



Digital Design

SPI Project



Provided by
Saeed Nabawy

Zeyad Abdallah Shaban
Mohamed Hassan Abdelkhalek
Submitted to
ENG / Kareem Waseem

Table of Contents

Introduction	5
Project Description.....	6
Overview.....	6
Top-Level Module: SPI_Wrapper	6
SPI_Slave Module	7
RAM Module.....	7
Data Conversion Modules.....	7
Serial_to_Parallel	7
parallel_to_serial	7
SPI_Wrapper_tb (Testbench)	8
Modules Code	9
Serial_to_Parallel Module	9
Parallel_to_Serial Module	10
RAM Module.....	11
SPI_Slave Module	12
SPI_Wrapper Module.....	14
Testbench.....	15
Do File	16
QuestaSim Simulation.....	17
Write Address.....	17
Write Data.....	17
Data Modified in Memory	18
Read Address	18
Data in Memory Address 'h255 ('d85)	18
Read Data & MISO output	19
Linting	19
Elaboration (seq encoding)	20
Message.....	20

Schematic	20
RAM Schematic	21
SPI Schematic	21
Synthesis (seq encoding)	22
Schematic.....	22
SPI_Slave Schematic.....	23
Message.....	24
Utilization Report.....	24
Timing Report.....	24
Critical path highlighted in the schematic	25
Implementation (seq encoding)	26
Device.....	26
Message.....	27
Utilization Report.....	27
Timing Report.....	27
Constrain File	28
Elaboration (gray encoding)	29
Message.....	29
Schematic.....	29
Synthesis (gray encoding)	30
Schematic.....	30
SPI_Slave Schematic.....	31
Message.....	32
Utilization Report.....	32
Timing Report.....	32
Implementation (gray encoding)	33
Device.....	33
Message.....	34
Utilization Report.....	34

Timing Report.....	34
Elaboration (one_hot encoding)	35
Message.....	35
Schematic.....	35
Synthesis (one_hot encoding)	36
Schematic.....	36
SPI_Slave Schematic.....	37
Message.....	38
Utilization Report.....	38
Timing Report.....	38
Implementation (one_hot encoding)	39
Device.....	39
Message.....	40
Utilization Report.....	40
Timing Report.....	40

Introduction

In the ever-evolving world of embedded systems, devices must speak a common language — one not built on human words, but on precise timing, pulses, and codes. Deep within the silicon silence of microcontrollers and peripheral chips, the Serial Peripheral Interface (SPI) is that language: a minimalist protocol, sharp and efficient, whispering commands and responses in synchronous rhythm.

This project delves into the hidden patterns of this digital dialogue. We design and implement a custom SPI Slave system in Verilog — a silent listener that not only interprets incoming serial data but acts on it intelligently. Like a watchful gatekeeper, it receives instructions, deciphers intent, and manipulates memory with precision.

The design merges state-machine logic with serial-to-parallel and parallel-to-serial data conversion. It features an internal RAM array, capable of storing and retrieving data based on encoded commands, all processed in real time through carefully synchronized hardware logic.

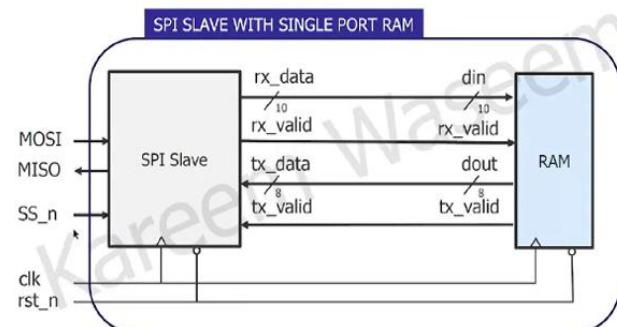
What begins as a stream of shifting bits over a single wire becomes, in this system, a fully responsive, memory-aware peripheral — crafted not just to understand the SPI protocol, but to embody it.

Project Description

Overview

This project implements a **complete SPI Slave system in Verilog**, with a built-in memory subsystem and full support for read/write communication initiated by an SPI Master. It includes all necessary components for:

- Serial-to-parallel conversion (to interpret incoming SPI data),
- Parallel-to-serial conversion (to send responses back),
- Command decoding,
- Addressable memory storage (RAM),
- And a testbench to simulate real-world usage.



Top-Level Module: SPI_Wrapper

This module acts as the central unit. It wires together:

- The SPI_Slave module, which handles protocol logic.
- The RAM module, which stores and retrieves data based on SPI commands.

Ports:

- Inputs: MOSI, SS_n, clk, rst_n
- Output: MISO

Internal Signals:

- rx_data [9:0] – full 10-bit command/data received from SPI
- rx_valid – signals when a valid 10-bit word has been received
- tx_data [7:0] – data to be transmitted over SPI
- tx_valid – signals when data is ready to be transmitted

SPI_Slave Module

This is the protocol engine. It:

- Uses an FSM with five states: IDLE, CHK_CMD, WRITE, READ_ADD, READ_DATA
- Receives 10-bit commands using the Serial_to_Parallel module
- Transmits 8-bit data using the parallel_to_serial module
- Signals data readiness with rx_valid, and waits for tx_valid to begin transmission

RAM Module

A simple 256-byte memory (8-bit wide, 8-bit addressable), supporting:

- Address selection via SPI
- Data storage and retrieval
- Behavioral simulation support via \$readmemh

It interprets the rx_data coming from the slave:

- 00 opcode sets the internal write address
- 01 opcode writes data to that address
- 10 opcode sets the read address
- 11 opcode reads data from that address and outputs it on dout

Data Conversion Modules

Serial_to_Parallel

- Converts incoming SPI stream into 10-bit parallel words
- Detects transaction completion and triggers done pulse

parallel_to_serial

- Converts 8-bit parallel data to serial format
- Manages busy signal to indicate transmission state

SPI_Wrapper_tb (Testbench)

The testbench simulates:

- Clock and reset behavior
- Command sequences for write and read operations
- Realistic SPI timing (bit-wise shifting over MOSI)
- Memory preload using \$readmemh("mem.dat", ...)

Test Phases:

1. Reset system
2. Write address
3. Write data
4. Read address
5. Read data
6. Wait for response, verify behavior

Modules Code

Serial_to_Parallel Module

```
 1 module Serial_to_Parallel(
 2     input clk,
 3     input rst_n,
 4     input serial_in,
 5     input enable,
 6     output reg [9:0] parallel_out,
 7     output reg done
 8 );
 9
10
11
12     reg [3:0] bit_count;
13     reg [9:0] shift_reg;
14
15     always @(posedge clk) begin
16         if (!rst_n) begin
17             shift_reg     <= 10'b0;
18             bit_count    <= 4'b0;
19             done          <= 1'b0;
20             parallel_out <= 10'b0;
21         end
22         else if (enable) begin
23             shift_reg <= {shift_reg[8:0], serial_in};
24             bit_count <= bit_count + 1;
25
26             if (bit_count == 4'b1001) begin
27                 parallel_out <= {shift_reg[8:0], serial_in};
28                 shift_reg <= 10'b0;
29                 done <= 1'b1;
30                 bit_count <= 4'b0;
31             end else begin
32                 done <= 1'b0;
33             end
34         end else begin
35             done <= 1'b0;
36         end
37     end
38
39 endmodule
40
```

Parallel_to_Serial Module

```
 1  module parallel_to_serial (
 2      input wire clk,
 3      input wire rst_n,
 4      input wire [7:0] parallel_in,
 5      input wire load,
 6      output reg serial_out,
 7      output reg busy
 8  );
 9
10    reg [7:0] shift_reg;
11    reg [2:0] bit_count;
12
13    always @(posedge clk) begin
14        if (!rst_n) begin
15            shift_reg  <= 8'd0;
16            bit_count <= 3'd0;
17            serial_out <= 1'bz;
18            busy       <= 1'b0;
19        end else if (load && !busy) begin
20            shift_reg  <= parallel_in;
21            bit_count <= 3'd0;
22            busy       <= 1'b1;
23        end
24        else if (busy) begin
25            serial_out <= shift_reg[7];
26            shift_reg  <= {shift_reg[6:0], 1'b0};
27            bit_count <= bit_count + 1;
28
29            if (bit_count == 3'd7) begin
30                busy <= 1'b0;
31            end
32        end
33        else begin
34            serial_out <= 1'bz;
35            busy       <= 1'b0;
36        end
37    end
38
39  endmodule
40
```

RAM Module

```
1 module RAM (
2     input [9:0] din,
3     input rx_valid,clk,rst_n,
4     output reg tx_valid,
5     output reg [7:0] dout
6 );
7
8 parameter MEM_DEPTH = 256;
9 parameter ADDR_SIZE = 8;
10
11 reg [ADDR_SIZE-1:0] write_address;
12 reg [ADDR_SIZE-1:0] read_address;
13
14 reg [7:0] mem [0:MEM_DEPTH-1];
15
16
17 always @(posedge clk) begin
18     if(!rst_n) begin
19         dout <= 0;
20         tx_valid <= 0;
21     end
22     else begin
23         if(rx_valid) begin
24             case(din[9:8])
25                 2'b00 : begin
26                     write_address <= din[7:0];
27                     dout <= 8'bzzzzzzz;
28                     tx_valid <= 1'bz;
29                 end
30                 2'b01 : begin
31                     mem [write_address] <= din[7:0];
32                     dout <= 8'bzzzzzzz;
33                     tx_valid <= 1'bz;
34                 end
35                 2'b10 : begin
36                     read_address <= din[7:0];
37                     dout <= 8'bzzzzzzz;
38                     tx_valid <= 1'bz;
39                 end
40                 2'b11 : begin
41                     dout <= mem [read_address];
42                     tx_valid <= 1;
43                 end
44                 default : begin
45                     dout <= 8'bzzzzzzz;
46                     tx_valid <= 1'bz;
47                 end
48             endcase
49         end
50         else begin
51             dout <= 0;
52             tx_valid <= 0;
53         end
54     end
55 end
56
57 endmodule //RAM
```

SPI_Slave Module

```
1 module SPI_Slave(
2     input MOSI,
3     input SS_n,
4     input clk,
5     input rst_n,
6     input [7:0] tx_data,
7     input tx_valid,
8     output MISO,
9     output reg [9:0] rx_data,
10    output reg rx_valid
11 );
12
13 parameter IDLE = 5'b000;
14 parameter CHK_CMD = 5'b001;
15 parameter WRITE = 5'b010;
16 parameter READ_ADD = 5'b011;
17 parameter READ_DATA = 5'b100;
18
19 // (* fsm_encoding = "gray" *)
20 reg [4:0] cs_ns;
21 reg addr_or_data ;
22 reg STP_enable;
23 wire STP_Done;
24 wire [9:0] STP_out;
25 wire PTS_busy, PTS_out;
26
27 Serial_to_Parallel STP (.clk(clk), .rst_n(rst_n), .serial_in(MOSI),
28 .enable(STP_enable), .parallel_out(STP_out), .done(STP_Done));
29
30 parallel_to_serial PTS (.clk(clk), .rst_n(rst_n), .parallel_in(tx_data),
31 .load(tx_valid), .serial_out_buff(MISO), .busy(PTS_busy));
32
33 always @(cs,SS_n) begin
34     case(cs)
35         IDLE: begin
36             if(SS_n) begin
37                 ns = IDLE;
38             end
39             else begin
40                 ns = CHK_CMD;
41             end
42         end
43         CHK_CMD: begin
44             if(SS_n) begin
45                 ns = IDLE;
46             end
47             else begin
48                 if(MOSI) begin
49                     if(addr_or_data) begin
50                         ns = READ_DATA;
51                     end
52                     else begin
53                         ns = READ_ADD;
54                     end
55                 end
56                 else begin
57                     ns = WRITE;
58                 end
59             end
60         end
61         WRITE: begin
62             if(SS_n) begin
63                 ns = IDLE;
64             end
65             else begin
66                 ns = WRITE;
67             end
68         end
69         READ_ADD: begin
70             if(SS_n) begin
71                 ns = IDLE;
72             end
73             else begin
74                 ns = READ_ADD;
75             end
76         end
77         READ_DATA: begin
78             if(SS_n) begin
79                 ns = IDLE;
80             end
81             else begin
82                 ns = READ_DATA;
83             end
84         end
85         default : ns = IDLE;
86     endcase
87 end
88
```

```
1      always @(posedge clk or negedge rst_n) begin
2          if(!rst_n) begin
3              cs <= IDLE;
4          end
5          else begin
6              cs <= ns;
7          end
8      end
9
10     always @(posedge clk) begin
11         if(cs == IDLE || cs == CHK_CMD) begin
12             STP_enable <= 0;
13             rx_data<= {10{1'b0}};
14             rx_valid <= 1'b0;
15         end
16         else if(cs == WRITE) begin
17             STP_enable <= 1;
18             if(STP_Done) begin
19                 rx_valid <= 1;
20                 rx_data <= STP_out;
21                 STP_enable <= 0;
22             end
23         end
24         else if(cs == READ_ADD) begin
25             STP_enable <= 1;
26             if(STP_Done) begin
27                 rx_valid <= 1;
28                 rx_data <= STP_out;
29                 addr_or_data <= 1;
30                 STP_enable <= 0;
31             end
32         end
33         else if(cs == READ_DATA) begin
34             STP_enable <= 1;
35             if(STP_Done) begin
36                 rx_valid <= 1;
37                 rx_data<= STP_out;
38                 addr_or_data <= 0;
39                 STP_enable <= 0;
40             end
41         end
42     end
43 endmodule
```

