BAB III
CODING REUSE

3.1 Tujuan

1. Praktikan memahami konsep kode Verilog.
2. Praktikan dapat membuat kode Verilog dan mengimplementasikan pada aplikasi Vivado.
3. Praktikan dapat memahami konsep dalam coding reuse dan menerapkan pada vivado.

3.2 Alat dan Bahan

1. Laptop
2. Papan Nexys A7

3.1 Dasar Teori

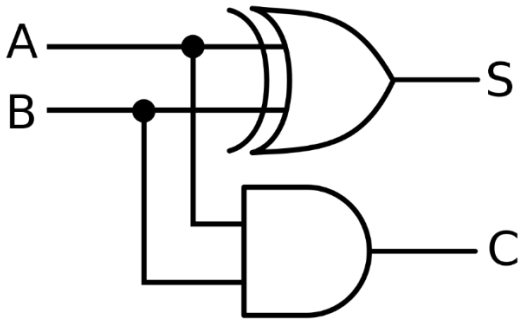
1. Coding Reuse

Code reuse adalah teknik pemrograman untuk mengurangi waktu dan sumber daya dalam mengembangkan perangkat lunak. Saat mengembangkan kode untuk digunakan kembali, kode tersebut dapat digunakan untuk berbagai tujuan. Teknik ini melibatkan modularisasi, yang memungkinkan banyak individu mengembangkan kode untuk berbagai komponen sistem secara mandiri dan paralel. Teknik ini juga menyederhanakan distribusi perangkat lunak.

(sumber : https://www.mathworks.com/help/rtw/ug/what-is-code-reuse-58d9ced3ba27.html)

2. Half Adder

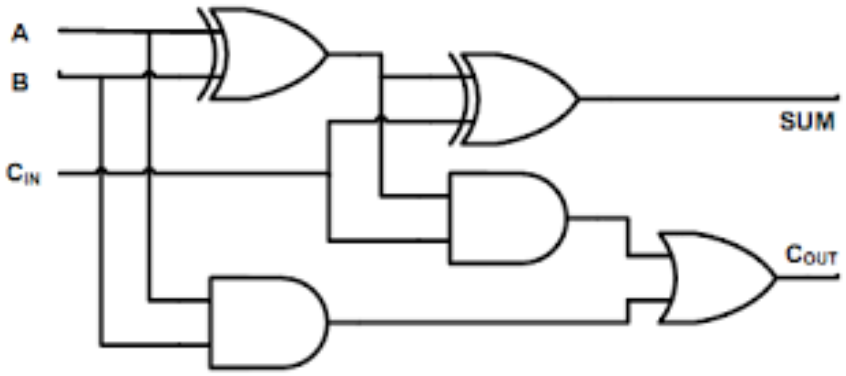
Half Adder adalah rangkaian elektronik yang bekerja melakukan perhitungan penjumlahan dari dua buah bilangan binary, yang masing-masing terdiri dari satu bit. Rangkaian ini memiliki dua input dan dua buah output, salah satu outputnya dipakai sebagai tempat nilai pindahan dan yang lain sebagai hasil dari penjumlahan.



(sumber : https://smkmuh2klaten.sch.id/blog/pengertian-half-adder-dan-full-adder/)

3. Full Adder

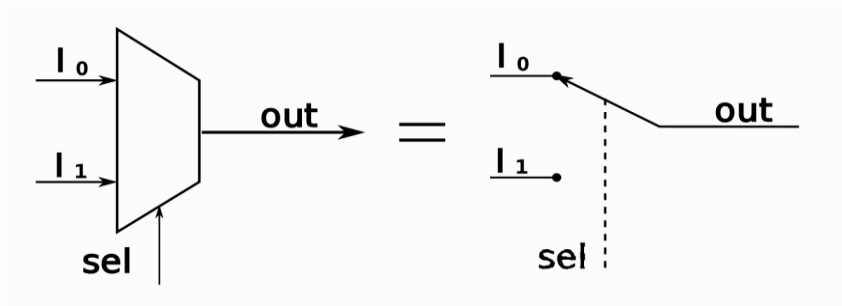
Full adder adalah rangkaian elektronik yang bekerja melakukan perhitungan penjumlahan sepenuhnya dari dua buah bilangan binary, yang masing-masing terdiri dari satu bit. Rangkaian ini memiliki tiga input dan dua buah output, salah satu input merupakan nilai dari pindahan penjumlahan, kemudian sama seperti pada half adder salah satu outputnya dipakai sebagai tempat nilai pindahan dan yang lain sebagai hasil dari penjumlahan



.

