CND 111: Introduction to Digital Design

Assignment #: 7

Section #: 16

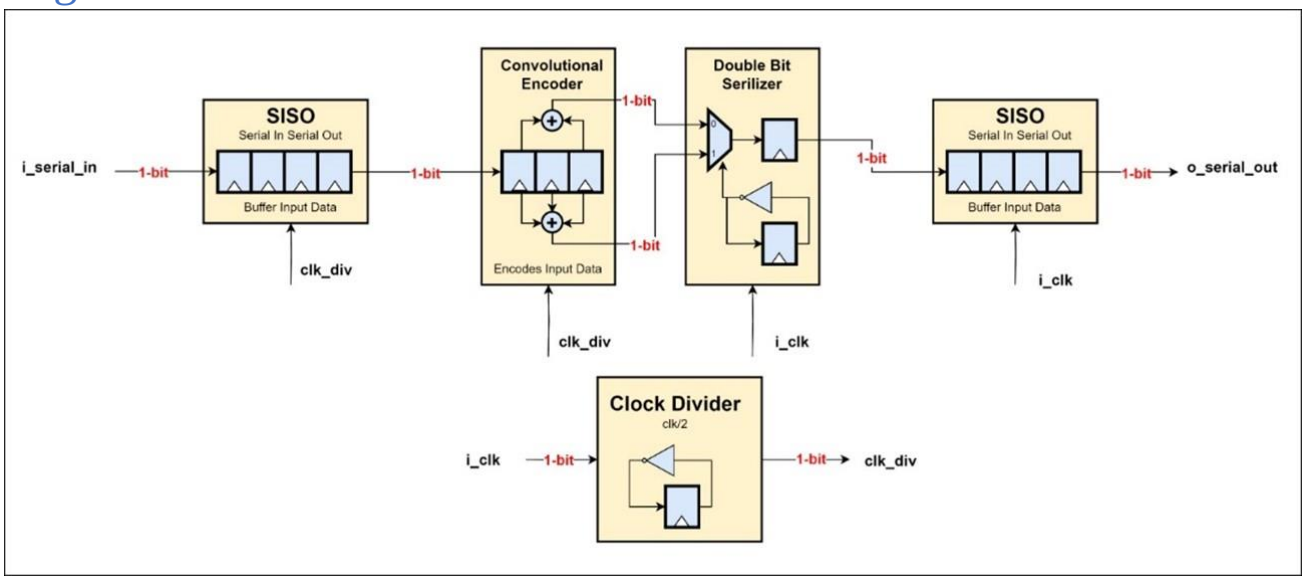
Submitted by:

Student Name	ID
Aya Ahmed Abdelrahman	23010284
Karim Mahmoud Kamal	V23010174
Tarek salah abdalhafeez	V23010337

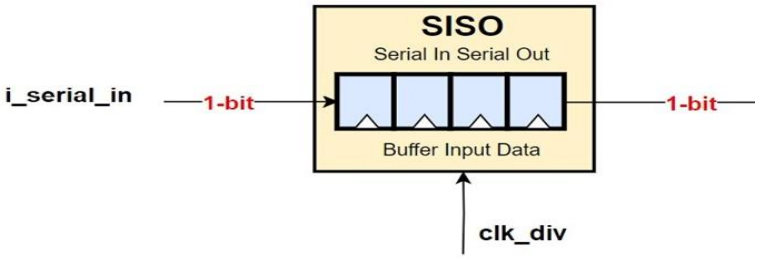
Submitted to TA: Mohamed Elshafey

Date: 15/11/2023

➢ Assignment 7



SISO (serial input serial output) design




```
1   module SISO(input IN,
2   input CLK,
3   input RST,
4   output reg OUT);
5   reg [3:0] SHIFT_REG;
6   always@(posedge CLK or negedge RST)
7    begin
8    if(!RST)
9     begin
10    SHIFT_REG<='b0;
11    OUT<=1'b0;
12    end
13    else
14     begin
15     SHIFT_REG[0]<=IN;
16     SHIFT_REG[1]<=SHIFT_REG[0];
17     SHIFT_REG[2]<=SHIFT_REG[1];
18     SHIFT_REG[3]<=SHIFT_REG[2];
19     OUT<=SHIFT_REG[3];
20     end
21   end
22
23
24
```

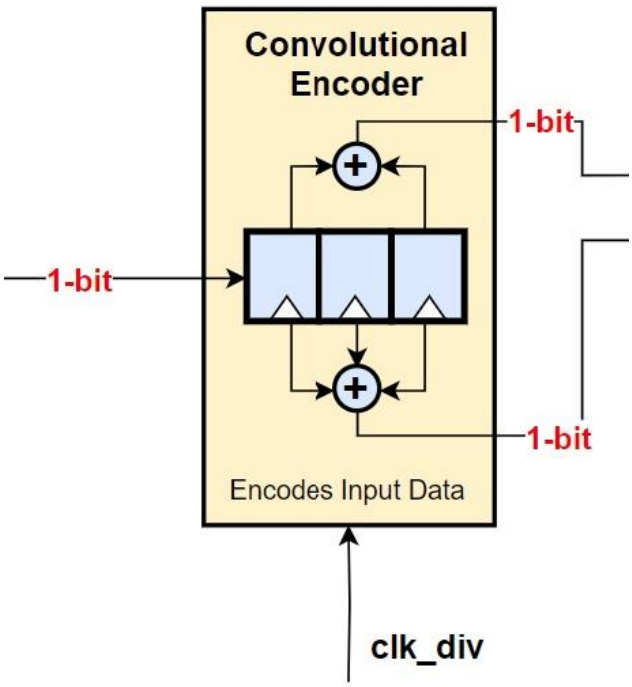
➢ Comments

The module has four ports:

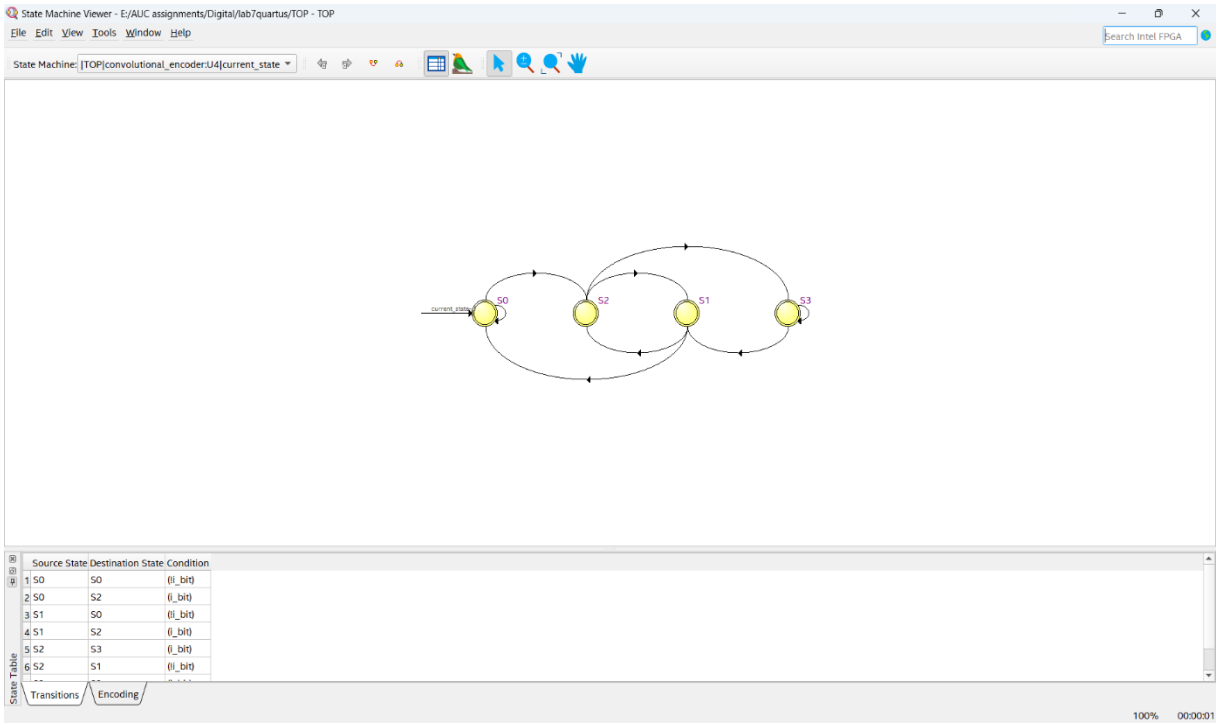
- IN: This is the input signal that represents the data to be shifted into the register. It is connected to the least significant bit (SHIFT_REG [0]) of the shift register.
- CLK: This is the clock input signal that controls the timing of the circuit. The shift register updates its values on the positive edge of the clock.
- RST: This is the reset input signal. When it is low (0 or negedge), the shift register is reset, and all its bits are set to 0. When it is high (1 or posedge), the shift register operates normally.
- OUT: This is the registered output signal. It represents the most significant bit (SHIFT_REG [3]) of the shift register and holds the value of the data that has been shifted out.