SPI_Wrapper Module

```
● ● ●
1 module SPI_Wrapper(
2     input MOSI,
3     input SS_n,
4     input clk,
5     input rst_n,
6     output MISO
7 );
8     wire [9:0] rx_data;
9     wire rx_valid;
10    wire [7:0] tx_data;
11    wire tx_valid;
12
13    SPI_Slave SPI (.MOSI(MOSI),.SS_n(SS_n),.clk(clk),.rst_n(rst_n),
14 .MISO(MISO),.rx_data(rx_data),.rx_valid(rx_valid),.tx_data(tx_data),
15 .tx_valid(tx_valid));
16
17    RAM u_ram(.din(rx_data),.rx_valid(rx_valid),.dout(tx_data),
18 .tx_valid(tx_valid),.clk(clk),.rst_n(rst_n));
19
20 endmodule
```

Testbench

```
1  module SPI_Wrapper_tb();
2
3   reg MOSI,SS_n,clk,rst_n;
4   wire MISO;
5
6   SPI_Wrapper SPI(.MOSI(MOSI),.SS_n(SS_n),.clk(clk),
7   .rst_n(rst_n),.MISO(MISO));
8
9   initial begin
10    clk = 0;
11    forever #1 clk = ~clk;
12  end
13
14  integer i;
15  initial begin
16    $readmemh("mem.dat", SPI.u_ram.mem);
17    // resq test
18    rst_n = 0;
19    @(posedge clk);
20    rst_n = 1;
21
22    // Write Address
23    @(posedge clk);
24    SS_n = 0;
25    @(posedge clk);
26    MOSI = 0;
27    @(posedge clk);
28    for(i = 0; i < 10; i = i + 1) begin
29      if(i < 2) begin
30        MOSI = 0;
31      end
32      else begin
33        MOSI = ~MOSI;
34      end
35      @(posedge clk);
36
37    end
38    SS_n = 1;
39
40    // Write Data
41    @(posedge clk);
42    SS_n = 0;
43    @(posedge clk);
44    MOSI = 0;
45    @(posedge clk);
46    for(i = 0; i < 10; i = i + 1) begin
47      if(i == 0) begin
48        MOSI = 0;
49      end begin
50        MOSI = 1;
51      end
52      @(posedge clk);
53    end
54    SS_n = 1;
55
56    // Read Address
57    @(posedge clk);
58    SS_n = 0;
59    @(posedge clk);
60    MOSI = 1;
61    @(posedge clk);
62    for(i = 0; i < 10; i = i + 1) begin
63      if(i == 0) begin
64        MOSI = 1;
65      end
66      else if(i == 1) begin
67        MOSI = 0;
68      end
69      else begin
70        MOSI = ~MOSI;
71      end
72      @(posedge clk);
73    end
74    SS_n = 1;
75
76    // Read Data
77    @(posedge clk);
78    SS_n = 0;
79    @(posedge clk);
80    MOSI = 1;
81    @(posedge clk);
82    for(i = 0; i < 10; i = i + 1) begin
83      MOSI = 1;
84      @(posedge clk);
85    end
86    for(i = 0; i < 20; i = i + 1) begin
87      @(posedge clk);
88    end
89    SS_n = 1;
90    $stop;
91  end
92 endmodule
```

Do File

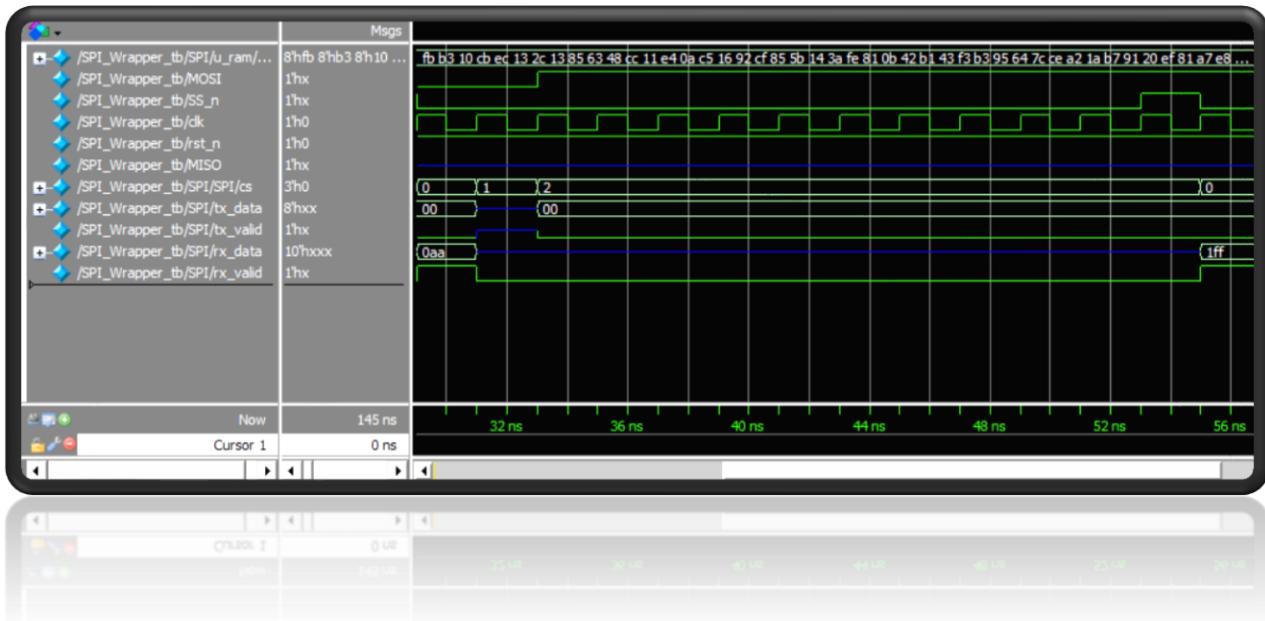
```
● ● ●  
1 vlib work  
2 vlog RAM.v Serial_to_Parallel.v Parallel_to_Serial.v SPI_Slave.v SPI_Wrapper.v  
3 vsim -voptargs=+acc work.SPI_Wrapper_tb  
4 add wave -position insertpoint sim:/SPI_Wrapper_tb/*  
5 add wave -position insertpoint \  
6 sim:/SPI_Wrapper_tb/SPI/rx_data \  
7 sim:/SPI_Wrapper_tb/SPI/rx_valid \  
8 sim:/SPI_Wrapper_tb/SPI/tx_data \  
9 sim:/SPI_Wrapper_tb/SPI/tx_valid  
10 add wave -position insertpoint \  
11 sim:/SPI_Wrapper_tb/SPI/SPI/cs  
12 add wave -position 0 sim:/SPI_Wrapper_tb/SPI/u_ram/mem  
13 run -all  
14 #quit -sim
```

QuestaSim Simulation

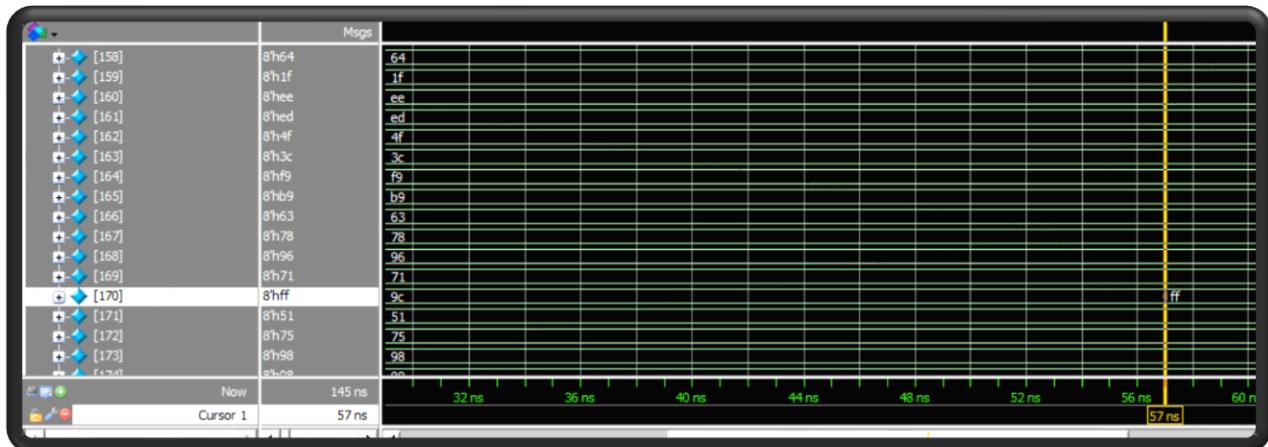
Write Address



Write Data



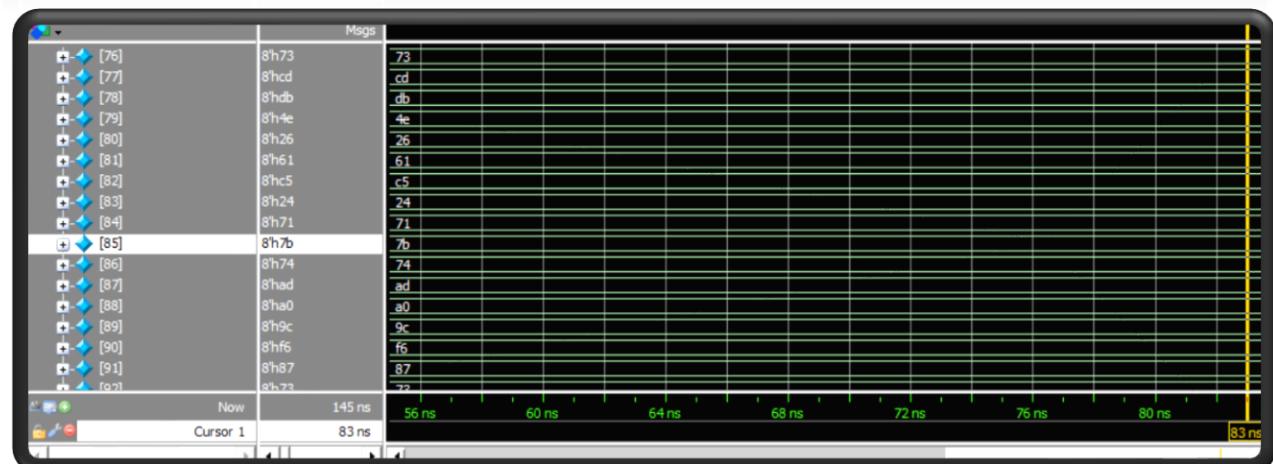
Data Modified in Memory



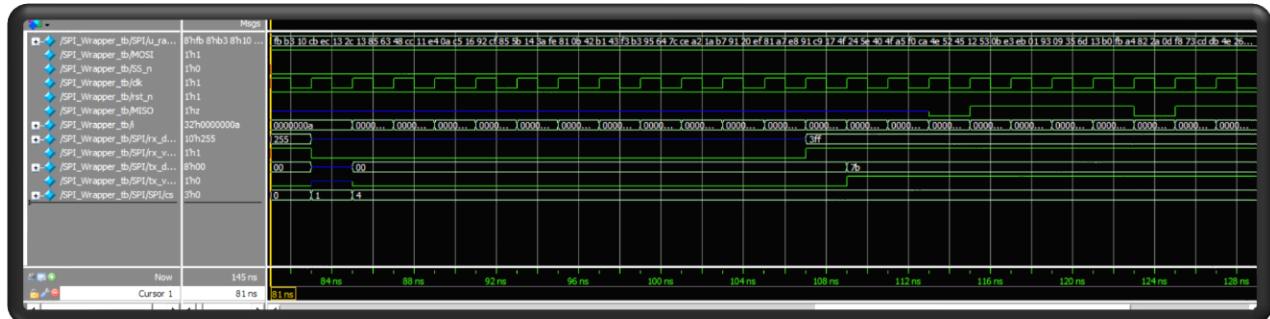
Read Address



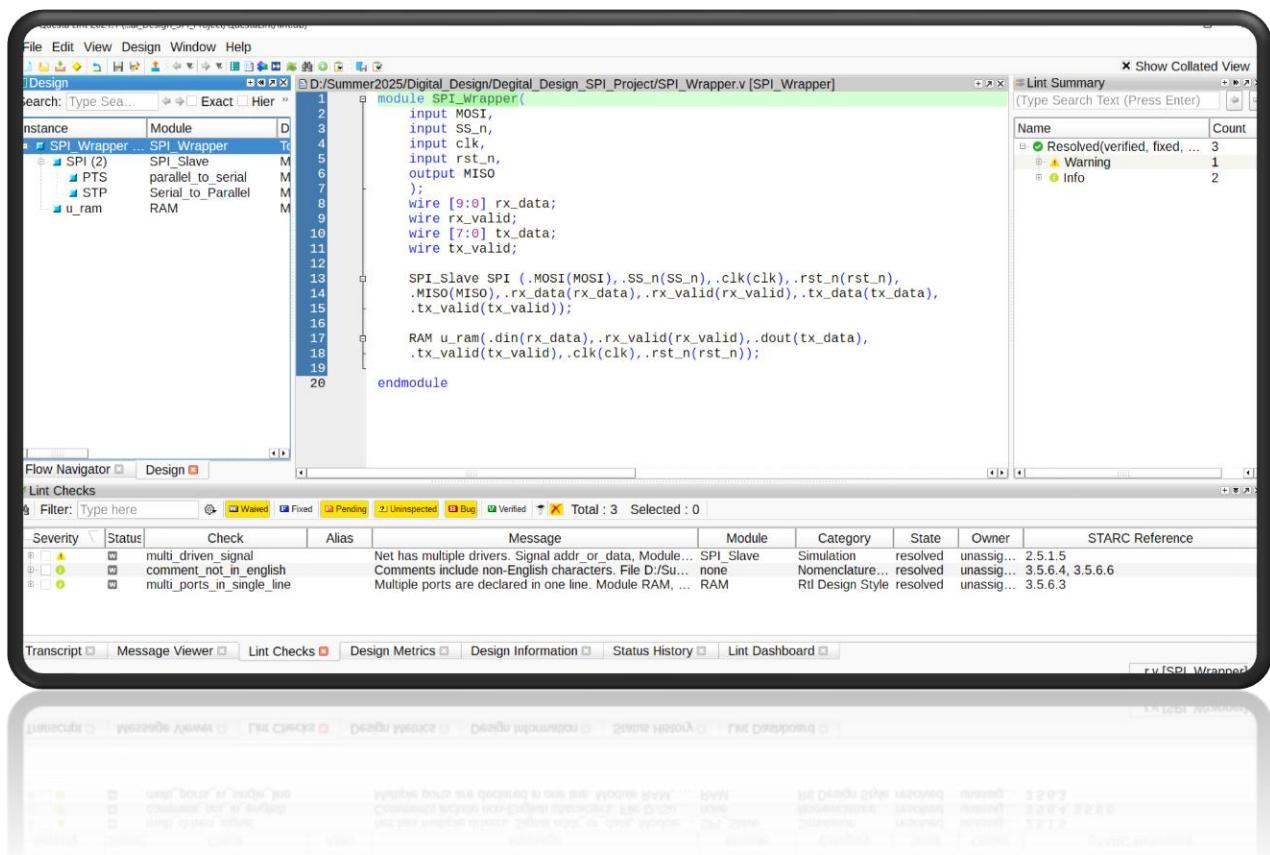
Data in Memory Address 'h255 ('d85)



