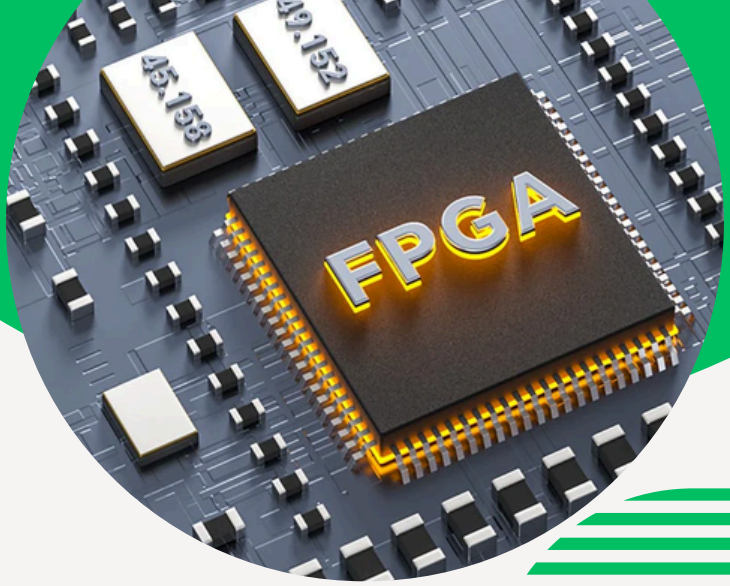


SPI Slave With Single Port RAM

PREPARED BY

Ashraf Ahmed

Table of

Content

01 Project Summary

02 Block Diagram

03 FSM state diagram

04 RTL Code

05 Testbench Code

06 Do File

07 Simulation

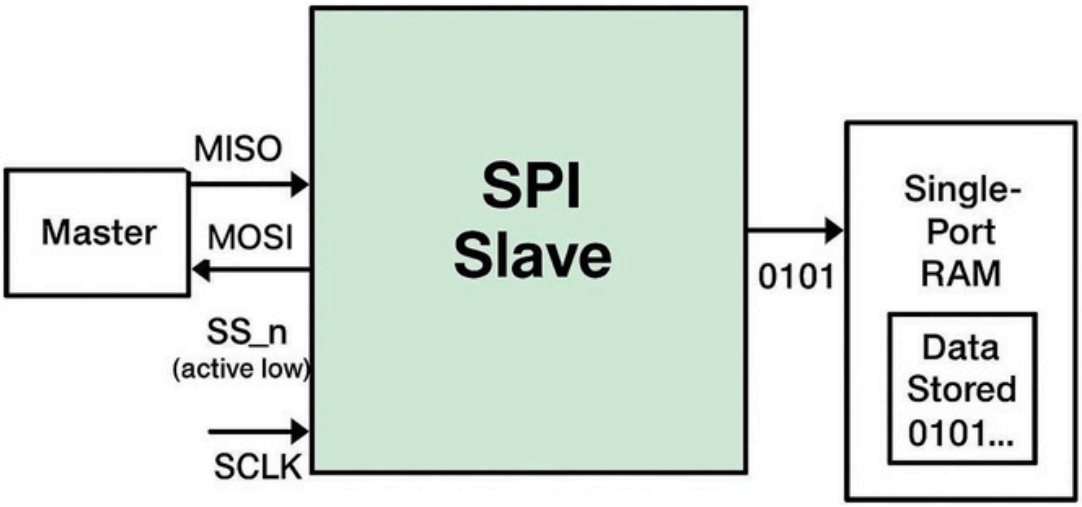
08 QustaLint

09 Constraints File

10 Vivado

Project

Summary



Integration of SPI Slave with Single-Port RAM

- ✅ Supports full-duplex communication via SPI protocol
- ✅ Reliable read and write operations with FSM control

Operation:

1. Master sends command
2. Address captured
3. Data written or read from RAM
4. Data exchanged via MOSI/MISO

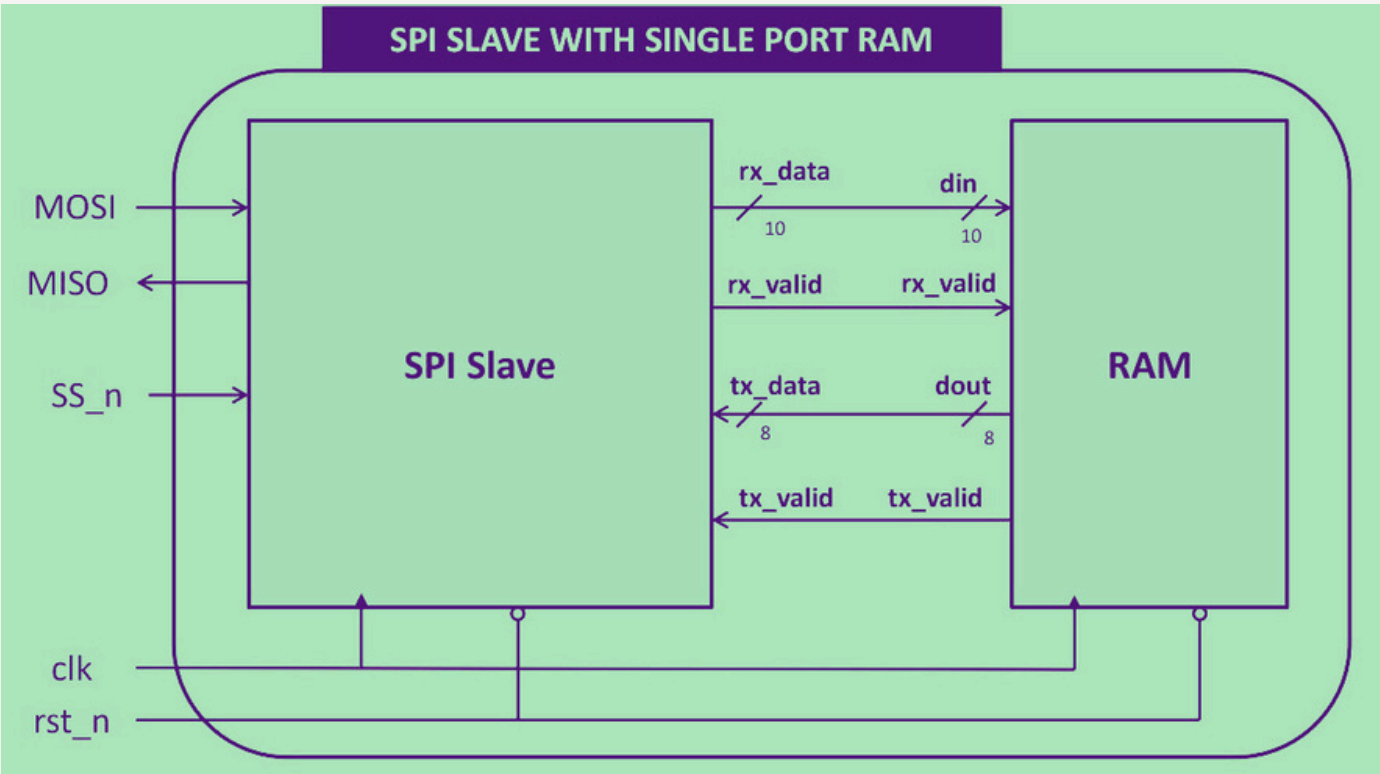
Applications:

- Embedded systems
- Sensors
- Memory chips
- Microcontroller-based devices

Importance:

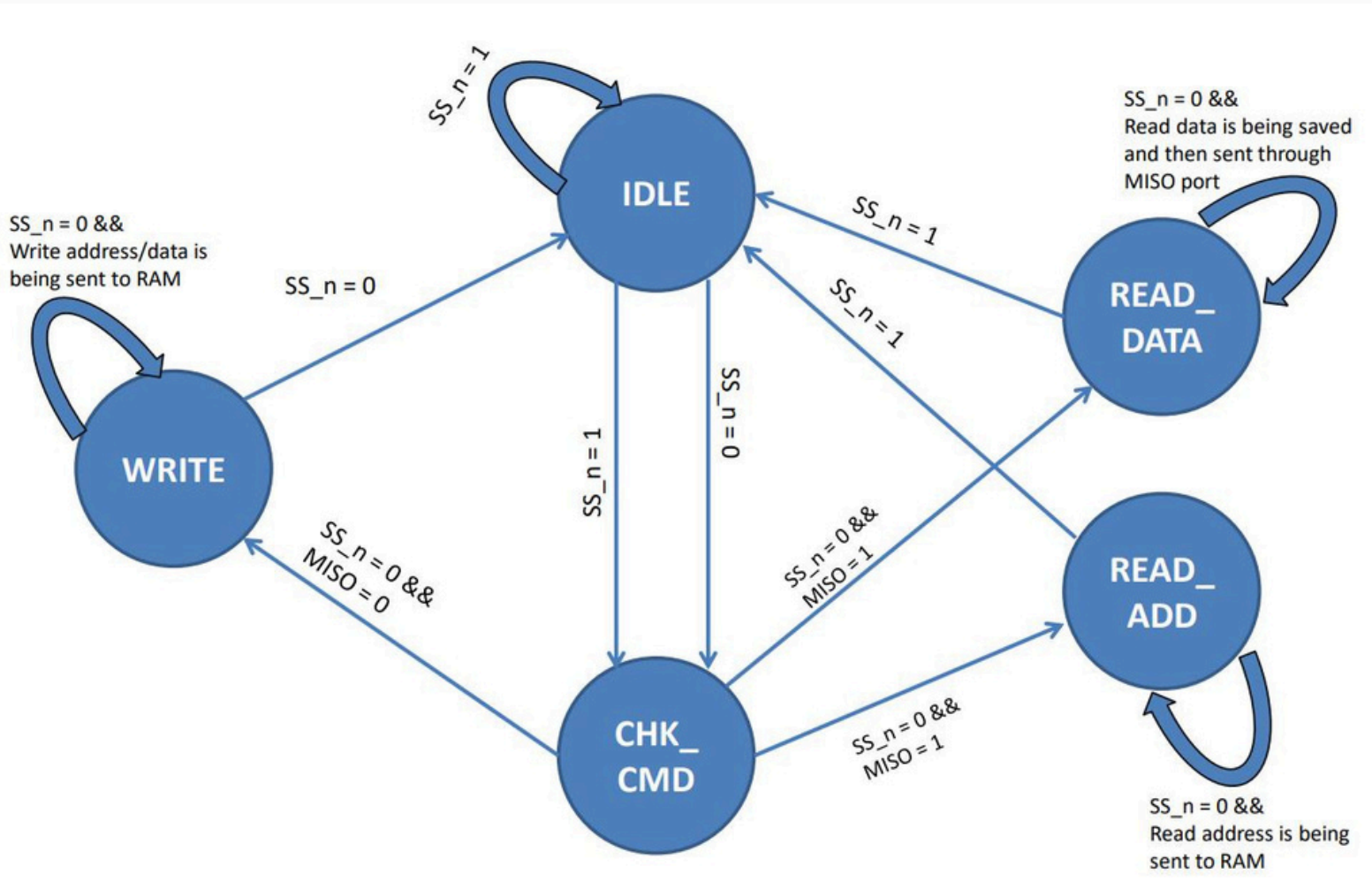
➡ Enables fast, short-distance data transfer with reliable storage and retrieval