Inside the module, it's important to note that the shift register updates its values only on the positive edge of the clock (posedge CLK) or the negative edge of the reset signal (negedge RST). This ensures that the register operates synchronously with the clock and maintains stability during the signal transitions. If RST is high (posedge), indicating normal operation, the code block inside the else statement executes. Here, the shift register is updated by shifting the existing bits to the right (from SHIFT_REG [0] to SHIFT_REG [3]), and the input data IN is loaded into the least significant bit (SHIFT_REG[0]). This shift operation effectively moves the data through the register.

➢ Convolutional encoder block design



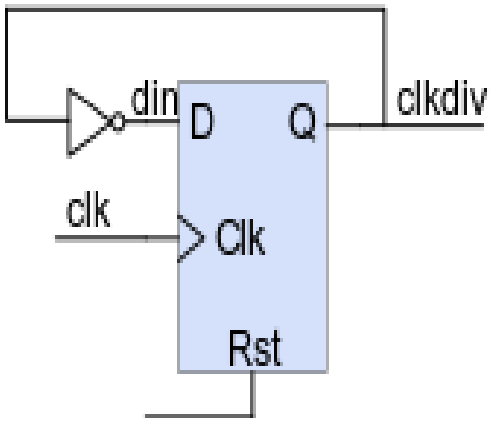
```
Ln#
 1  module convolutional_encoder(input i_bit,
 2  input CLK,
 3  input RST,
 4  output reg out1,out2);
 5
 6  localparam S0=2'b00,
 7                          S1=2'b01,
 8                          S2=2'b10,
 9                          S3=2'b11;
10  reg [1:0] current_state,next_state;
11  always @(posedge CLK or negedge RST)
12   begin
13   if(!RST)
14    begin
15                  current_state<=S0;
16    end
17    else
18     begin
19          current_state<=next_state;
20          end
21   end
22
23  always @(*)
24  begin
25  case(current_state)
26   S0:begin
27                  if(i_bit)
28                   begin
29                          next_state=S2;
30                          out1=1'b1;
31                          out2=1'b1;
32                   end
33                   else
34                   begin
35                    next_state=S0;
36                          out1=1'b0;
37                          out2=1'b0;
38                   end
39       end
40  S1:begin
41                  if(i_bit)
42                   begin
43                          next_state=S2;
44                          out1=1'b0;
45                          out2=1'b0;
46                     end
47                     else
48                     begin
49                      next_state=S0;
50                      out1=1'b1;
51                      out2=1'b1;
52                     end
53       end
54  S2:begin
55                    if(i_bit)
56                     begin
57                            next_state=S3;
58                            out1=1'b0;
59                            out2=1'b1;
60                     end
61                     else
62                     begin
63                      next_state=S1;
64                      out1=1'b1;
65                      out2=1'b0;
66                     end
67       end
68  S3:begin
69                    if(i_bit)
70                     begin
71                            next_state=S3;
72                            out1=1'b1;
73                            out2=1'b0;
74                     end
75                     else
76                     begin
77                      next_state=S1;
78                      out1=1'b0;
79                      out2=1'b1;
80                     end
81       end
82  endcase
83  end
84
85
86  endmodule
```

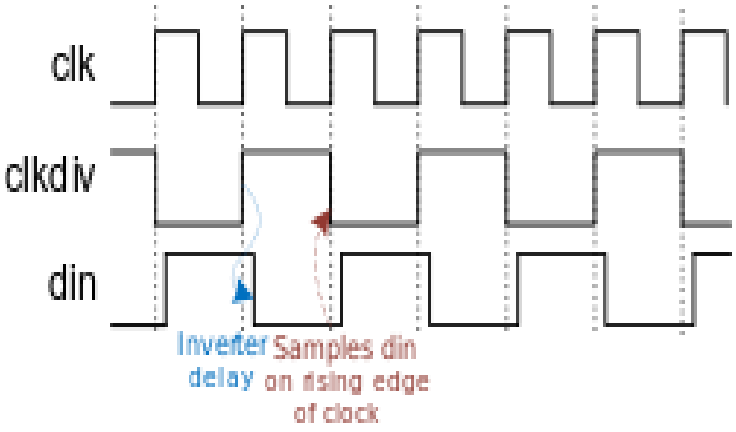


➢ Comments

Convolutional encoding is a technique used in digital communication systems for error detection and correction. It adds redundancy to the transmitted data, which helps in detecting and correcting errors at the receiver side. The convolutional encoder works by encoding the input data bit (i_bit) based on the current state of the encoder. The encoder has four states: S0, S1, S2, and S3. These states are represented by 2-bit values (reg [1:0] current_state, next_state;). By following these encoding rules, the convolutional encoder produces two encoded output bits (out1 and out2) based on the input bit (i_bit) and the current state of the encoder. Convolutional codes, like the ones implemented by this encoder, are widely used in various communication systems, including wireless communication, satellite communication, and error-correction techniques such as Viterbi decoding. The encoded data can be transmitted over a noisy channel, and at the receiver side, decoding techniques are used to recover the original data by reversing the encoding process.

