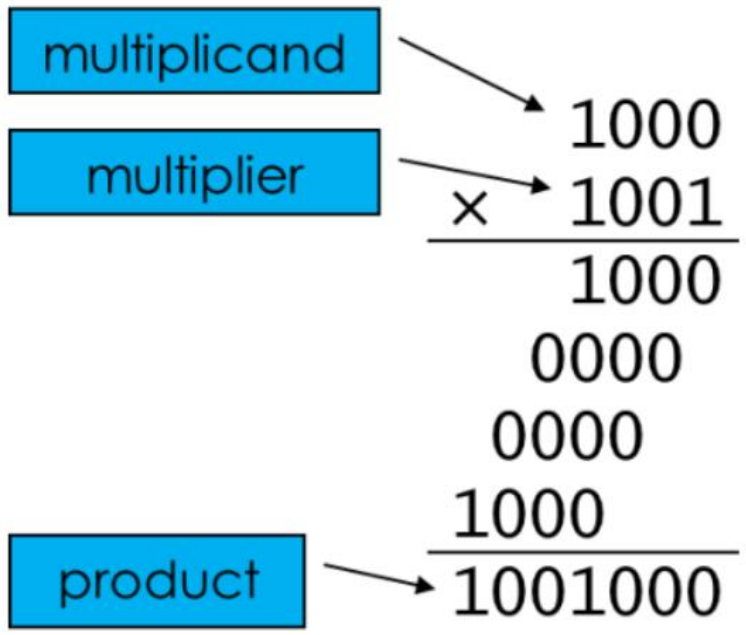
پروژه سوم
آزمایشگاه مدار منطقی و معماری

سپهر جعفری - عباس یزدان مهر

الف) پیاده سازی ماژول ضرب 6 بیتی

هدف از انجام این پروژه پیاده سازی یک ضرب کننده با استفاده از روش جمع و شیفت می باشد.
در این روش دو عدد را با نام های multiplicand یا ضرب شونده و multiplier یا ضرب کننده می نامیم. یک مثال از این روش با 4 بیت در روبه رو قابل مشاهده است.
توسط شیفت به راست multiplier و استفاده از اولین بیت آن در هر مرحله به عنوان شرط، multiplicand را با result جمع می کنیم.
و برای مرحله ی بعد multiplier را به چپ شیفت می دهیم تا در زمان استفاده مثل شکل رو به رو شود.



شکل ۱ نمونه ضرب به روش جمع و شیفت

```
1  module multiplier #(parameter N = 6)(
2      input clk,
3      input start,
4      input [5:0] a,
5      input [5:0] b,
6      output reg [11:0] out
7  );
```

برای نوشتن ماژول با ورودی ها و خروجی ها آغاز می کنیم:
ورودی هایمان شامل کلاک، زمان شروع ماژول، a(multiplicand)، b(multiplier) است.
خروجی ما شامل حاصل ضرب (out) است.
N یک متغیر محلی است که در یک شرط از آن استفاده می کنیم.

```
 9  reg [5:0] multiplier;
10  reg [11:0] multiplicand, result;
11  reg [5:0] counter;
12  reg initialValue;
```

سپس متغیر های کمکی را تعریف می کنیم:
Multiplier: برای نگه داری شیفت خورده ی عدد b.
Multiplicand: برای نگه داری شیفت خورده ی عدد a.
Result: برای نگه داشتن مجموع در هر مرحله.
Counter: برای توقف بعد از 6 بار شیفت و کلاک.
initialValue: برای شروع اصل ماژول.

```
16      if(start) begin
17
18          if(initialValue) begin
```

بعد از یک شدن start ماژول ما در بلاک always از If ابتدایی می گذرد و در غیر این صورت برنامه ی قبل را متوقف می کند.
بعد از آن initial value است که در اولین کلاک یک می شود.

```
if(N > counter)    begin
    if(multiplier[0] == 1) begin
        result = result + multiplicand;
    end
    multiplicand = multiplicand << 1;
    multiplier = multiplier >> 1;
    counter = counter + 1;
end    else begin
    initialValue = 0;
    out = result;
end
```

قسمت اصلی ماژول ما اینجاست.
توسط شیفت به راست multiplier و استفاده از اولین بیت آن در هر مرحله به عنوان شرط، multiplicand را با result جمع می کنیم.
و برای مرحله ی بعد multiplier را به چپ شیفت می دهیم تا در زمان استفاده مثل شکل رو به رو شود.