Block Diagram



- **SPI SLAVE : Interfaces with MOSI, MISO, SCLK, SS_n**
- **FSM CONTROLLER : Decodes commands and manages operations**
- **SINGLE-PORT RAM : Stores and retrieves data efficiently**
- **MODULAR DESIGN : Ensures smooth communication and reliable memory access**

FSM state diagram



- **IDLE :** Waits for master to start communication
- **CHK_CMD :** Detects read or write command
- **WRITE_ADDR / WRITE_DATA :** Capture address and write data into RAM
- **READ_ADDR / READ_DATA :** Capture address and send data to master
- **FSM ROLE :** Ensures proper sequencing, synchronization and error-free operation

RTL Code

Sub Module SPI Slave (Instantiated in the main design)

```
module SPI_Slave (MOSI , MISO , SS_n , clk , rst_n , rx_data
 , rx_valid , tx_data , tx_valid );
// define parameters to used in states
parameter   IDLE     = 3'b000 ;
parameter  CHK_CMD   = 3'b001 ;
parameter  WRITE     = 3'b010 ;
parameter  READ_ADD  = 3'b011 ;
parameter  READ_DATA = 3'b100 ;
// define all ports
input  clk , rst_n , SS_n ;
input  MOSI , tx_valid ;
input  [7:0] tx_data ;
output reg MISO , rx_valid ;
output reg [9:0] rx_data ;
// define internal signal in the design
  (*fsm_encoding = "sequential"*)
reg [2:0] cs , ns ;
reg read_addr_recieved ;
reg [3:0] counter ;
// next state Logic
always @(*) begin
    case (cs)
       IDLE    :    if (SS_n == 1'b0 ) begin
                         ns = CHK_CMD ;
                    end
                    else begin
                         ns = IDLE ;
                    end

       CHK_CMD :    if (SS_n) begin
                          ns = IDLE ;
                    end
                    else begin
                        if (~MOSI) begin
                            ns = WRITE ;
                        end
                        else begin
                            if (~read_addr_recieved ) begin
                                ns = READ_ADD ;
                            end
                            else begin
                                ns = READ_DATA ;
                            end
                        end
                    end

       WRITE   :    if (SS_n) begin
                         ns = IDLE ;
                    end
                    else begin
                         ns = WRITE ;
                    end

       READ_ADD:    if (SS_n) begin
                         ns = IDLE ;
                    end
                    else begin
                         ns = READ_ADD ;
                    end

       READ_DATA :  if (SS_n) begin
                         ns = IDLE ;
                    end
                    else begin
                         ns = READ_DATA ;
                    end
    endcase
end
```

RTL Code

Sub Module SPI Slave (Instantiated in the main design)

```
// state memory
always @(posedge clk ) begin
      if (~ rst_n) begin
            cs <= IDLE ;
      end
      else begin
            cs <= ns ;
      end
end
// output Logic
always @(posedge clk ) begin
    if (~rst_n) begin
        rx_data <= 0 ;
        rx_valid <= 0 ;
        counter <= 0 ;
        MISO <= 0 ;
        read_addr_recieved <= 0 ;
    end
    else begin
    case (cs)
       IDLE    :   begin
                        counter <= 0 ;
                        MISO <= 0 ;
                        rx_valid <= 0 ;
                    end
       CHK_CMD :   begin
                        counter <= 0 ;
                        rx_valid <= 0  ;
                        MISO <= 0 ;
                    end
       WRITE   :   begin
                       if (counter <= 9 ) begin
                       rx_data <= {rx_data[8:0] , MOSI }  ;
                       counter <= counter + 1 ;
                       rx_valid <= 0 ;
                       end
                       if (counter >= 9) begin
                          rx_valid <= 1 ;
                       end
                    end
       READ_ADD :  begin
                         if (counter <= 9 ) begin
                         rx_data <= {rx_data[8:0] , MOSI } ;
                         counter <= counter + 1 ;
                         read_addr_recieved <= 1 ;
                         rx_valid <=  0 ;
                    end
                    if (counter >= 9 ) begin
                         rx_valid <= 1 ;
                    end
                    end
      READ_DATA :  begin
                        if (counter >= 3 && rx_valid == 1'b1)
begin
                            MISO <= tx_data [counter-3 ] ;
                            counter <= counter - 1 ;
                        end
                        else begin
                        if (counter <= 9 ) begin
                            rx_data <= {rx_data[8:0] , MOSI
 } ;
                            counter <= counter + 1 ;
                        end
                        if (counter >= 9) begin
                            rx_valid <= 1 ;
                            read_addr_recieved <= 0 ;
                        end
                        end
                  end
    endcase
    end
end
endmodule
```

RTL Code

Synchronous Single Port RAM (Instantiated in the main design)

```
module spr_sync (din , rx_valid , dout , tx_valid
 , clk , rst_n );
// define all ports
parameter MEM_DEPTH = 256 ;
parameter ADDR_SIZE = 8 ;
input [9:0] din ;
input clk , rst_n , rx_valid  ;
output reg [7:0] dout ;
output reg tx_valid ;
// create internal signal (RAM)
reg [ADDR_SIZE-1 : 0] mem [MEM_DEPTH-1:0] ;
reg [ADDR_SIZE-1 : 0] addr ;
// design using Behavioral Moduling
always @(posedge clk) begin
      if (~rst_n) begin
           dout <= 8'd0 ;
           tx_valid <= 1'b0 ;
      end
      else begin
           if (rx_valid) begin
                 case (din[9:8])
                     2'b00 :  addr <= din[7:0
] ;        // write address
                     2'b01 :  mem[addr] <= din[7:0
] ;   // write data
                     2'b10 :  addr <= din[7:0
] ;        // read address
                     2'b11 :  begin
                                 dout <= mem[addr
] ;       // read data
                                 tx_valid <= 1'b1
 ;
                              end
                 endcase
           end
      end
end
endmodule
```

RTL Code

SPI_Wrapper (Top Module)

```
module SPI_Wrapper (MOSI , MISO , SS_n , clk ,
rst_n);
// define all ports
input   clk , rst_n ;
input   MOSI , SS_n ;
output  MISO ;
// define internal signals
wire [9:0] rx_data ;
wire rx_valid , tx_valid ;
wire [7:0] tx_data ;
// instantiate the first design (SPI Slave)
SPI_Slave Block_1 (.clk(clk) , .SS_n(SS_n) , .
rst_n(rst_n) , .MISO(MISO) , .MOSI(MOSI) , .
rx_data(rx_data)
, .rx_valid(rx_valid) , .tx_data(tx_data) , .
tx_valid(tx_valid) ) ;
// instantiate the secound design (RAM)
spr_sync  Block_2 (.clk(clk) , .rst_n(rst_n) , .
din(rx_data) , .rx_valid(rx_valid) , .dout(tx_data
) , .tx_valid(tx_valid)) ;

endmodule
```

Testbench Code

```
module SPI_Wrapper_tb ();
reg    clk , rst_n ;
reg    MOSI , SS_n ;
wire   MISO ;
// instantiate the design to  under test
SPI_Wrapper DUT (.clk(clk) , .rst_n(rst_n) , .MOSI
(MOSI) , .SS_n(SS_n) , .MISO(MISO)) ;
// Clock Generation
initial begin
     clk = 0 ;
     forever begin
        #1  clk = ~ clk ;
     end
end
// Stimulus Generation
initial begin
    // Load initial Value in RAM
    $readmemb ("mem.dat" , DUT.Block_2.mem);
    $display ("Start Simulation ");
    // check reset
    rst_n = 1'b0 ;
    MOSI =  1'b0 ;
    SS_n = 1'b0 ;
    @(negedge clk ) ;
    rst_n = 1'b1 ;

//-----------------------------------------------
----------------------//

//---------------------------- Write Address-----
----------------------//
    repeat (2)  @(negedge clk) ;

//-----------sent 2'b00 to rx_data[9:8] bits  to s
elect the Write address ------///
    MOSI = 0 ;
    repeat (2) @(negedge clk ) ;
    //---------- sent the address ------ //
    MOSI = 1'b1 ;
    repeat (7)  @(negedge clk ) ;
    MOSI = 1'b0 ;
    @(negedge clk ) ;
    //-----------------------------------//
    SS_n = 1'b1 ;
    repeat (2)  @(negedge clk) ;
    $display("Write address Operation Sucessfully"
);
```

Testbench Code

```
//---------------------------- Write Data--------
----------------------//
    SS_n = 1'b0 ;
    @(negedge clk ) ;
    SS_n = 1'b0 ;
    MOSI = 1'b0 ;
    @(negedge clk ) ;
    MOSI = 1'b0 ;
    @(negedge clk ) ;
    MOSI = 1'b1 ;
    @(negedge clk ) ;
    repeat (8) begin
      MOSI = $random ;
      @(negedge clk ) ;
    end
    SS_n = 1'b1 ;
    repeat (2) @(negedge clk) ;
    $display("Write data Operation Sucessfully");

//-----------------------------------------------
------------------------//

//---------------------------- Read Address -----
------------------------//
    SS_n = 1'b0 ;
    @(negedge clk ) ;
    MOSI = 1'b1 ;
    @(negedge clk ) ;
    MOSI = 1'b1 ;
    @(negedge clk ) ;
    MOSI = 1'b0 ;
    @(negedge clk ) ;
    //---------- sent the address ------ //
    MOSI = 1'b1 ;
    repeat (7)  @(negedge clk ) ;
    MOSI = 1'b0 ;
    @(negedge clk ) ;
    //----------------------------------//
    SS_n = 1'b1 ;
    repeat (2) @(negedge clk ) ;
    $display("Read address Operation Sucessfully"
);

//-----------------------------------------------
------------------------//

//---------------------------- Read Data  -------
------------------------//
    SS_n = 1'b0 ;
    @(negedge clk) ;
    MOSI = 1'b1 ;
    @(negedge clk ) ;
    repeat (20) begin
     @(negedge clk ) ;
    end
    SS_n = 1 ;
    repeat (2) @(negedge clk ) ;
    $display("Read Data Operation Sucessfully");
    $display("The design is done successfully") ;
    $stop ;
end
// Generate monitor
initial begin
   $monitor ("MOSI= %b , MISO =%b , SS_n =%b
 , counter=%d" , MOSI , MISO , SS_n , DUT.Block_1.
counter ) ;
end
endmodule
```

Do File

run_SPI_Wrapper.do

```
vlib work
vlog SPI_Wrapper.v SPI_Wrapper_tb.v
vsim -voptargs=+acc work.SPI_Wrapper_tb
add wave *
add wave -position insertpoint  \
sim:/SPI_Wrapper_tb/DUT/rx_data \
sim:/SPI_Wrapper_tb/DUT/rx_valid \
sim:/SPI_Wrapper_tb/DUT/tx_data \
sim:/SPI_Wrapper_tb/DUT/tx_valid
add wave -position insertpoint  \
sim:/SPI_Wrapper_tb/DUT/Block_1/counter
add wave -position insertpoint  \
sim:/SPI_Wrapper_tb/DUT/Block_1/cs \
sim:/SPI_Wrapper_tb/DUT/Block_1/ns \
sim:/SPI_Wrapper_tb/DUT/Block_1/read_addr_recieved
add wave -position insertpoint  \
sim:/SPI_Wrapper_tb/DUT/Block_2/addr \
sim:/SPI_Wrapper_tb/DUT/Block_2/mem
run -all
#quit -sim
```

