دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش تمرین چهارم

درس طراحی سیستم های دیجیتال برنامه‌پذیر

استاد درس

دکتر صاحب الزمانی

نگارش

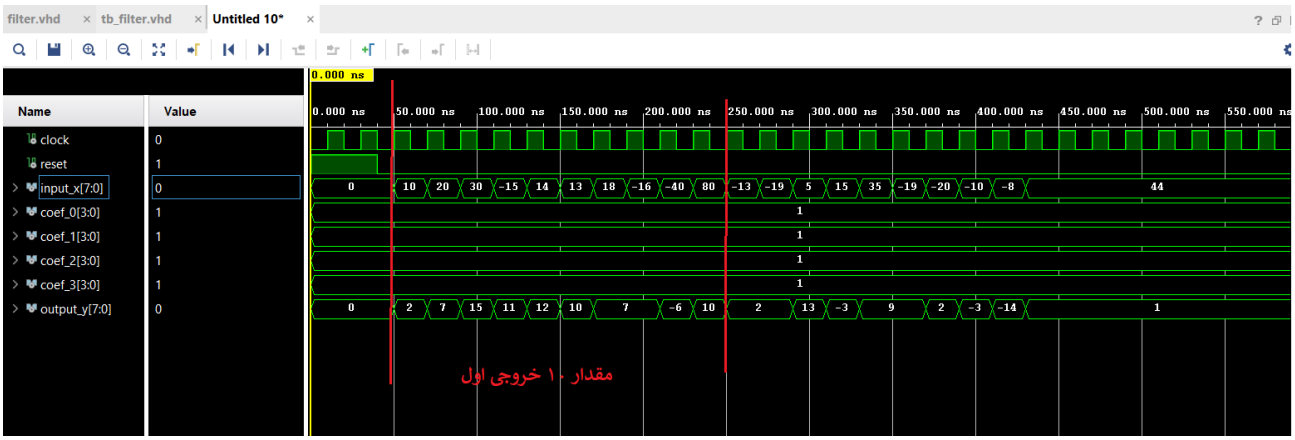
آرش حاجی صفی - ۹۶۳۱۰۱۹

خرداد ۱۳۹۹

گزارش:

سوال ۴-۲:

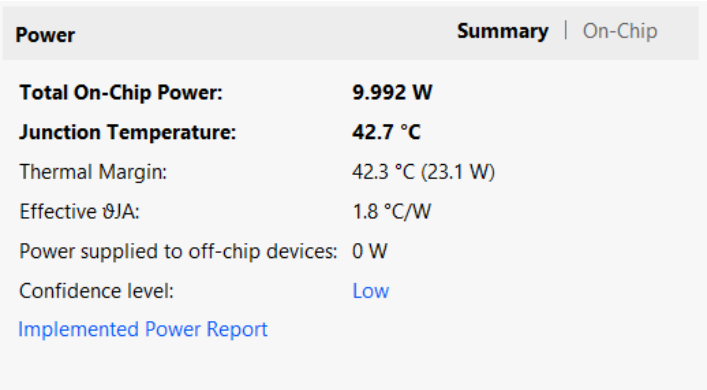
ب) شکل موج ۱۰ خروجی اول به صورت زیر می‌باشد:

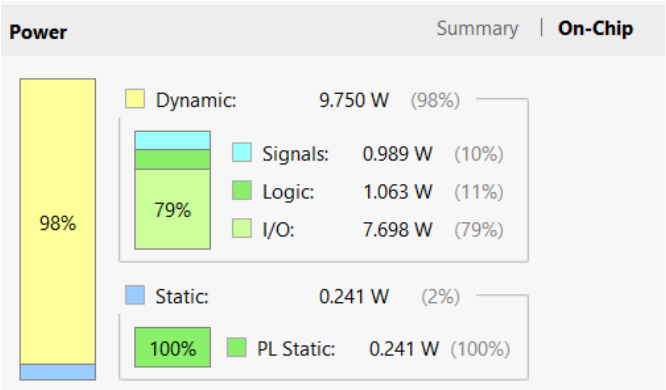


ج)

توان مصرفی:

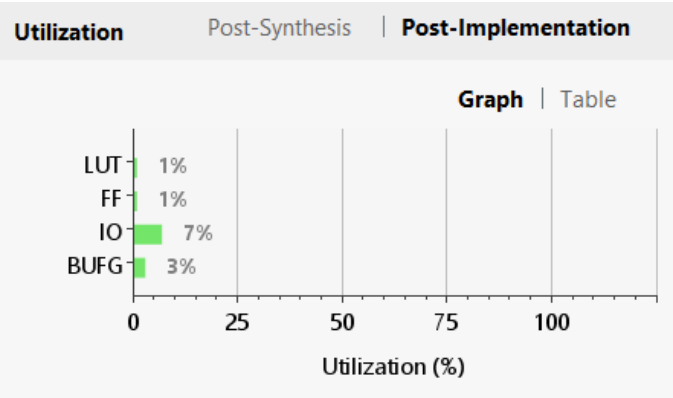
در مجموع مصرف توان چیپ، **۹.۹۹۲ وات**، به شرح زیر است:

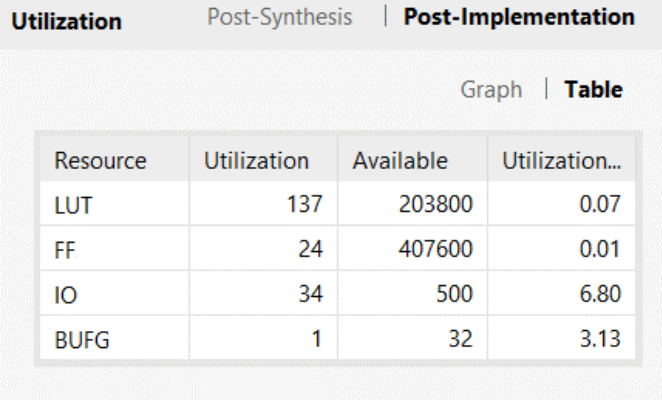




منابع مصرفی: تعداد منابع مصرفی به شرح زیر است:

نوع منبع	تعداد منابع مصرفی
	پس از پیاده سازی
Look-up Table	137
Flip-Flop	24
BRAM	0
DSP	0
IO	34
BUFF-Gates	1



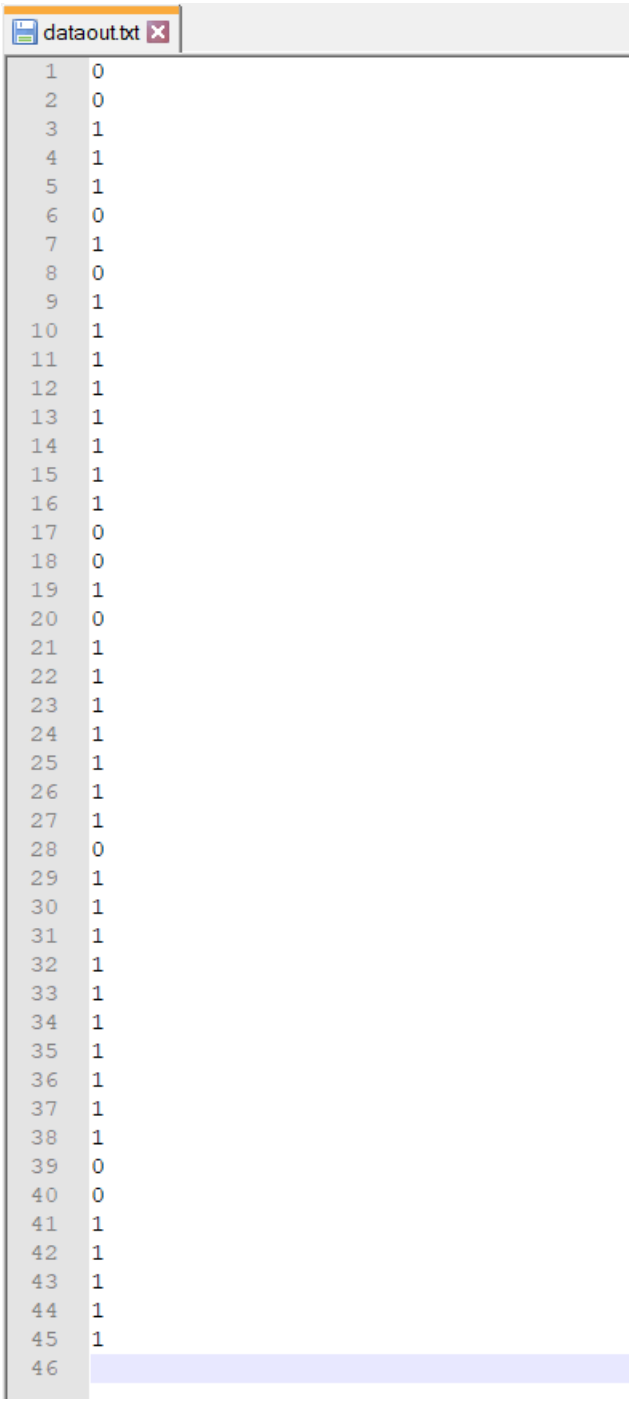


Resource	Utilization	Available	Utilization...
LUT	137	203800	0.07
FF	24	407600	0.01
IO	34	500	6.80
BUFG	1	32	3.13

سوال ۴-۳:

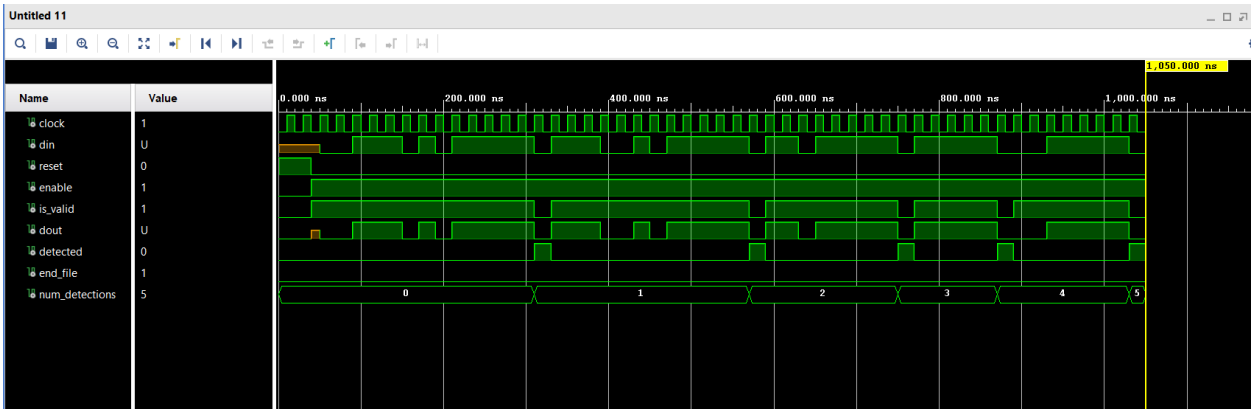
ج)