Read Data & MISO output

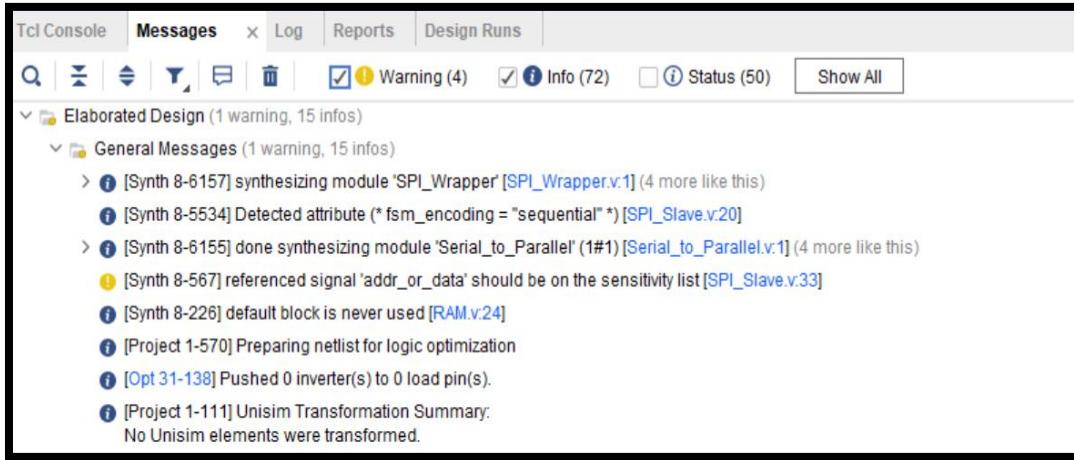


Linting

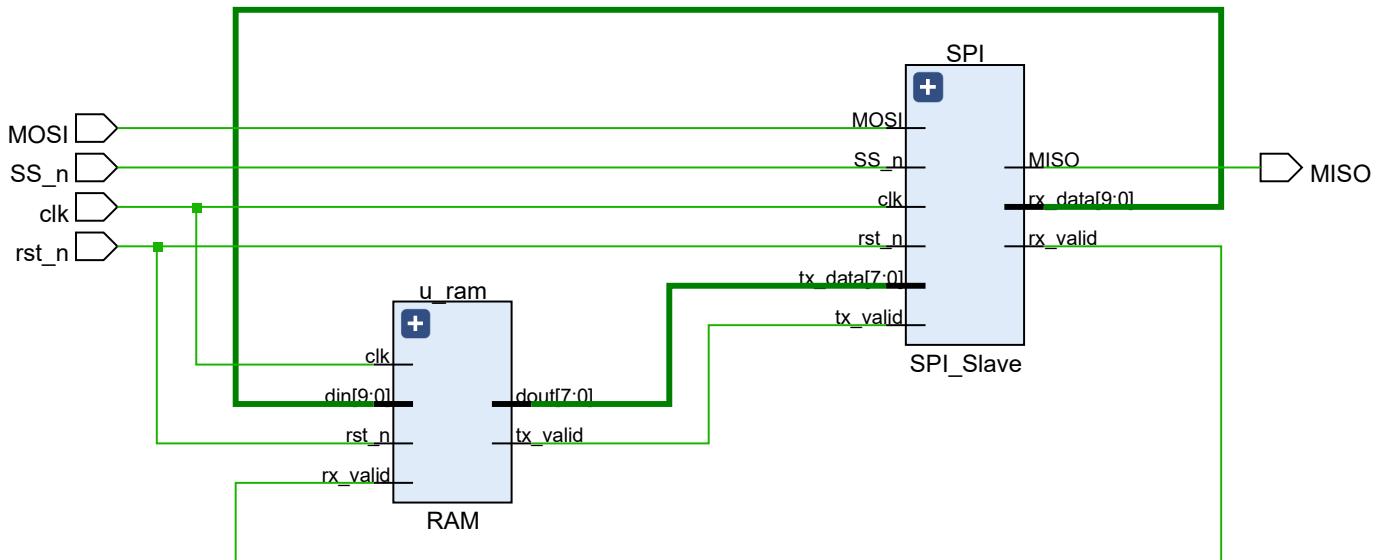


Elaboration (seq encoding)

