BAB IV

Coding Latches

2.1. Tujuan

1. Praktikan memahami konsep d-latch dan sr-latch
2. Praktikan memahami konsep SR flip-flop dan JK flip-flop
3. Praktikan dapat mengubah semua konsep latch dan flip flop ke bahasa pemrograman Verilog
4. Praktikan dapat mengimplementasi konsep latch dan flip-flop ke papan FPGA Nexys A7

2.2. Alat dan bahan

1. Laptop
2. Papan Nexys A7

2.3. Dasar Teori

1. Latches

Latch adalah sirkuit digital yang menyimpan satu bit informasi dan mempertahankan nilainya tersebut hingga diperbarui dengan sinyal masukkan. Latch digunakan pada sistem digital sebagai elemen penyimpanan sementara untuk menyimpan informasi biner. Latch dapat diimplementasikan dengan berbagai macam gerbang logika seperti AND, OR, NOT, NAND, dan NOR.

Terdapat 2 macam Latch, yaitu:

a. SR (Set Reset) Latch

SR-Latch merupakan latch yang paling sederhana yang menggunakan dua masukkan dimana masukkan SET akan membuat sinyal keluaran menjadi informasi 1 dan masukkan RESET akan mengembalikan sinyal keluaran menjadi 0 Berikut adalah tabel kebenarannya.

S	**R**	$\bar{Y}$	**Y**
0	0	latch	latch
0	1	1	0
1	0	0	1
1	1	Invalid/0	Invalid/0

b. D (Data) Latch

D-Latch, atau dapat disebut juga sebagai *transparent latches*, adalah latch yang hanya mengambil 1 masukkan dan sinyal keluarannya bergantung pada satu masukkan ini. Berikut adalah tabel kebenarannya

D	$\overline{Y}$	**Y**
0	1	0
1	0	1

2. Flip-Flops

Flip-flop adalah suatu rangkaian elektronika yang memiliki dua kondisi stabil dan dapat digunakan untuk menyimpan informasi. Flip Flop merupakan pengaplikasian gerbang logika yang bersifat Multivibrator Bistabil. Dikatakan Multibrator Bistabil karena kedua tingkat tegangan keluaran pada Multivibrator tersebut adalah stabil dan hanya akan mengubah situasi tingkat tegangan keluarannya saat dipicu (trigger). Flip-flop mempunyai dua Output (Keluaran) yang salah satu outputnya merupakan komplemen Output yang lain.

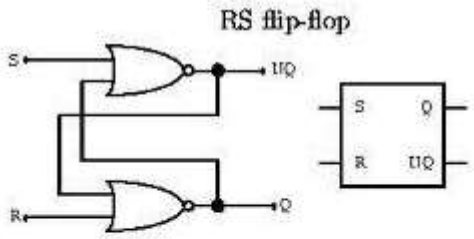
Flip-flop Elektronik yang pertama kali ditemukan oleh dua orang ahli fisika Inggris William Eccles and F. W. Jordan pada tahun 1918 ini merupakan dasar dari penyimpan data memory pada komputer maupun Smartphone. Flip-flop juga dapat digunakan sebagai penghitung detak dan sebagai penyinkronsasian input sinyal waktu variabel untuk beberapa sinyal waktu referensi.

- RS Flip-Flop

RS FF ini adalah dasar dari semua Flip-flop yang memiliki 2 gerbang inputan / masukan yaitu R dan S. R artinya “RESET” dan S artinya “SET”. Flip-flop yang satu ini mempunyai 2 keluaran / outputyaitu Q dan Q`.

Bila S diberi logika 1 dan R diberi logika 0, maka output Q akan berada pada logika 0 dan Q not pada logika 1. Bila R diberi logika 1 dan S diberi logika 0 maka keadaan output akan berubah menjadi Q berada pada logik 1 dan Q not pada logika 0. Sifat paling penting dari Flip-Flop adalah bahwa sistem ini dapat menempati salah satu dari dua keadaan stabil yaitu stabil I diperoleh saat Q =1 dan Q not = 0, stabil ke II diperoleh saat Q=0 dan Q not .