(sumber : https://smkmuh2klaten.sch.id/blog/pengertian-half-adder-dan-full-adder/)

4. Multiplexer



Multiplexer adalah perangkat yang dapat membolehkan satu atau lebih sinyal analog atau digital untuk berjalan pada satu sambungan transmisi komunikasi. Tujuan dari multiplexer adalah untuk menggabungkan dan mengirimkan sinyal pada media yang tergabung untuk mengoptimasi efisiensi dan mengurangi biaya komunikasi

(sumber: https://www.techopedia.com/definition/24124/multiplexer-mux)

5. Papan FPGA

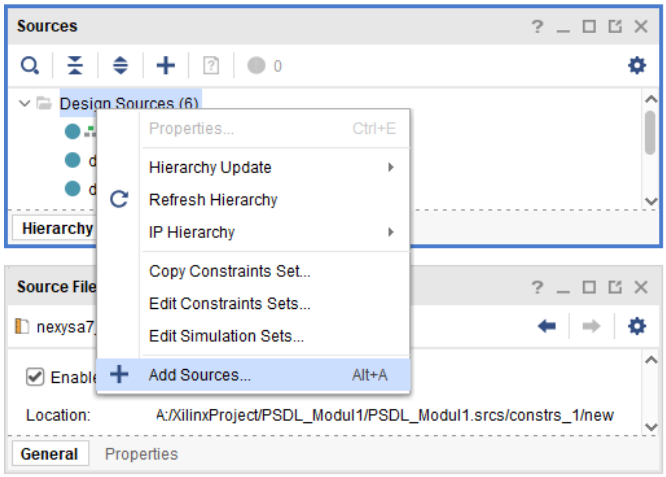
FPGA (Field Programmable Gate Arrays) adalah sirkuit terintegrasi yang perangkat kerasnya dapat dikonfigurasi untuk memenuhi kebutuhan spesifik dari pengguna setelah proses manufaktur. Hal ini membolehkan peningkatan fitur dan perbaikan kerusakan langsung di tempat.

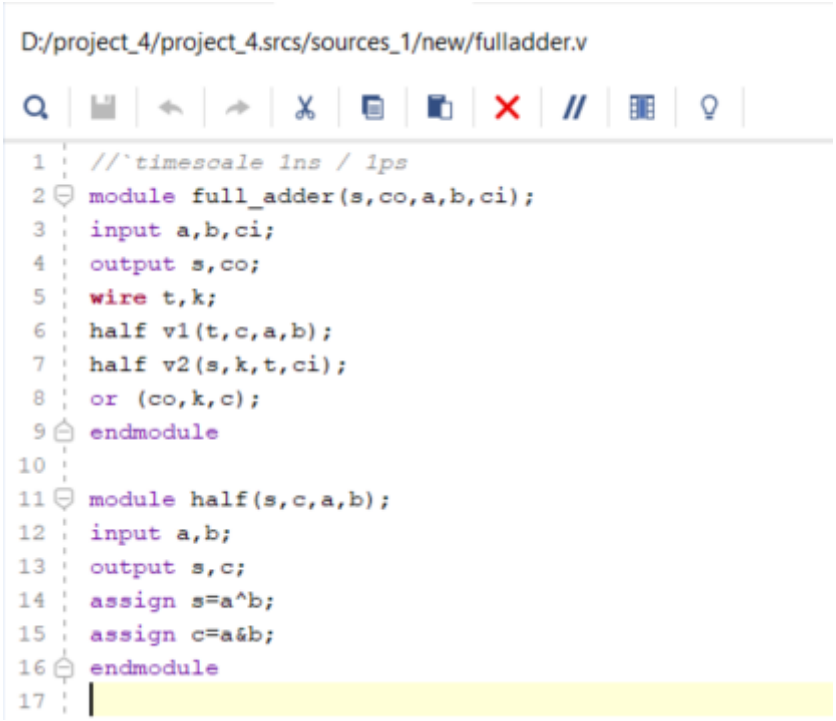
(Sumber: https://www.arm.com/glossary/fpga#:~:text=Field%20Programmable%20Gate%20Arrays%20(FPGAs,requirements%20after%20the%20manufacturing%20process.)