خروجی فایل تکست:

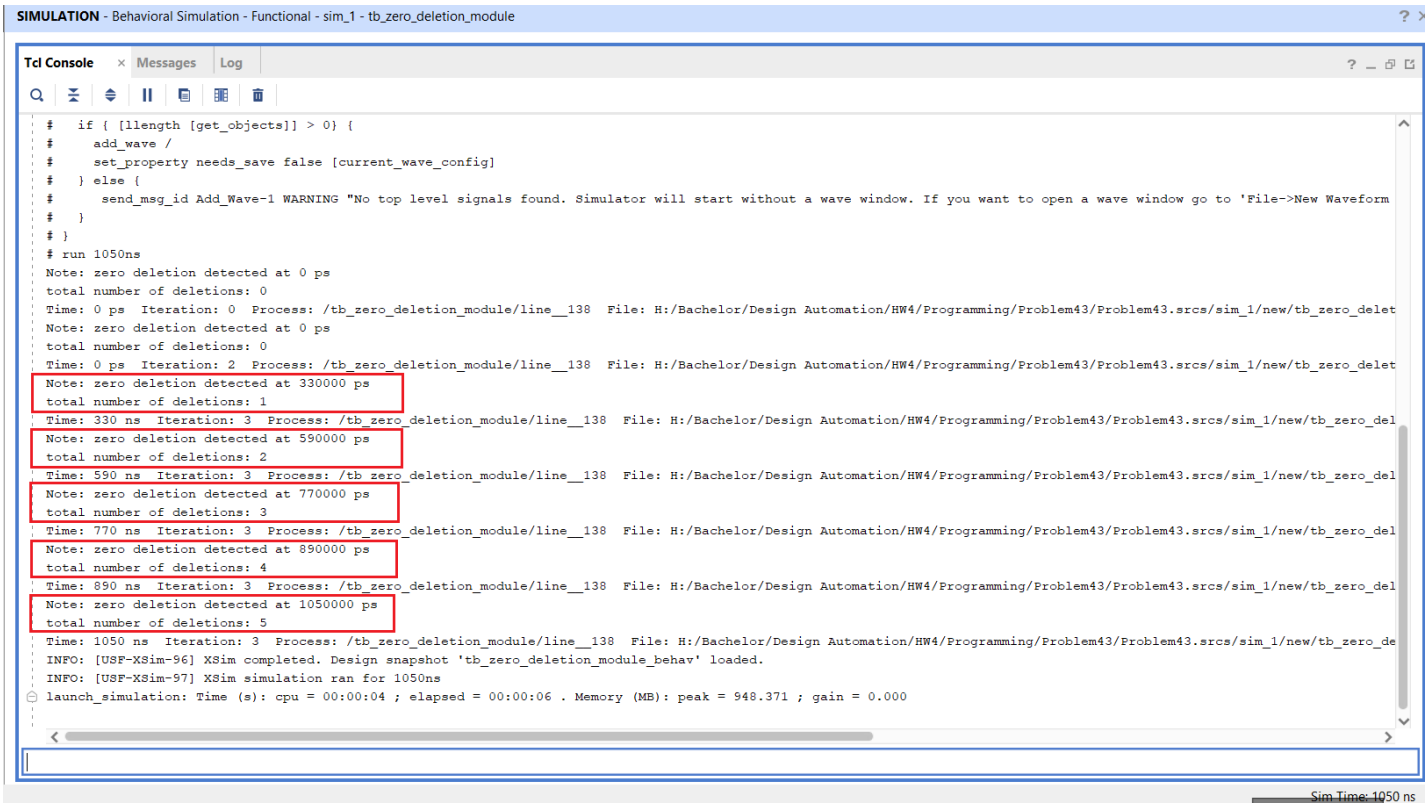

dataout.txt

```
0
0
1
1
1
0
1
0
1
1
1
1
1
1
1
1
0
0
1
0
1
1
1
1
1
1
1
0
1
1
1
1
1
1
1
1
0
0
1
1
1
1
1
```

شکل موج خروجی:

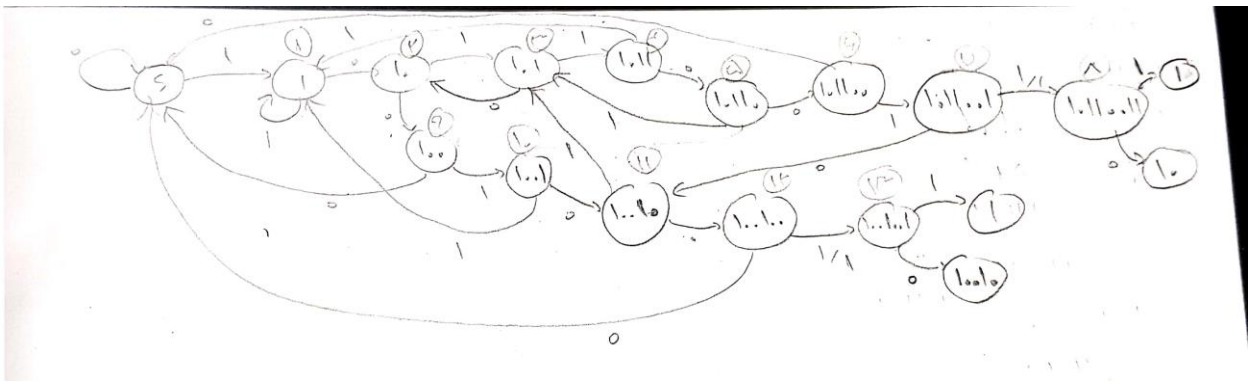


نتیجه ی دستور assert در کنسول:

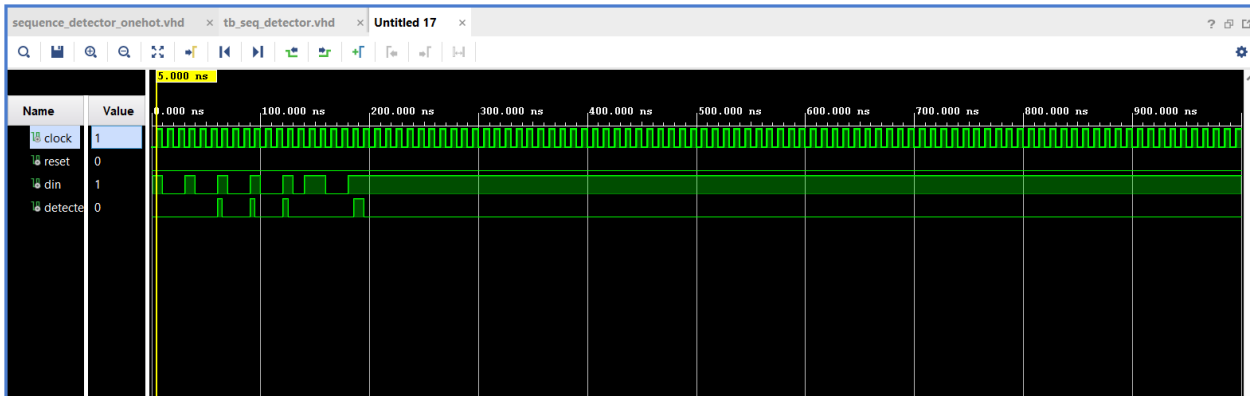


سوال ۴-۴:

ماشین حالت طراحی شده که هردو دنباله را به صورت هم‌پوشان تشخیص می‌دهد:



شکل موج خروجی این ماشین برای دنباله ی 10010010010010110011:



الف) با دستور زیر، نوع state encoding را one-hot قرار دادم:

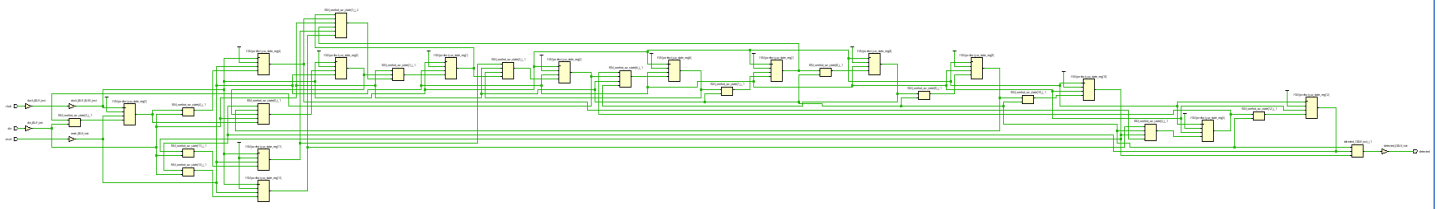
```
attribute fsm_encoding : string;
attribute fsm_encoding of cur_state : signal is "one-hot";
```

خروجی log سنتز را به صورت فایل one-hot synthesis log.txt در فولدر تمرین 4-4 ضمیمه کردم، قسمت قابل توجه که نوع encoding را نشان می‌دهد به این صورت می‌باشد:

Tcl Console | Messages | Log | Reports | Design Runs

```
INFO: [Device 21-403] Loading part xc7k325tffg900-2
INFO: [Synth 8-802] inferred FSM for state register 'cur_state_reg' in module 'sequence_detector'
---------------------------------------------------------------------------------------------------
                   State |                     New Encoding |                Previous Encoding
---------------------------------------------------------------------------------------------------
                   start |                  00000000000001 |                            0000
                      s1 |                  00000000000010 |                            0001
                      s2 |                  00000000000100 |                            0010
                      s9 |                  00000000001000 |                            1001
                     s10 |                  00000000010000 |                            1010
                     s11 |                  00000000100000 |                            1011
                      s3 |                  00000001000000 |                            0011
                      s4 |                  00000010000000 |                            0100
                      s5 |                  00000100000000 |                            0101
                      s6 |                  00001000000000 |                            0110
                      s7 |                  00010000000000 |                            0111
                      s8 |                  00100000000000 |                            1000
                     s12 |                  01000000000000 |                            1100
                     s13 |                  10000000000000 |                            1101
---------------------------------------------------------------------------------------------------
INFO: [Synth 8-3354] encoded FSM with state register 'cur_state_reg' using encoding 'one-hot' in module 'sequence_detector'
```

خروجی شماتیک هم به صورت one-hot schematic.sch در فولدر این تمرین ضمیمه کردم، شماتیک این مدار به این صورت می‌باشد:



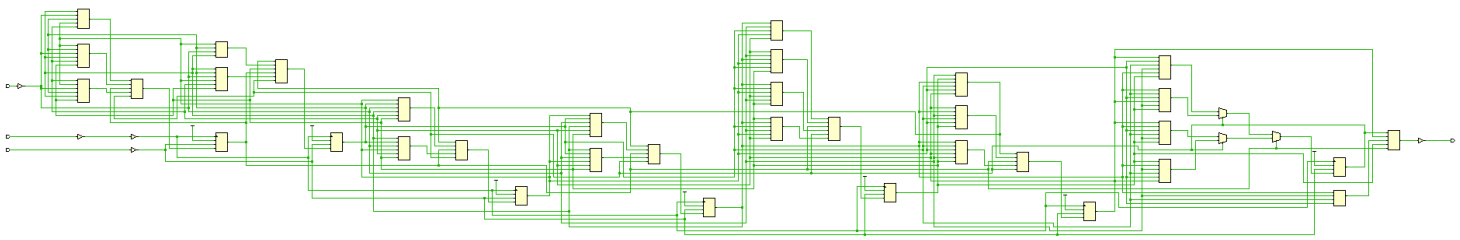
ب) با دستور زیر، نوع state encoding را johnson قرار دادم:

```
attribute fsm_encoding : string;
attribute fsm_encoding of cur_state : signal is "johnson";
```

خروجی log سنتز را به صورت فایل johnson synthesis log.txt در فولدر تمرین 4-4 ضمیمه کردم، قسمت قابل توجه که نوع encoding را نشان می‌دهد به این صورت می‌باشد:

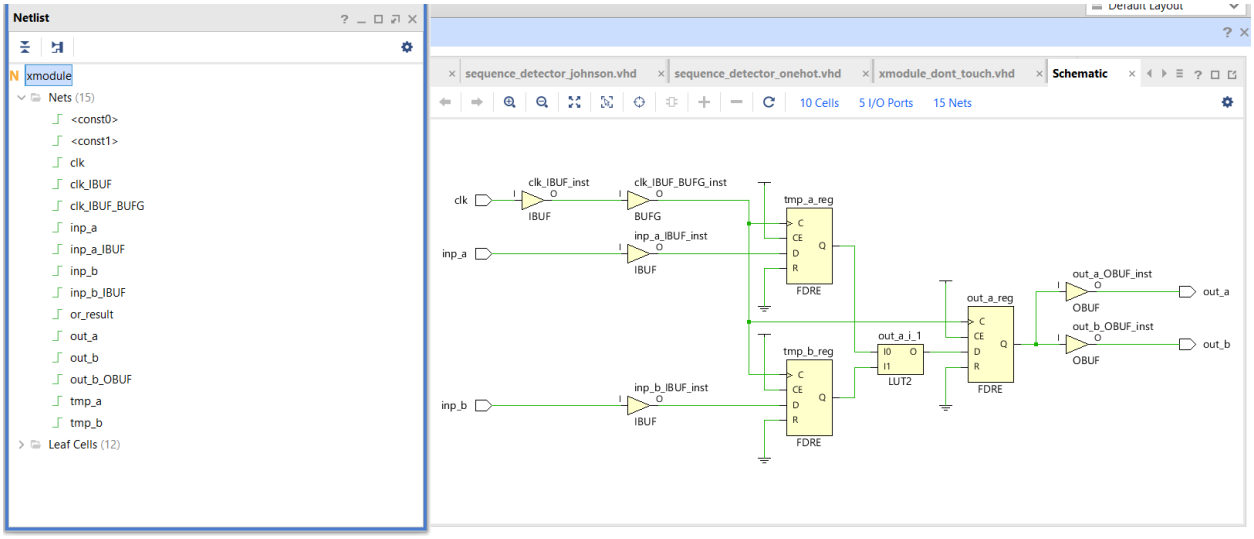
```
INFO: [Device 21-403] Loading part xc7k325tffg900-2
INFO: [Synth 8-802] inferred FSM for state register 'cur_state_reg' in module 'sequence_detector'
---------------------------------------------------------------------------------------------------
                   State |                     New Encoding |               Previous Encoding
---------------------------------------------------------------------------------------------------
                   start |                          0000000 |                             0000
                      s1 |                          1000000 |                             0001
                      s2 |                          1100000 |                             0010
                      s9 |                          1110000 |                             1001
                     s10 |                          1111000 |                             1010
                     s11 |                          1111100 |                             1011
                      s3 |                          1111110 |                             0011
                      s4 |                          1111111 |                             0100
                      s5 |                          0111111 |                             0101
                      s6 |                          0011111 |                             0110
                      s7 |                          0001111 |                             0111
                      s8 |                          0000111 |                             1000
                     s12 |                          0000011 |                             1100
                     s13 |                          0000001 |                             1101
---------------------------------------------------------------------------------------------------
INFO: [Synth 8-3354] encoded FSM with state register 'cur_state_reg' using encoding 'johnson' in module 'sequence_detector'
---------------------------------------------------------------------------------
Finished RTL Optimization Phase 2 : Time (s): cpu = 00:00:07 ; elapsed = 00:00:06 . Memory (MB): peak = 622.824 ; gain = 321.949
---------------------------------------------------------------------------------
```

خروجی شماتیک هم به صورت johnson schematic.sch در فولدر این تمرین ضمیمه کردم، شماتیک این مدار به این صورت می‌باشد:



ج)