Berikut adalah Symbol dan Tabel kebenaran dari RS Flip-Flop:

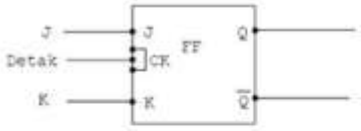


S	R	Q	$\overline{Q}$
0	0	no change	
0	1	0	1
1	0	1	0
1	1	undefined	

- JK Flip-Flop

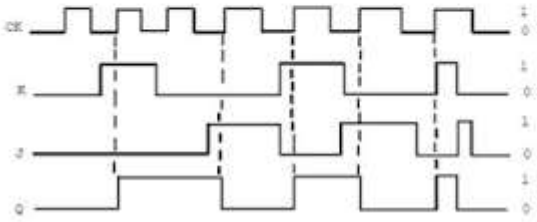
Jenis lain dari flip – flop adalah JK-FF. Input – inputnya J dan K dari JK-FFmengontrol keadaan output FF dengan cara yang sama seperti S dan R dari SFF. Kecuali bahwa pada keadaan J = K = 1 tidak menghasilkan keadaan tak menentu melainkan keadaan yang berlawanan dengan keadaan sebelumnya bila terjadi transisi CK yang sesuai. Keadaan seperti ini dikatakan terjadinya operasi toogle. Seperti halnya D-FF, flip – flop ini juga mempunyai input asinkron seperti set dan reset (clear). JK-FF dapat dibangun dengan gerbang logika.

Kelebihan JK Flip-flop adalah tidak adanya kondisi terlarang atau yang berarti di beri berapapun inputan asalkan terdapat clock maka akan terjadi perubahan pada keluarannya / outputnya. berikut adalah symbol dan tabel kebenaran dari JK Flip-Flop.



Mode Operasi	Masukan			Keluaran	
	CK	J	K	Q	Q'
NC	⎍	0	0	Tidak berubah	
Reset	⎍	0	1	0	1
Set	⎍	1	0	1	0
Togel	⎍	1	1	Keadaan berlawanan	

Timing Diagram



(Sumber : https://binus.ac.id/bandung/2019/12/flip-flop-dan-jenis-jenisnya/)

3. Clocking

Clocking merupakan sinyal dalam sirkuit digital yang menentukan seberapa cepat flip flop berjalan. Sinyal clock terhubung ke semua flipflop dan blok RAM untuk mengaktifkannya sesuai frekuensi clock yang diatur. Semakin cepat clock, semakin cepat juga desain program berjalan. Oleh karena itu, FPGA dengan kecepatan clock yang tinggi akan melakukan fungsi yang diinginkan lebih cepat. FPGA biasanya terdiri dari bebrapa sinyal clock untuk memungkinkan area berbeda dalam FPGA beroperasi dalam kecepatan yang berbeda.

(Sumber : https://hardwarebee.com/ultimate-guide-fpga-clock/)

4. Papan FPGA

FPGA (Field Programmable Gate Arrays) adalah sirkuit terintegrasi yang perangkat kerasnya dapat dikonfigurasi untuk memenuhi kebutuhan spesifik dari pengguna setelah proses manufaktur. Hal ini membolehkan peningkatan fitur dan perbaikan kerusakan langsung di tempat.

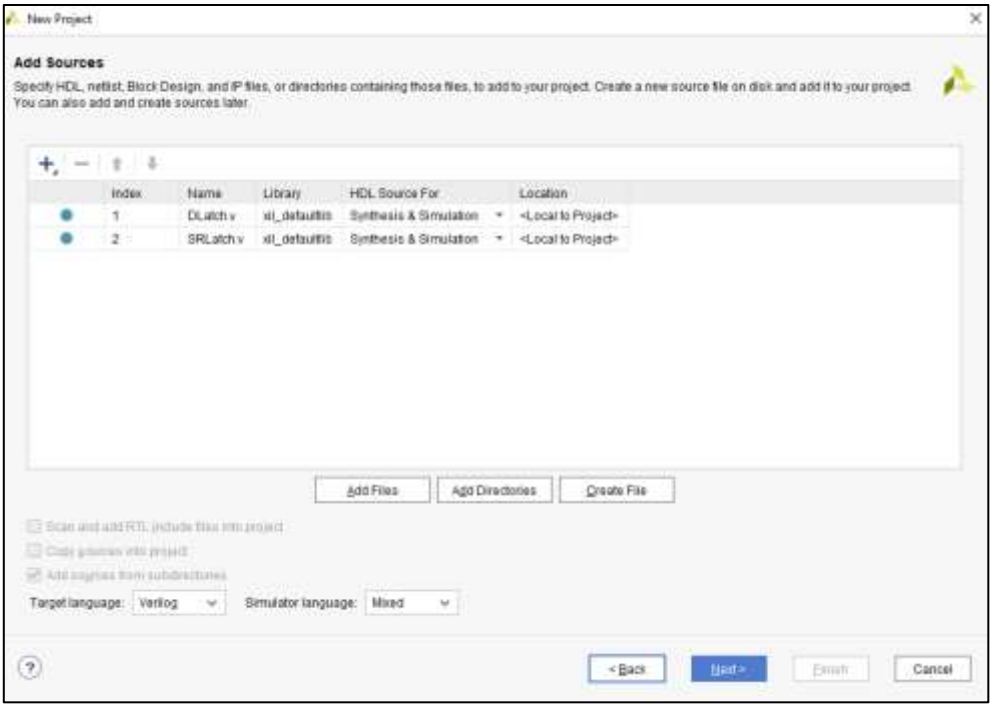
(Sumber: https://www.arm.com/glossary/fpga#:~:text=Field%20Programmable%20Gate%20Arrays%20(FPGAs,requirements%20after%20the%20manufacturing%20process.)

