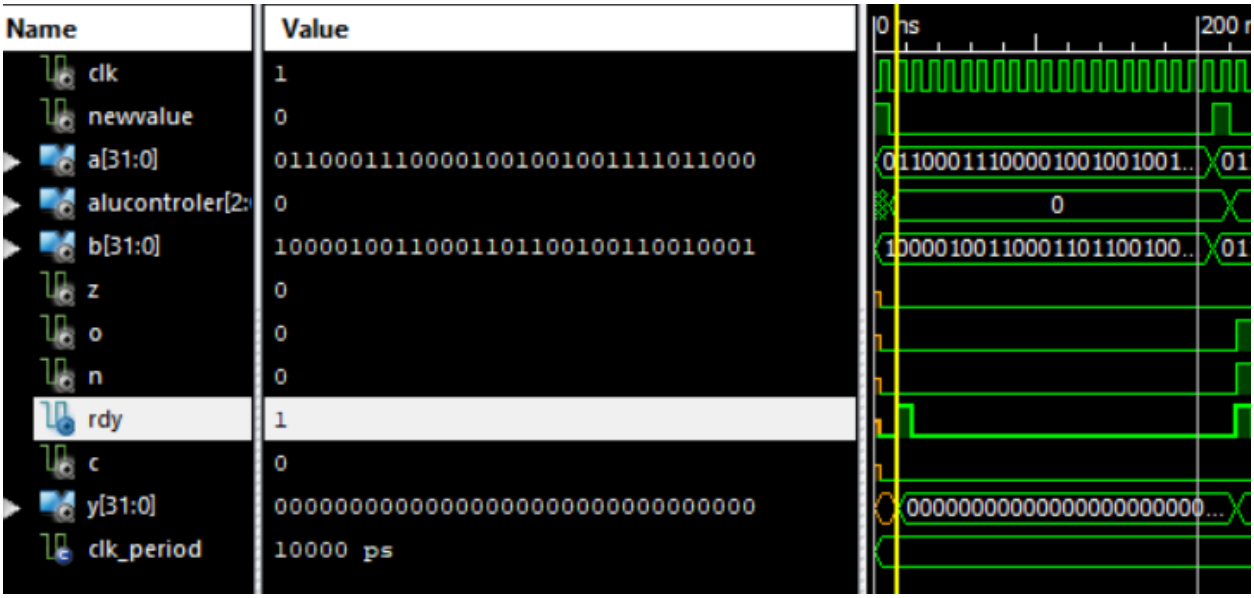
```
inst_mult: multi_16
    Port map (
            clk => clk,
            a  => a,
            b  => b,
            start => start,
            rdy   => rdy_m,
            o  => mult_op
             );
```

به سادگی دستورات and و or و xor را انجام میدهیم

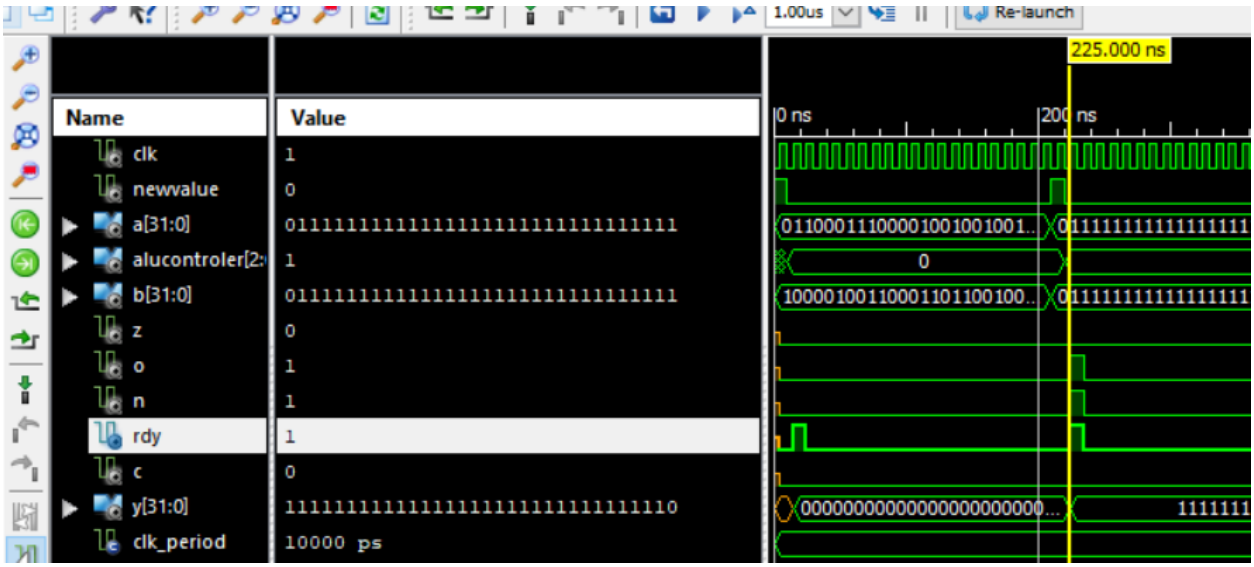
```
And_Op <= a and b ;
or_op <=  a or b ;
xor_op <= a xor b ;
```