Transcript

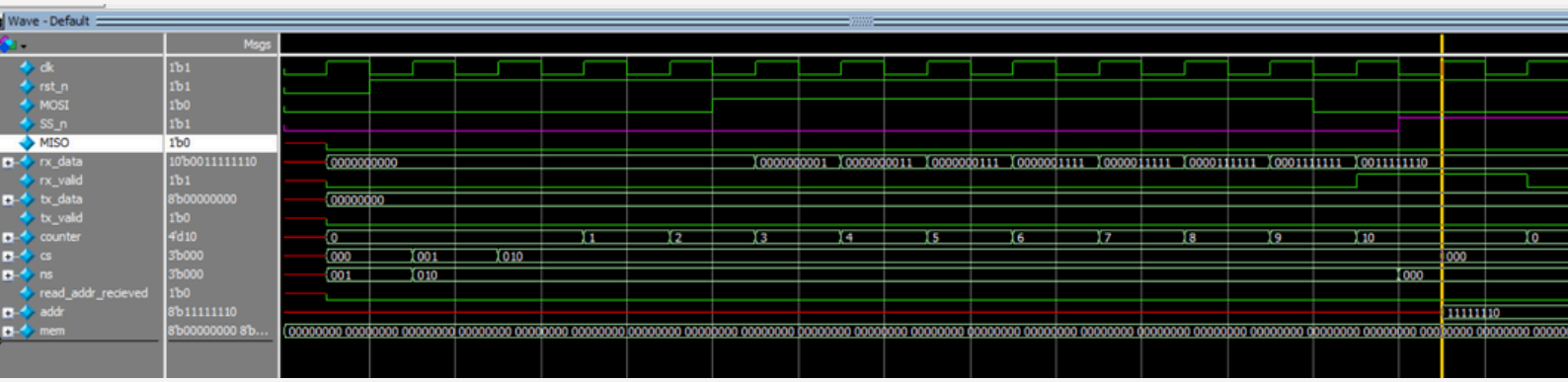
```
# Start Simulation
# MOSI= 0 , MISO =x , SS_n =0 , counter= x
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 0
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 1
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 2
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 2
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 3
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 4
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 5
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 6
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 7
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 8
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 9
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 9
# MOSI= 0 , MISO =0 , SS_n =0 , counter=10
# MOSI= 0 , MISO =0 , SS_n =1 , counter=10
# MOSI= 0 , MISO =0 , SS_n =1 , counter= 0
# Write address Operation Sucessfully
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 0
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 1
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 1
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 2
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 2
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 3
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 3
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 4
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 5
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 6
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 7
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 8
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 9
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 9
# MOSI= 0 , MISO =0 , SS_n =0 , counter=10
# MOSI= 0 , MISO =0 , SS_n =1 , counter=10
# MOSI= 0 , MISO =0 , SS_n =1 , counter= 0
```

Transcript

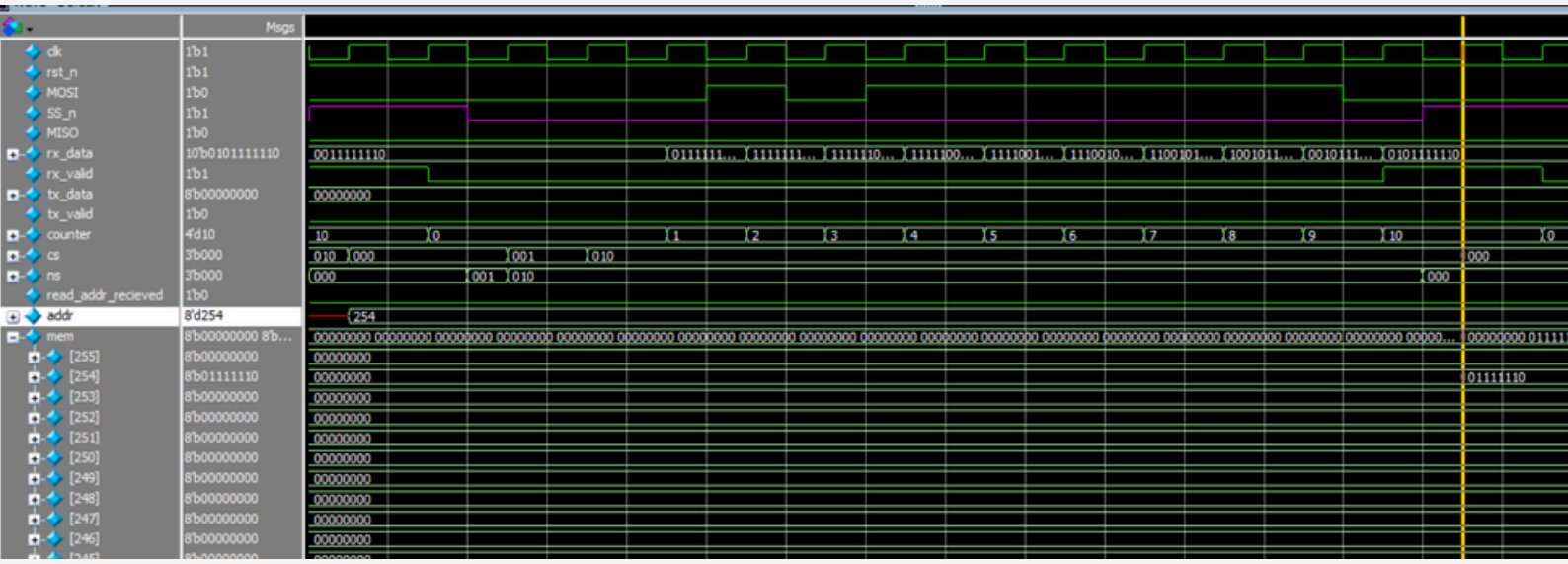
```
# Write data Operation Sucessfully
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 0
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 0
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 1
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 1
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 2
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 2
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 3
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 4
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 5
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 6
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 7
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 8
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 9
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 9
# MOSI= 0 , MISO =0 , SS_n =0 , counter=10
# MOSI= 0 , MISO =0 , SS_n =1 , counter=10
# MOSI= 0 , MISO =0 , SS_n =1 , counter= 0
# Read address Operation Sucessfully
# MOSI= 0 , MISO =0 , SS_n =0 , counter= 0
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 0
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 1
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 2
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 3
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 4
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 5
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 6
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 7
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 8
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 9
# MOSI= 1 , MISO =0 , SS_n =0 , counter=10
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 9
# MOSI= 1 , MISO =1 , SS_n =0 , counter= 8
# MOSI= 1 , MISO =1 , SS_n =0 , counter= 7
# MOSI= 1 , MISO =1 , SS_n =0 , counter= 6
# MOSI= 1 , MISO =1 , SS_n =0 , counter= 5
# MOSI= 1 , MISO =1 , SS_n =0 , counter= 4
# MOSI= 1 , MISO =1 , SS_n =0 , counter= 3
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 2
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 3
# MOSI= 1 , MISO =0 , SS_n =0 , counter= 2
# MOSI= 1 , MISO =0 , SS_n =1 , counter= 2
# MOSI= 1 , MISO =0 , SS_n =1 , counter= 3
# MOSI= 1 , MISO =0 , SS_n =1 , counter= 0
# Read Data Operation Sucessfully
# The design is done successfully
```