3.2 Langkah Kerja

3.4.1 Percobaan Pertama FULL ADDER using HALF ADDER

1. Buat file design source dengan source code di bawah ini

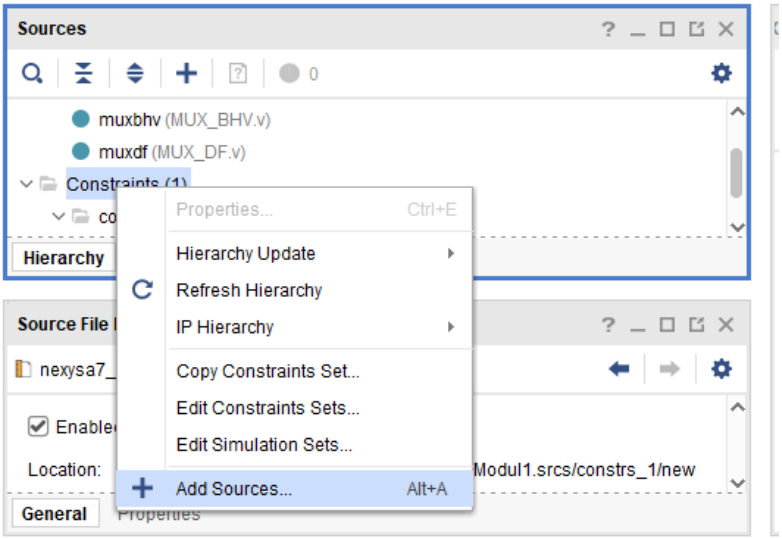





```
//`timescale 1ns / 1ps
module full_adder(s,co,a,b,ci);
input a,b,ci;
output s,co;
wire t,k;
half v1(t,c,a,b);
```

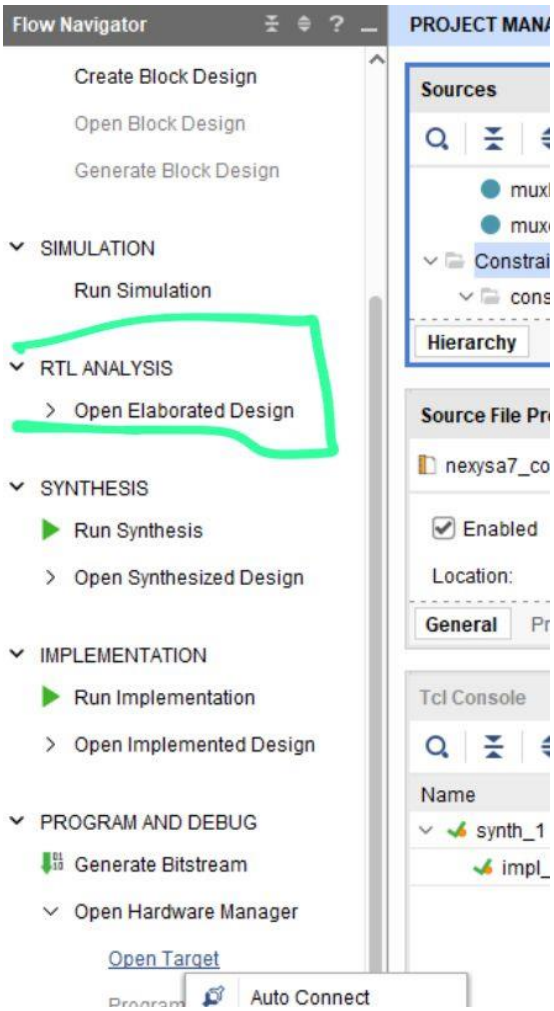
```
half v2(s,k,t,ci);
or (co,k,c);
endmodule

module half(s,c,a,b);
input a,b;
output s,c;
assign s=a^b;
assign c=a&b;
endmodule
```