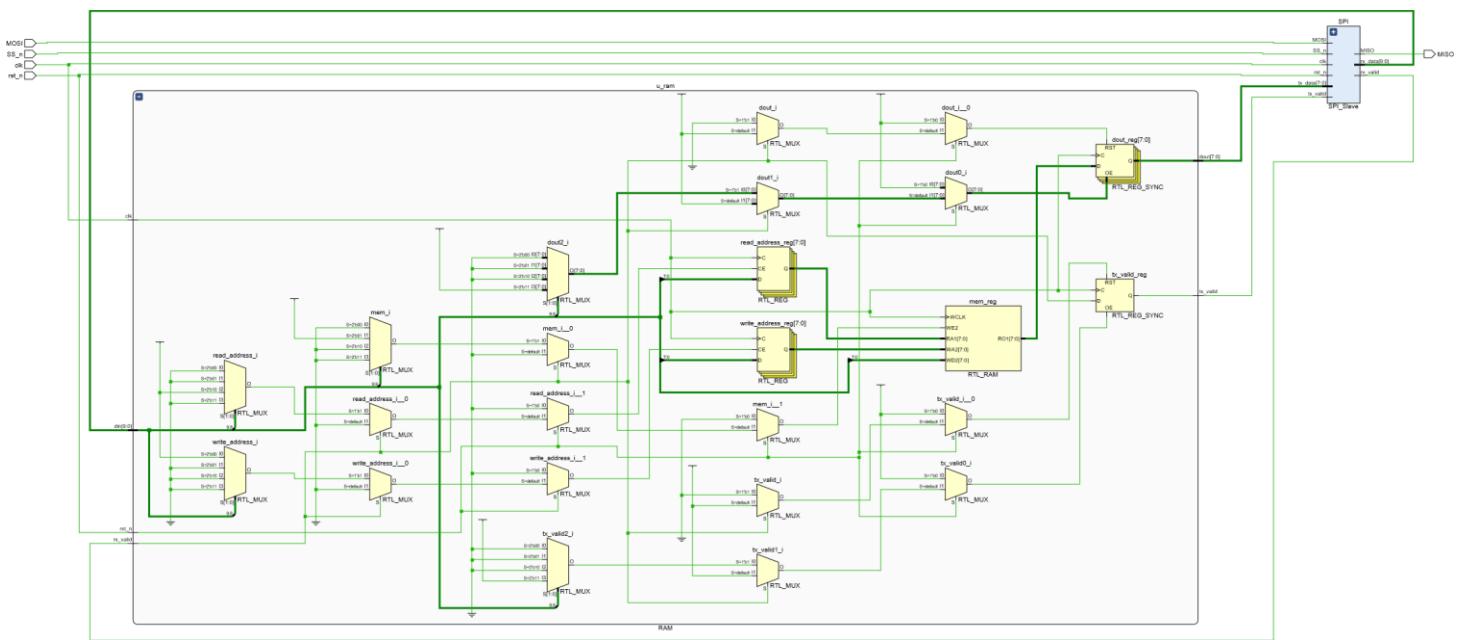
Message



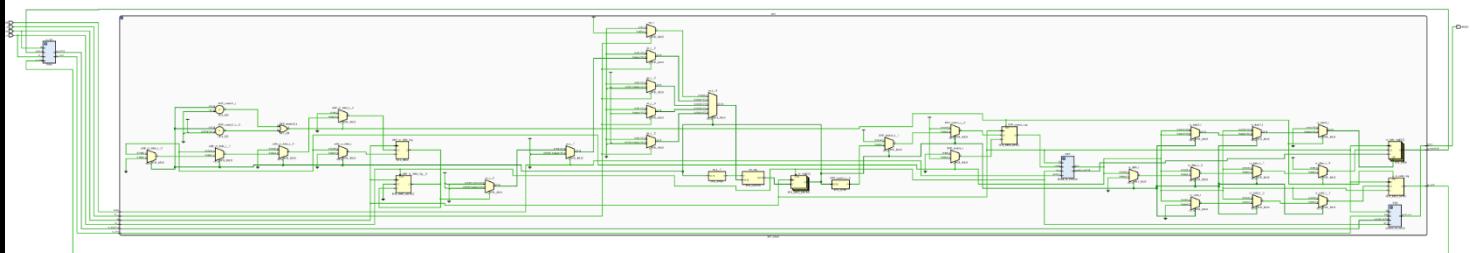
Schematic



RAM Schematic



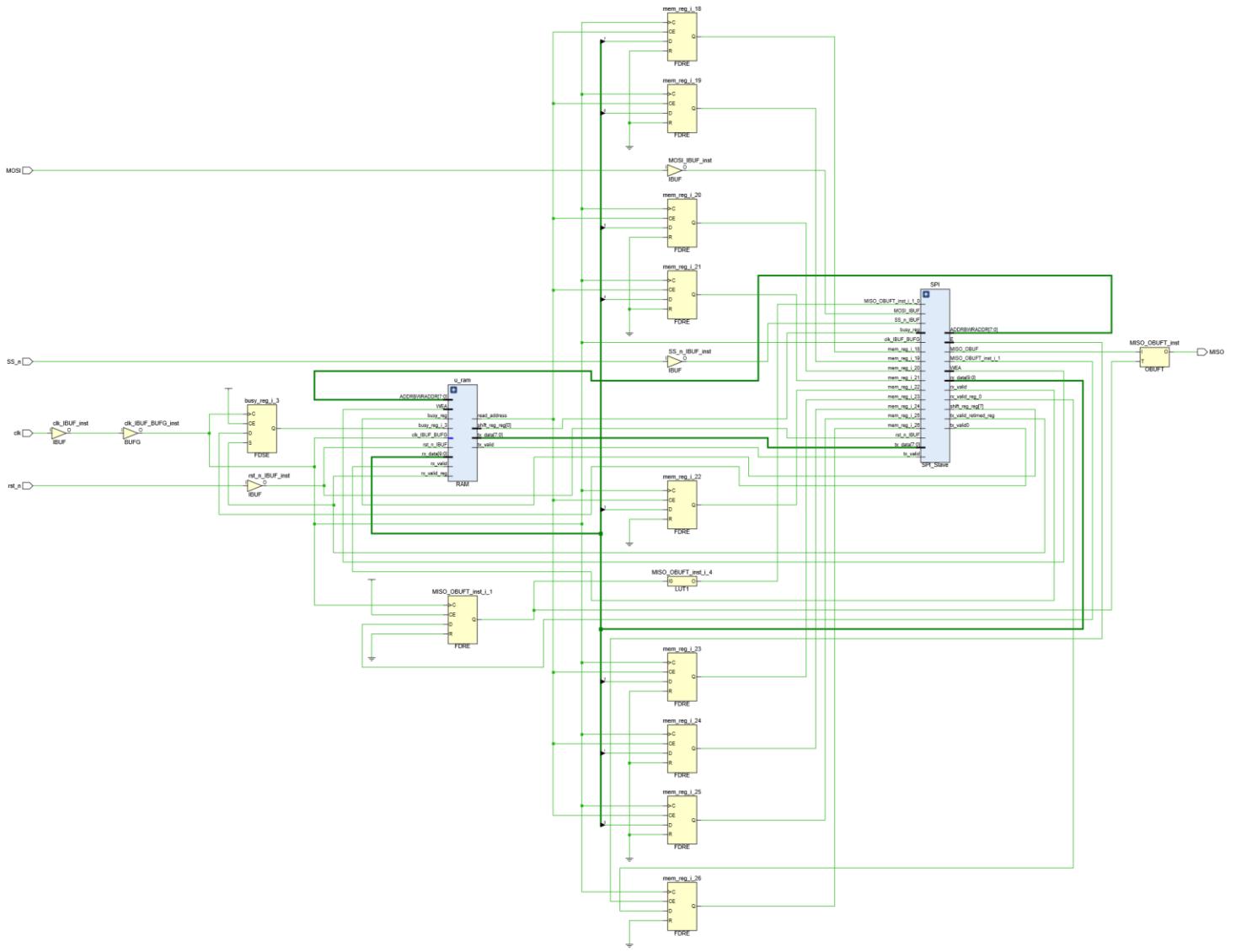
SPI Schematic



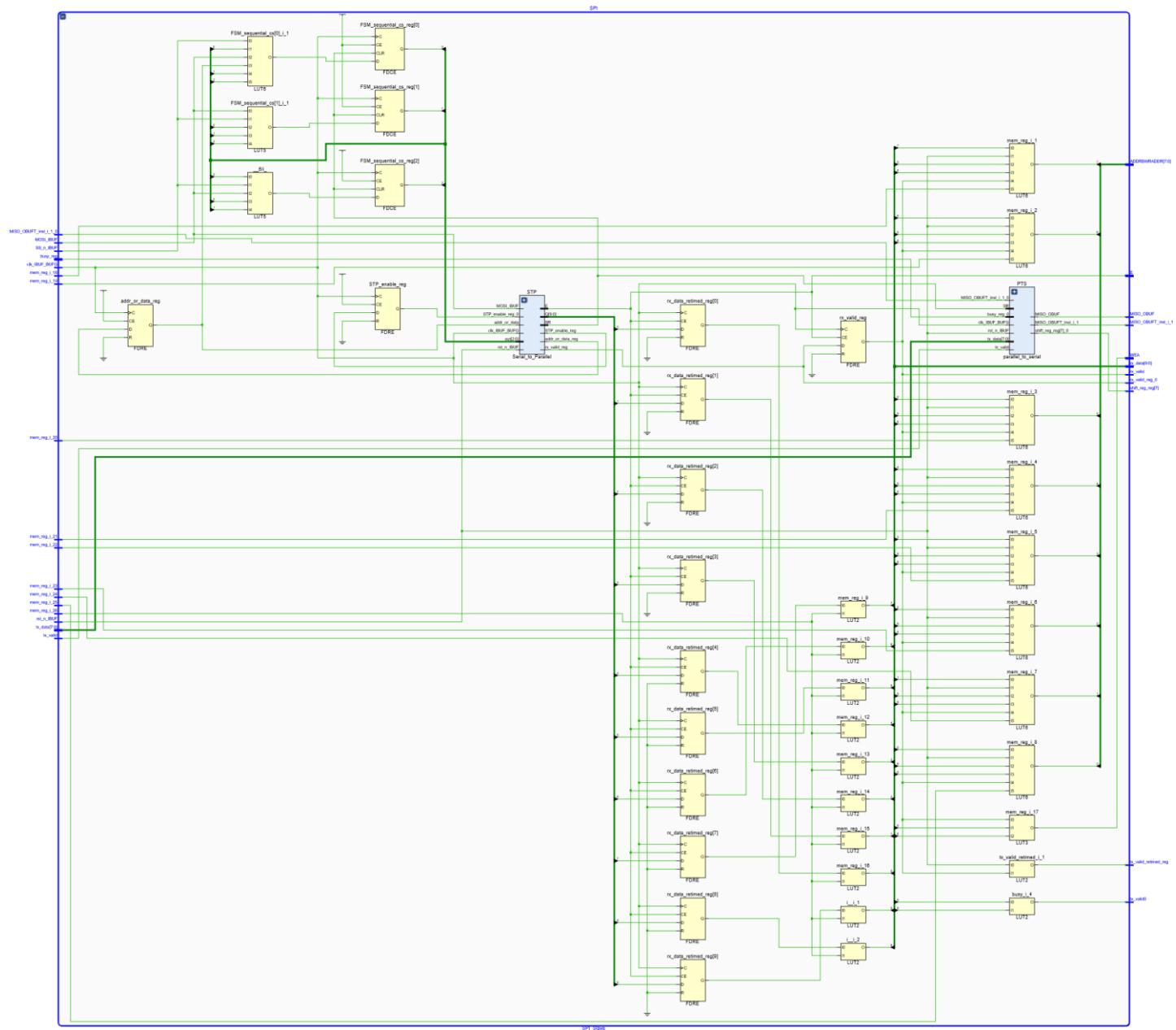
Synthesis (seq encoding)

State	Encoding
IDLE	5'b000
CHK_CMD	5'b001
WRITE	5'b010
READ_ADD	5'b011
READ_DATA	5'b100

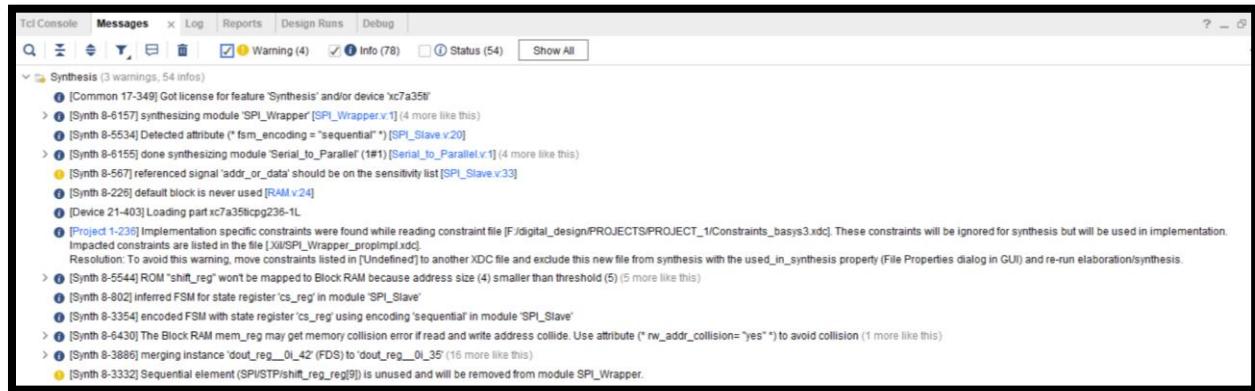
Schematic



SPI_Slave Schematic



Message



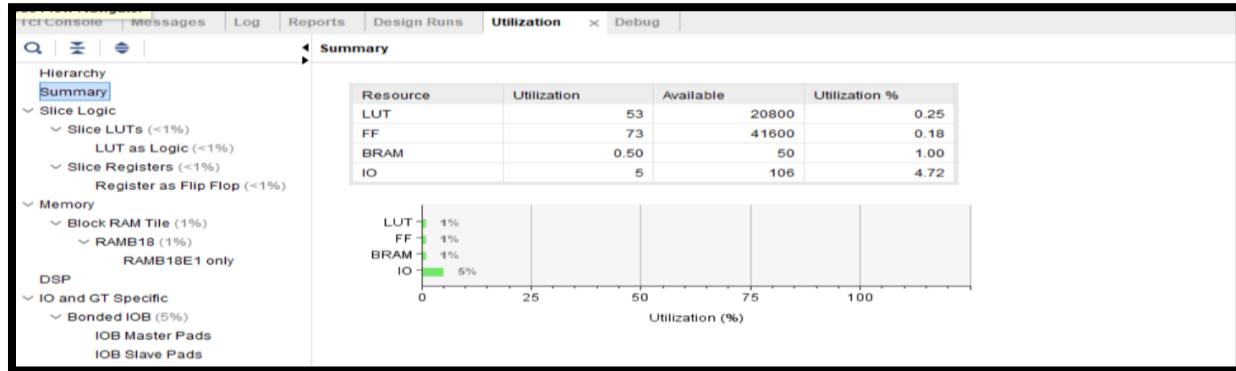
Tcl Console Messages Log Reports Design Runs Debug

Q W E F T M H B W Warning (4) I Info (78) S Status (54) Show All

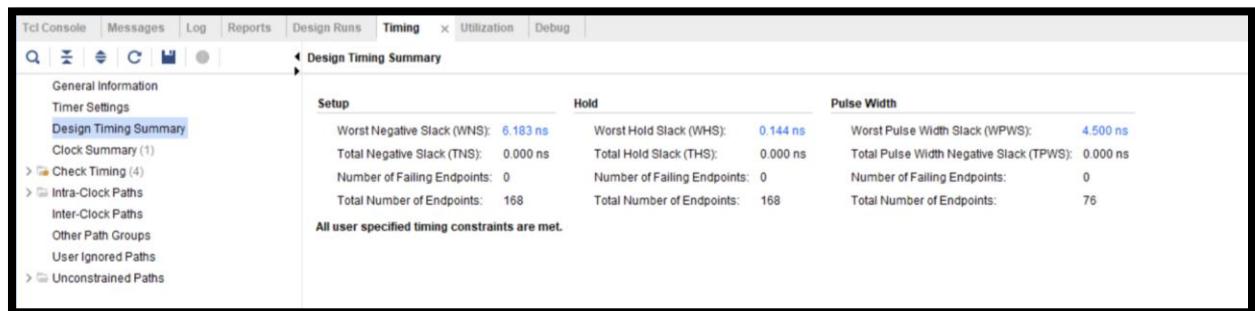
Synthesis (3 warnings, 54 infos)

- [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35f'
- [Synth 8-6-157] synthesizing module 'SPI_Wrapper [SPI_Wrapper.v1]' (4 more like this)
- [Synth 8-5534] Detected attribute (' fsm_encoding = "sequential" ') [SPI_Slave.v20]
- [Synth 8-6155] done synthesizing module 'Serial_to_Parallel (#1) [Serial_to_Parallel.v1]' (4 more like this)
- [Synth 8-567] referenced signal 'addr_or_data' should be on the sensitivity list [SPI_Slave.v33]
- [Synth 8-226] default block is never used [RAM.v24]
- [Device 21-403] Loading part xc7a35fclg236-1L
- [Project 1-236] Implementation specific constraints were found while reading constraint file [F:/digital_design/PROJECTS/PROJECT_1/Constraints_basys3.xdc]. These constraints will be ignored for synthesis but will be used in implementation. Impacted constraints are listed in the file [XILINX/SPI_Wrapper_propimpl.xdc]. Resolution: To avoid this warning, move constraints listed in [Undefined] to another XDC file and exclude this new file from synthesis with the used_in_synthesis property (File Properties dialog in GUI) and re-run elaboration/synthesis.
- [Synth 8-5544] ROM 'shift_reg' won't be mapped to Block RAM because address size (4) smaller than threshold (5) (5 more like this)
- [Synth 8-802] inferred FSM for state register 'cs_reg' in module 'SPI_Slave'
- [Synth 8-3354] Encoded FSM with state register 'cs_reg' using encoding 'sequential' in module 'SPI_Slave'
- [Synth 8-6430] The Block RAM mem_req may get memory collision error if read and write address collide. Use attribute (* rw_addr_collision="yes" *) to avoid collision (1 more like this)
- [Synth 8-3886] merging instance 'dout_reg_0_42' (FDS) to 'dout_reg_0_35' (16 more like this)
- [Synth 8-3332] Sequential element (SPI/STP/shift_reg/reg9) is unused and will be removed from module SPI_Wrapper.