Simulation

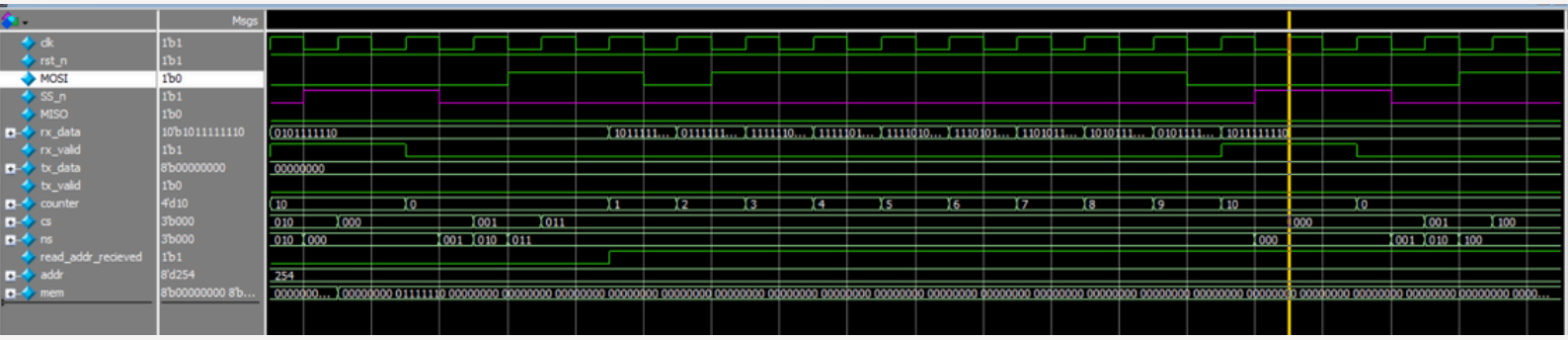
Write Address



Write Data

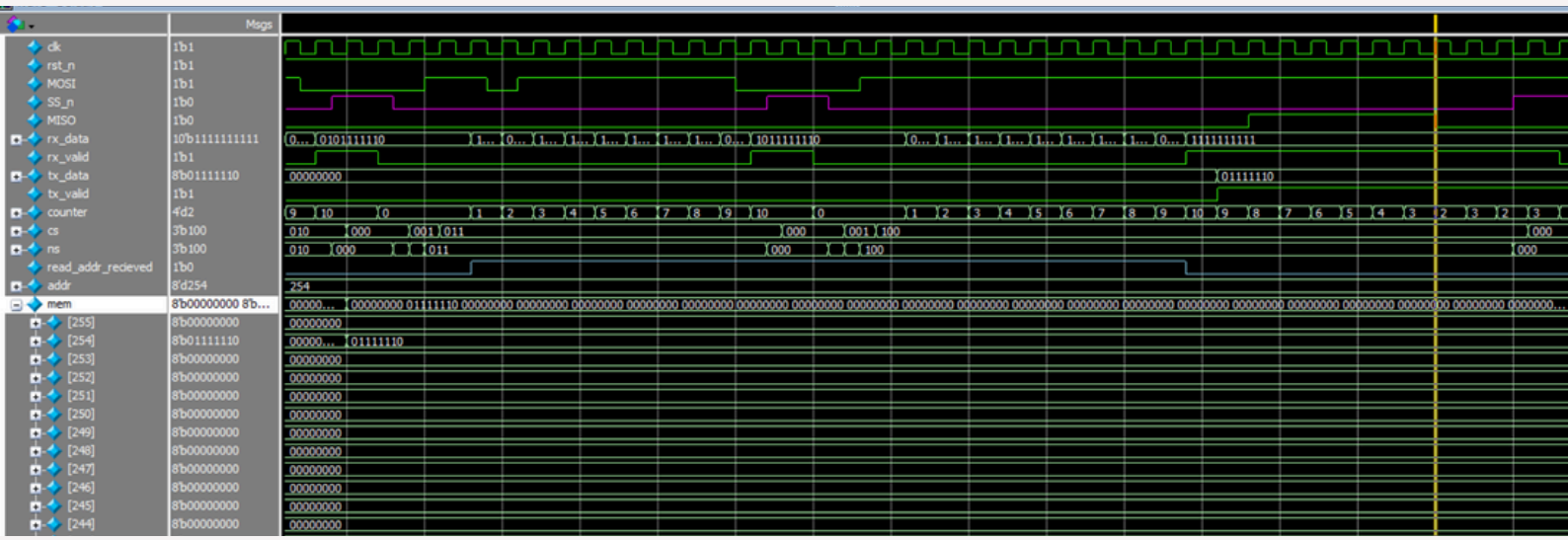


Read Address

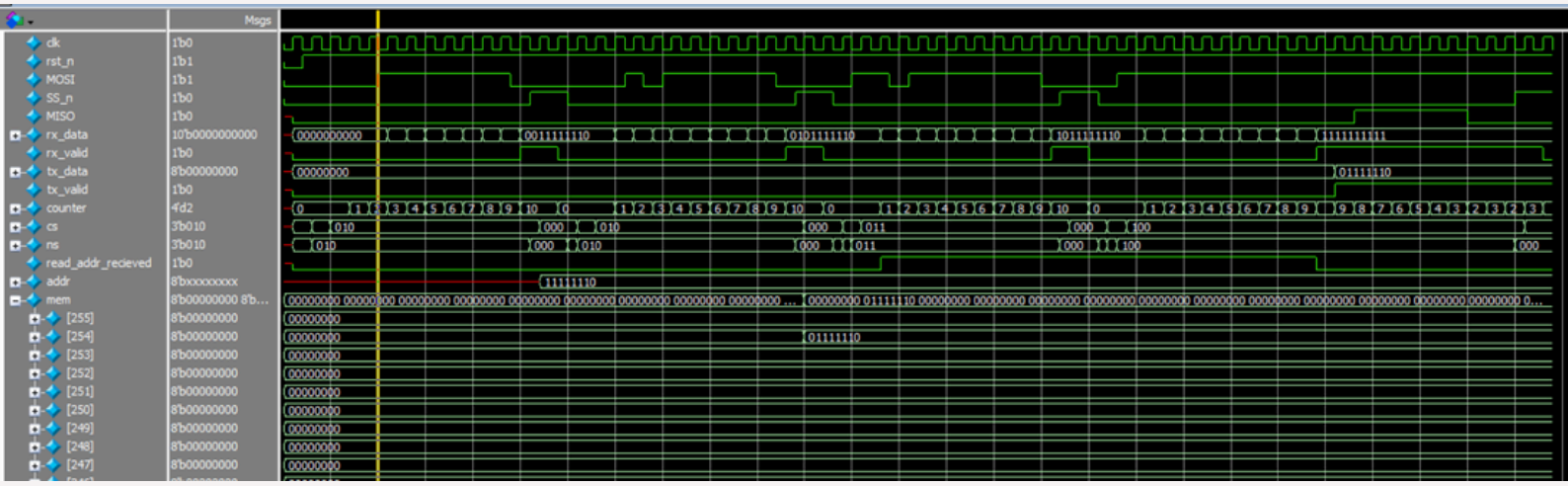


Simulation

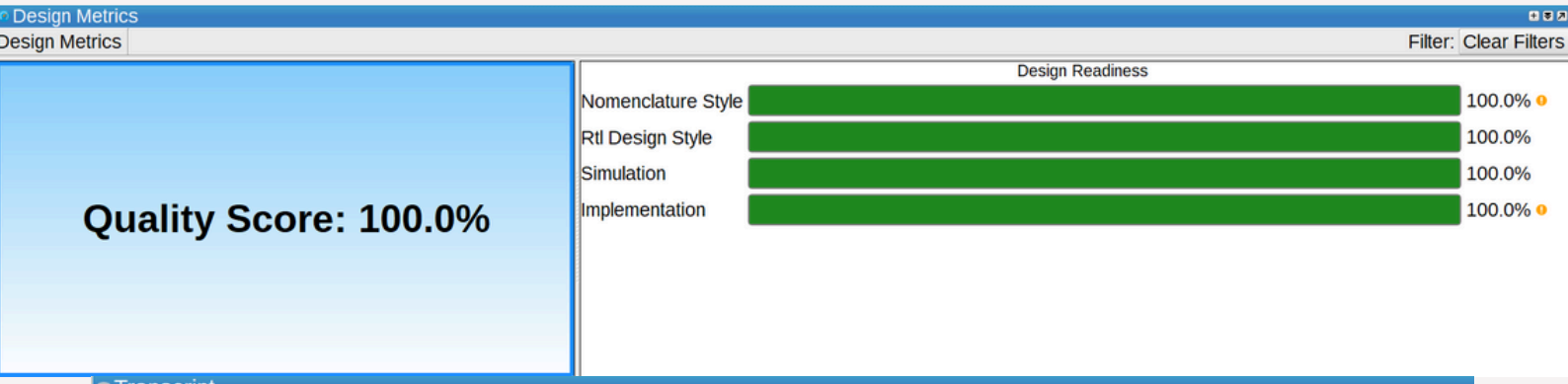
Read Data



Full Wave



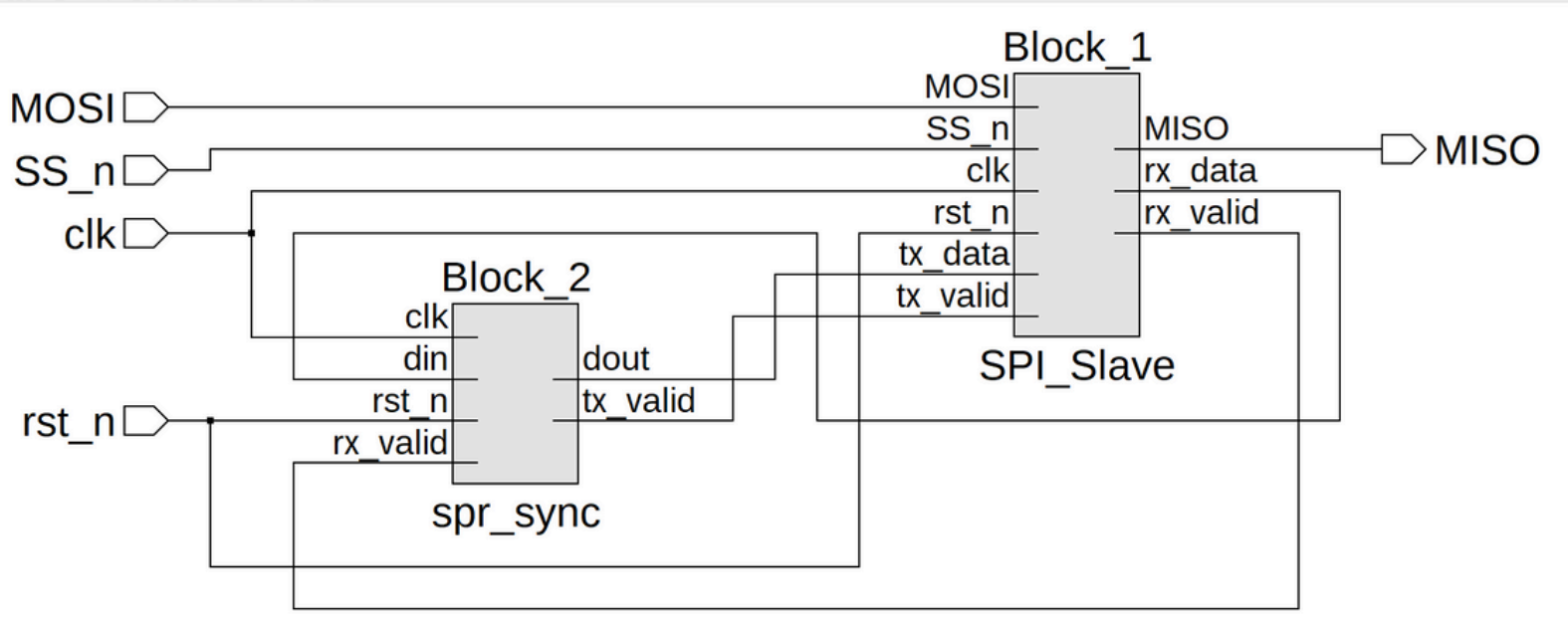
Qustalint



Transcript

```
# ------------------------------------------------------------
# Warning (0)

# ------------------------------------------------------------

# ------------------------------------------------------------
# Info (4)

# ------------------------------------------------------------
#   line_char_large                          :1
#   multi_ports_in_single_line               :3

# ------------------------------------------------------------
# Resolved (0)
```