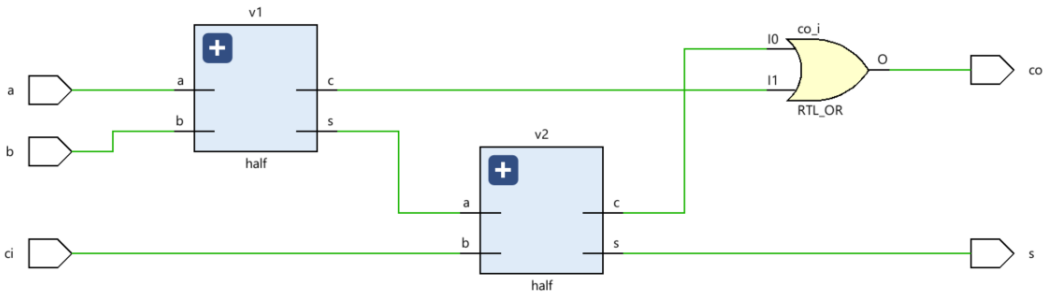
2. Membuat constraint dan memasukkan source code di bawah ini



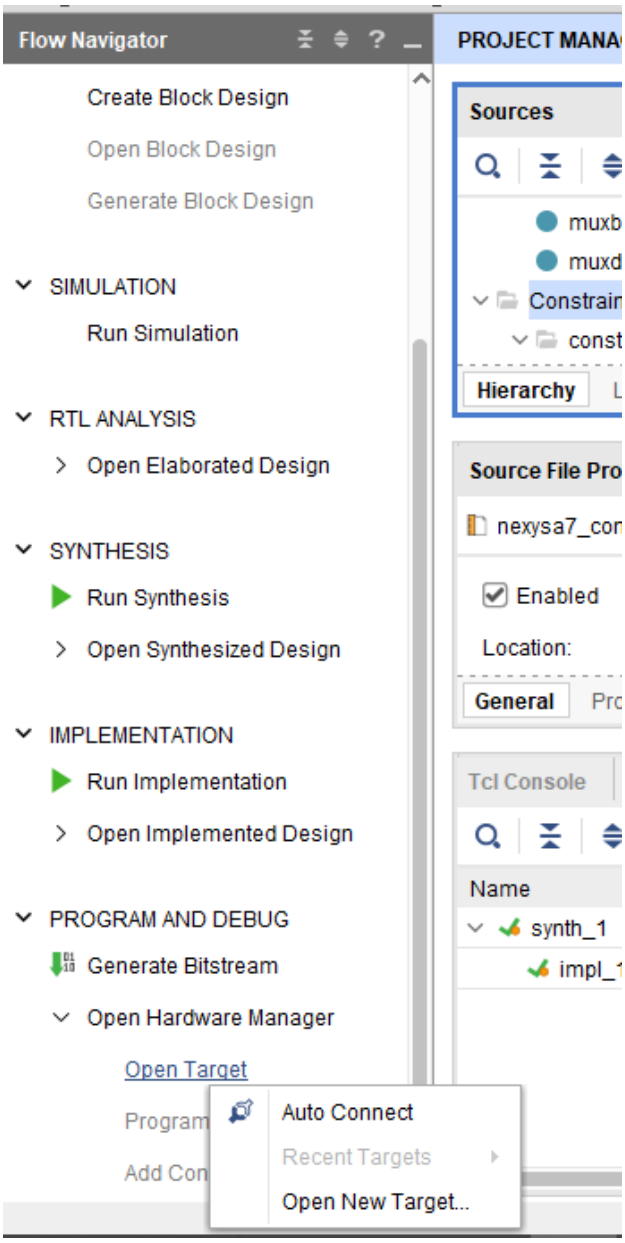

```
(Input a,b,ci)
set_property  -dict  {  PACKAGE_PIN  J15      IOSTANDARD  LVCMOS33  }
[get_ports { a }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property  -dict  {  PACKAGE_PIN  L16      IOSTANDARD  LVCMOS33  }
[get_ports { b }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property  -dict  {  PACKAGE_PIN  M13      IOSTANDARD  LVCMOS33  }
[get_ports { ci }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]

(Output s,co,c)
set_property  -dict  {  PACKAGE_PIN  H17      IOSTANDARD  LVCMOS33  }
[get_ports { s }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property  -dict  {  PACKAGE_PIN  K15      IOSTANDARD  LVCMOS33  }
[get_ports { co }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property  -dict  {  PACKAGE_PIN  J13      IOSTANDARD  LVCMOS33  }
[get_ports { c }]; #IO_L17N_T2_A25_15 Sch=led[2]
```