در صورت nop خروجی ها را مشخص میکنیم

```
case (alucontroler) is
when "000" =>  -- NOp
                y <= x"00000000";
                z <= '0';
                o <= '0';
                c <= '0';
                n <= '0';
                rdy <= '1';
                i <= 0;
```



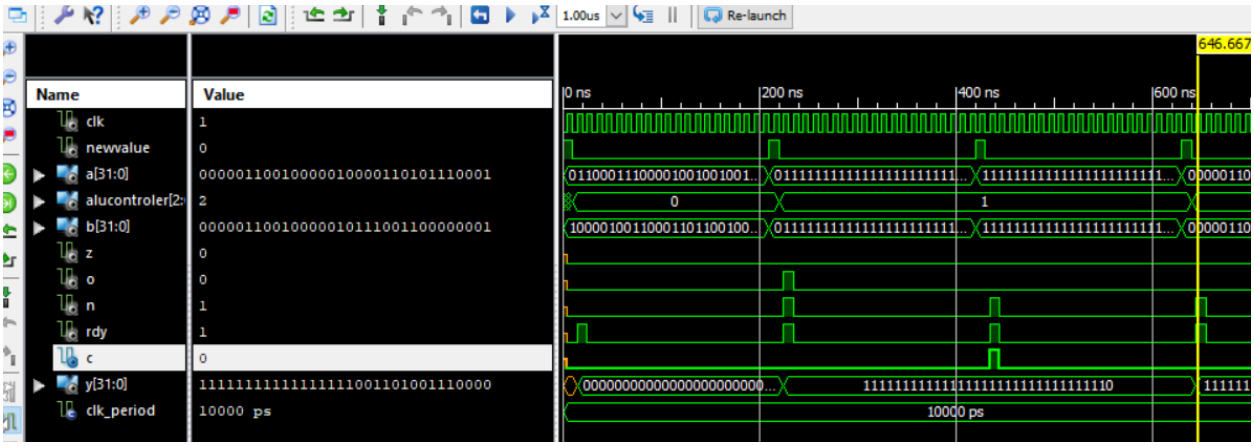

عملیات Add



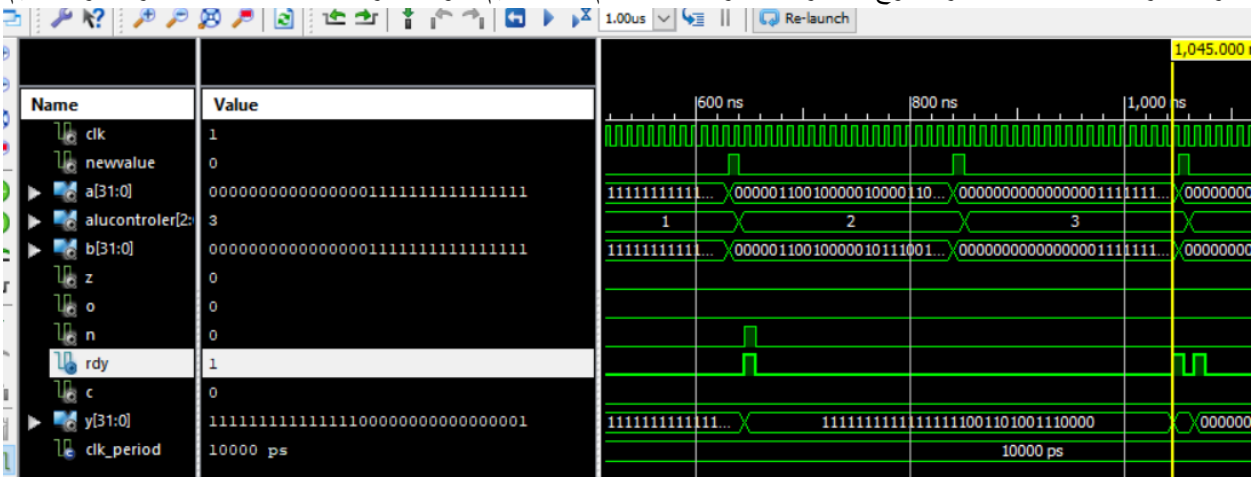

```
y <= add_op(32-1 downto 0);
rdy <= '1';
i <= 0;
if  ( signed (add_op(32-1 downto 0)) /= signed(add_op))   then --( sign
    o <= '1' ;
end if;
if add_op(32) = '1'  then
    c <= '1' ;
end if ;
if add_op(32-1 downto 0) = x"00000000" then
    z <= '0';
end if;
if   add_op(31) = '1' then
    n <= '1' ;
end if;
```