Schematic



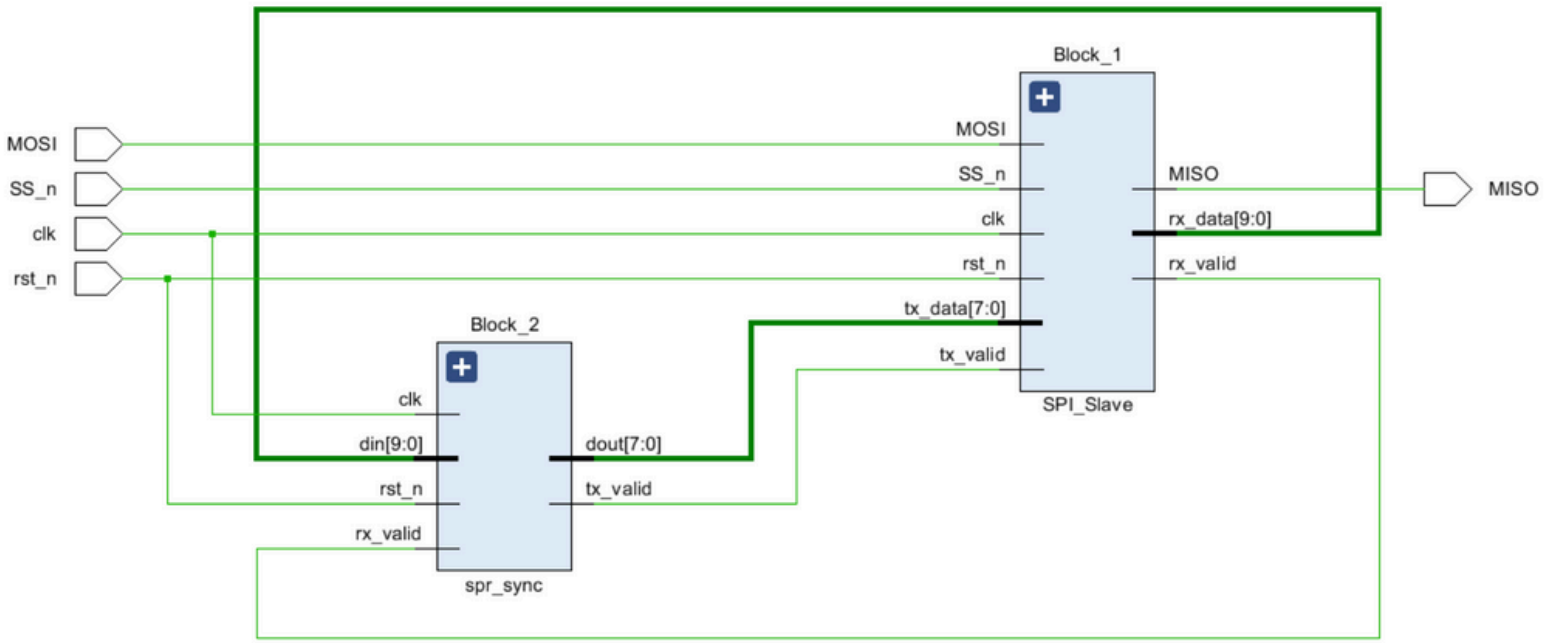
Constraints File

Constraints_basys3.xdc

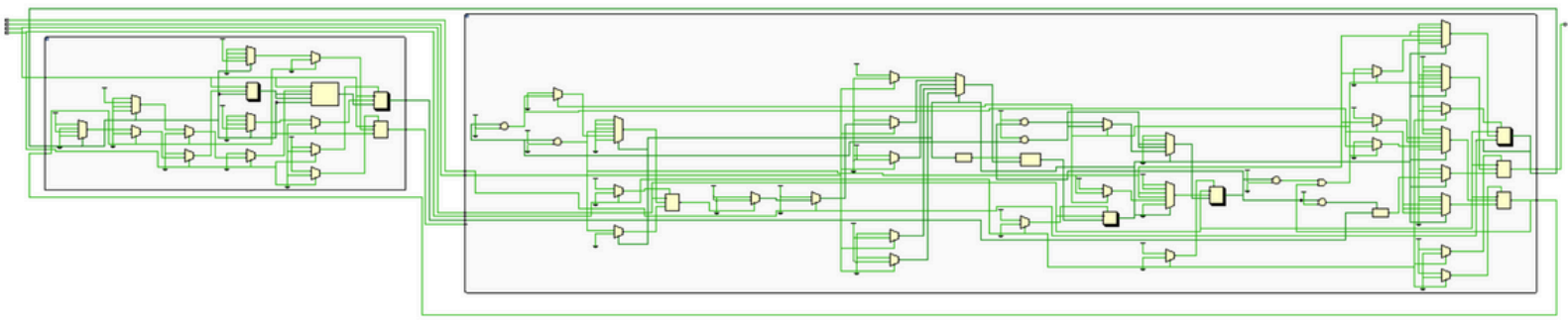
```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signa

## Clock signal
set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports clk]
create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports clk]


## Switches
set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports MOSI]
set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports SS_n]
## LEDs
set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports MISO]


##7 Segment Display
#set_property -dict { PACKAGE_PIN W7   IOSTANDARD LVCMOS33 } [get_ports {seg[0]}]
#set_property -dict { PACKAGE_PIN W6   IOSTANDARD LVCMOS33 } [get_ports {seg[1]}]
#set_property -dict { PACKAGE_PIN U8   IOSTANDARD LVCMOS33 } [get_ports {seg[2]}]
#set_property -dict { PACKAGE_PIN V8   IOSTANDARD LVCMOS33 } [get_ports {seg[3]}]
#set_property -dict { PACKAGE_PIN U5   IOSTANDARD LVCMOS33 } [get_ports {seg[4]}]
#set_property -dict { PACKAGE_PIN V5   IOSTANDARD LVCMOS33 } [get_ports {seg[5]}]
#set_property -dict { PACKAGE_PIN U7   IOSTANDARD LVCMOS33 } [get_ports {seg[6]}]

#set_property -dict { PACKAGE_PIN V7   IOSTANDARD LVCMOS33 } [get_ports dp]

#set_property -dict { PACKAGE_PIN U2   IOSTANDARD LVCMOS33 } [get_ports {an[0]}]
#set_property -dict { PACKAGE_PIN U4   IOSTANDARD LVCMOS33 } [get_ports {an[1]}]
#set_property -dict { PACKAGE_PIN V4   IOSTANDARD LVCMOS33 } [get_ports {an[2]}]
#set_property -dict { PACKAGE_PIN W4   IOSTANDARD LVCMOS33 } [get_ports {an[3]}]


##Buttons
set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports rst_n]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports btnU]
```

Vivado

RTL ANALYSIS



RTL ANALYSIS_Internal

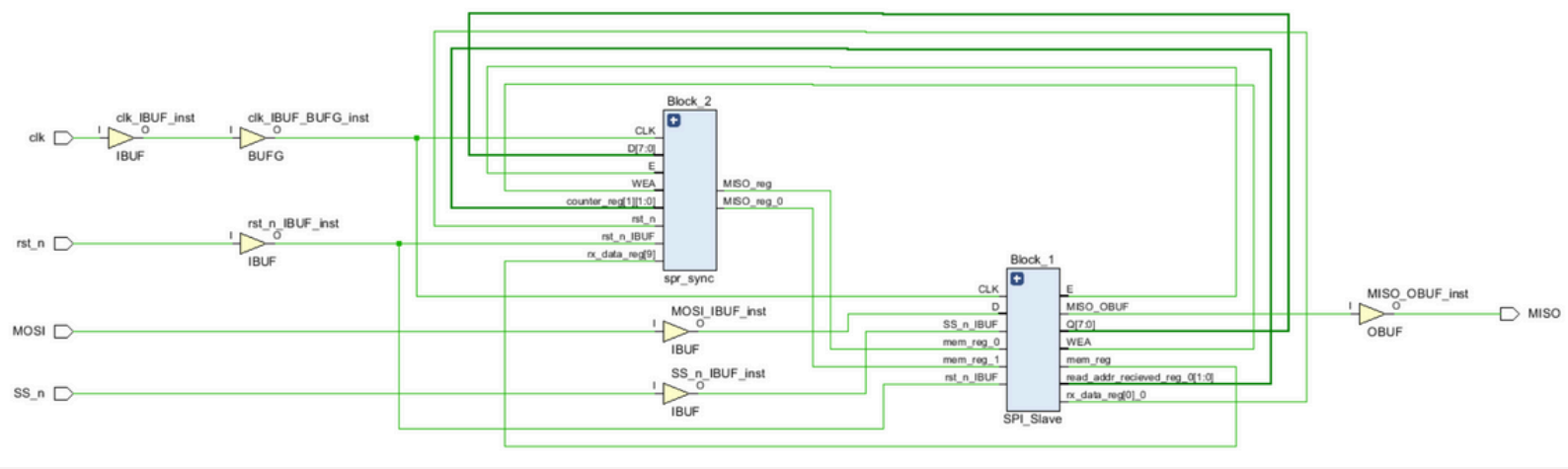


Vivado
Gray Encoding

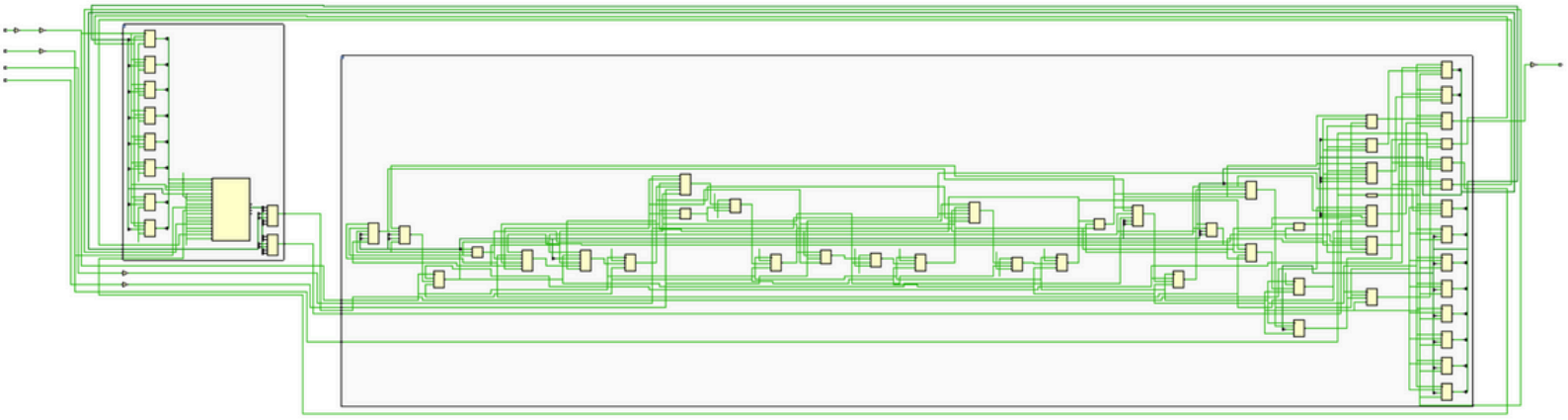
Report

State	New Encoding	Previous Encoding
IDLE	000	000
CHK_CMD	001	001
WRITE	011	010
READ_ADD	010	011
READ_DATA	111	100

Synthesis



Synthesis_Internal



Vivado
Gray Encoding

Report Timing Summary after synthesis

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.445 ns	Worst Hold Slack (WHS):	0.154 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:	74	Total Number of Endpoints:	74	Total Number of Endpoints:	31

All user specified timing constraints are met.

Report Utilization after synthesis

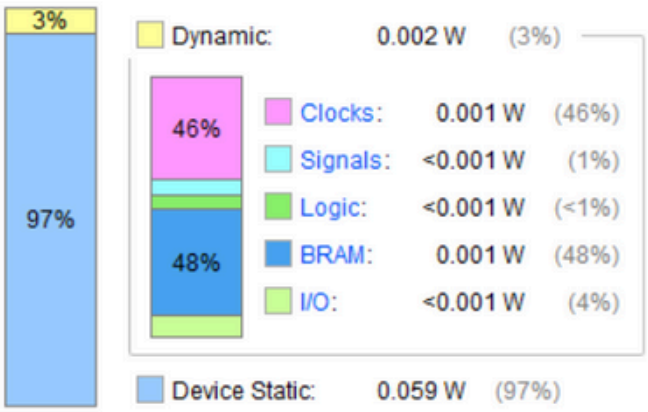
Name	Slice LUTs (20800)	Slice Registers (41600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
SPI_Wrapper	25	31	0.5	5	1
Block_1 (SPI_Slave)	23	23	0	0	0
Block_2 (spr_sync)	2	8	0.5	0	0