3. Untuk melihat rangkaian, klik open elaborated design kemudian pilih schematic

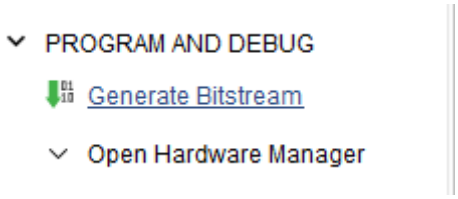




4. Sambungkan papan dengan laptop dan sambungkan ke vivado



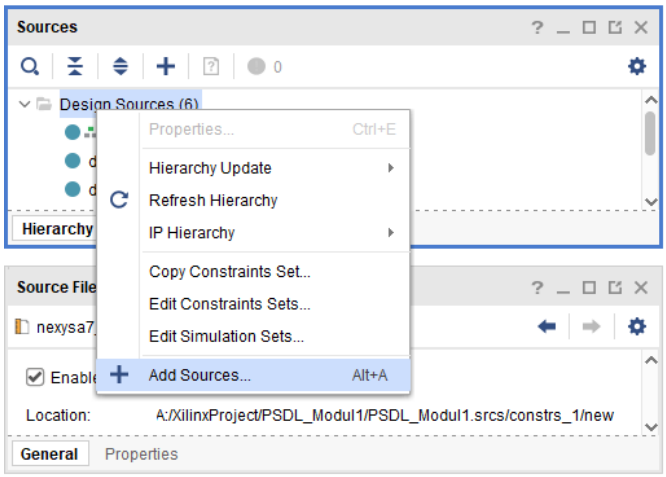
5. Generate Bitstream



6. Program device
7. Bolak-balikkan switch yang diprogram dan amati perbedaannya

3.4.2 Percobaan Kedua MUX 8-1 using MUX 2-1

1. Buat file design source dengan source code di bawah ini

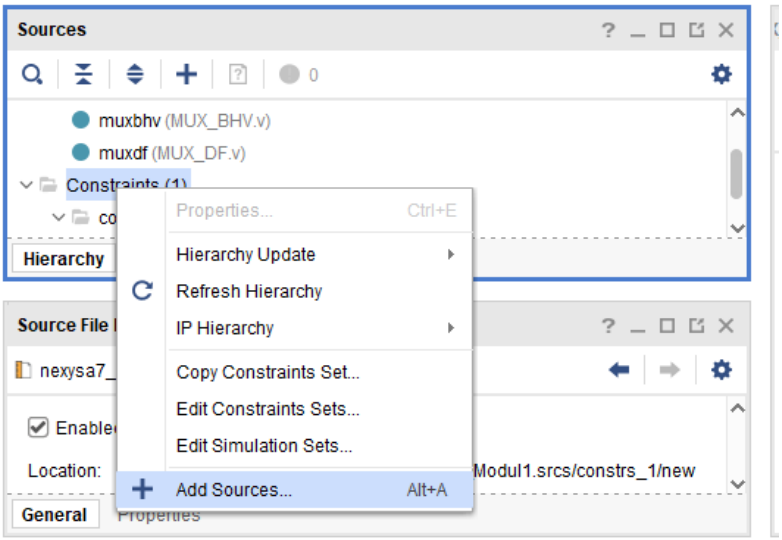



```
1  `timescale 1ns / 1ps
2
3  module mux_case(out,in,s);
4  output reg out;
5  input [1:0]in;
6  input s;
7  always @ (*)
8  casex(s)
9  1'b0 : out = in[0];
10 1'b1 : out = in[1];
11 default : out = 1'bx;
12 endcase
13 endmodule
14
15 module mux_8_1 (o,i,s);
16 output o;
17 input [7:0]i;
18 input [2:0]s;
19 wire [6:1]k;
20 mux_case vas1(k[1], i[1:0], s[0]);
21 mux_case vas2(k[2], i[3:2], s[0]);
22 mux_case vas3(k[3], i[5:4], s[0]);
23 mux_case vas4(k[4], i[7:6], s[0]);
24 mux_case vas5(k[5], k[2:1], s[1]);
25 mux_case vas6(k[6], k[4:3], s[1]);
26 mux_case vas7(o,k[6:5], s[2]);
27 endmodule
```

```
//`timescale 1ns / 1ps

module mux_case(out,in,s);
output reg out;
input [1:0]in;
input s;
always @ (*)
casex(s)
1'b0 : out = in[0];
1'b1 : out = in[1];
default : out = 1'bx;
endcase
endmodule
```

```
module mux_8_1 (o,i,s);
output o;
input [7:0]i;
input [2:0]s;
wire [6:1]k;
mux_case vas1(k[1], i[1:0], s[0]);
mux_case vas2(k[2], i[3:2], s[0]);
mux_case vas3(k[3], i[5:4], s[0]);
mux_case vas4(k[4], i[7:6], s[0]);
mux_case vas5(k[5], k[2:1], s[1]);
mux_case vas6(k[6], k[4:3], s[1]);
mux_case vas7(o,k[6:5], s[2]);
endmodule
```