2.4. Langkah kerja

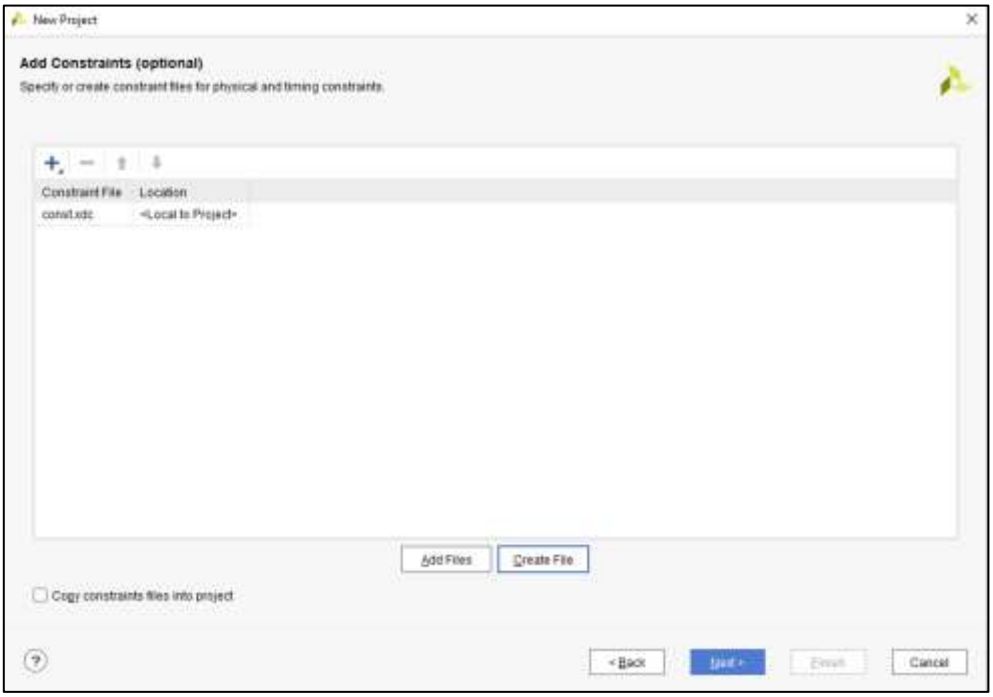
2.4.1. Percobaan 1

1. Buat project baru menggunakan Vivado
2. Masukkan nama project
3. Pilih RTL project sesuai gambar di bawah ini dan tekan next