Report Power after synthesis

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

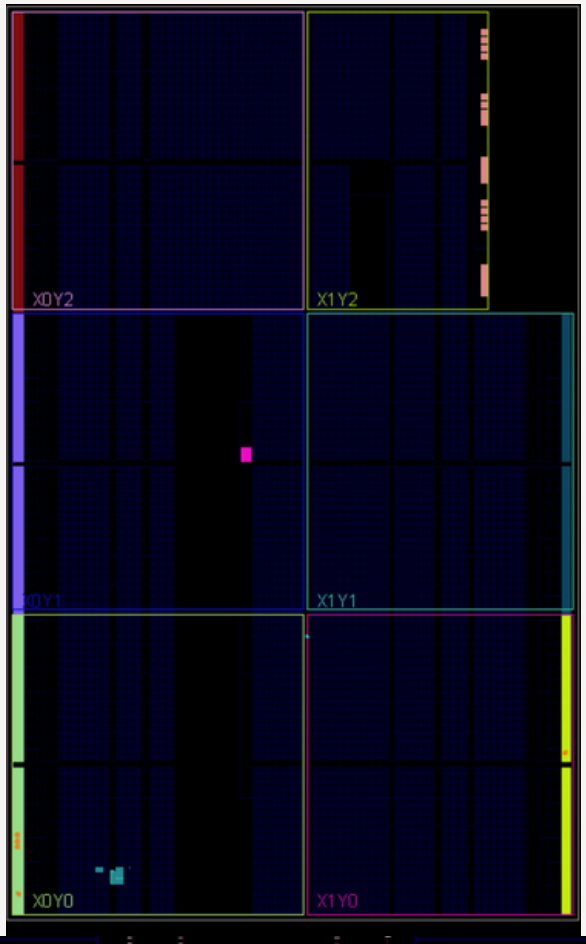
Total On-Chip Power:	**0.06 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	74.7°C (14.9 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

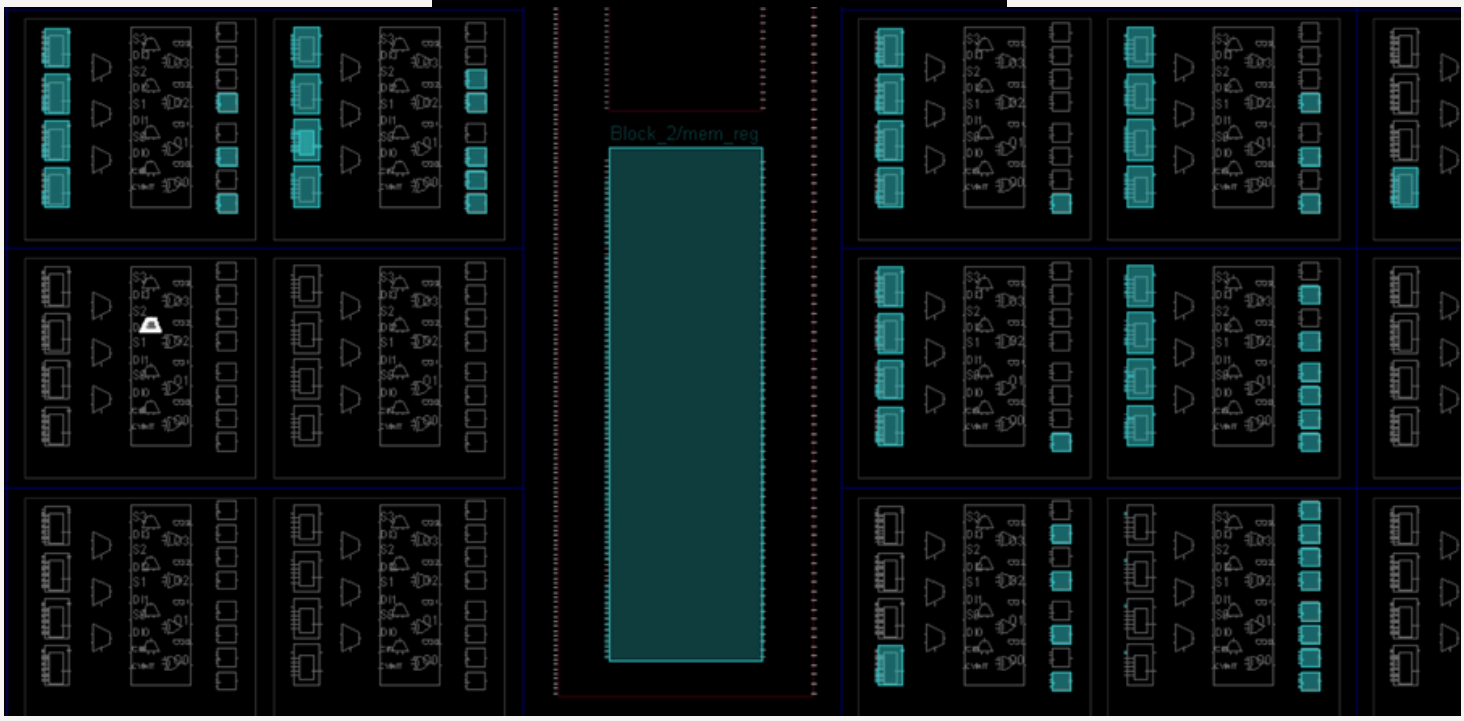
On-Chip Power



Vivado
Gray Encoding

IMPLEMENTATION





Vivado
Gray Encoding

Report Timing Summary after implementation

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.670 ns	Worst Hold Slack (WHS):	0.048 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:	75	Total Number of Endpoints:	75	Total Number of Endpoints:	31

All user specified timing constraints are met.

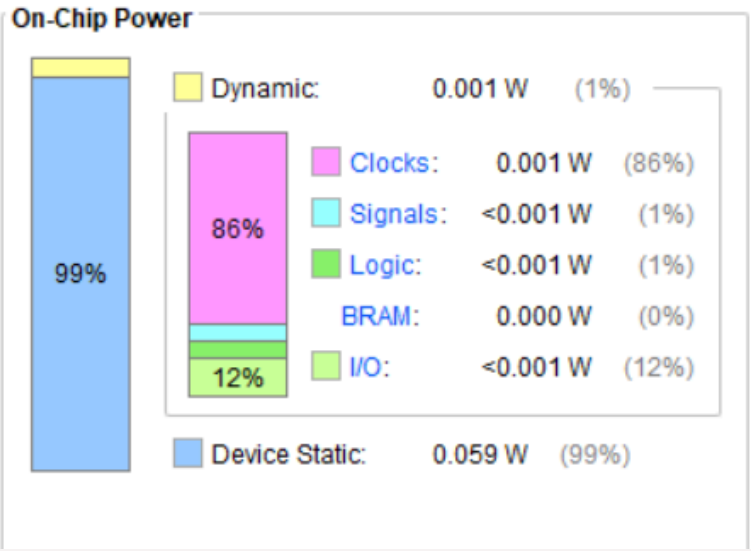
Report Utilization after implementation

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
SPI_Wrapper	26	31	9	26	14	0.5	5	1
Block_1 (SPI_Slave)	23	23	8	23	13	0	0	0
Block_2 (spr_sync)	3	8	4	3	0	0.5	0	0

Report Power after implementation

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.059 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	74.7°C (14.9 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

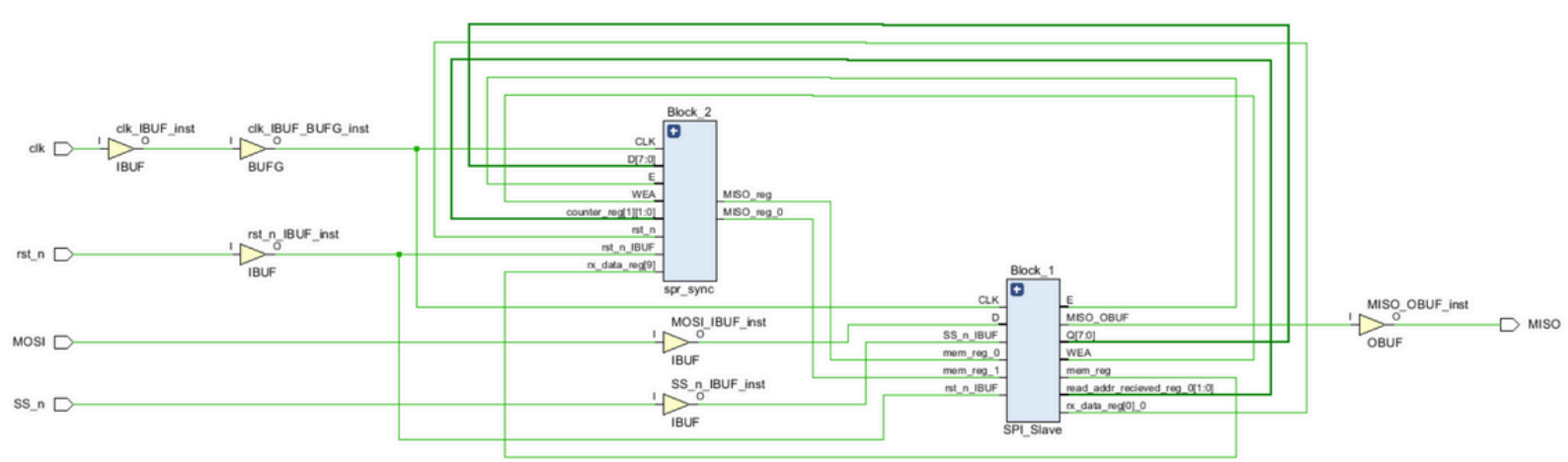


Vivado
one-hot Encoding

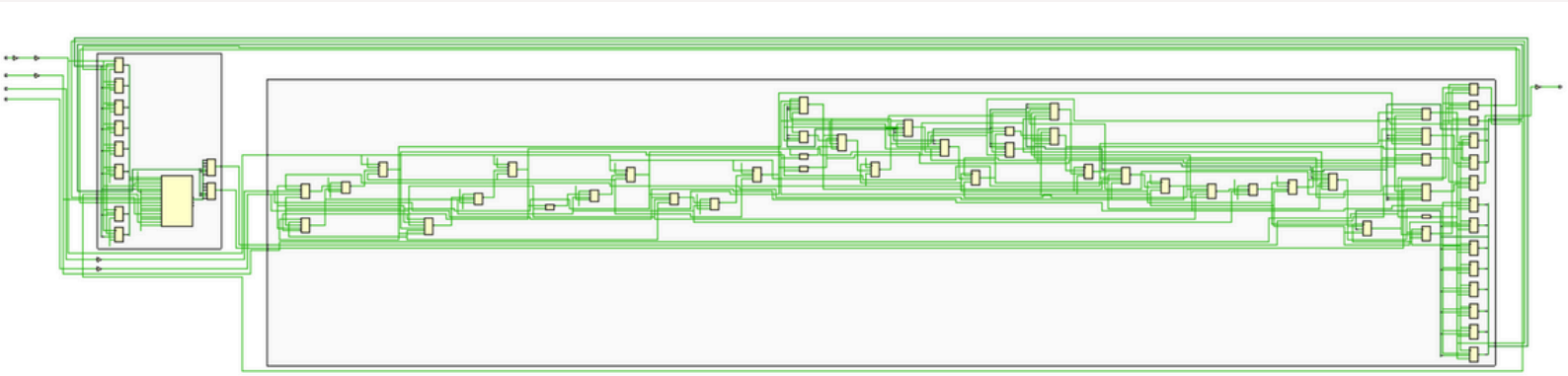
Report

State	New Encoding	Previous Encoding
IDLE	00001	000
CHK_CMD	00010	001
WRITE	00100	010
READ_ADD	01000	011
READ_DATA	10000	100

Synthesis



Synthesis_Internal



Vivado

one-hot Encoding

Report Timing Summary after synthesis

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.898 ns	Worst Hold Slack (WHS):	0.153 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:	74	Total Number of Endpoints:	74	Total Number of Endpoints:	33

All user specified timing constraints are met.