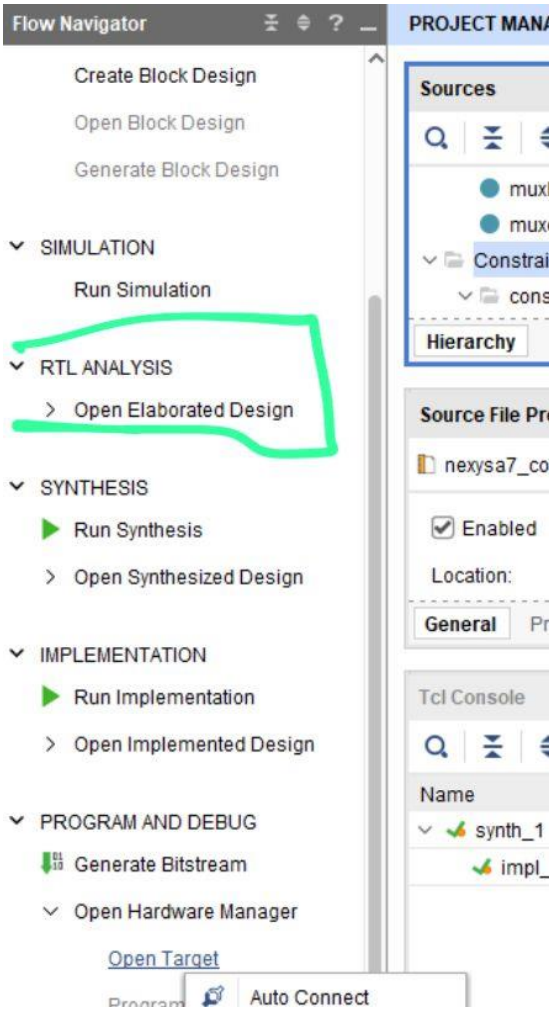
2. Membuat constraint dan memasukkan source code di bawah ini