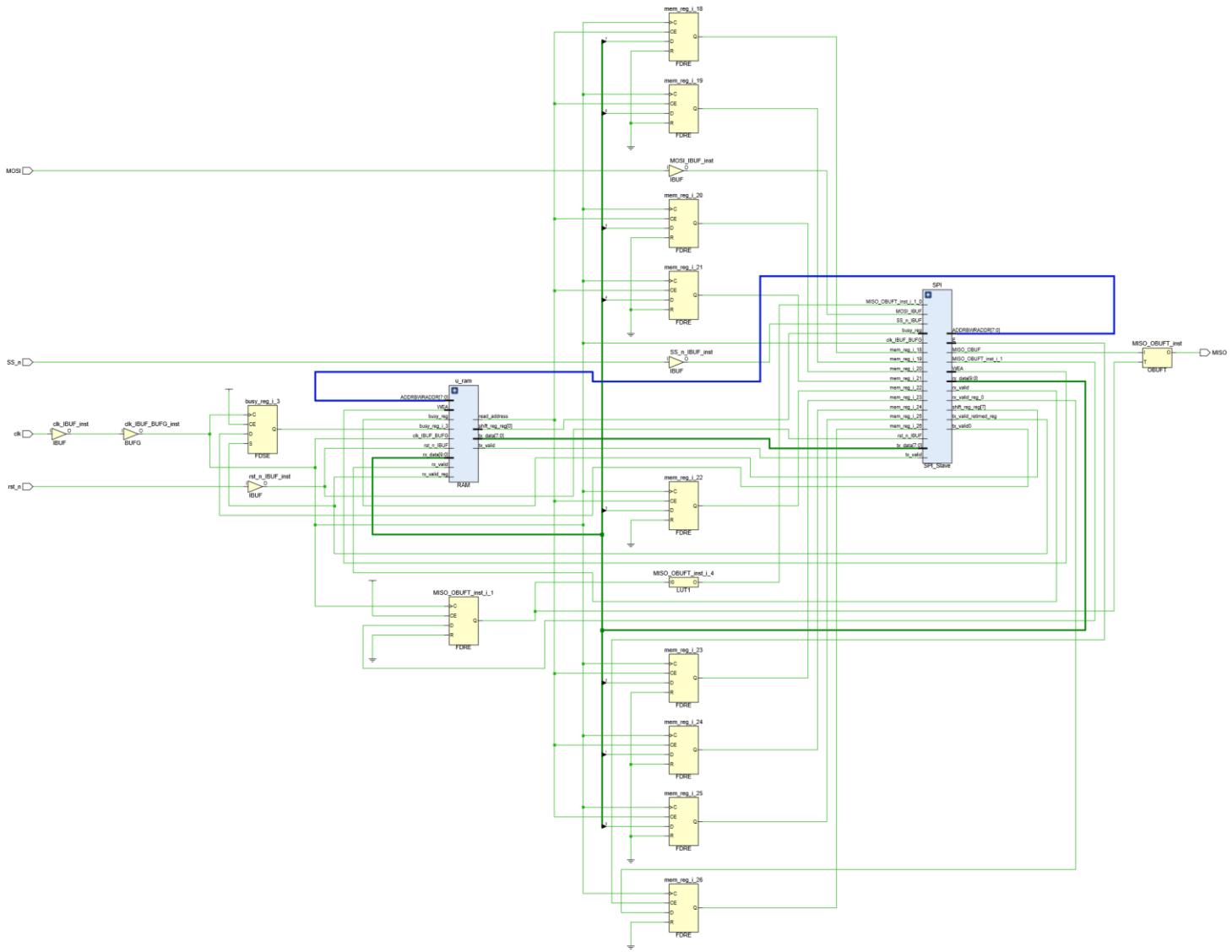
Utilization Report



Timing Report

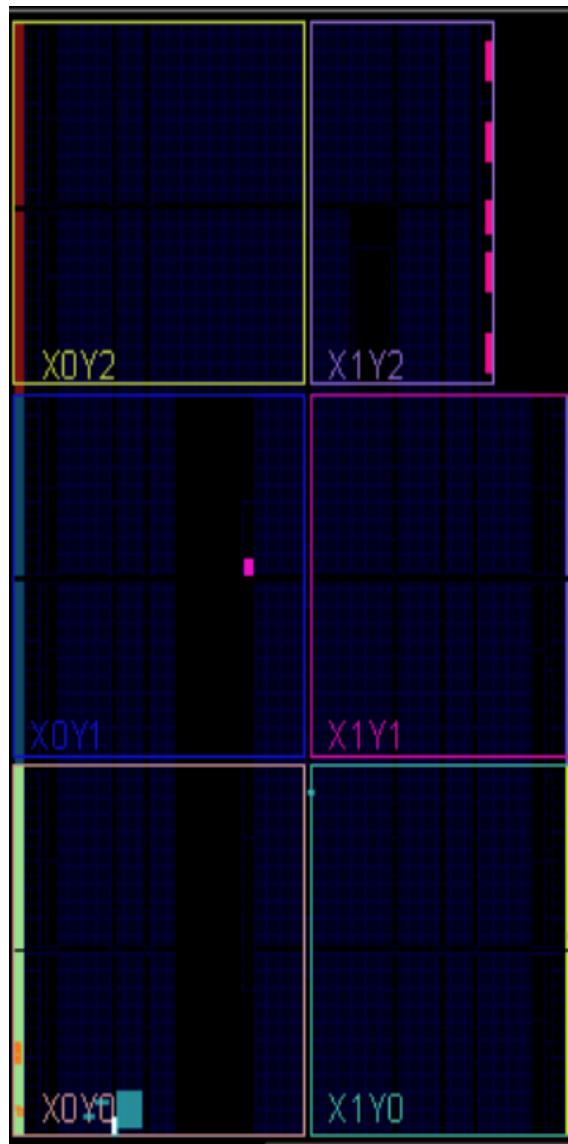


Critical path highlighted in the schematic



Implementation (seq encoding)

Device

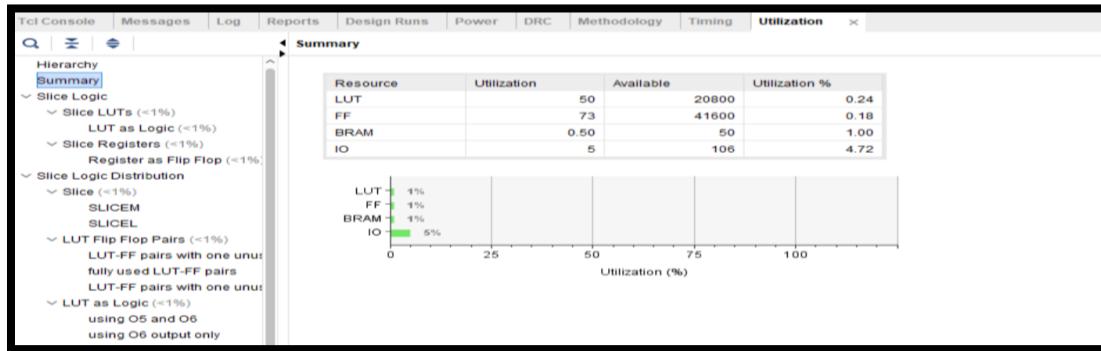


Message

The screenshot shows the 'Messages' tab in the Vivado interface. It displays a list of messages under 'Implemented Design (9 infos)'. The messages include:

- [Netlist 29-17] Analyzing 5 Unisim elements for replacement
- [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
- [Project 1-479] Netlist was created with Vivado 2018.2
- [Project 1-570] Preparing netlist for logic optimization
- [Timing 38-478] Restoring timing data from binary archive.
- [Timing 29-470] Binary timing data restore complete.

Utilization Report



Timing Report

The screenshot shows the 'Timing' tab in the Vivado interface. It displays a 'Design Timing Summary' table. The table shows the following timing constraints:

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.819 ns	Worst Hold Slack (WHS): 0.109 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWNS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 169	Total Number of Endpoints: 169	Total Number of Endpoints: 76

A note at the bottom states: "All user specified timing constraints are met."

Constrain File

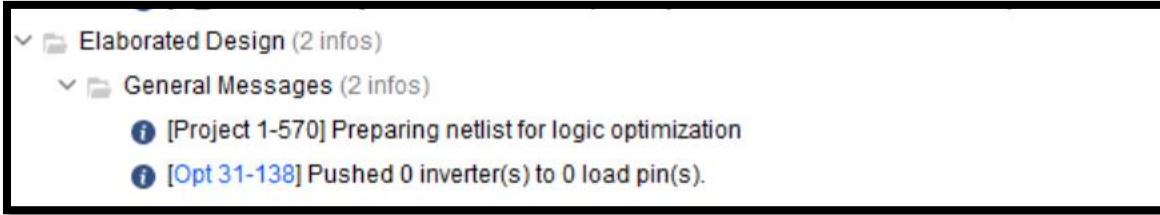
```

1  ## This file is a general .xco for the Basys3 rev B board
2  ## To use it in a project:
3  ## 1. Copy this file to your project directory
4  ## 2. rename the used ports (in each line, after set_property) according to the top level signal names in the project
5
6  # Clock signals
7  set_property -dict { PACKAGE_PIN W5  IOSTANDARD LVCROS33 } [get_ports clk]
8  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 4} [get_ports clk]
9
10
11 ## Switches
12 set_property -dict { PACKAGE_PIN V17  IOSTANDARD LVCROS33 } [get_ports (ext_n)]
13 set_property -dict { PACKAGE_PIN V16  IOSTANDARD LVCROS33 } [get_ports (ss_n)]
14 set_property -dict { PACKAGE_PIN W16  IOSTANDARD LVCROS33 } [get_ports (HOST1)]
15 set_property -dict { PACKAGE_PIN W17  IOSTANDARD LVCROS33 } [get_ports (HOST2)]
16 set_property -dict { PACKAGE_PIN W15  IOSTANDARD LVCROS33 } [get_ports (swd[4])];
17 set_property -dict { PACKAGE_PIN V15  IOSTANDARD LVCROS33 } [get_ports (swd[5])];
18 set_property -dict { PACKAGE_PIN V14  IOSTANDARD LVCROS33 } [get_ports (sw[0])];
19 set_property -dict { PACKAGE_PIN W13  IOSTANDARD LVCROS33 } [get_ports (sw[1])];
20 set_property -dict { PACKAGE_PIN V2  IOSTANDARD LVCROS33 } [get_ports (sw[2])];
21 set_property -dict { PACKAGE_PIN I3  IOSTANDARD LVCROS33 } [get_ports (sw[3])];
22 set_property -dict { PACKAGE_PIN T7  IOSTANDARD LVCROS33 } [get_ports (sw[4])];
23 set_property -dict { PACKAGE_PIN R5  IOSTANDARD LVCROS33 } [get_ports (sw[5])];
24 set_property -dict { PACKAGE_PIN W2  IOSTANDARD LVCROS33 } [get_ports (sw[6])];
25 set_property -dict { PACKAGE_PIN V3  IOSTANDARD LVCROS33 } [get_ports (sw[7])];
26 set_property -dict { PACKAGE_PIN T3  IOSTANDARD LVCROS33 } [get_ports (sw[8])];
27 set_property -dict { PACKAGE_PIN R2  IOSTANDARD LVCROS33 } [get_ports (sw[9])];
28
29
30 ## LEDs
31 set_property -dict { PACKAGE_PIN V16  IOSTANDARD LVCROS33 } [get_ports {M20[1]}];
32 set_property -dict { PACKAGE_PIN C59  IOSTANDARD LVCROS33 } [get_ports {led[0]}];
33 set_property -dict { PACKAGE_PIN U39  IOSTANDARD LVCROS33 } [get_ports {led[1]}];
34 set_property -dict { PACKAGE_PIN U38  IOSTANDARD LVCROS33 } [get_ports {led[2]}];
35 set_property -dict { PACKAGE_PIN W38  IOSTANDARD LVCROS33 } [get_ports {led[3]}];
36 set_property -dict { PACKAGE_PIN U35  IOSTANDARD LVCROS33 } [get_ports {led[4]}];
37 set_property -dict { PACKAGE_PIN V35  IOSTANDARD LVCROS33 } [get_ports {led[5]}];
38 set_property -dict { PACKAGE_PIN V34  IOSTANDARD LVCROS33 } [get_ports {led[6]}];
39 set_property -dict { PACKAGE_PIN V15  IOSTANDARD LVCROS33 } [get_ports {led[7]}];
40 set_property -dict { PACKAGE_PIN V31  IOSTANDARD LVCROS33 } [get_ports {led[8]}];
41 set_property -dict { PACKAGE_PIN V30  IOSTANDARD LVCROS33 } [get_ports {led[9]}];
42 set_property -dict { PACKAGE_PIN U3  IOSTANDARD LVCROS33 } [get_ports {led[10]}];
43 set_property -dict { PACKAGE_PIN P3  IOSTANDARD LVCROS33 } [get_ports {led[11]}];
44 set_property -dict { PACKAGE_PIN P2  IOSTANDARD LVCROS33 } [get_ports {led[12]}];
45 set_property -dict { PACKAGE_PIN P4  IOSTANDARD LVCROS33 } [get_ports {led[13]}];
46 set_property -dict { PACKAGE_PIN L2  IOSTANDARD LVCROS33 } [get_ports {led[14]}];
47
48
49 ## Segment Display
50 set_property -dict { PACKAGE_PIN V7  IOSTANDARD LVCROS33 } [get_ports {seg[0]}];
51 set_property -dict { PACKAGE_PIN W6  IOSTANDARD LVCROS33 } [get_ports {seg[1]}];
52 set_property -dict { PACKAGE_PIN U8  IOSTANDARD LVCROS33 } [get_ports {seg[2]}];
53 set_property -dict { PACKAGE_PIN V9  IOSTANDARD LVCROS33 } [get_ports {seg[3]}];
54 set_property -dict { PACKAGE_PIN V5  IOSTANDARD LVCROS33 } [get_ports {seg[4]}];
55 set_property -dict { PACKAGE_PIN V6  IOSTANDARD LVCROS33 } [get_ports {seg[5]}];
56 set_property -dict { PACKAGE_PIN U7  IOSTANDARD LVCROS33 } [get_ports {seg[6]}];
57
58 ##set_property -dict { PACKAGE_PIN V7  IOSTANDARD LVCROS33 } [get_ports d0]
59
60
61 set_property -dict { PACKAGE_PIN U2  IOSTANDARD LVCROS33 } [get_ports {an[0]}];
62 set_property -dict { PACKAGE_PIN U4  IOSTANDARD LVCROS33 } [get_ports {an[1]}];
63 set_property -dict { PACKAGE_PIN W6  IOSTANDARD LVCROS33 } [get_ports {an[2]}];
64 set_property -dict { PACKAGE_PIN W4  IOSTANDARD LVCROS33 } [get_ports {an[3]}];
65
66
67 ##Buttons
68 set_property -dict { PACKAGE_PIN U18  IOSTANDARD LVCROS33 } [get_ports rst_n];
69 set_property -dict { PACKAGE_PIN T18  IOSTANDARD LVCROS33 } [get_ports btrn];
70 set_property -dict { PACKAGE_PIN T17  IOSTANDARD LVCROS33 } [get_ports btrn];
71 set_property -dict { PACKAGE_PIN U17  IOSTANDARD LVCROS33 } [get_ports btrn];
72
73
74 ##FPGA Header JA