لیست اتصالات پس از سنتز به صورت xmodule_not_changed_netlist و xmodule_not_changed_netlist.v در فولدر این تمرین ذخیره گردید. شماتیک هم به صورت xmodule_not_changed_schematic.sch در فولدر این تمرین ذخیره شد. نتیجه ی هردو با هم به صورت زیر است که می‌بینیم wire_a و wire_b حذف شده اند:

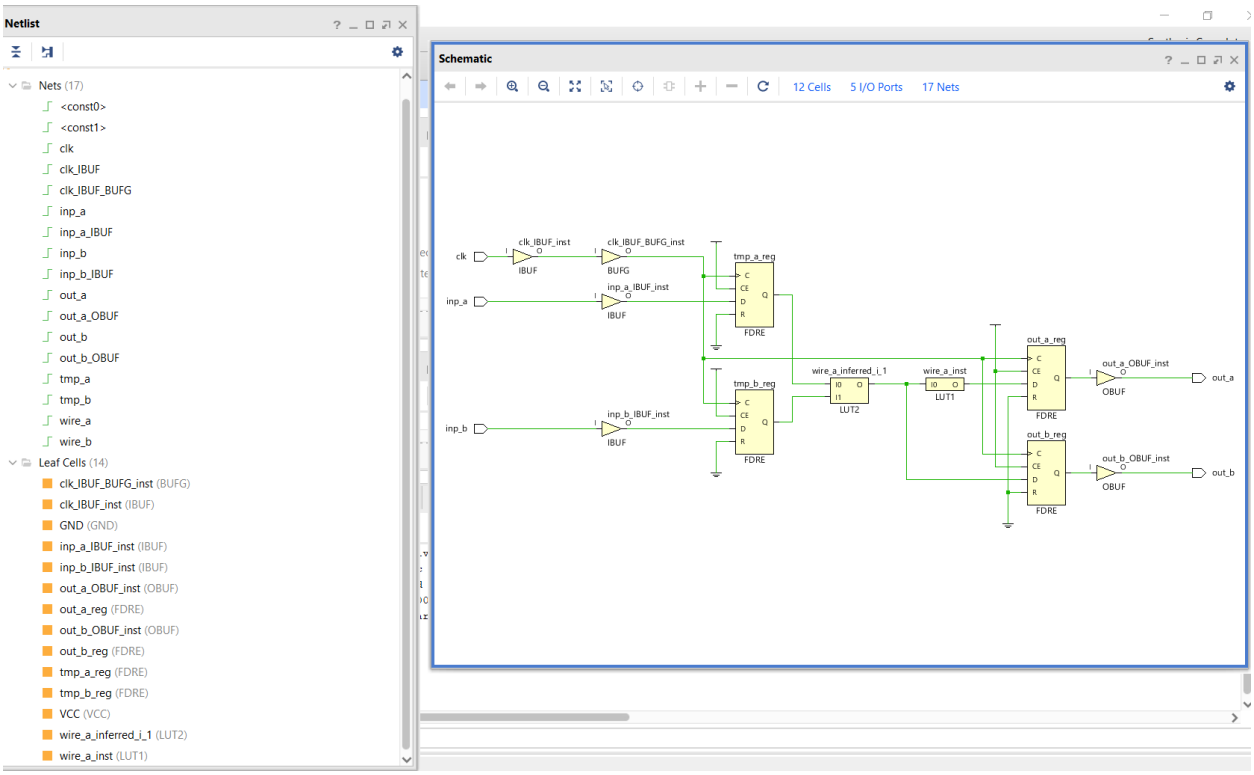


د)

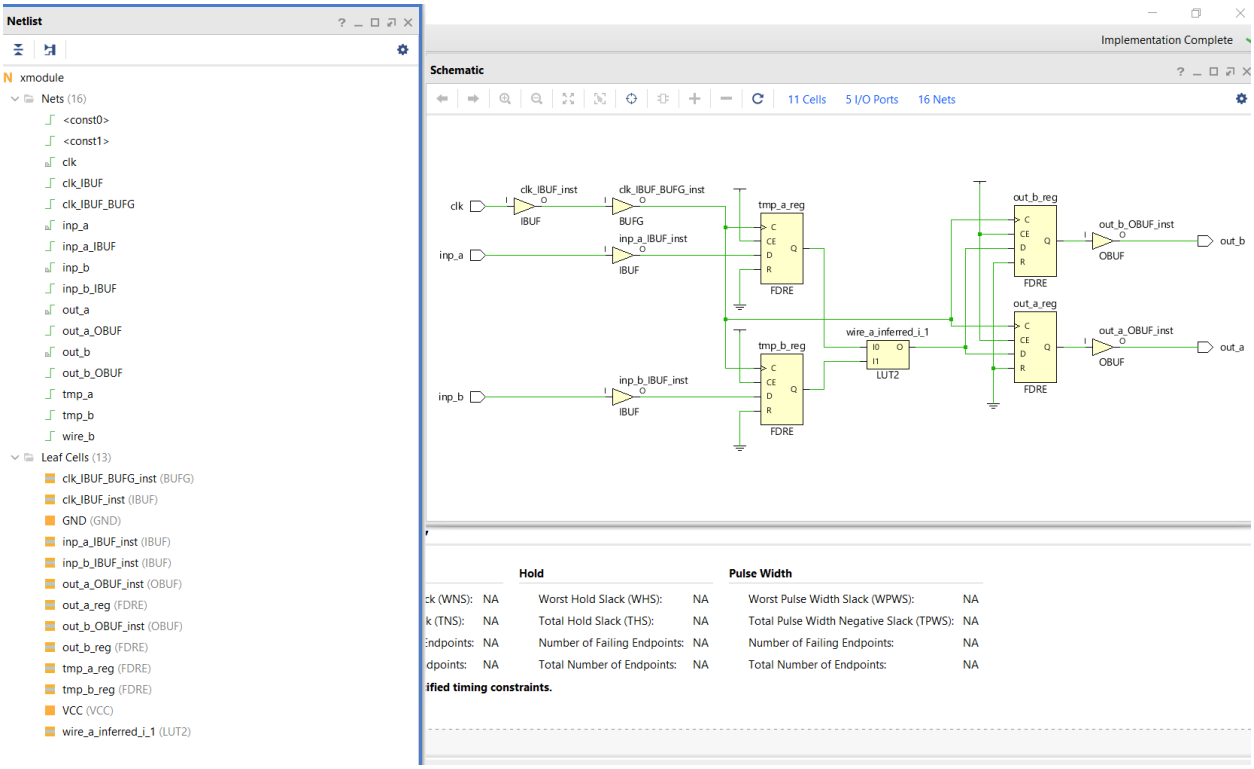
با دستور زیر، keep را برای wire_a و wire_b اعمال می‌کنیم:

```
-- write your code here, add keep attribute for wire_a and wire_b signals
attribute keep : string;
attribute keep of wire_a : signal is "true";
attribute keep of wire_b : signal is "true";
```