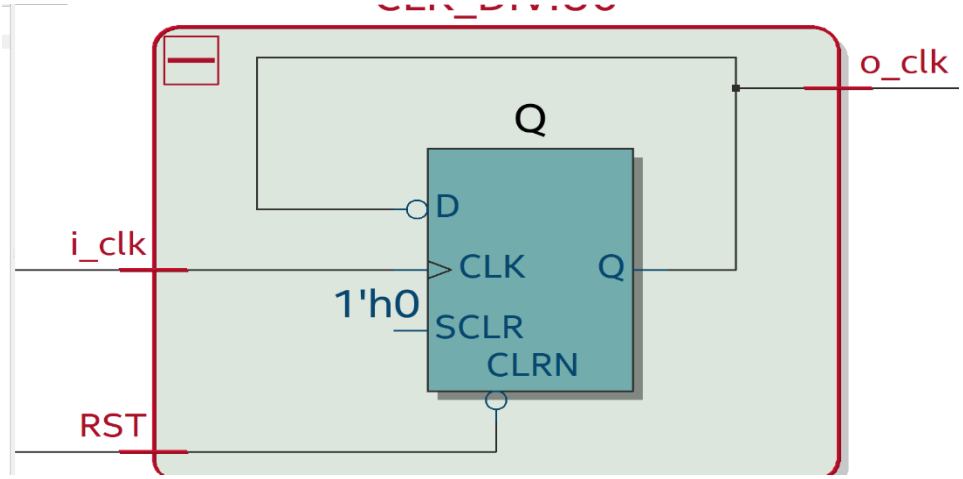
➢ Clock divider





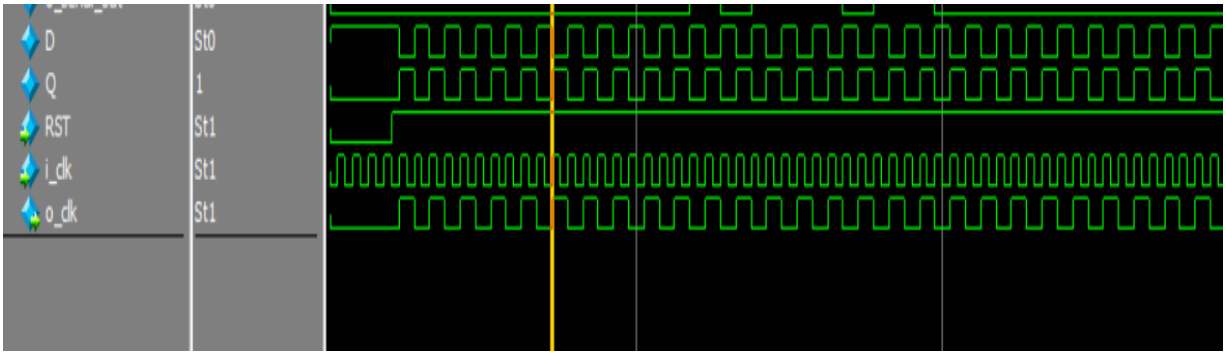
E:/AUC assignments/Digital/conv_encoder/CLK_DIV.v - Default

```
Ln#
 1  module CLK_DIV(input i_clk,
 2  input RST,
 3  output o_clk);
 4  reg Q;
 5  wire D;
 6  always @(posedge i_clk or negedge RST)
 7   if(!RST)
 8    begin
 9          Q<=1'b0;
10    end
11    else
12     begin
13          Q<=D;
14     end
15  assign D = ~Q;
16  assign o_clk = Q;
17  endmodule
```

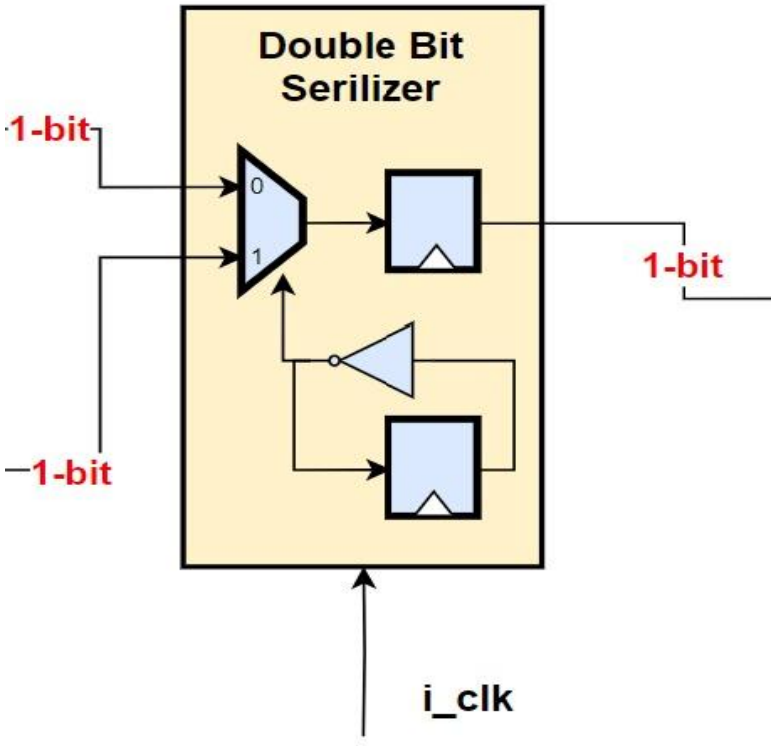


➢ Comments

In this module, it takes an input clock signal (i_clk) and produces a divided output clock signal (o_clk). The clock division is achieved by toggling the state of the Q register on each rising edge of the input clock (divided clock frequency by 2). The division factor is determined by the relationship between Q and its complement D. When Q is high, D is low, and vice versa. The output clock signal o_clk follows the state of Q, resulting in a divided frequency compared to the input clock. The reset signal (RST) is used to initialize the clock divider to a known state.



➢ Double bit serializer



E:/AUC assignments/Digital/conv_encoder/DOUBLE_BIT_serializer.v (/tb/U0/U3) - Default

```
Ln#
 1  module DOUBLE_BIT_serializer(input [1:0] IN,
 2  input CLK,
 3  input RST,
 4  output reg OUT);
 5  reg SER_FLAG;
 6  always@(posedge CLK or negedge RST)
 7   begin
 8   if(!RST)
 9    begin
10    OUT<=1'b0;
11    SER_FLAG<=1'b0;
12    end
13    else
14     begin
15           if(!SER_FLAG)
16            begin
17                  OUT<=IN[0];
18                  SER_FLAG<=1'b1;
19            end
20           else
21            begin
22            OUT<=IN[1];
23                  SER_FLAG<=1'b0;
24            end
25     end
26  end
```

➢ comments

In this module, it represents a double-bit serializer that takes a 2-bit input (IN) and produces a serialized output bit (OUT). If the RST signal is low (negedge), indicating a reset condition, the code block inside the if statement executes. In this case, OUT is set to logic 0, representing the initial state of the serializer. Additionally, SER_FLAG is set to logic 0 to indicate that the serializer is not in the middle of serializing bits.If the RST signal is high (posedge), indicating normal operation, the code block inside the else statement executes. The value of OUT and SER_FLAG is updated based on the current state of the serializer.When SER_FLAG is low (indicating that the serializer is not in the middle of serializing bits), the code block inside the nested if statement executes. In this case, OUT is assigned the value of the first bit (IN [0]) from the input IN, and SER_FLAG is set to logic 1 to indicate that the next bit should be serialized. When SER_FLAG is high, the code block inside the nested else statement executes. In this case, OUT is assigned the value of the second bit (IN [1]) from the input IN, and SER_FLAG is set to logic 0 to indicate that the serialization of bits is complete.