75 set_property -dict { PACKAGE_PIN J3  IOSTANDARD LVCROS33 } [get_ports {JA[0]}];#sch name = JA3
76 set_property -dict { PACKAGE_PIN L2  IOSTANDARD LVCROS33 } [get_ports {JA[1]}];#sch name = JA2
77 set_property -dict { PACKAGE_PIN J2  IOSTANDARD LVCROS33 } [get_ports {JA[2]}];#sch name = JA3
78 set_property -dict { PACKAGE_PIN P15  IOSTANDARD LVCROS33 } [get_ports {JA[3]}];#sch name = JA4
79 set_property -dict { PACKAGE_PIN M15  IOSTANDARD LVCROS33 } [get_ports {JA[4]}];#sch name = JA5
80 set_property -dict { PACKAGE_PIN K2  IOSTANDARD LVCROS33 } [get_ports {JA[5]}];#sch name = JA6
81 set_property -dict { PACKAGE_PIN H2  IOSTANDARD LVCROS33 } [get_ports {JA[6]}];#sch name = JA9
82 set_property -dict { PACKAGE_PIN G8  IOSTANDARD LVCROS33 } [get_ports {JA[7]}];#sch name = JA10
83
84 ##FPGA Header JB
85 set_property -dict { PACKAGE_PIN A14  IOSTANDARD LVCROS33 } [get_ports {JB[0]}];#sch name = JB1
86 set_property -dict { PACKAGE_PIN A16  IOSTANDARD LVCROS33 } [get_ports {JB[1]}];#sch name = JB2
87 set_property -dict { PACKAGE_PIN B15  IOSTANDARD LVCROS33 } [get_ports {JB[2]}];#sch name = JB3
88 set_property -dict { PACKAGE_PIN B16  IOSTANDARD LVCROS33 } [get_ports {JB[3]}];#sch name = JB4
89 set_property -dict { PACKAGE_PIN A15  IOSTANDARD LVCROS33 } [get_ports {JB[4]}];#sch name = JB5
90 set_property -dict { PACKAGE_PIN A17  IOSTANDARD LVCROS33 } [get_ports {JB[5]}];#sch name = JB6
91 set_property -dict { PACKAGE_PIN C19  IOSTANDARD LVCROS33 } [get_ports {JB[6]}];#sch name = JB7
92 set_property -dict { PACKAGE_PIN C10  IOSTANDARD LVCROS33 } [get_ports {JB[7]}];#sch name = JB8
93
94 ##ramid Header JC
95 set_property -dict { PACKAGE_PIN K17  IOSTANDARD LVCROS33 } [get_ports {JC[0]}];#sch name = JC1
96 set_property -dict { PACKAGE_PIN M18  IOSTANDARD LVCROS33 } [get_ports {JC[1]}];#sch name = JC2
97 set_property -dict { PACKAGE_PIN N17  IOSTANDARD LVCROS33 } [get_ports {JC[2]}];#sch name = JC3
98 set_property -dict { PACKAGE_PIN N16  IOSTANDARD LVCROS33 } [get_ports {JC[3]}];#sch name = JC4
99 set_property -dict { PACKAGE_PIN L17  IOSTANDARD LVCROS33 } [get_ports {JC[4]}];#sch name = JC7
100 set_property -dict { PACKAGE_PIN M19  IOSTANDARD LVCROS33 } [get_ports {JC[5]}];#sch name = JC6
101 set_property -dict { PACKAGE_PIN P12  IOSTANDARD LVCROS33 } [get_ports {JC[6]}];#sch name = JC9
102 set_property -dict { PACKAGE_PIN R18  IOSTANDARD LVCROS33 } [get_ports {JC[7]}];#sch name = JC10
103
104 ##FPGA Header JXADC
105 set_property -dict { PACKAGE_PIN J2  IOSTANDARD LVCROS33 } [get_ports {JXADC[0]}];#sch name = XA1_P
106 set_property -dict { PACKAGE_PIN L3  IOSTANDARD LVCROS33 } [get_ports {JXADC[1]}];#sch name = XA2_P
107 set_property -dict { PACKAGE_PIN H2  IOSTANDARD LVCROS33 } [get_ports {JXADC[2]}];#sch name = XA3_P
108 set_property -dict { PACKAGE_PIN N2  IOSTANDARD LVCROS33 } [get_ports {JXADC[3]}];#sch name = XA4_P
109 set_property -dict { PACKAGE_PIN K3  IOSTANDARD LVCROS33 } [get_ports {JXADC[4]}];#sch name = XA1_N
110 set_property -dict { PACKAGE_PIN M3  IOSTANDARD LVCROS33 } [get_ports {JXADC[5]}];#sch name = XA2_N
111 set_property -dict { PACKAGE_PIN P3  IOSTANDARD LVCROS33 } [get_ports {JXADC[6]}];#sch name = XA3_N
112 set_property -dict { PACKAGE_PIN N1  IOSTANDARD LVCROS33 } [get_ports {JXADC[7]}];#sch name = XA4_N
113
114
115 ##LVGA Connector
116 set_property -dict { PACKAGE_PIN G19  IOSTANDARD LVCROS33 } [get_ports {VidRed[0]}];
117 set_property -dict { PACKAGE_PIN H19  IOSTANDARD LVCROS33 } [get_ports {VidRed[1]}];
118 set_property -dict { PACKAGE_PIN J19  IOSTANDARD LVCROS33 } [get_ports {VidRed[2]}];
119 set_property -dict { PACKAGE_PIN N19  IOSTANDARD LVCROS33 } [get_ports {VidRed[3]}];
120 set_property -dict { PACKAGE_PIN P19  IOSTANDARD LVCROS33 } [get_ports {VidRed[4]}];
121 set_property -dict { PACKAGE_PIN L18  IOSTANDARD LVCROS33 } [get_ports {VgaBlue[1]}];
122 set_property -dict { PACKAGE_PIN K18  IOSTANDARD LVCROS33 } [get_ports {VgaBlue[2]}];
123 set_property -dict { PACKAGE_PIN M18  IOSTANDARD LVCROS33 } [get_ports {VgaBlue[3]}];
124 set_property -dict { PACKAGE_PIN J17  IOSTANDARD LVCROS33 } [get_ports {VgaGreen[1]}];
125 set_property -dict { PACKAGE_PIN H17  IOSTANDARD LVCROS33 } [get_ports {VgaGreen[2]}];
126 set_property -dict { PACKAGE_PIN P17  IOSTANDARD LVCROS33 } [get_ports {VgaGreen[3]}];
127 set_property -dict { PACKAGE_PIN D17  IOSTANDARD LVCROS33 } [get_ports {VgaGreen[4]}];
128 set_property -dict { PACKAGE_PIN P19  IOSTANDARD LVCROS33 } [get_ports {VSync}];
129 set_property -dict { PACKAGE_PIN R19  IOSTANDARD LVCROS33 } [get_ports {VSync}]
130
131
132 ##PS2/RS232 Serial
133 set_property -dict { PACKAGE_PIN B18  IOSTANDARD LVCROS33 } [get_ports RsRx];
134 set_property -dict { PACKAGE_PIN A18  IOSTANDARD LVCROS33 } [get_ports RsTx];
135
136
137 ##USB HID (PS/2)
138 set_property -dict { PACKAGE_PIN C17  IOSTANDARD LVCROS33 } [get_ports PS2C1K];
139 set_property -dict { PACKAGE_PIN R17  IOSTANDARD LVCROS33 } [get_ports PS2Data];
140
141
142 ##Quad SPI Flash
143 ##Note that CCLK 0 cannot be placed in 7 series devices. You can access it using the
144 ##STARTUP2 primitive.
145 set_property -dict { PACKAGE_PIN D18  IOSTANDARD LVCROS33 } [get_ports {QspiIO[0]}];
146 set_property -dict { PACKAGE_PIN D19  IOSTANDARD LVCROS33 } [get_ports {QspiIO[1]}];
147 set_property -dict { PACKAGE_PIN G18  IOSTANDARD LVCROS33 } [get_ports {QspiIO[2]}];
148 set_property -dict { PACKAGE_PIN F18  IOSTANDARD LVCROS33 } [get_ports {QspiIO[3]}];
149 set_property -dict { PACKAGE_PIN K18  IOSTANDARD LVCROS33 } [get_ports {QspiIO[4]}];
150
151
152 ##Configuration options, can be used for all designs
153 set_property CONFIG_VOLTAGE 3.3 [current_design];
154 set_property CFGWVS VCCO [current_design];
155
156 ## SPI configuration mode options for QSPI boot, can be used for all designs
157 set_property BTSTREAM_GENERAL_COMPRESS TRUE [current_design];
158 set_property BTSTREAM_CONFIG_CONFIGURE_3S [current_design];
159 set_property CONFIG_MODE_S2PM [current_design];

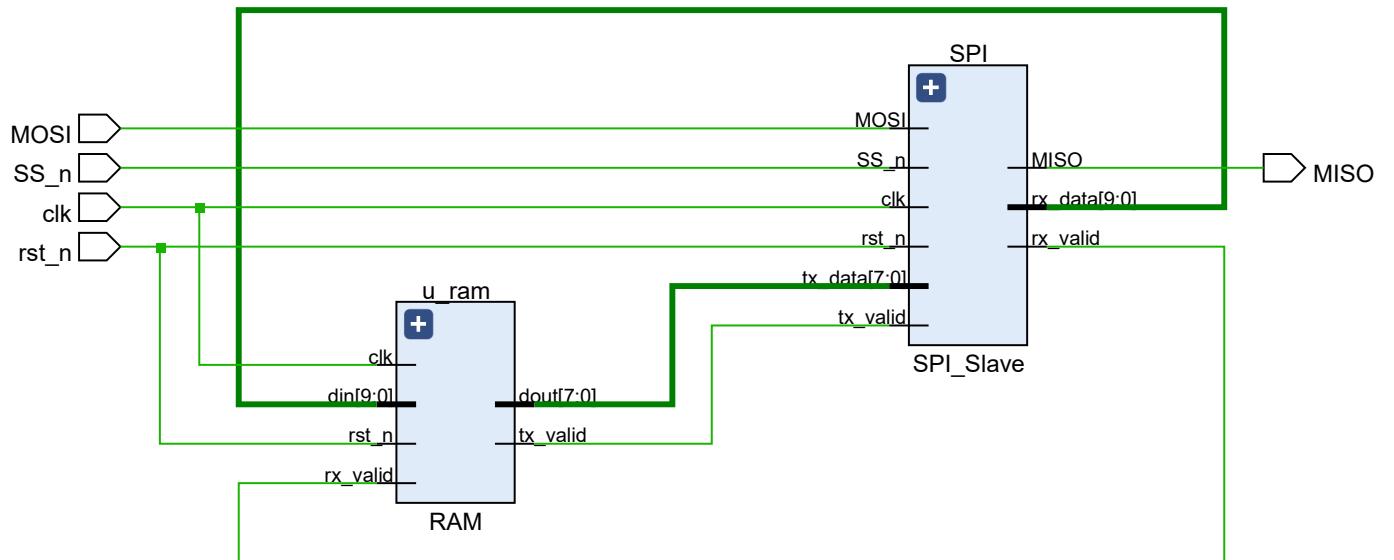
```