لیست اتصالات و شماتیک پس از سنتز به صورت زیر است:



لیست اتصالات و شماتیک پس از پیاده سازی به صورت زیر است:



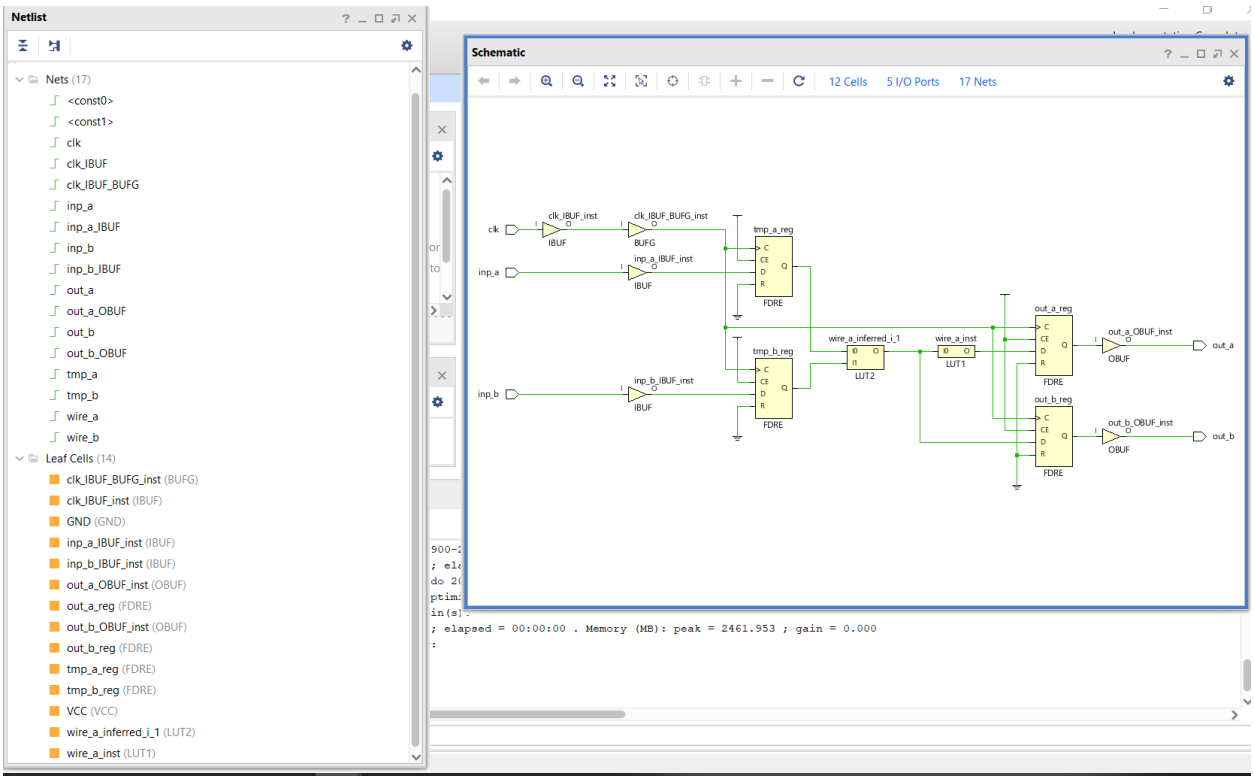
فایل های شماتیک و netlist در فولدر سوال ذخیره شدند.