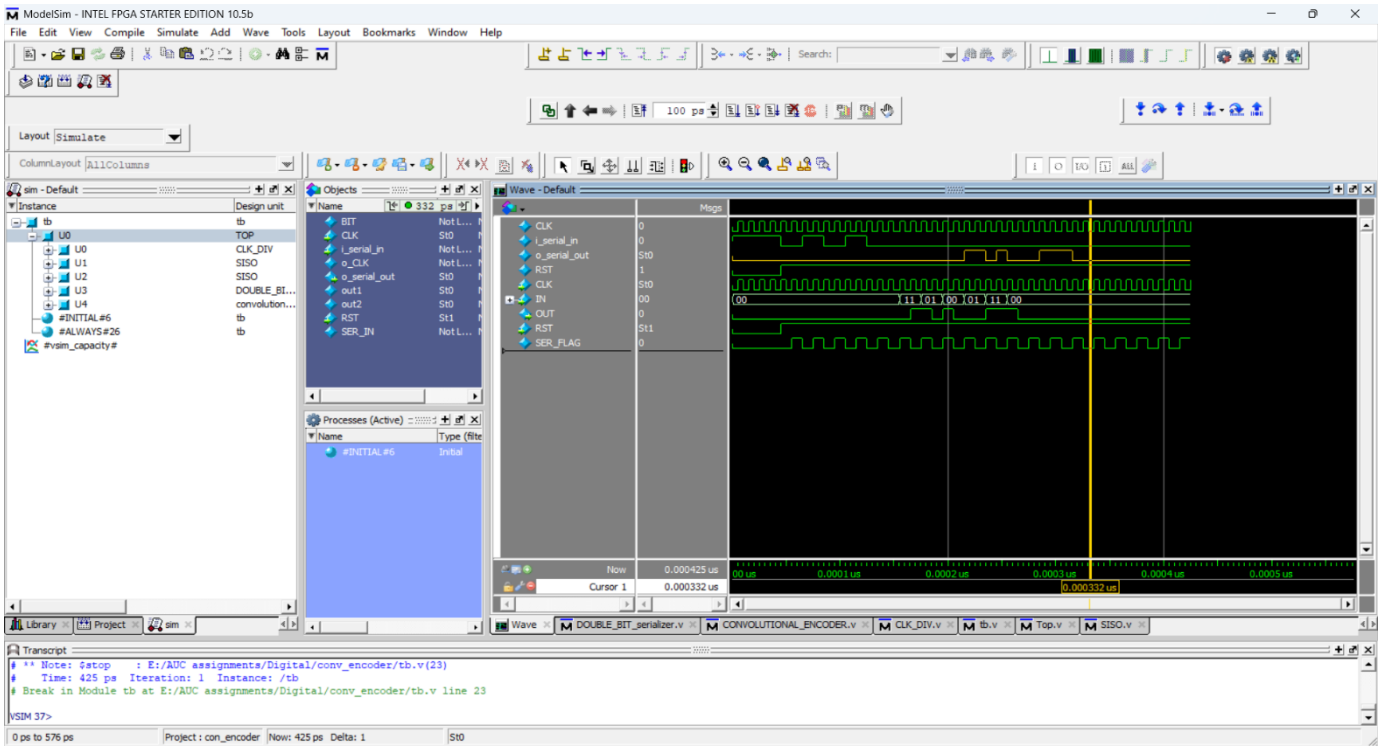
➢ TOP module and test bench

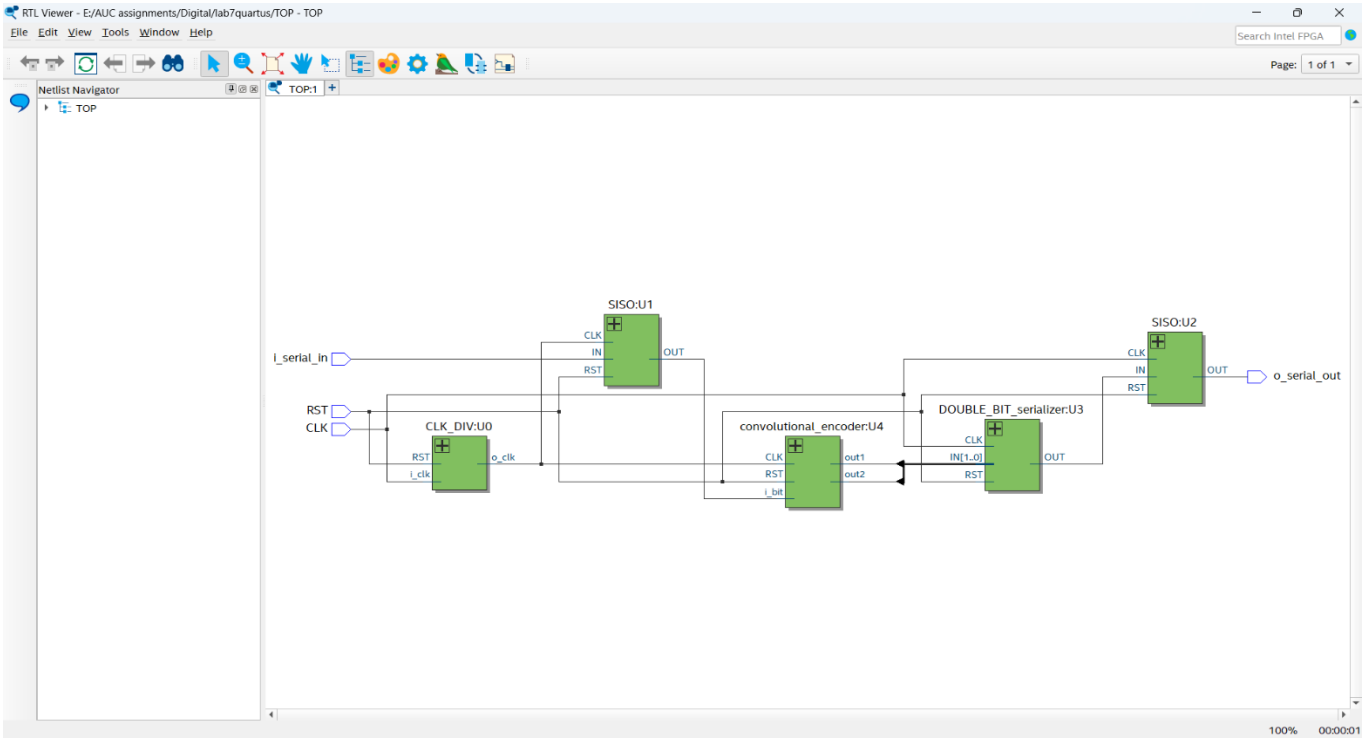
```
module TOP(input i_serial_in,
input CLK,
input RST,
output o_serial_out);
wire o_CLK,BIT,out1,out2,SER_IN;
CLK_DIV U0 (.i_clk(CLK),.RST(RST),.o_clk(o_CLK));

SISO U1 (.IN(i_serial_in),.CLK(o_CLK),.RST(RST),.OUT(BIT));


SISO U2 (.IN(SER_IN),.CLK(CLK),.RST(RST),.OUT(o_serial_out));

DOUBLE_BIT_serializer U3 (.IN({out2,out1}),.CLK(CLK),.RST(RST),.OUT(SER_IN));

convolutional_encoder U4 (.i_bit(BIT),.CLK(o_CLK),.RST(RST),.out1(out1),.out2(out2));



endmodule
```

➢ Test bench

E:/AUC assignments/Digital/conv_encoder/tb.v (/tb) - Default

```
Ln#
 1  module tb();
 2  reg i_serial_in,CLK,RST;
 3  wire o_serial_out;
 4  TOP U0 (.i_serial_in(i_serial_in),.CLK(CLK),.RST(RST),.o_serial_out(o_serial_out));
 5
 6  initial
 7   begin
 8   i_serial_in=1'b1;
 9   CLK=1'b0;
10   RST=1'b0;
11   repeat(5)  @(posedge CLK);
12   RST=1'b1;
13   i_serial_in=1'b0;
14   repeat(2)  @(posedge CLK);
15   i_serial_in=1'b1;
16   repeat(2)  @(posedge CLK);
17   i_serial_in=1'b0;
18   repeat(2)  @(posedge CLK);
19   i_serial_in=1'b1;
20   repeat(2)  @(posedge CLK);
21  i_serial_in=1'b0;
22   repeat(30)  @(posedge CLK);
23   $stop;
24   end
25
26  always #5 CLK=~CLK;
27
28  endmodule
```


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
sim - Default
Instance
Design unit
tb
U0
CLK_DIV
U1
SISO
U2
SISO
U3
DOUBLE_BI...
U4
convolution...
#INITIAL#6
#ALWAYS#26
#vsim_capacity#
Objects
BIT
CLK
i_serial_in
o_CLK
o_serial_out
out1
out2
RST
SER_IN
Processes (Active)
#INITIAL#6
Wave - Default
CLK
i_serial_in
o_serial_out
RST
CLK
IN
OUT
RST
SER_FLAG
Now
0.000425 us
Cursor 1
0.000332 us
Wave
DOUBLE_BIT_serializer.v
CONVOLUTIONAL_ENCODER.v
CLK_DIV.v
tb.v
Top.v
SISO.v
Transcript
** Note: $stop : E:/AUC assignments/Digital/conv_encoder/tb.v(23)
Time: 425 ps Iteration: 1 Instance: /tb
Break in Module tb at E:/AUC assignments/Digital/conv_encoder/tb.v line 23
VSIM 37>
0 ps to 576 ps
Project : con_encoder
Now: 425 ps Delta: 1
St0


RTL Viewer - E:/AUC assignments/Digital/lab7quartus/TOP - TOP
File Edit View Tools Window Help
Netlist Navigator
TOP
TOP:1
SISO:U1
SISO:U2
i_serial_in
o_serial_out
RST
CLK
CLK_DIV:U0
convolutional_encoder:U4
DOUBLE_BIT_serializer:U3
CLK
IN
OUT
RST
o_clk
i_clk
out1
out2
i_bit
IN[1..0]
100%
00:00:01