Report Utilization after synthesis

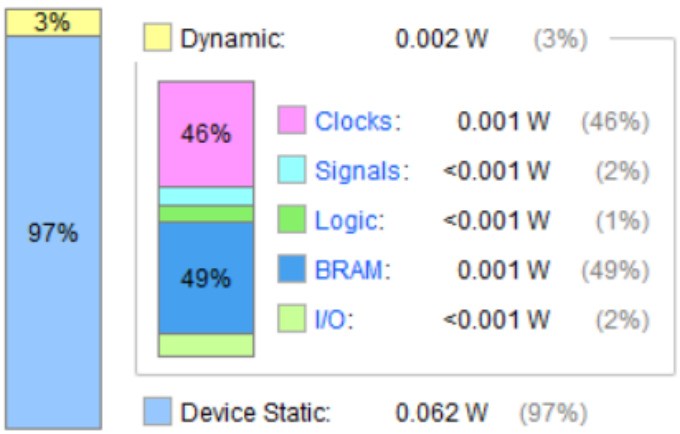
Name	Slice LUTs (20800)	Slice Registers (41600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
SPI_Wrapper	28	35	0.5	5	1
Block_1 (SPI_Slave)	26	27	0	0	0
Block_2 (spr_sync)	2	8	0.5	0	0

Report Power after synthesis

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

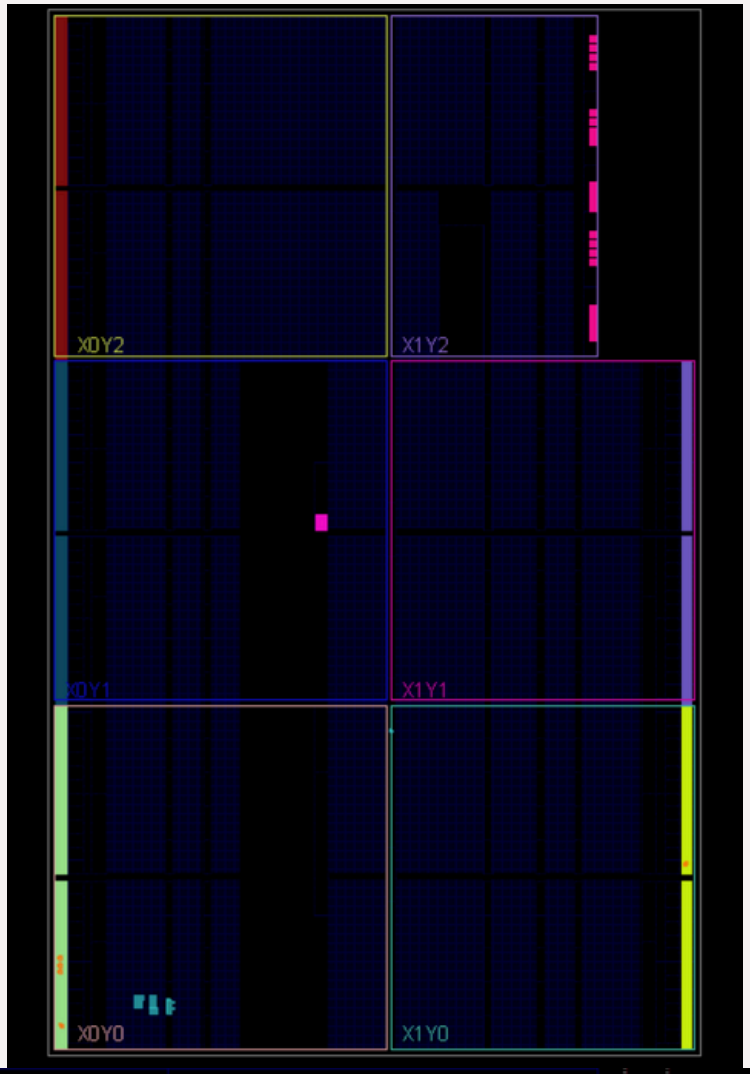
Total On-Chip Power:	**0.064 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	74.7°C (14.9 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

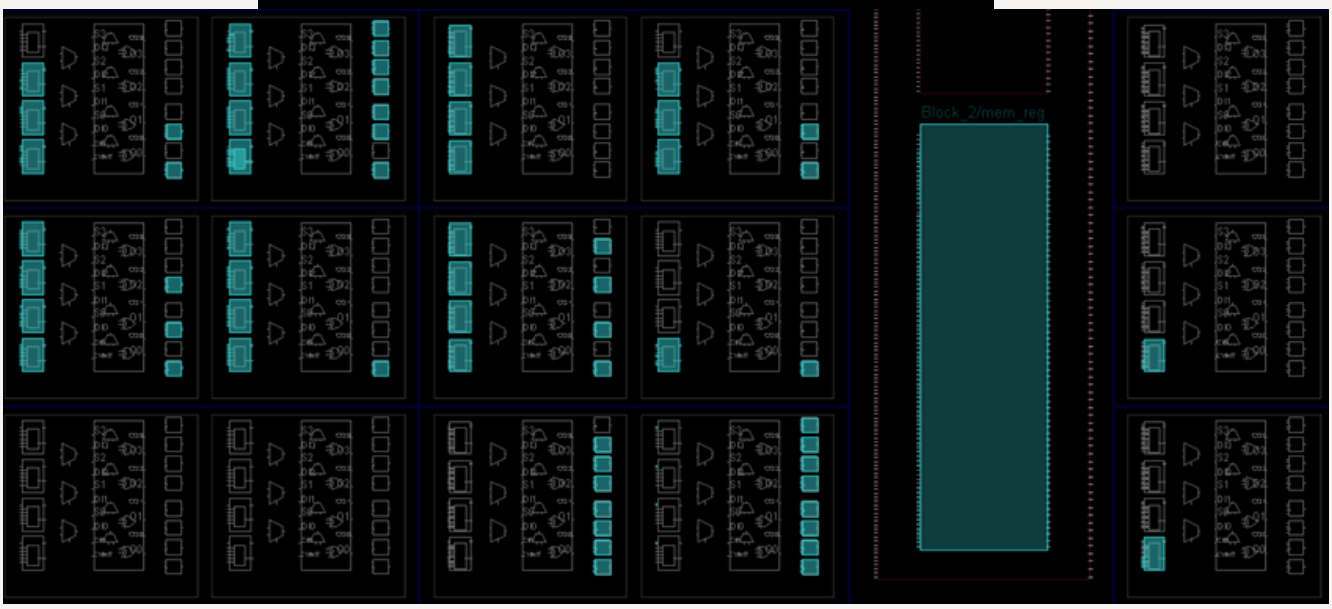
On-Chip Power



Vivado
one-hot Encoding

IMPLEMENTATION





Vivado
one-hot Encoding

Report Timing Summary after implementation

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.394 ns	Worst Hold Slack (WHS):	0.177 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:	75	Total Number of Endpoints:	75	Total Number of Endpoints:	33

All user specified timing constraints are met.

Report Utilization after implementation

Hierarchy

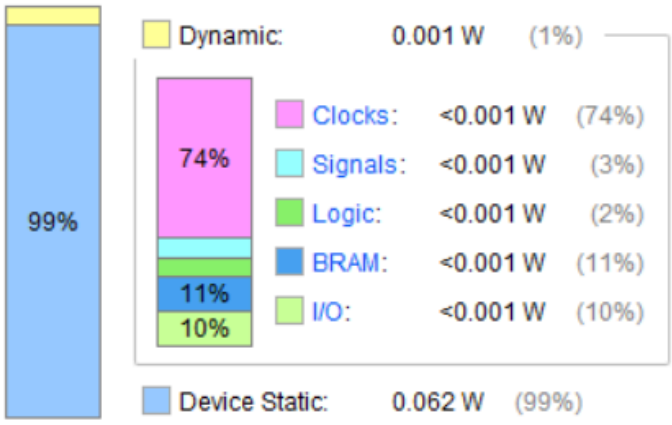
Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
SPI_Wrapper	29	35	12	29	13	0.5	5	1
Block_1 (SPI_Slave)	26	27	9	26	12	0	0	0
Block_2 (spr_sync)	3	8	4	3	0	0.5	0	0

Report Power after implementation

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.063 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	74.7°C (14.9 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