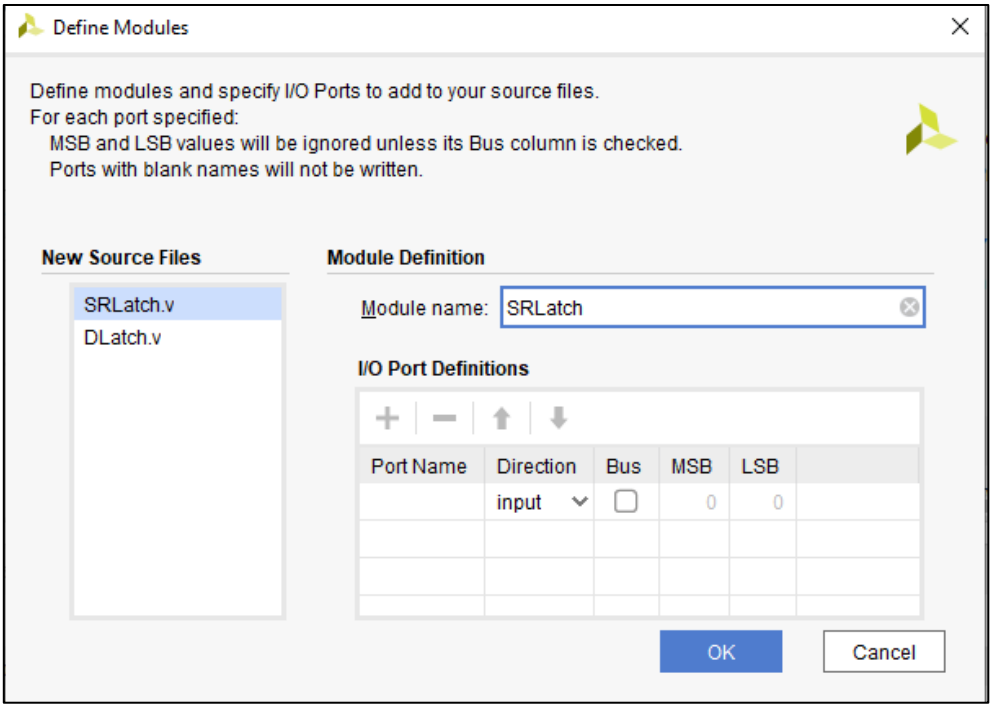
4. Buat 2 file untuk d-latch dan sr-latch dan tekan next



5. Buat 1 file yang nantinya berisi constraint untuk percobaan ini dan tekan next



6. Selesaikan pembuatan project seperti biasa hingga ke tampilan vivado utama
7. Tekan OK dan YES jika muncul tampilan seperti berikut



8. Masukkan kode berikut untuk setiap file yang sudah dibuat

D-LATCH:

```
`timescale 1ns / 1ps
module dlat (output x, not_x, input d);

wire not_d;
not(not_d, d);

nor(not_x, d, x);
nor(x, not_d ,not_x);
endmodule
```

SR-LATCH:

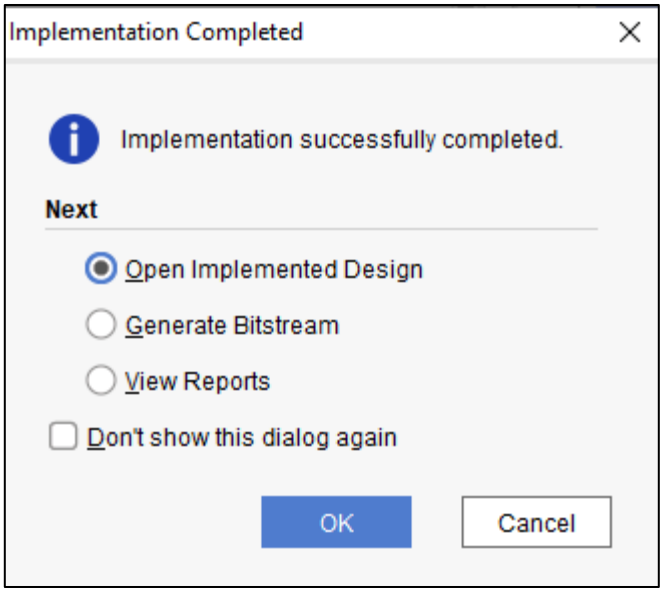
```
`timescale 1ns / 1ps
module srlat (output x, not_x, input s, r);

nor(not_x, s, x);
nor(x, r, not_x);

endmodule
```