➢ Comments

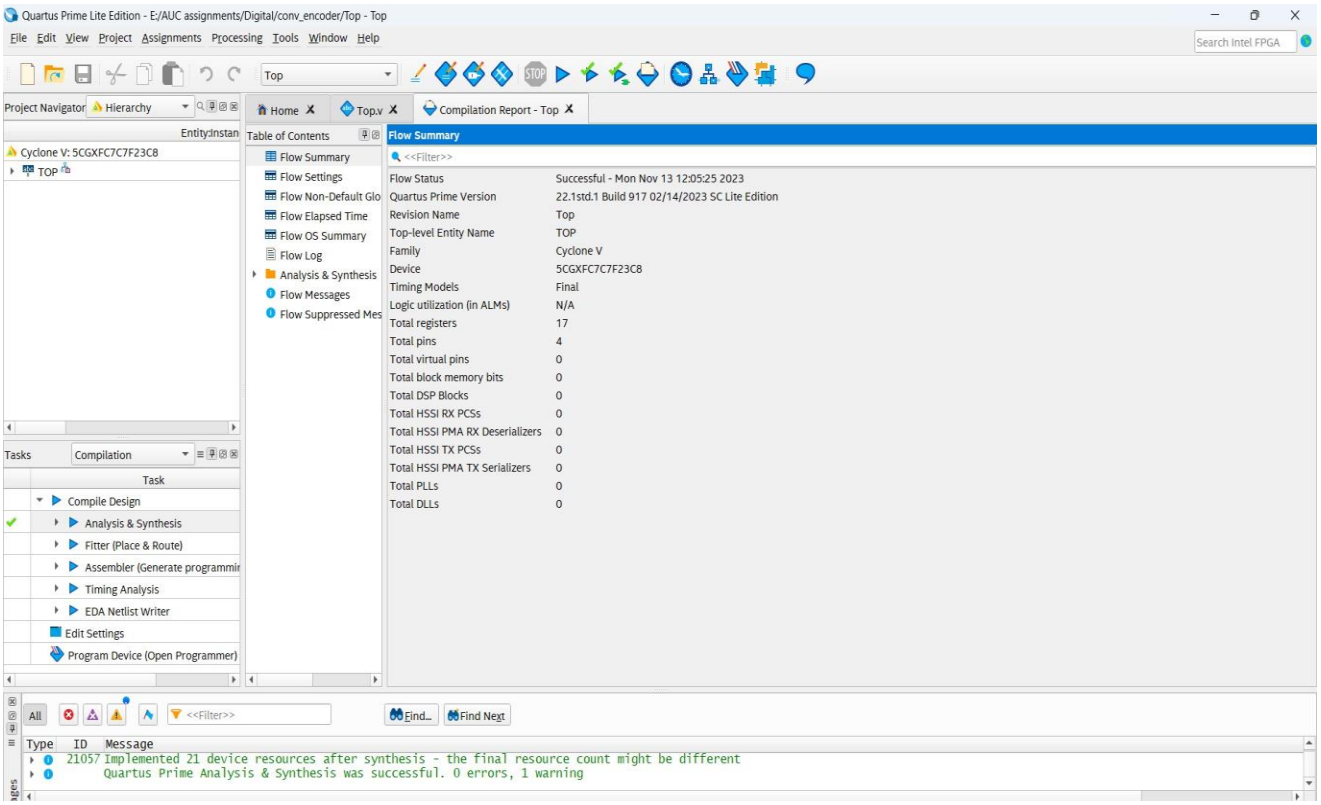
the TOP module integrates various sub-modules to implement a data transmission system. It includes a clock divider (CLK_DIV), a serial-in, serial-out shift register (SISO), a double-bit serializer (DOUBLE_BIT_serializer), and a convolutional encoder (convolutional_encoder). These modules work together to receive an input serial data stream, perform serialization, shift register operations, encoding, and produce the final serialized output data. The clock divider ensures proper synchronization and timing of the system. The top module integrates the following blocks:

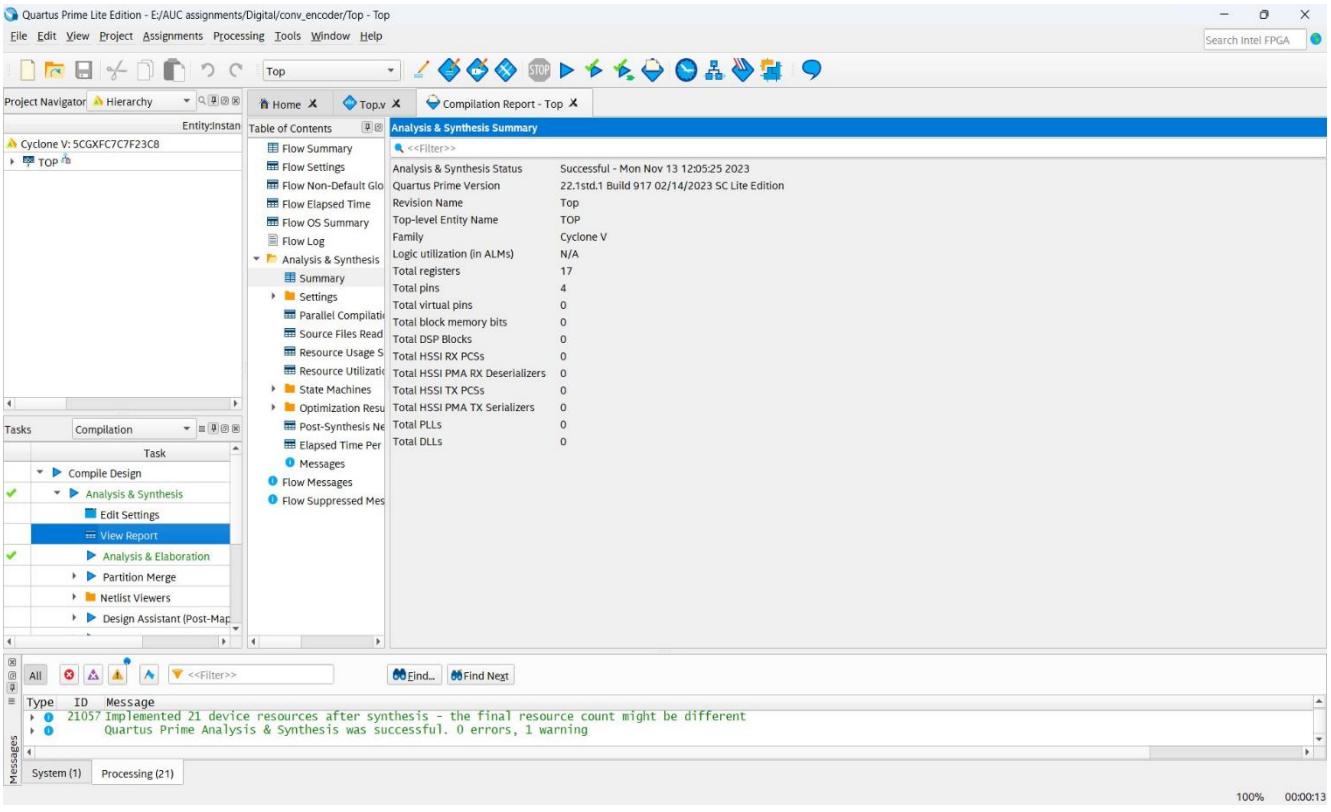
- U0 is an instance of the CLK_DIV module, which represents a clock divider. It takes the input clock signal (CLK) and produces a divided clock signal (o_CLK).
- U1 is an instance of the SISO module, which represents a serial-in, serial-out shift register. It takes the input serial data (i_serial_in), clock signal (o_CLK), and reset signal (RST), and produces the shifted output data (BIT).
- U2 is another instance of the SISO module. It takes the input serial data (SER_IN), the original clock signal (CLK), and reset signal (RST), and produces the shifted output data (o_serial_out).
- U3 is an instance of the DOUBLE_BIT_serializer module, which represents a double bit serializer. It takes the input data bits (out2 and out1), the clock signal (CLK), and the reset signal (RST), and produces the serialized output bit (SER_IN).
- U4 is an instance of the convolutional_encoder module, which represents a convolutional encoder. It takes the input data bit (BIT), the divided clock signal (o_CLK), the reset signal (RST), and produces two encoded output bits (out1 and out2).