اورفلو را با بررسی بیت آخر چک میکنیم دقت داریم که ما یه بیت بیشتر تعریف کردیم که اگه اور رخ داد جواب رو داشته باشیم

تفریق



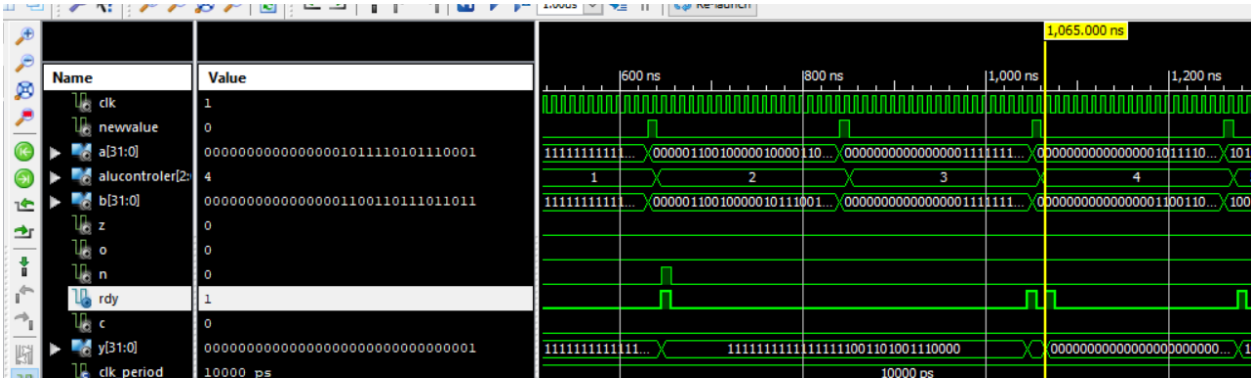

ضرب را با استفاده از نوع ضرب فاز 1 انجام دادیم و صرفا باعث شد ماهر تر بشیم



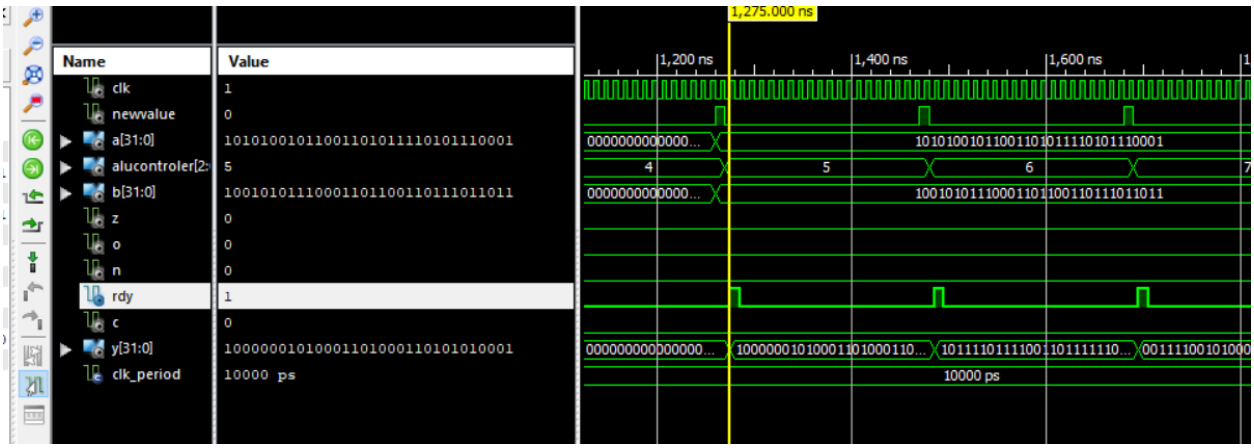

```
when "011" =>  -- Mult
                start <= '1' ;
                if rdy_m ='1' then
                    y <= mult_op ;
                    rdy <= '1';
                    start <= '0';
                    i <= 0;
                    if mult_op(32-1 downto 0) = x"00000000" then
                        z <= '0';
                    end if;
                  --  if  mult_op(31) = '1' then
                    --    n <= '1' ;
                   -- end if;
                end if;
               -- if  mult_op (32) = '1'  then
                --    o <= '1' ;
              --  end if;
```