شرط (N > counter) باعث می شود که قسمت جلوی if در 6 کلاک اجرا شود و پس 6 کلاک، مقدار result در out ریخته می شود و مقدار initial value صفر شده که باعث می شود از این مراحل از ابتدا آغاز به کار کند.

پس متوجه شدیم که ضرب در این ماژول به اندازه 7 کلاک و به درستی به طول خواهد انجامید.

ب) فایل آزمون و شبیه سازی

ابتدا ورودی ها و خروجی ها را به صورت ذخیره کننده و سیم تعریف می کنیم.

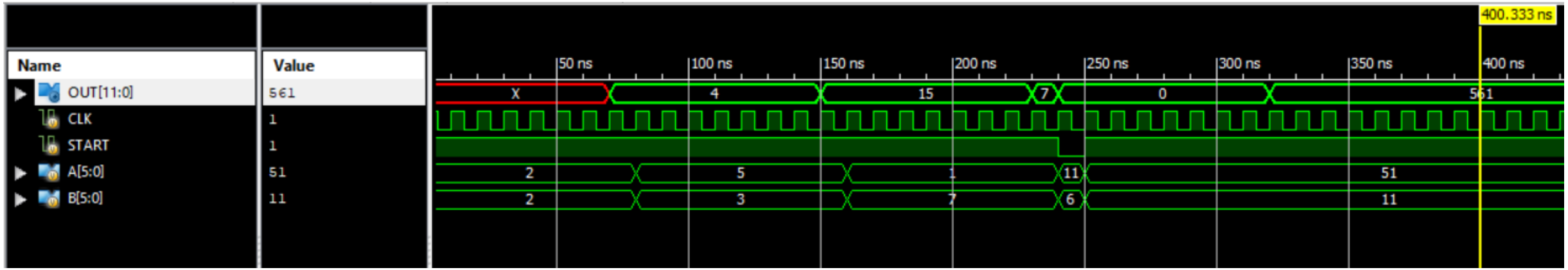
```
1  module multiplier_tb;
2  reg CLK, START;
3  reg [5:0] A, B;
4
5  wire [11:0] OUT;
6
7  multiplier UTT(.clk(CLK), .start(START), .a(A), .b(B),.out(OUT) );
8
```

سپس مقدار دهی را آغاز می کنیم.
ابتدا نوبت کلاک است که در آغاز آن را صفر قرار می دهیم و تعیین می کنیم که در هر 5 ثانیه یک بار کلاک نوسان کند و دوره ی تناوب کلاک 10 خواهد بود.

```
9  initial begin
10     CLK = 1;
11        forever #5 CLK = ~CLK;
12   end
13
```

در ادامه چند تست بررسی شده است که در قسمت شبیه سازی به آنها می پردازیم. دقت می کنیم که ماژول پس از 7 کلاک پاسخ را می دهد پس زمان مقدار دهی تا گرفتن پاسخ 7 کلاک طول کشیده(تاخیر دارد) و ما در کلاک 8 پاسخ مقدار دهی در کلاک یک را دریافت می کنیم.