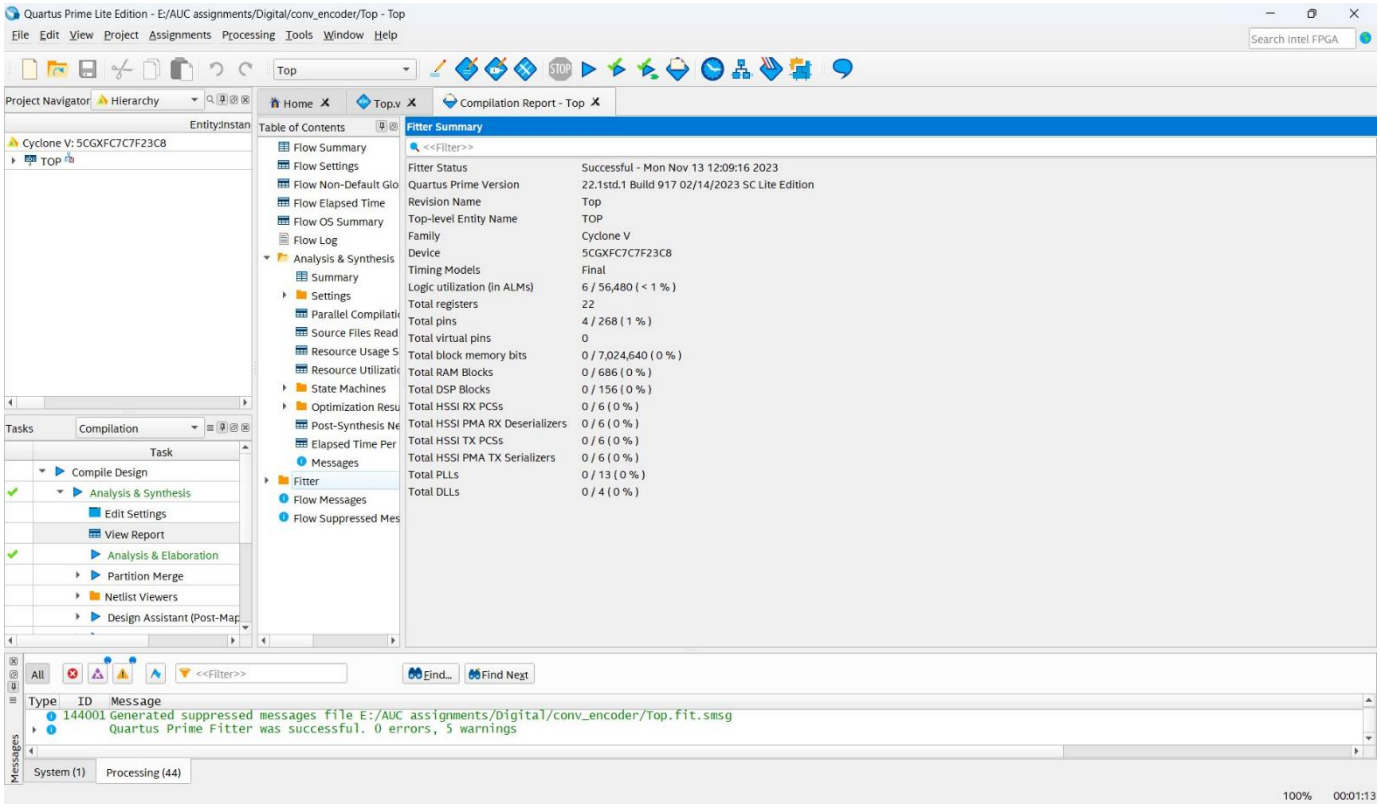
Finally, it gives me sequence pattern that are sent to the receiver by transmitter to reduce error in the data after sent it to receiver by using convolutional encoding method.