٥. عملياتSLT

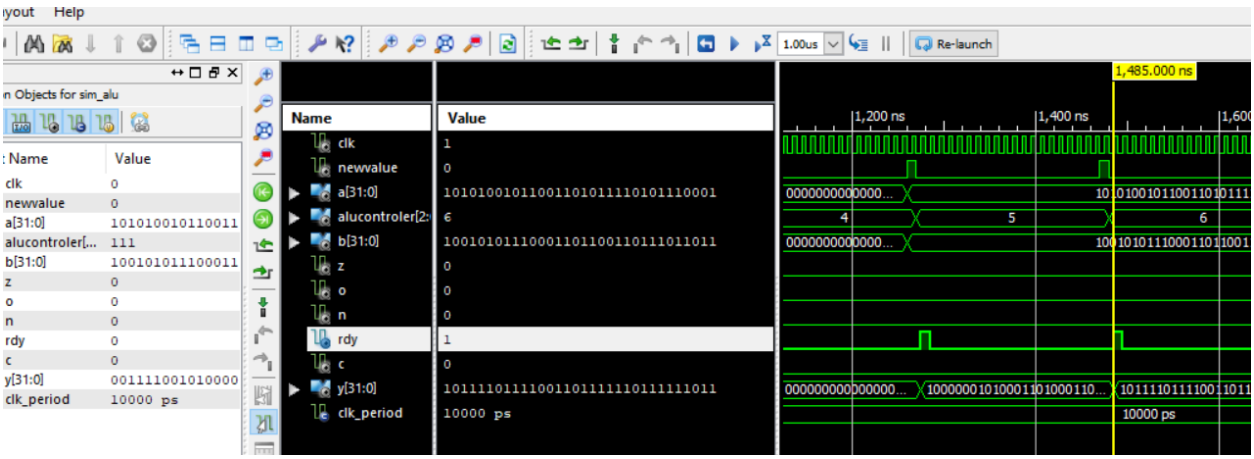
```
when "100" =>  -- Slt
                y <= slt_op;
                rdy <= '1';
                i <= 0;


                rdy <= '1';
            --    if slt_op (32) = '1'  then
              --      o <= '1' ;
                --end if;

                if slt_op(32-1 downto 0) = x"00000000" then
                    z <= '0';
                end if;
                if  slt_op(31) = '1' then
                    n <= '1' ;
                end if;
```



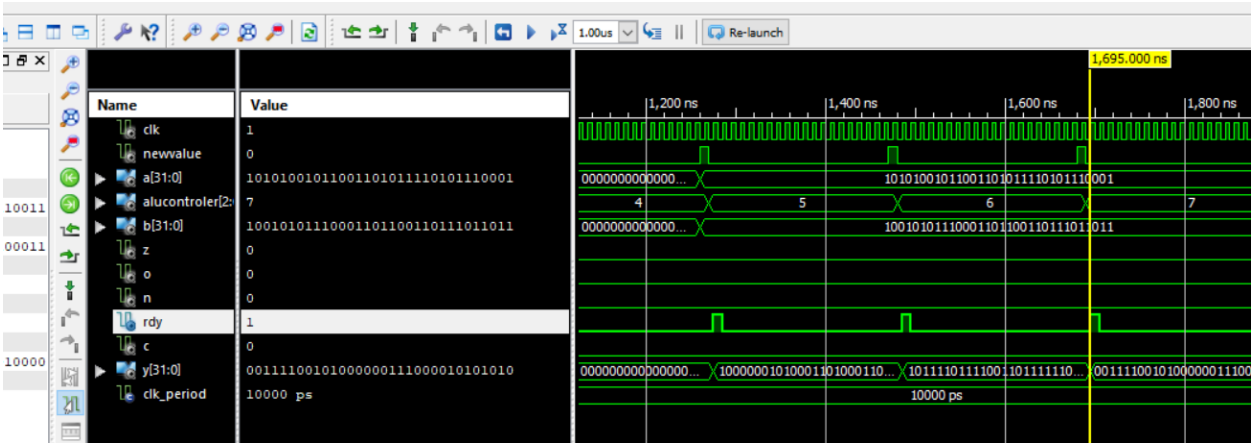

٦. عملياتAND




٧. عمليات OR




٨. عمليات XOR



یه متغییر درونی که وقتی ایک بشه یعنی ورودی جدید اومده و وقتی ورودی جدید کار باهاش تموم بشه صفر میکنیم

در کل این پروژه الگوریتمش ساده بود اما چون مراحلش زیاد بود یه کم طول کشید