۵)

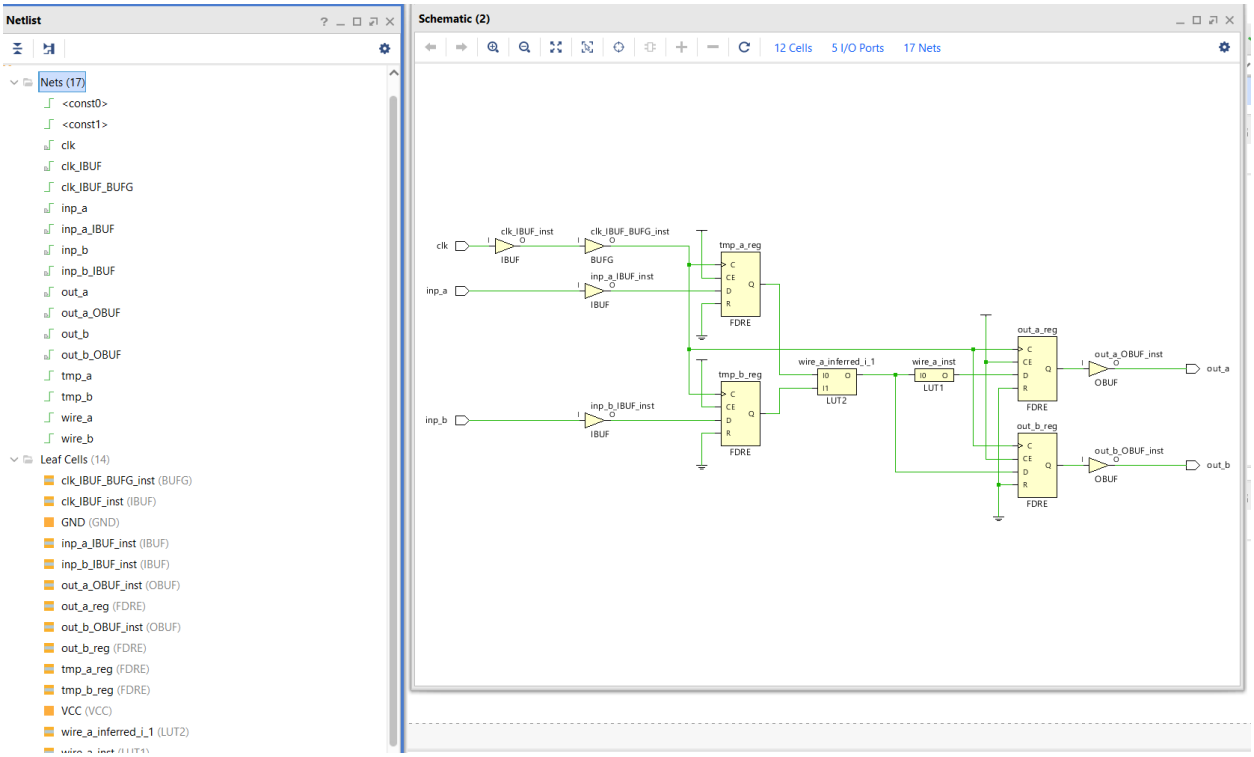
با دستور زیر، dont_touch را برای wire_a و wire_b اعمال می‌کنیم:

```
-- write your code here, add don't touch attribute for wire_a and wire_b signals
attribute dont_touch : string;
attribute dont_touch of wire_a : signal is "true";
attribute dont_touch of wire_b : signal is "true";
```

لیست اتصالات و شماتیک پس از سنتز به صورت زیر است:



لیست اتصالات و شماتیک پس از پیاده سازی به صورت زیر است:



فایل های شماتیک و netlist در فولدر سوال ذخیره شدند.

و)

همانطور که از نتایج مشخص است، KEEP از بهینه سازی روی سیگنال ها در مرحله ی سنتز جلوگیری می‌کند و نمی‌گذارد ابزار سنتز سیگنالهای تکراری را به عنوان مثال در این xmodule حذف کند و روی logic block جذب کند ولی در مرحله ی place and route که در implementation اتفاق می‌افتد، بهینه سازی ممکن است صورت بگیرد (بهینه سازی روی منطق) و سیگنالی حذف یا بهینه شود و keep جلوی آنرا نمی‌گیرد.

DONT_TOUCH هم به طور مشابه جلوی بهینه سازی روی سیگنالها را در مرحله ی سنتز می‌گیرد ولی تفاوت آن با KEEP در این است که DONT_TOUCH جلوی بهینه سازی در مرحله ی place and route که در implementation اتفاق می‌افتد را هم می‌گیرد و از بهینه سازی روی logic جلوگیری می‌کند و هیچ سیگنالی حذف یا بهینه نمی‌شود.

به عنوان مثال در همین xmodule، بدون اعمال هیچکدام از KEEP و DONT_TOUCH می‌بینیم که بهینه سازی صورت گرفته و سیگنالهای wire_a و wire_b از بین رفته اند چراکه نیازی به آنها نبوده و تکراری بوده اند.

در حالتی که از KEEP استفاده کرده ایم جلوی هرگونه بهینه سازی روی این سیگنالها در فاز سنتز گرفته شده است، در حالت استفاده از DONT_TOUCH هم دقیقاً مشابه همین است در فاز سنتز، ولی پس از implementation می‌بینیم که در حالتی که از KEEP استفاده کرده ایم یک بهینه سازی اتفاق افتاده و یک LUT که به نام wire_a_ins است حذف شده، البته هنوز هم جلوی بعضی بهینه سازی های دیگر که اگر از KEEP و DONT_TOUCH استفاده نمی‌کردیم، اعمال می‌شد، گرفته شده. اما در زمان استفاده از DONT_TOUCH این بهینه سازی اتفاق نیفتاده و این LUT حذف نشده و نتیجه ی بعد از سنتز و بعد از implementation دقیقاً یکی است.