Elaboration (gray encoding)

Message



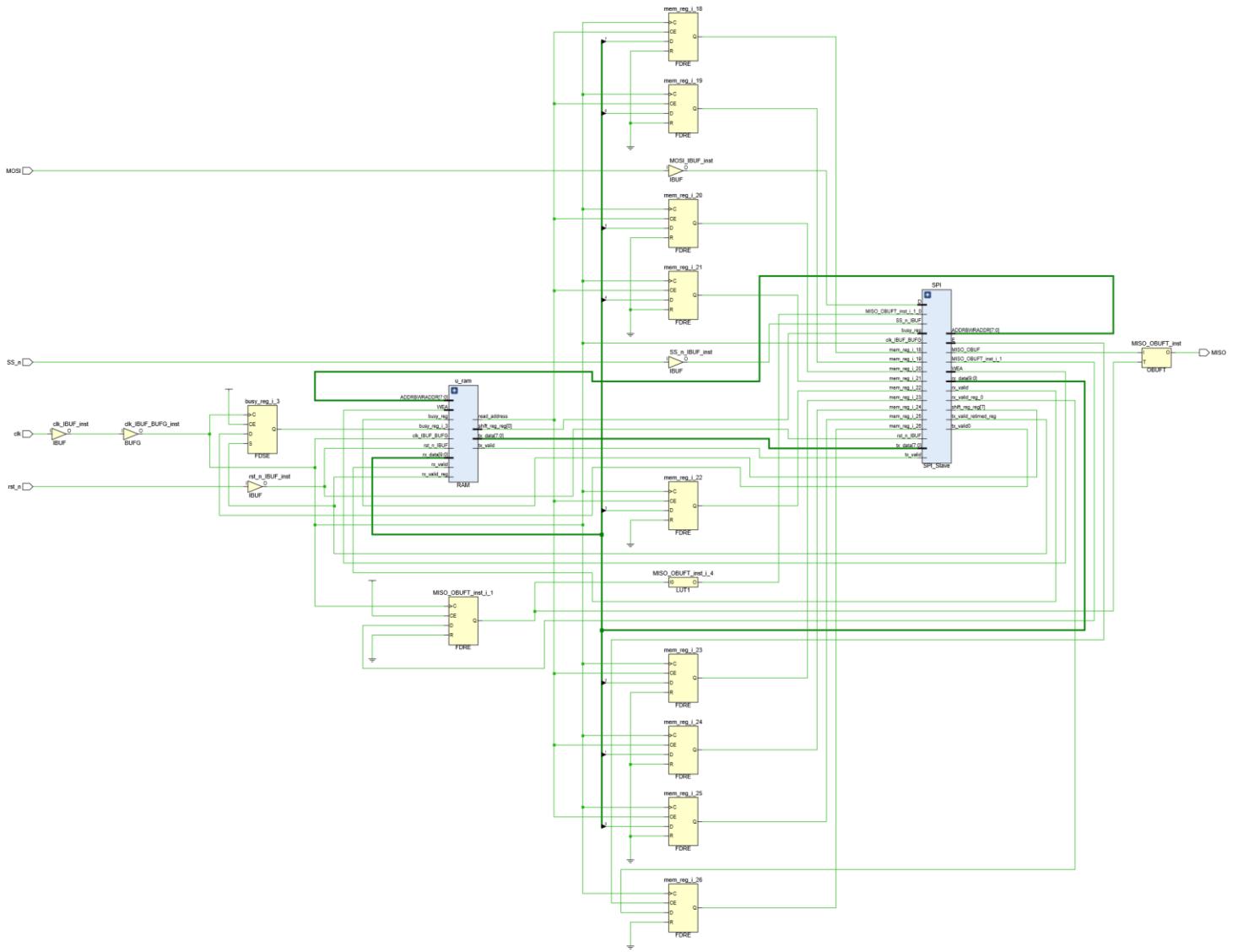
Schematic



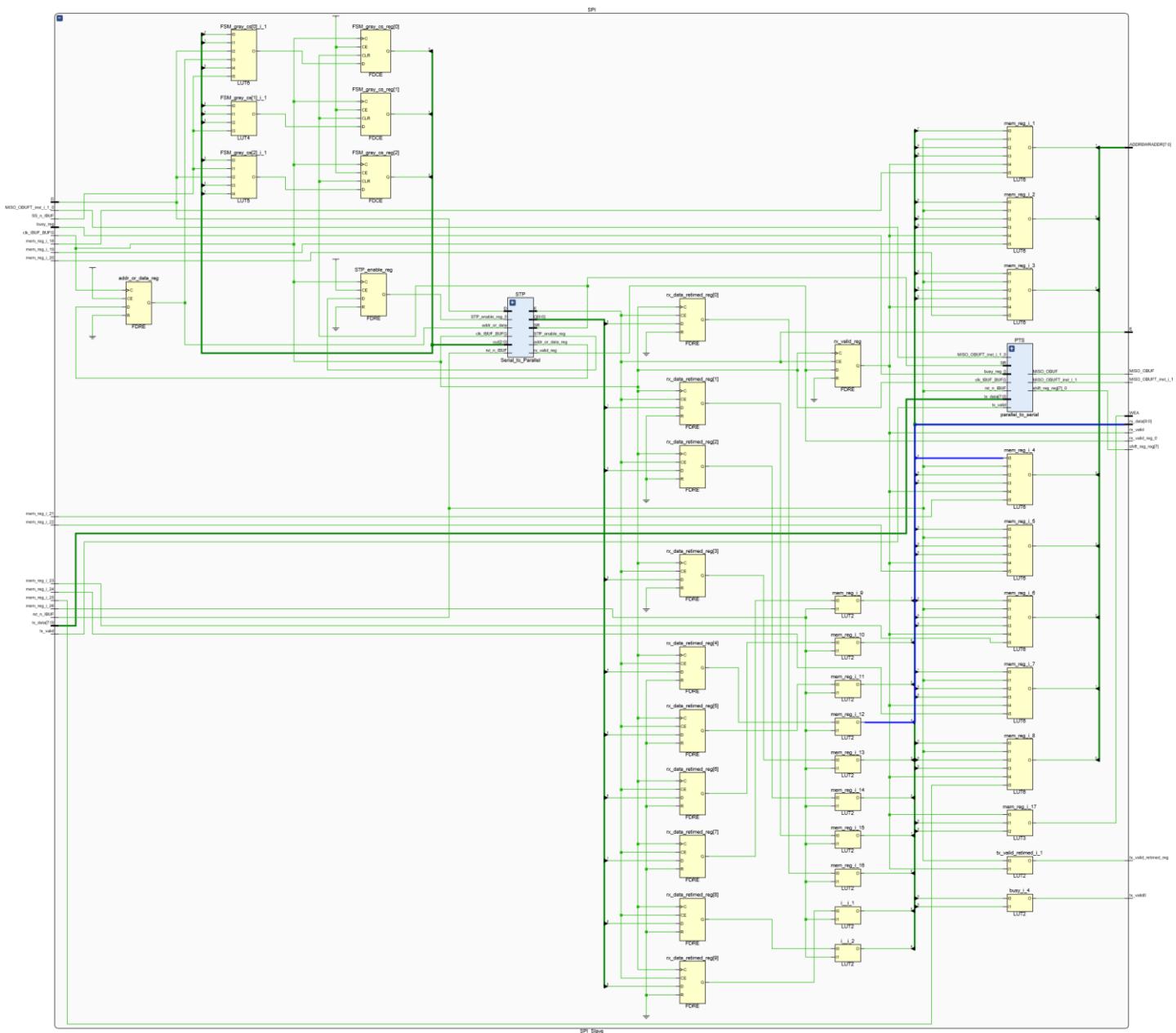
Synthesis (gray encoding)

State	Encoding
IDLE	5'b000
CHK_CMD	5'b001
WRITE	5'b011
READ_ADD	5'b010
READ_DATA	5'b110

Schematic



SPI_Slave Schematic



Message

Tcl Console Messages Log Reports Design Runs Debug

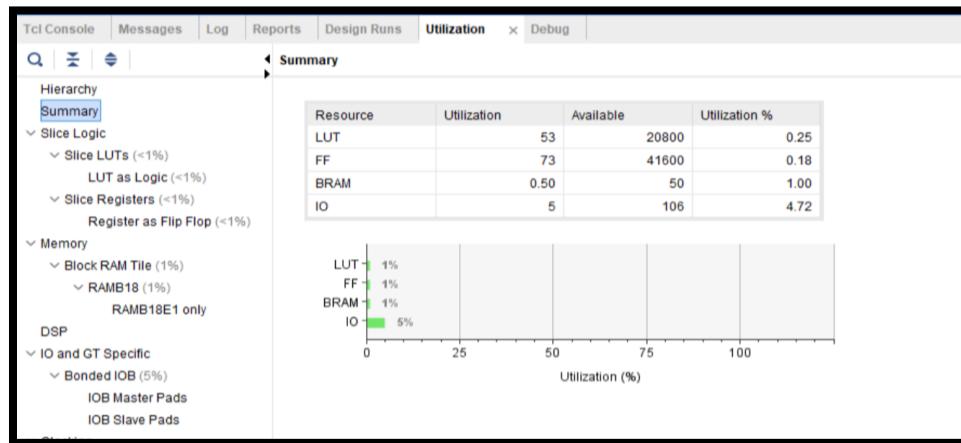
Q | X | D | Y | M | B | Show All

Warning (3) Info (65) Status (52)

▼ Synthesis (3 warnings, 54 infos)

- [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35t'
- [Synth 8-6157] synthesizing module 'SPI_Wrapper' [SPI_Wrapper.v:1] (4 more like this)
- [Synth 8-5534] Detected attribute (* fsm_encoding = "gray"*) [SPI_Slave.v:20]
- [Synth 8-6155] done synthesizing module 'Serial_to_Parallel' (1#1) [Serial_to_Parallel.v:1] (4 more like this)
- [Synth 8-567] referenced signal 'addr_or_data' should be on the sensitivity list [SPI_Slave.v:33]
- [Synth 8-226] default block is never used [RAM.v:24]
- [Device 21-403] Loading part xc7a35ticpg236-1L
- [Project 1-236] Implementation specific constraints were found while reading constraint file /F/digital_design/PROJECTS/PROJECT_1/Constraints.basys3.xdc. These constraints will be ignored for synthesis.

Utilization Report



Timing Report

Tcl Console Messages Log Reports Design Runs Timing Utilization Debug

Q | X | D | C | B | Design Timing Summary

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (4)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths

Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.183 ns	Worst Hold Slack (WHS): 0.144 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 168	Total Number of Endpoints: 168	Total Number of Endpoints: 76

All user specified timing constraints are met.

Activate Window Go to Settings

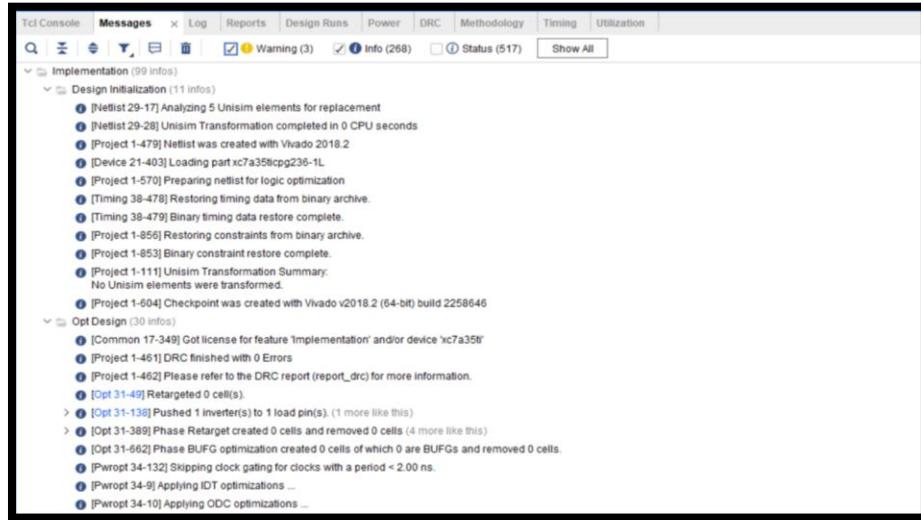
Timing Summary - timing_156

Implementation (gray encoding)

Device



Message

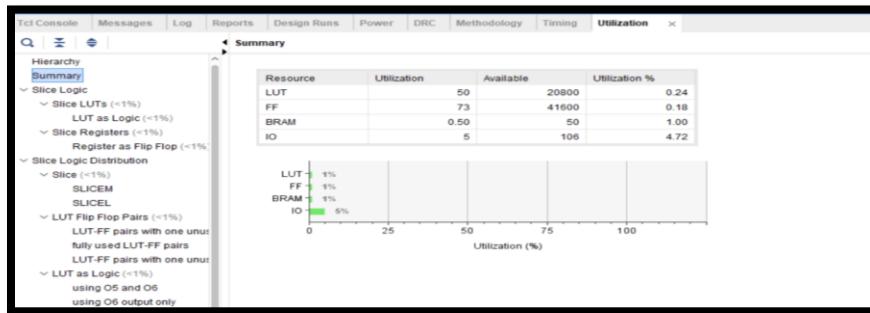


The screenshot shows the 'Messages' tab in the Vivado interface. The 'Implementation' section contains 99 info messages. Key messages include:

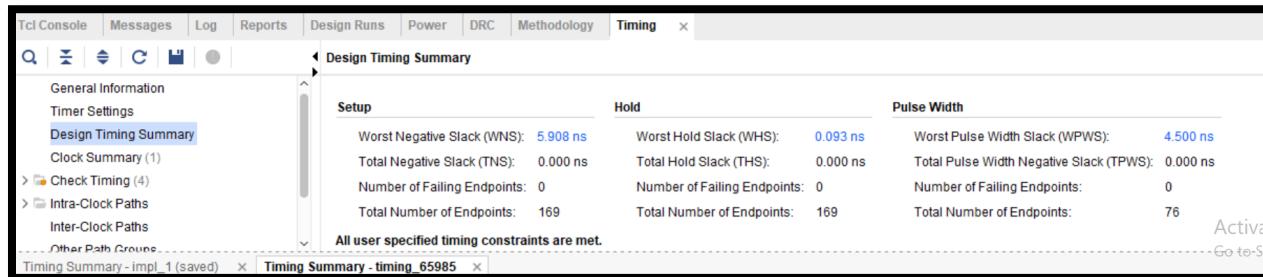
- [Netlist 29-17] Analyzing 5 Unisim elements for replacement
- [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
- [Project 1-479] Netlist was created with Vivado 2018.2
- [Device 21-403] Loading part xc7a35tcpg236-1L
- [Project 1-570] Preparing netlist for logic optimization
- [Timing 38-478] Restoring timing data from binary archive.
- [Timing 38-479] Binary timing data restore complete.
- [Project 1-856] Restoring constraints from binary archive.
- [Project 1-853] Binary constraint restore complete.
- [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.
- [Project 1-604] Checkpoint was created with Vivado v2018.2 (64-bit) build 2258646

The 'Opt Design' section contains 30 info messages, including DRC finished with 0 errors and various optimization reports.

Utilization Report



Timing Report

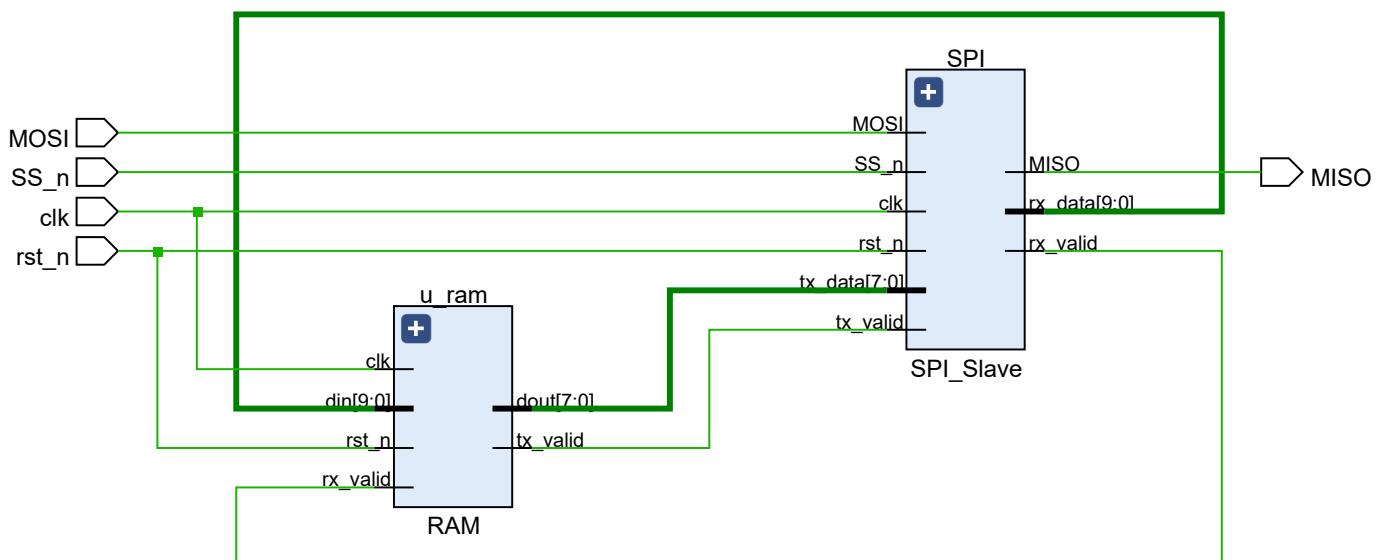


Elaboration (one_hot encoding)