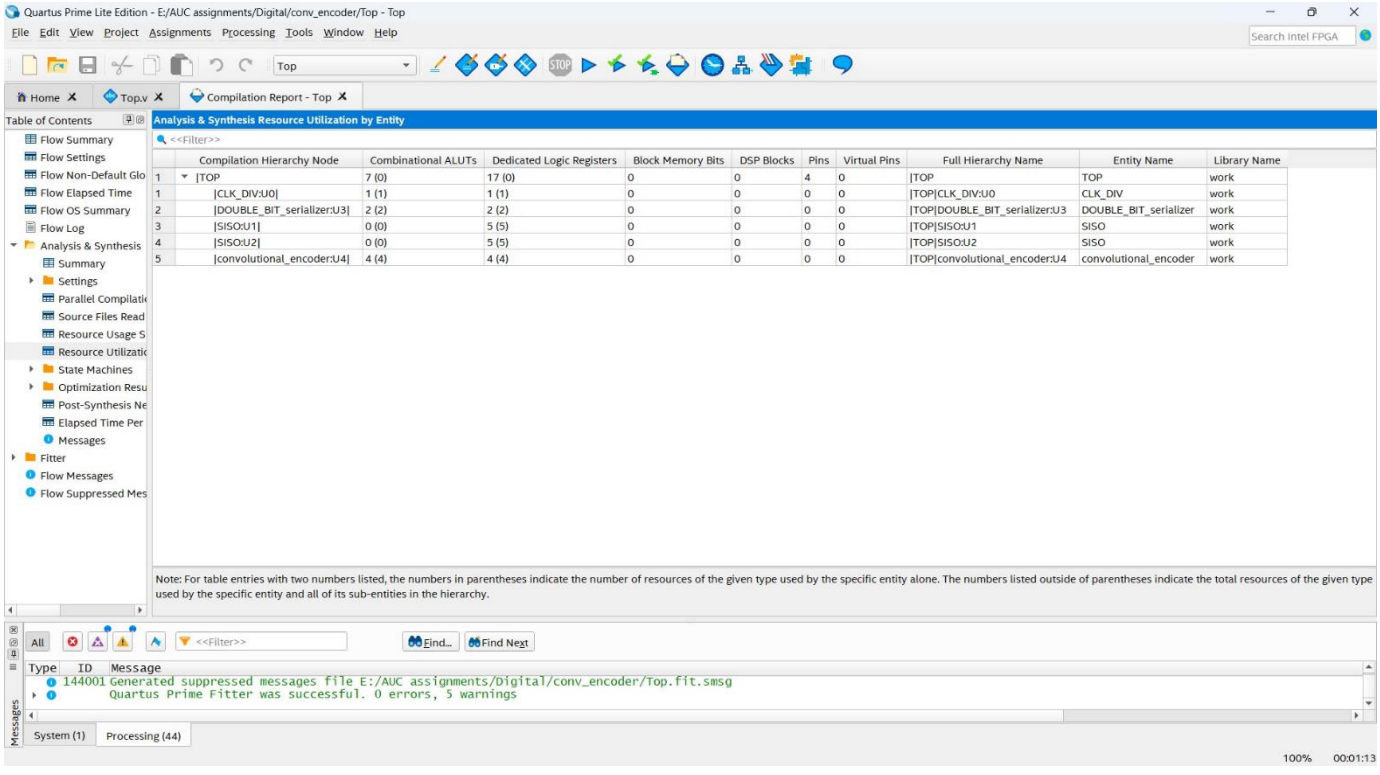
➢ Generated reports

Before edit clock properties and put constrains

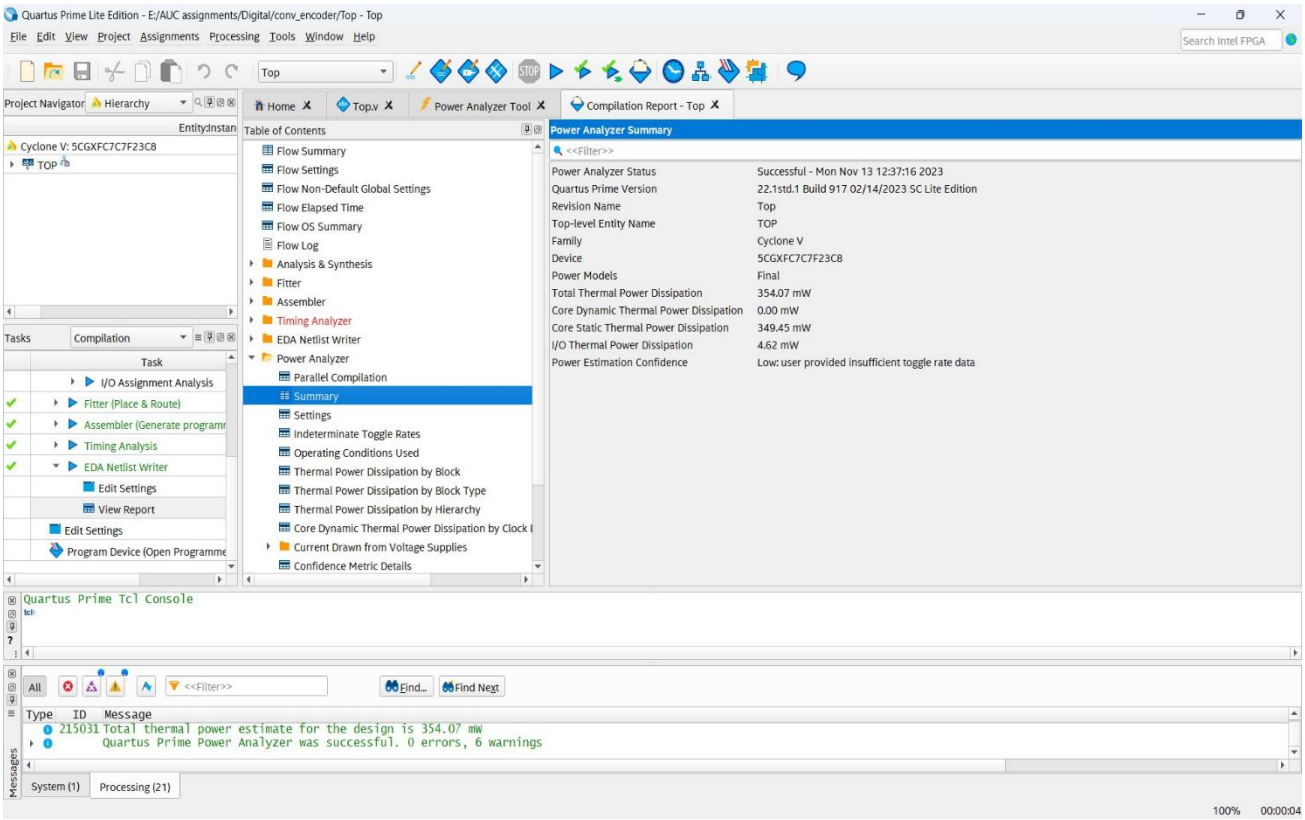




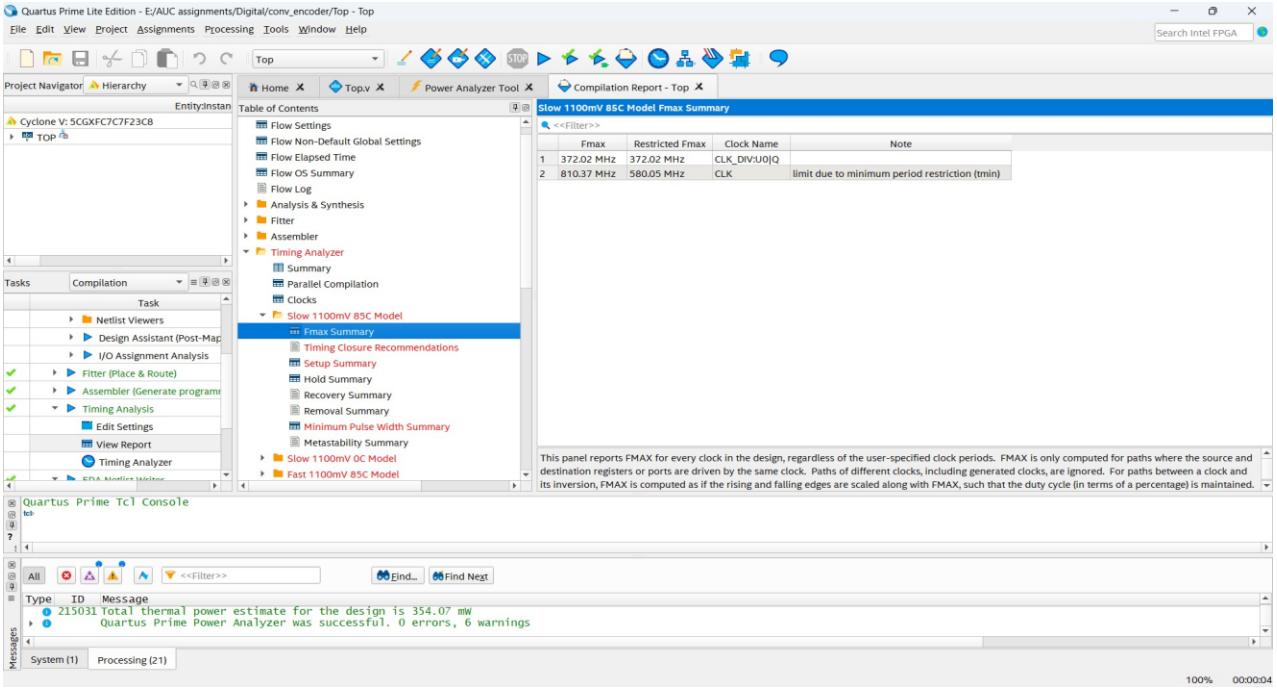

Quartus Prime Lite Edition - E:/AUC assignments/Digital/conv_encoder/Top - Top
File Edit View Project Assignments Processing Tools Window Help
Search Intel FPGA
Top
Project Navigator Hierarchy
Entity:Instan
Cyclone V: 5CGXFC7C7F23C8
TOP
Home
Top.v
Compilation Report - Top
Table of Contents
Flow Summary
Flow Settings
Flow Non-Default Glo
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Summary
Settings
Parallel Compilatio
Source Files Read
Resource Usage S
Resource Utilizatio
State Machines
Optimization Resu
Post-Synthesis Ne
Elapsed Time Per
Messages
Fitter
Flow Messages
Flow Suppressed Mes
Fitter Summary
<<Filter>>
Fitter Status Successful - Mon Nov 13 12:09:16 2023
Quartus Prime Version 22.1std.1 Build 917 02/14/2023 SC Lite Edition
Revision Name Top
Top-level Entity Name TOP
Family Cyclone V
Device 5CGXFC7C7F23C8
Timing Models Final
Logic utilization (in ALMs) 6 / 56,480 (< 1 %)
Total registers 22
Total pins 4 / 268 (1 %)
Total virtual pins 0
Total block memory bits 0 / 7,024,640 (0 %)
Total RAM Blocks 0 / 686 (0 %)
Total DSP Blocks 0 / 156 (0 %)
Total HSSI RX PCSs 0 / 6 (0 %)
Total HSSI PMA RX Deserializers 0 / 6 (0 %)
Total HSSI TX PCSs 0 / 6 (0 %)
Total HSSI PMA TX Serializers 0 / 6 (0 %)
Total PLLs 0 / 13 (0 %)
Total DLLs 0 / 4 (0 %)
Tasks Compilation
Task
Compile Design
Analysis & Synthesis
Edit Settings
View Report
Analysis & Elaboration
Partition Merge
Netlist Viewers
Design Assistant (Post-Map
All
<<Filter>>
Find...
Find Next
Type ID Message
144001 Generated suppressed messages file E:/AUC assignments/Digital/conv_encoder/Top.fit.smsg
Quartus Prime Fitter was successful. 0 errors, 5 warnings
Messages
System (1)
Processing (44)
100%
00:01:13