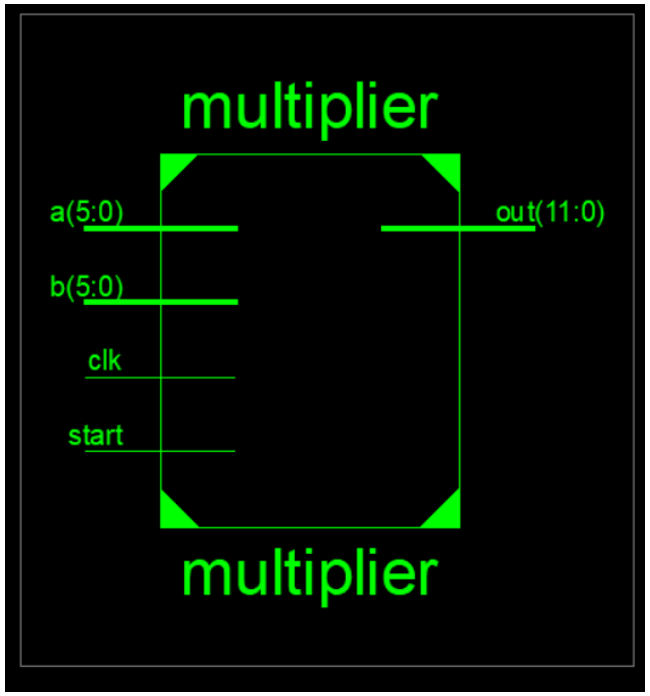
شبیه سازی زیر با توجه به تست بنچ نوشته شده است. ابتدا 2*2 ضرب شده است که پاسخ آن در زمان 70 نانو ثانیه یا 7 کلاک در out قرار گرفته است و قبل از آن out مقدار ندارد و x را نشان می دهد. و به همین صورت پیش می رویم و چندین حاصل را حساب می کنیم که مشاهده می شود که تست ها به درستی پاسخ می دهند. در زمان 240ns مقدار start را صفر می کنیم و همان طور که مشاهده می شود out در اولین کلاک برابر با صفر می شود و نه برابر حاصل ضرب 11 * 6 و به اندازه ی هفت کلاک صفر می ماند. و سپس مقدار

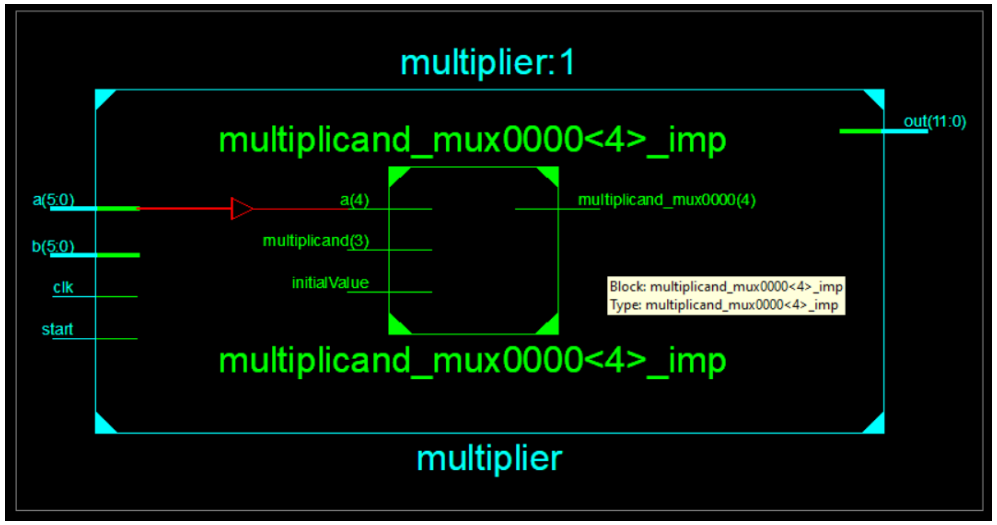


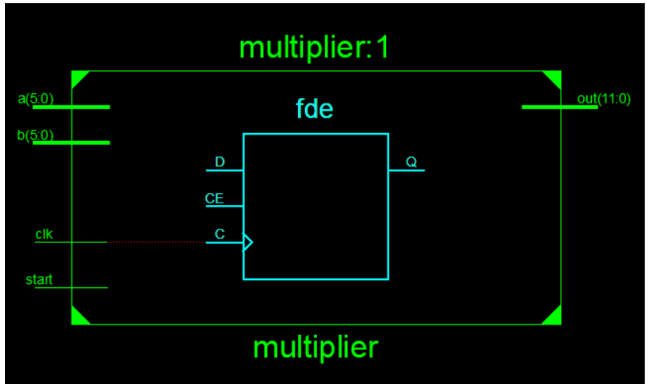
دقت می کنیم که ماژول پس از 7 کلاک پاسخ را می دهد پس زمان مقدار دهی تا گرفتن پاسخ 7 کلاک طول کشیده(تاخیر دارد) و ما در کلاک 8 پاسخ مقدار دهی در کلاک یک را دریافت می کنیم.