Message



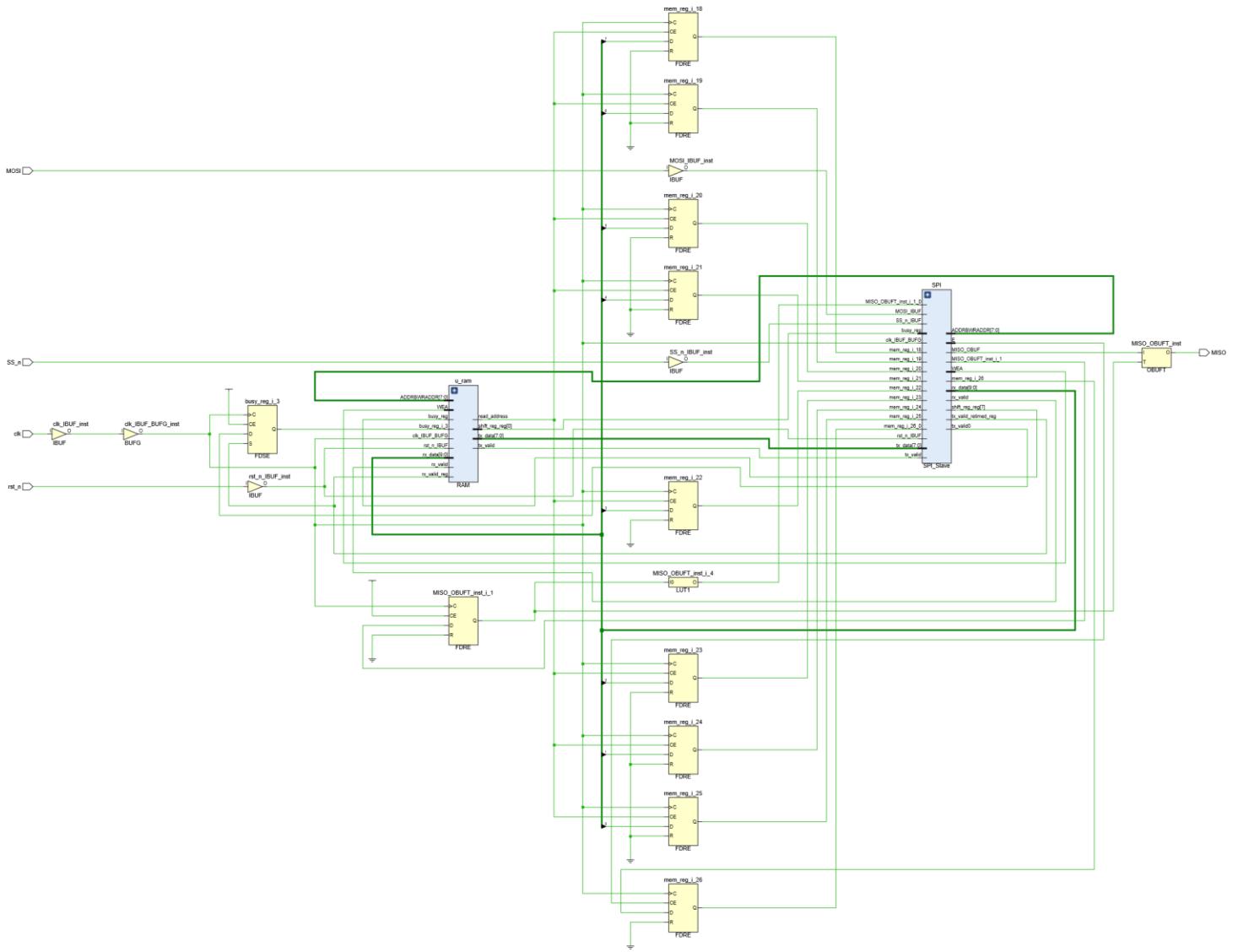
Schematic



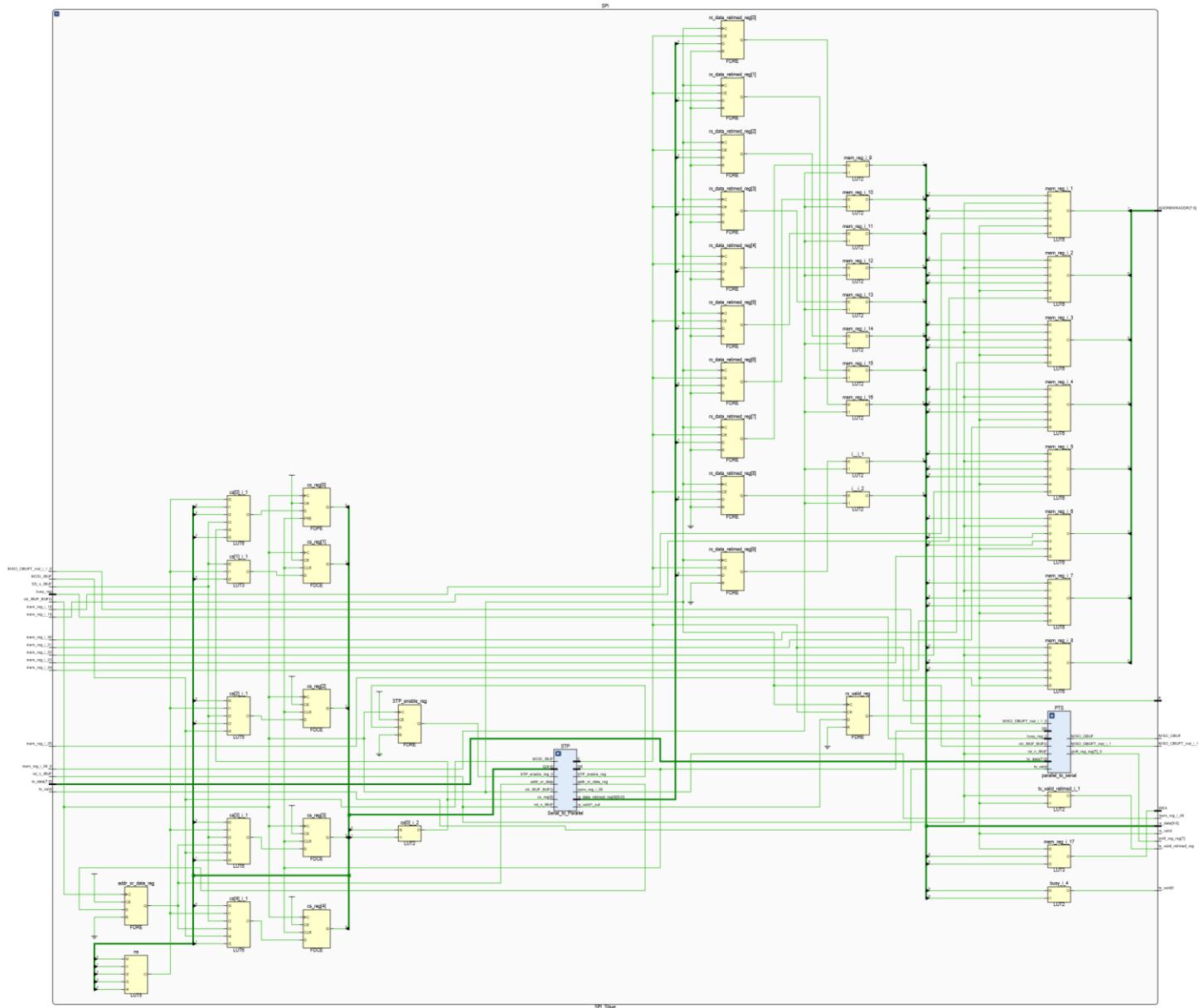
Synthesis (one_hot encoding)

State	Encoding
IDLE	5'b00001
CHK_CMD	5'b00010
WRITE	5'b00100
READ_ADD	5'b01000
READ_DATA	5'b10000

Schematic



SPI_Slave Schematic



Message

Tcl Console Messages Log Reports Design Runs Debug

Q | X | D | T | M | B | Warning (4) Info (79) Status (58) Show All

✓ Synthesis (3 warnings, 55 infos)

- ⓘ [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a35t'
- ⓘ [Synth 8-6157] synthesizing module 'SPI_Wrapper' [SPI_Wrapper.v1] (4 more like this)
- ⓘ [Synth 8-5534] Detected attribute (* fsm_encoding = "one_hot") [SPI_Slave.v20]
- ⓘ [Synth 8-6155] done synthesizing module 'Serial_to_Parallel' (#1) [Serial_to_Parallel.v1] (4 more like this)
- ⚠ [Synth 8-567] referenced signal 'addr_or_data' should be on the sensitivity list [SPI_Slave.v33]
- ⓘ [Synth 8-226] default block is never used [RAM.v24]
- ⓘ [Device 21-403] Loading part xc7a35ticpg236-1L
- ⓘ [Project 1-236] Implementation specific constraints were found while reading constraint file [E:/digital_design/PROJECTS/PROJECT_1/Constraints.hsv3.yo1]. These constraints will be ignored.

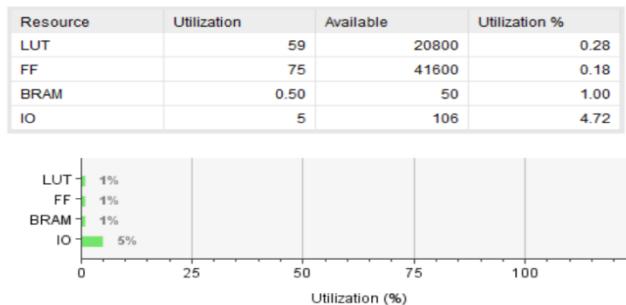
Utilization Report

Tcl Console Messages Log Reports Design Runs Utilization Debug

Q | X | D | T | M | B | Summary

Hierarchy

- ✓ Summary
- ✓ Slice Logic
 - ✓ Slice LUTs (<1%)
 - LUT as Logic (<1%)
 - ✓ Slice Registers (<1%)
 - Register as Flip Flop (<1%)
- ✓ Memory
 - ✓ Block RAM Tile (1%)
 - ✓ RAMB18 (1%)
 - RAMB18E1 only
- DSP
- ✓ IO and GT Specific
 - ✓ Bonded IOB (5%)
 - IOB Master Pads
 - IOB Slave Pads



Timing Report

Tcl Console Messages Log Reports Design Runs Timing Utilization Debug

Q | X | D | C | B | S | Design Timing Summary

General Information

Timer Settings

Design Timing Summary

Clock Summary (1)

> Check Timing (4)

> Intra-Clock Paths

Inter-Clock Paths

Other Path Groups

User Ignored Paths

> Unconstrained Paths

Timing Summary - timing_98

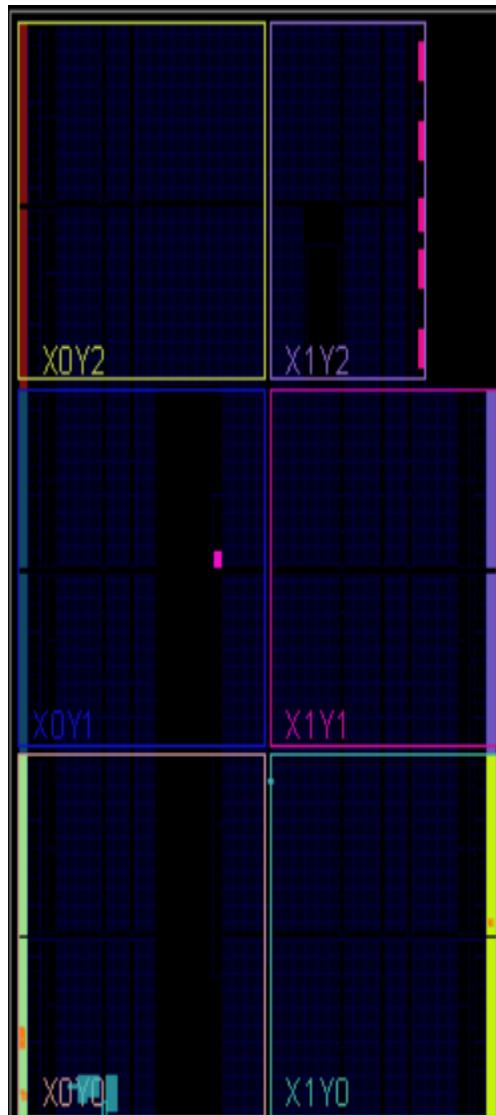
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 6.183 ns	Worst Hold Slack (WHS): 0.146 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 170	Total Number of Endpoints: 170	Total Number of Endpoints: 78

All user specified timing constraints are met.

Activate Windows
Go to Settings to activate Windows

Implementation (one_hot encoding)

Device

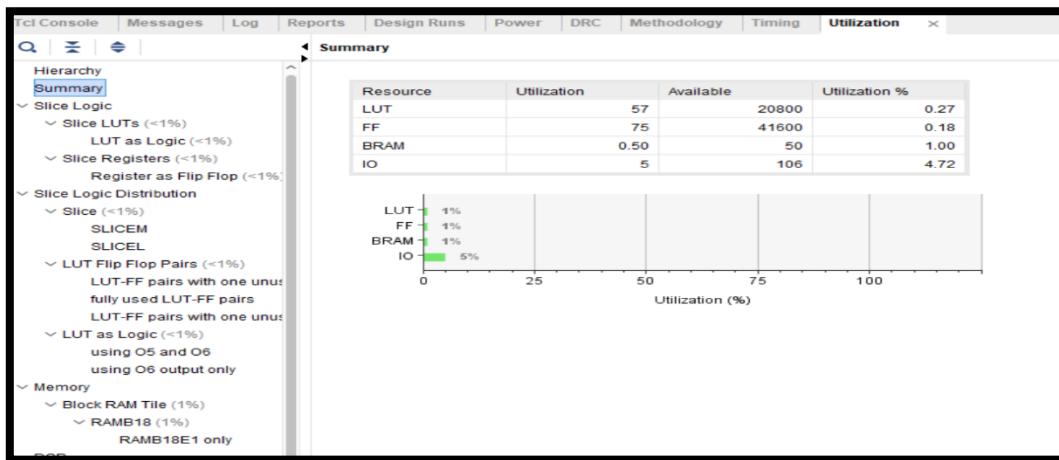


Message

The screenshot shows the Vivado Tcl Console window with the 'Messages' tab selected. The 'Implementation' section contains 11 warning messages related to netlist analysis and device loading:

- [Netlist 29-17] Analyzing 5 Unisim elements for replacement
- [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
- [Project 1-479] Netlist was created with Vivado 2018.2
- [Device 21-403] Loading part xc7a35tciapg236-1L
- [Project 1-570] Preparing netlist for logic optimization
- [Timing 38-478] Restoring timing data from binary archive.
- [Timing 38-479] Binary timing data restore complete.

Utilization Report



Timing Report