Constraint (aktifkan baris kode sesuai kebutuhan):

```
set_property SEVERITY {Warning} [get_drc_checks LUTLP-1];

set_property ALLOW_COMBINATORIAL_LOOPS TRUE;
```

```
##Switches

##D-LATCH

#set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 }
[get_ports { d }]; #IO_L24N_T3_RS0_15 Sch=sw[0]

##SR-LATCH

#set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 }
[get_ports { r }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]

#set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 }
[get_ports { s }]; #IO_L21P_T3_DQS_14 Sch=sw[15]



## LEDs

##D-LATCH

#set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 }
[get_ports { not_x }]; #IO_L18P_T2_A24_15 Sch=led[0]

#set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 }
[get_ports { x }]; #IO_L24P_T3_RS1_15 Sch=led[1]



##SR-LATCH

#set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 }
[get_ports { x }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]

#set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 }
[get_ports { not_x }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]
```

9. Implementasikan ke board dengan cara biasa, ada jug acara lain dengan mengikuti langkah 10-13
10. Setelah memasukkan sumber kode, jalankan implementasi dengan menekan “Run Implementation” pada jendela “Flow Navigator” (di sebelah kiri, di atas generate bitstream)
11. Jika sudah selesai, buka desain implementasinya



12. Buka jendela “Tcl Console” di jendela bagian bawah (tab paling kiri), dan jalankan kode di bawah ini

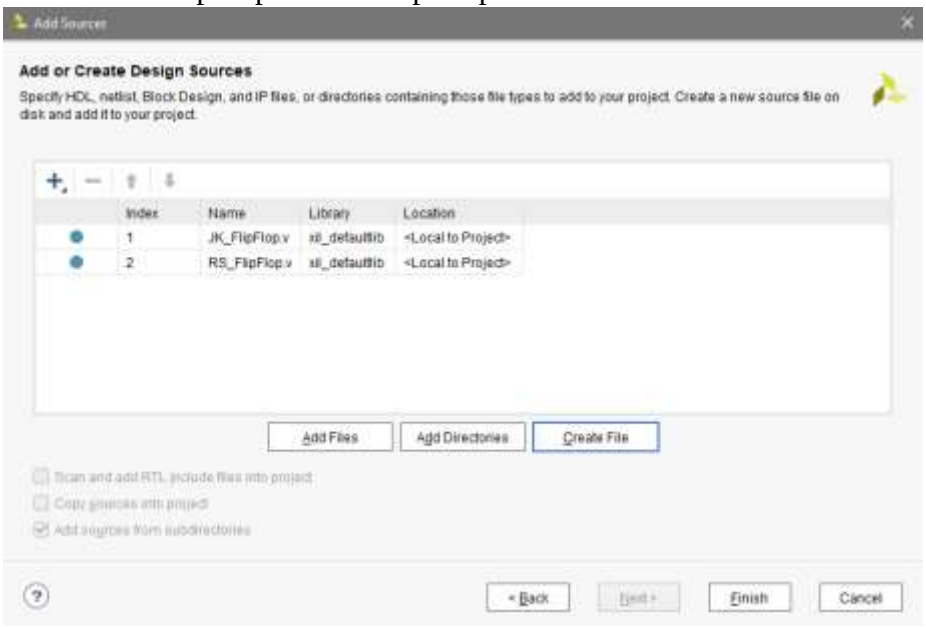
```
set_property SEVERITY {Warning} [get_drc_checks LUTLP-1]
set_property IS_ENABLED 0 [get_drc_checks {CSCL-1}]
write_bitstream <MASUKKAN_PATH
_PROJECT_KALIAN_DISINI>.runs/impl_1/design1.bit

Contoh:
write_bitstream
A:/XilinxProject/PSDL_Modul4_Test/PSDL_Modul4_Test.runs/impl_1/design1.
bit
```