Quartus Prime Lite Edition - E:/AUC assignments/Digital/conv_encoder/Top - Top
File Edit View Project Assignments Processing Tools Window Help
Search Intel FPGA
Top
Home
Top.v
Compilation Report - Top
Table of Contents
Flow Summary
Flow Settings
Flow Non-Default Glo
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Summary
Settings
Parallel Compilatio
Source Files Read
Resource Usage S
Resource Utilizatio
State Machines
Optimization Resu
Post-Synthesis Ne
Elapsed Time Per
Messages
Fitter
Flow Messages
Flow Suppressed Mes
Analysis & Synthesis Resource Utilization by Entity
<<Filter>>
Compilation Hierarchy Node | Combinational ALUTs | Dedicated Logic Registers | Block Memory Bits | DSP Blocks | Pins | Virtual Pins | Full Hierarchy Name | Entity Name | Library Name
1 |TOP | 7 (0) | 17 (0) | 0 | 0 | 4 | 0 | |TOP | TOP | work
1 |CLK_DIV:U0| | 1 (1) | 1 (1) | 0 | 0 | 0 | 0 | |TOP|CLK_DIV:U0 | CLK_DIV | work
2 |DOUBLE_BIT_serializer:U3| | 2 (2) | 2 (2) | 0 | 0 | 0 | 0 | |TOP|DOUBLE_BIT_serializer:U3 | DOUBLE_BIT_serializer | work
3 |SISO:U1| | 0 (0) | 5 (5) | 0 | 0 | 0 | 0 | |TOP|SISO:U1 | SISO | work
4 |SISO:U2| | 0 (0) | 5 (5) | 0 | 0 | 0 | 0 | |TOP|SISO:U2 | SISO | work
5 |convolutional_encoder:U4| | 4 (4) | 4 (4) | 0 | 0 | 0 | 0 | |TOP|convolutional_encoder:U4 | convolutional_encoder | work
Note: For table entries with two numbers listed, the numbers in parentheses indicate the number of resources of the given type used by the specific entity alone. The numbers listed outside of parentheses indicate the total resources of the given type used by the specific entity and all of its sub-entities in the hierarchy.
All
<<Filter>>
Find...
Find Next
Type ID Message
144001 Generated suppressed messages file E:/AUC assignments/Digital/conv_encoder/Top.fit.smsg
Quartus Prime Fitter was successful. 0 errors, 5 warnings
Messages
System (1)
Processing (44)
100%
00:01:13

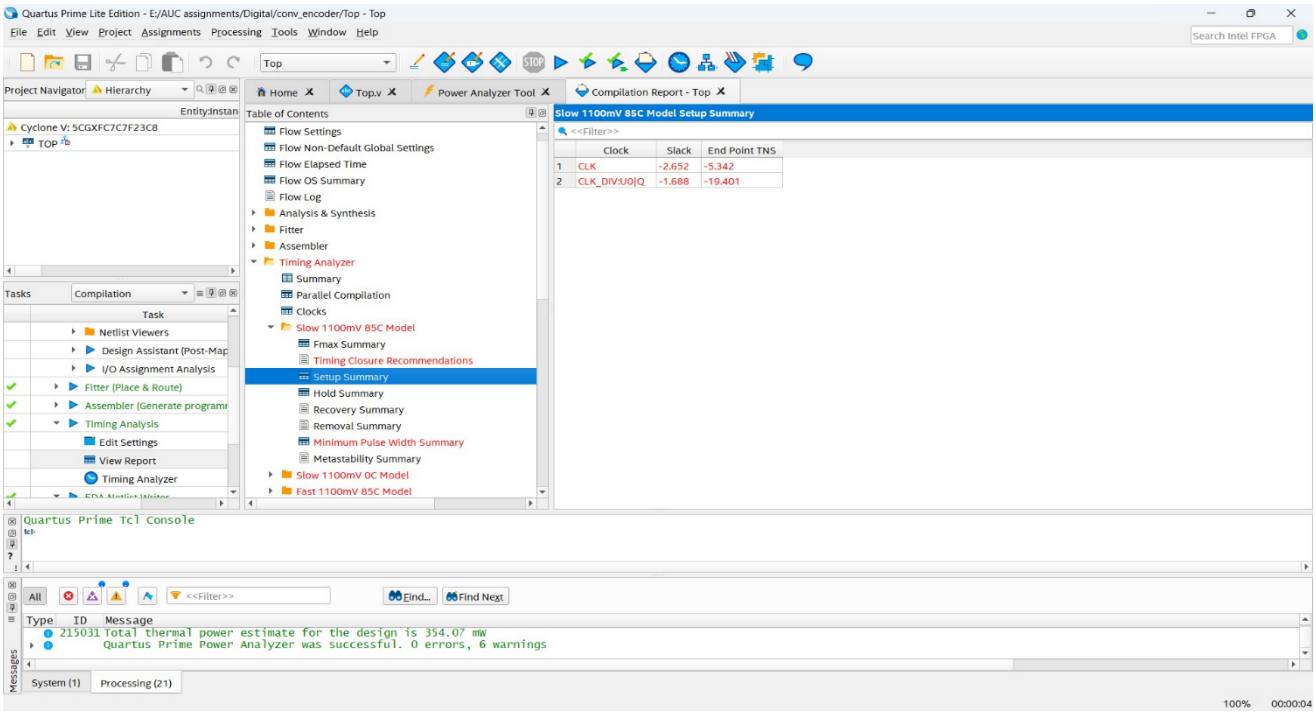
Total power dissipation



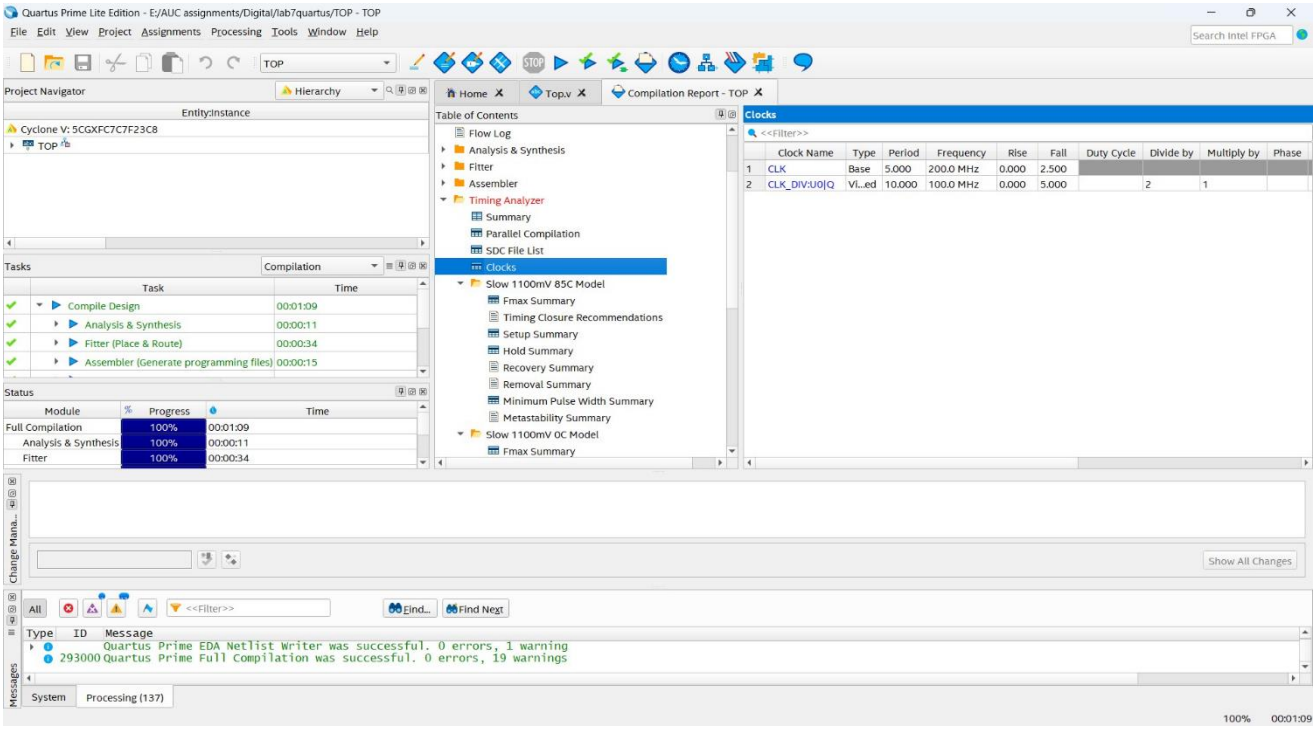
Maximum clock frequency



setup time -ve slack before editing clock frequency.



After edit clock properties and put constrains



Setup time and hold time after solving