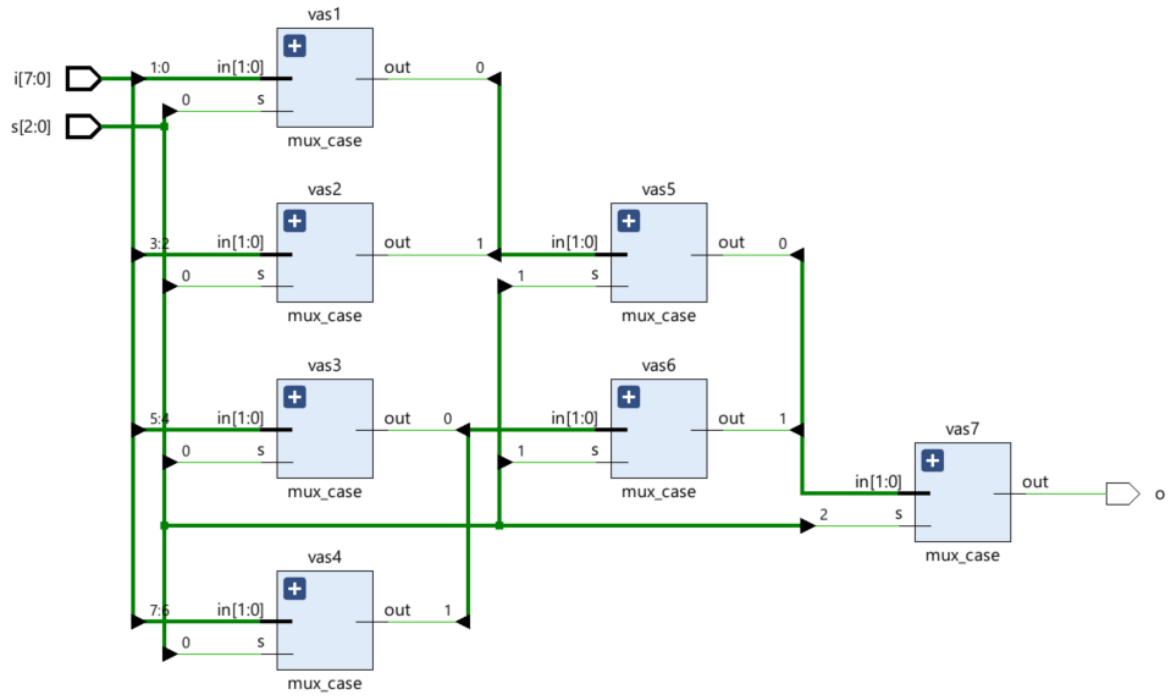



```
(Inputan i0-i7)
set_property  -dict  {  PACKAGE_PIN  J15      IOSTANDARD  LVCMOS33  }
[get_ports { i[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property  -dict  {  PACKAGE_PIN  L16      IOSTANDARD  LVCMOS33  }
[get_ports { i[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property  -dict  {  PACKAGE_PIN  M13      IOSTANDARD  LVCMOS33  }
[get_ports { i[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property  -dict  {  PACKAGE_PIN  R15      IOSTANDARD  LVCMOS33  }
[get_ports { i[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property  -dict  {  PACKAGE_PIN  R17      IOSTANDARD  LVCMOS33  }
[get_ports { i[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property  -dict  {  PACKAGE_PIN  T18      IOSTANDARD  LVCMOS33  }
[get_ports { i[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property  -dict  {  PACKAGE_PIN  U18      IOSTANDARD  LVCMOS33  }
[get_ports { i[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property  -dict  {  PACKAGE_PIN  R13      IOSTANDARD  LVCMOS33  }
[get_ports { i[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]

(inputan Selector s0-s2)
set_property  -dict  {  PACKAGE_PIN  U12      IOSTANDARD  LVCMOS33  }
[get_ports { s[2] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
set_property  -dict  {  PACKAGE_PIN  U11      IOSTANDARD  LVCMOS33  }
[get_ports { s[1] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
```

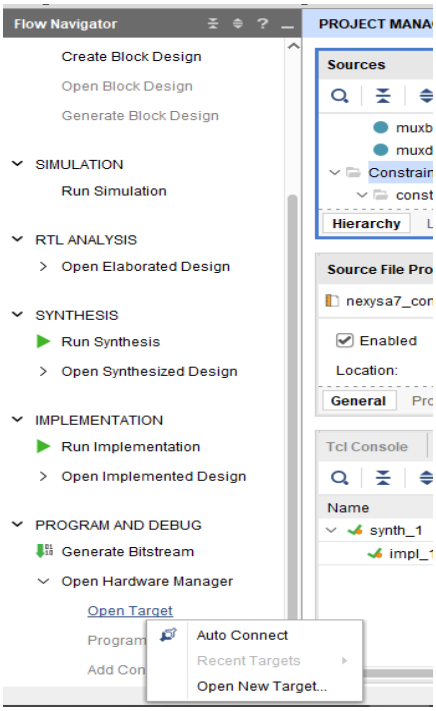
```
set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 }
[get_ports { s[0] }]; #IO_L21P_T3_DQS_14 Sch=sw[15

(output LED)
set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 }
[get_ports { o }]; #IO_L18P_T2_A24_15 Sch=led[0]set_property -dict {
PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { s }];
#IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 }
[get_ports { co }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33 }
[get_ports { c }]; #IO_L17N_T2_A25_15 Sch=led[2]
```

3. Untuk melihat rangkaian, klik open elaborated design kemudian pilih schematic

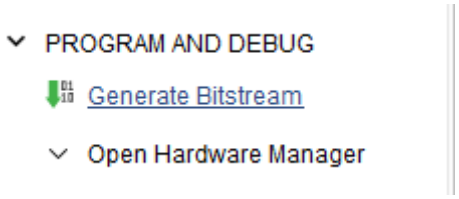




4. Sambungkan papan dengan laptop dan sambungkan ke vivado



5. Generate Bitstream



6. Program device

7. Bolak-balikkan switch yang diprogram dan amati perbedaannya