13. Jika bitstream sudah selesai di-generate, silahkan implementasi ke papan dengan program device

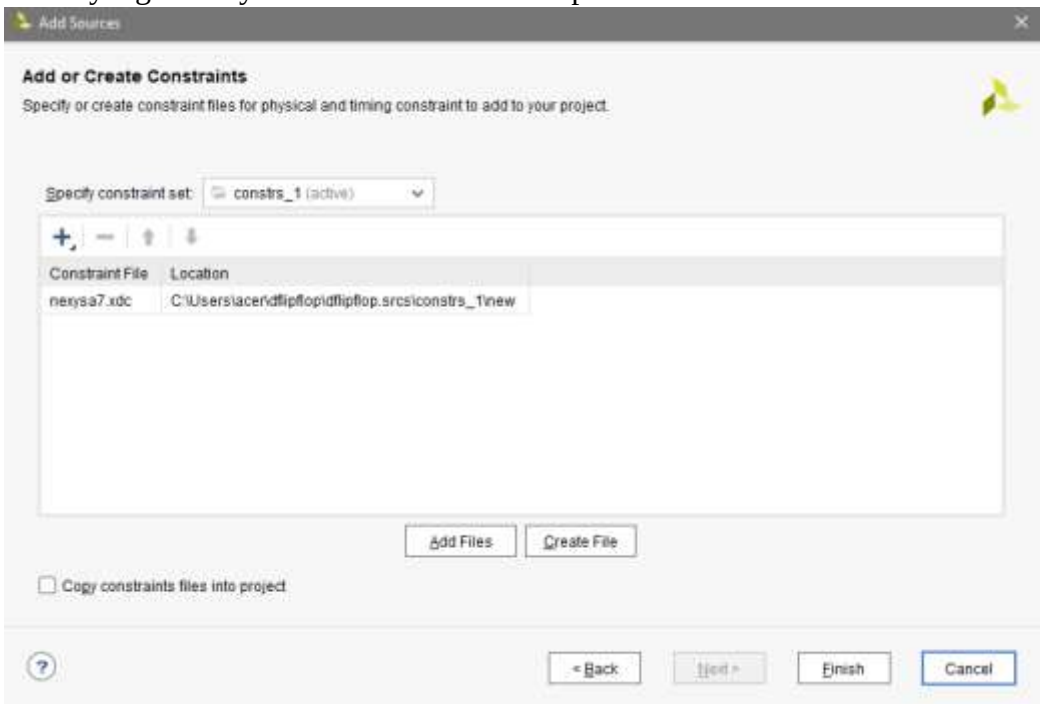
2.4.2. Percobaan 2

1. Buat project baru menggunakan Vivado
2. Masukkan nama project
3. Pilih RTL project sesuai gambar di bawah ini dan tekan next

4. Buat 2 file untuk SR-FlipFlop dan JK Flip Flop dan tekan next



5. Buat 1 file yang nantinya berisi constraint untuk percobaan ini dan tekan next.



6. Selesaikan pembuatan project seperti biasa hingga ke tampilan vivado utama
7. Masukkan Source code berikut untuk setiap file design yang telah dibuat.

RS_FlipFlop

```
module srff_behave(s,r,clk, q, qbar);

input s,r,clk;
output reg q, qbar;
```

```
      always@(posedge clk)
      begin

      if(s == 1)
      begin
      q = 1;
      qbar = 0;
      end
      else if(r == 1)
      begin
      q = 0;
      qbar =1;
      end
      else if(s == 0 & r == 0)
      begin
      q <= q;
      qbar <= qbar;
      end
      end
endmodule
```

JK_FlipFlop

```
      module jkff_behave(clk,j,k,q,qbar);

      input clk,j,k;
      output reg q,qbar;
      reg [26:0] one_second_counter;
      wire one_second_enable;

      always @(posedge clk)
       begin
       if(one_second_counter>=99999999)
          one_second_counter <= 0;
       else
          one_second_counter <= one_second_counter + 1;
       end
       assign one_second_enable = (one_second_counter==99999999)?1:0;

      always@(posedge one_second_enable)
      begin
      if(j == 0 & k == 0)
         begin
          q = q;
          qbar = qbar;
         end
          else if(j == 1 & k == 1)
         begin
          q = ~q;
          qbar = ~qbar;
         end
         else if(k == 1)
         begin
          q = 0;
          qbar = 1;
         end
        else if(j == 1)
          begin
```

```
          q = 1;
          qbar = 0;
         end

      end

endmodule
```

8. Masukkan constrain ke file yang telah dibuat sebelumnya.

```
## This file is a general .xdc for the Nexys A7-100T
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports)
according to the top level signal names in the project

## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 }
[get_ports { clk }]; IO_L12P_T1_MRCC_35 Sch=clk
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5}
[get_ports {clk}];



##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 }
[get_ports { j }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 }
[get_ports { k }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 }
[get_ports { q }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 }
[get_ports { qbar }]; #IO_L24P_T3_RS1_15 Sch=led[1]
```

9. Implementasikan SR_FlipFlop maupun JK_FlipFlop ke board seperti biasanya.