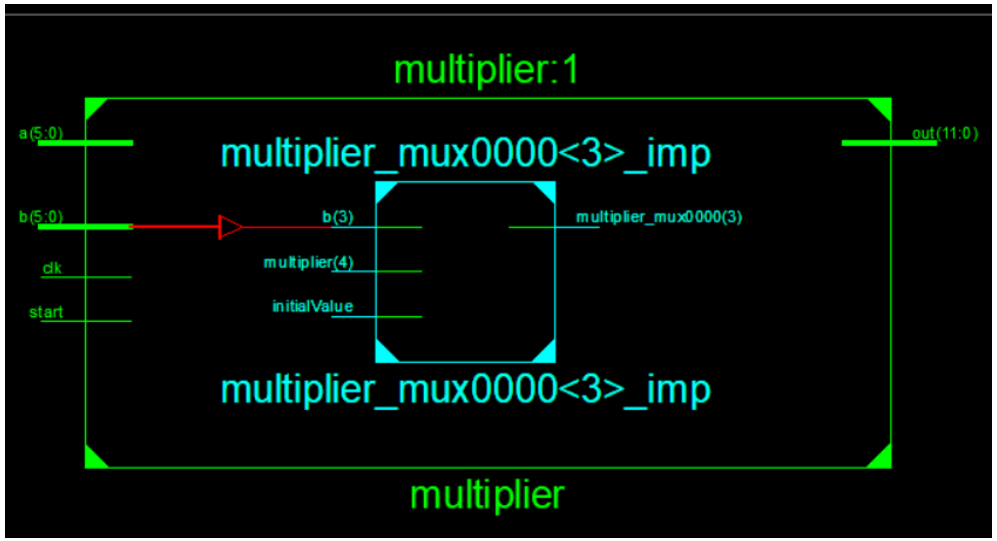
پ) سنتز با ISE

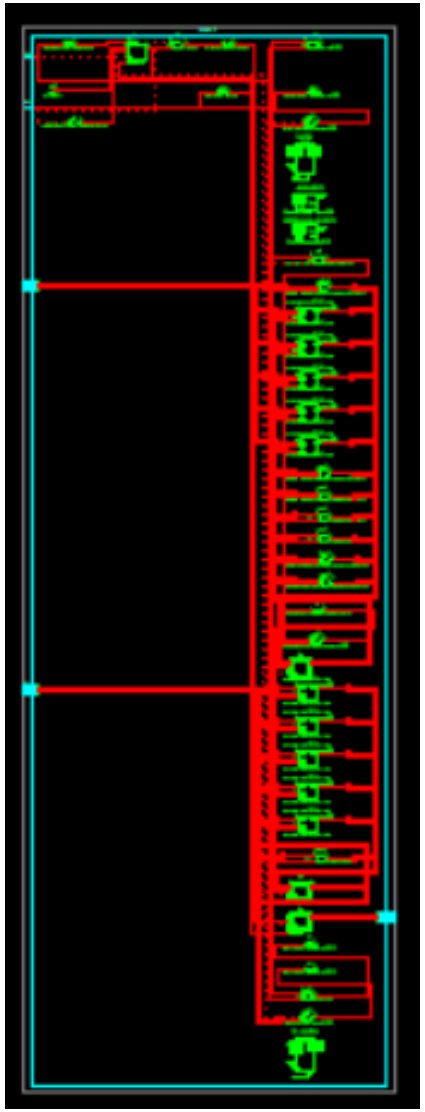
عکس های سنتز:

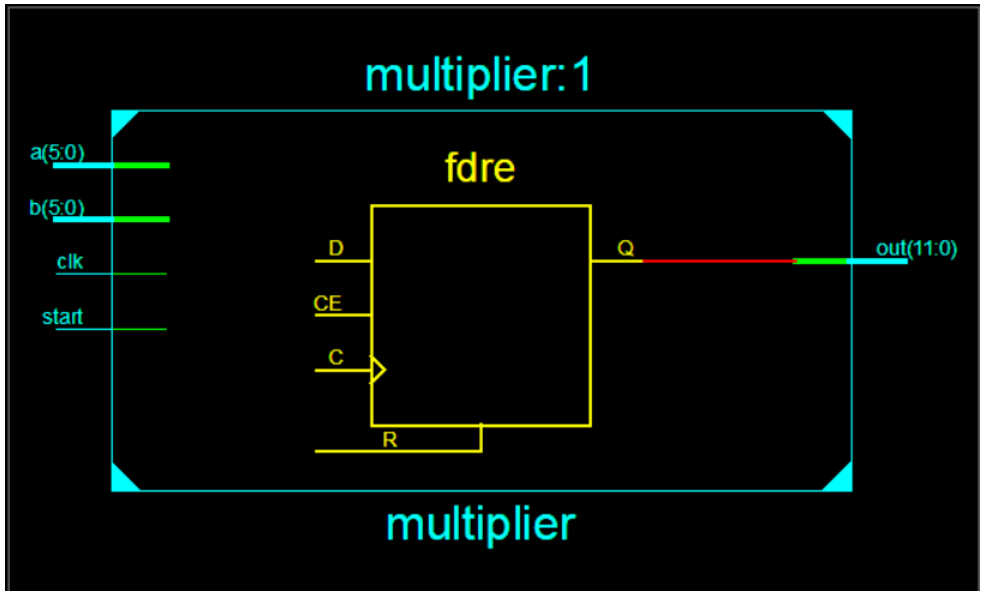

multiplier
a(5:0)
b(5:0)
clk
start
out(11:0)
multiplier

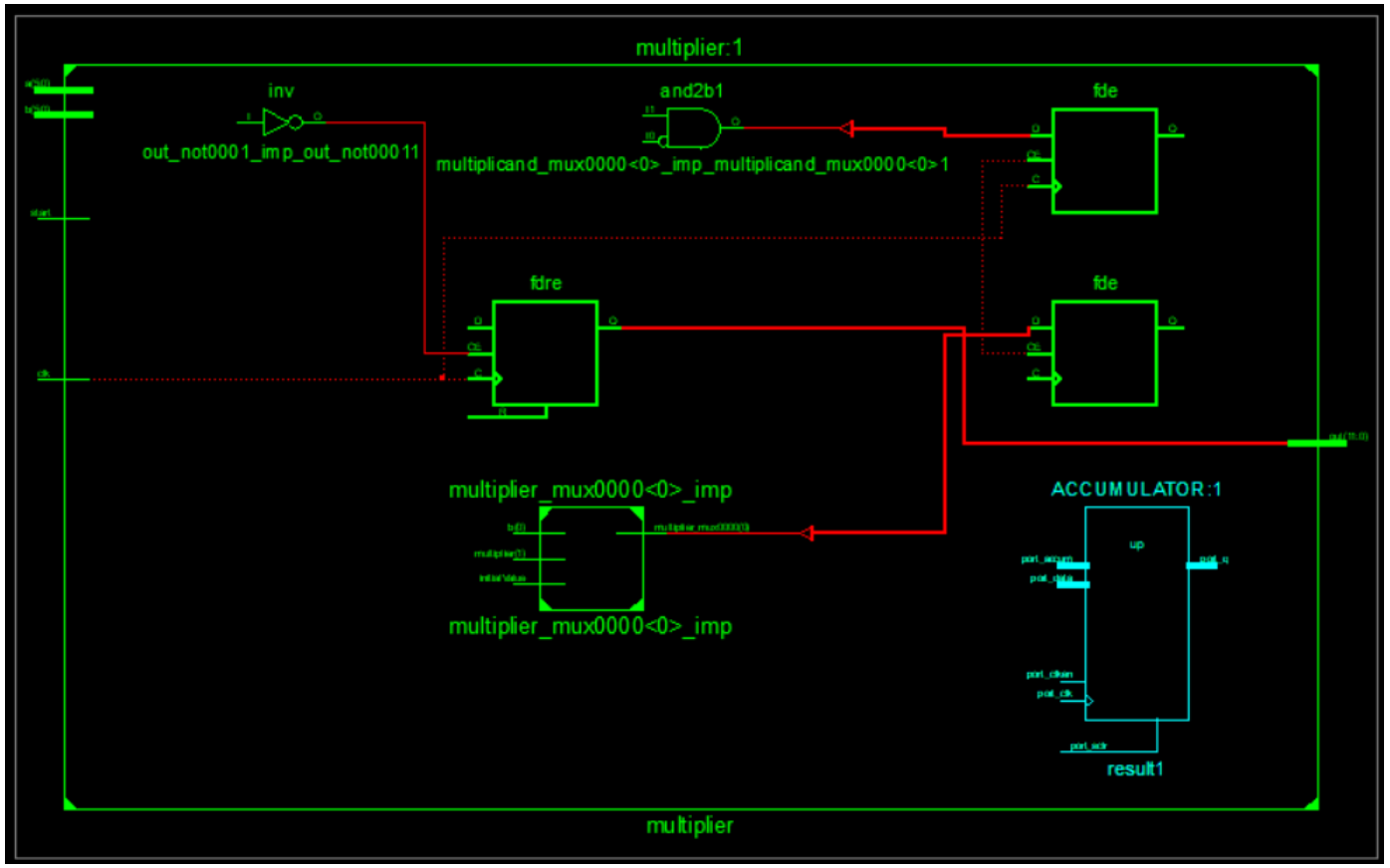

multiplier:1
multiplicand_mux0000<4>_imp
a(5:0)
b(5:0)
clk
start
a(4)
multiplicand(3)
initialValue
multiplicand_mux0000(4)
out(11:0)
Block: multiplicand_mux0000<4>_imp
Type: multiplicand_mux0000<4>_imp
multiplicand_mux0000<4>_imp
multiplier


multiplier:1
a(5:0)
b(5:0)
fde
D
CE
C
Q
clk
start
out(11:0)
multiplier


multiplier:1
a(5:0)
b(5:0)
clk
start
multiplier_mux0000<3>_imp
b(3)
multiplier(4)
initialValue
multiplier_mux0000(3)
out(11:0)
multiplier_mux0000<3>_imp
multiplier




multiplier:1
fdre
a(5:0)
b(5:0)
clk
start
D
CE
C
R
Q
out(11:0)
multiplier


multiplier:1
inv
out_not0001_imp_out_not00011
and2b1
multiplicand_mux0000<0>_imp_multiplicand_mux0000<0>1
fde
fdre
fde
multiplier_mux0000<0>_imp
multiplier_mux0000<0>_imp
ACCUMULATOR:1
up
result1
multiplier

Design Summary (ج

تاخیر زمانی:

```
Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 3.952ns (frequency: 253.008MHz)
  Total number of paths / destination ports: 595 / 116
-------------------------------------------------------------------------
Delay:               3.952ns (Levels of Logic = 2)
  Source:            counter_5 (FF)
  Destination:       multiplicand_0 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: counter_5 to multiplicand_0
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDRE:C->Q             6   0.514   0.721  counter_5 (counter_5)
     LUT4_L:I0->LO         1   0.612   0.103  multiplicand_not00011_SW0 (N17)
     LUT4:I3->O           18   0.612   0.908  multiplicand_not00011 (multiplicand_not0001)
     FDE:CE                    0.483          multiplicand_0
    ----------------------------------------
    Total                      3.952ns (2.221ns logic, 1.731ns route)
                                       (56.2% logic, 43.8% route)
```

انواع منابع موجود در FPGA:

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	49	1,920	2%	
Number of 4 input LUTs	48	1,920	2%	
Number of occupied Slices	32	960	3%	
Number of Slices containing only related logic	32	32	100%	
Number of Slices containing unrelated logic	0	32	0%	
Total Number of 4 input LUTs	48	1,920	2%	
Number of bonded IOBs	26	66	39%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	2.98			

End...