On-Chip Power



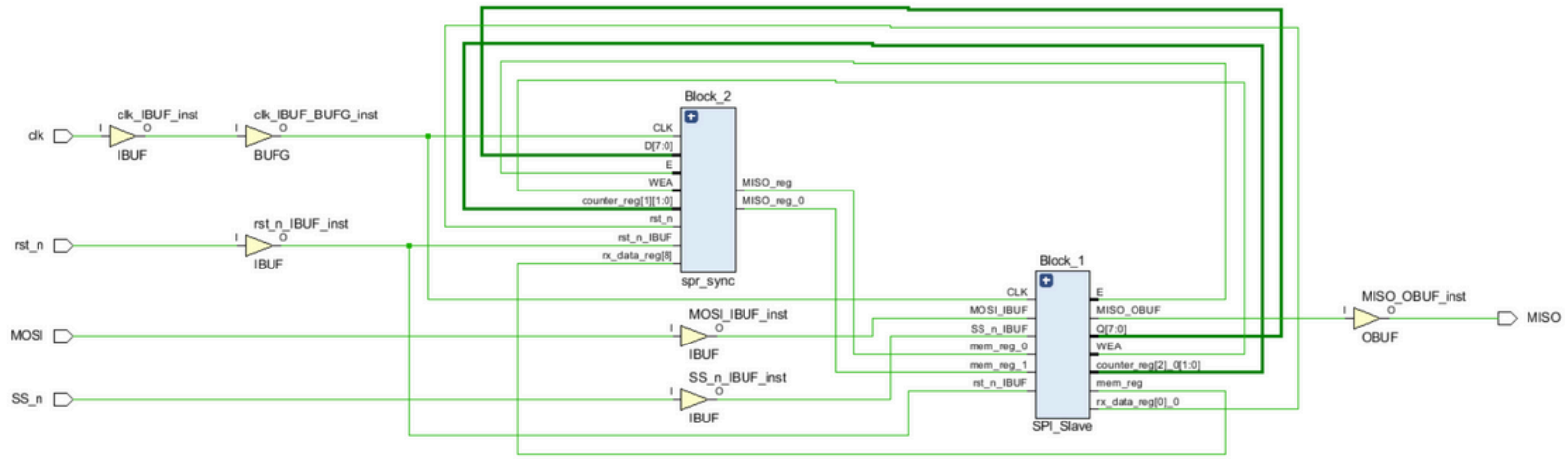
Vivado

Sequential Encoding

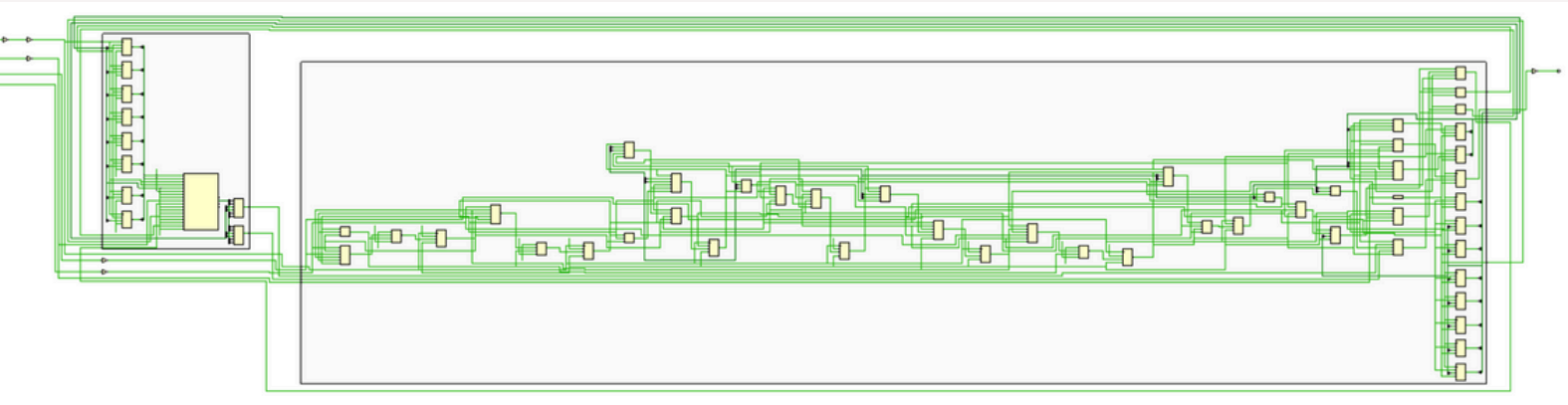
Report

State	New Encoding	Previous Encoding
IDLE	000	000
CHK_CMD	001	001
WRITE	010	010
READ_ADD	011	011
READ_DATA	100	100

Synthesis



Synthesis_Internal



Vivado

Sequential Encoding

Report Timing Summary after synthesis

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.445 ns	Worst Hold Slack (WHS):	0.152 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:	74	Total Number of Endpoints:	74	Total Number of Endpoints:	31

All user specified timing constraints are met.

Report Utilization after synthesis

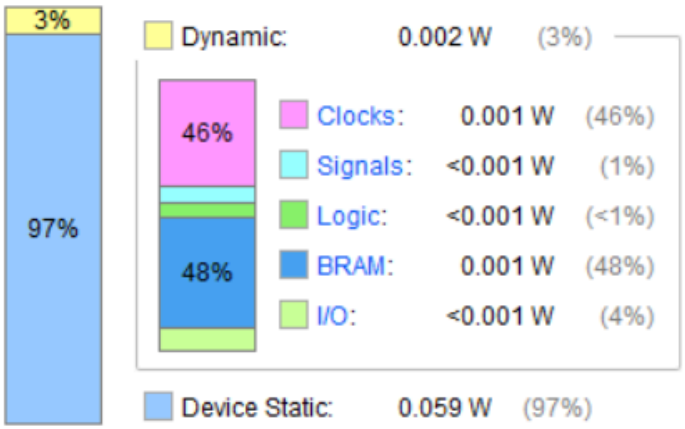
Name	Slice LUTs (20800)	Slice Registers (41600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
SPI_Wrapper	26	31	0.5	5	1
Block_1 (SPI_Slave)	24	23	0	0	0
Block_2 (spr_sync)	2	8	0.5	0	0

Report Power after synthesis

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

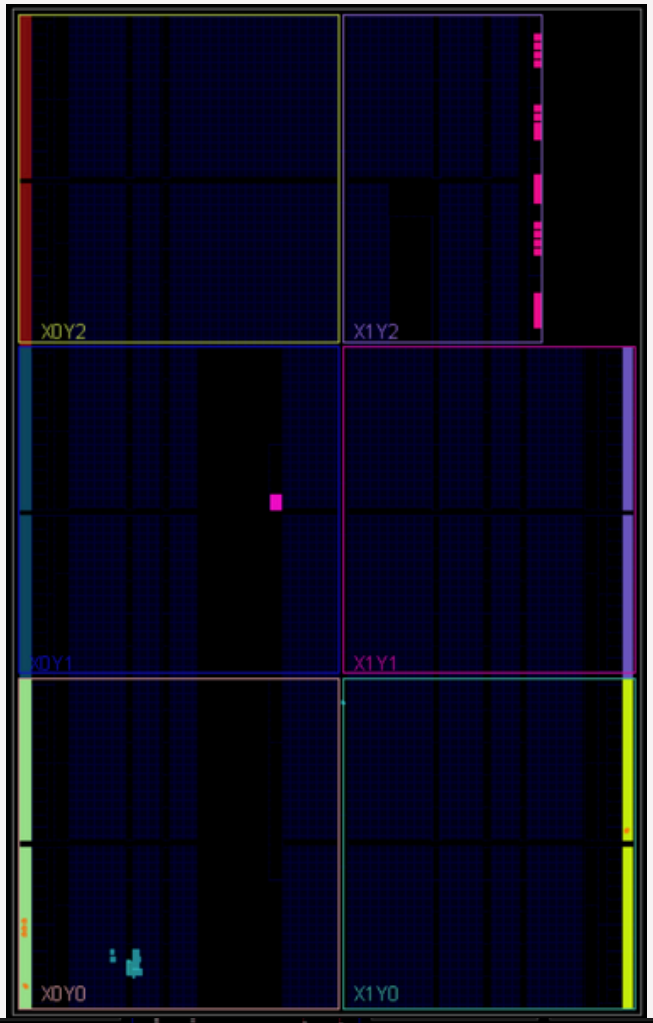
Total On-Chip Power:	**0.06 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	74.7°C (14.9 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

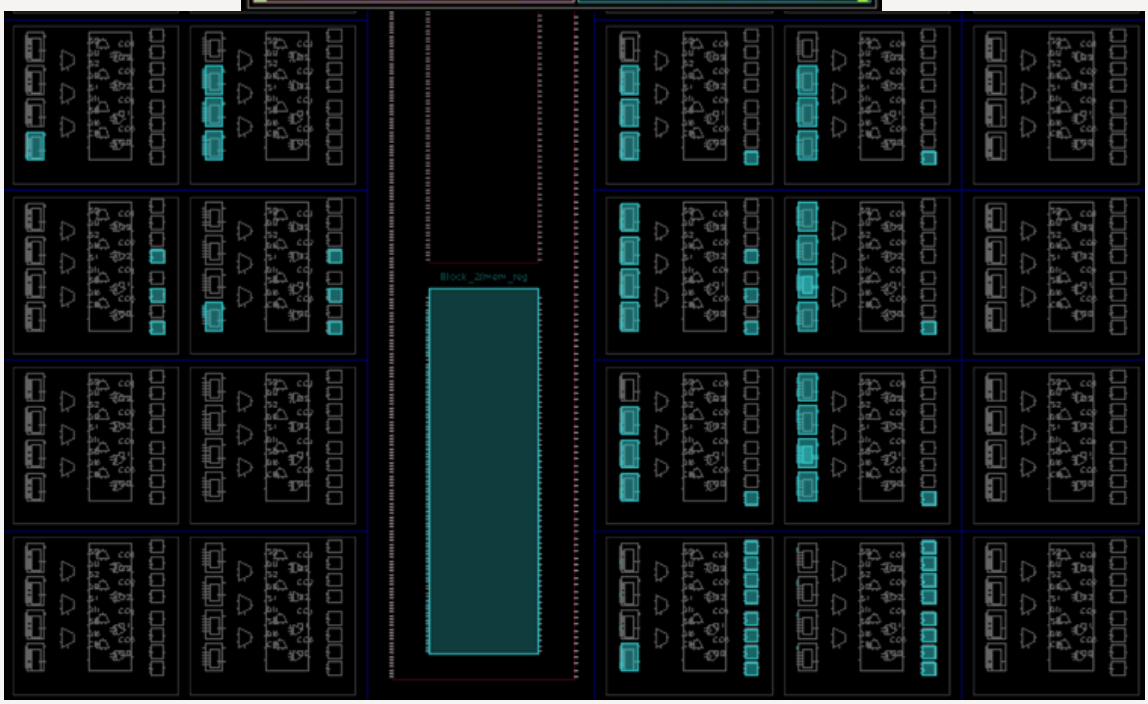
On-Chip Power



Vivado
Sequential Encoding

IMPLEMENTATION





Vivado
Sequential Encoding

Report Timing Summary after implementation

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.559 ns	Worst Hold Slack (WHS):	0.049 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:	75	Total Number of Endpoints:	75	Total Number of Endpoints:	31

All user specified timing constraints are met.

Report Utilization after implementation

Hierarchy

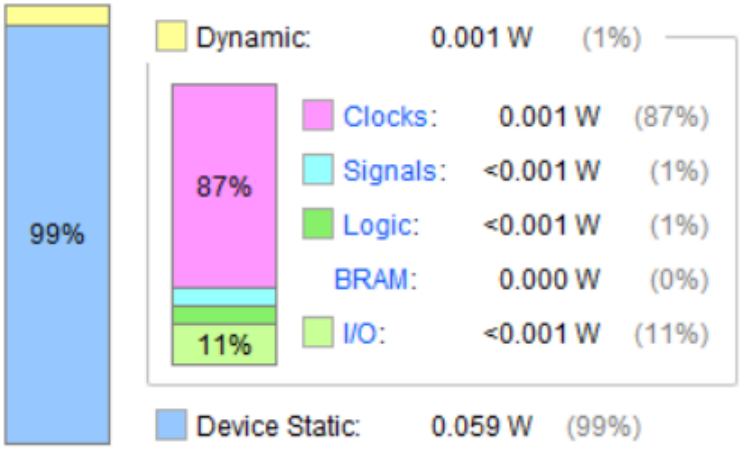
Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
SPI_Wrapper	27	31	13	27	7	0.5	5	1
Block_1 (SPI_Slave)	24	23	13	24	7	0	0	0
Block_2 (spr_sync)	3	8	3	3	0	0.5	0	0

Report Power after implementation

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.059 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	74.7°C (14.9 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

On-Chip Power



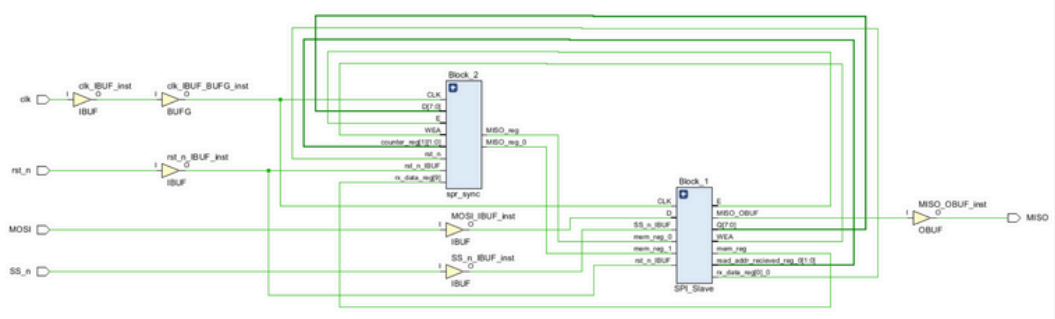
Vivado

Debug Core

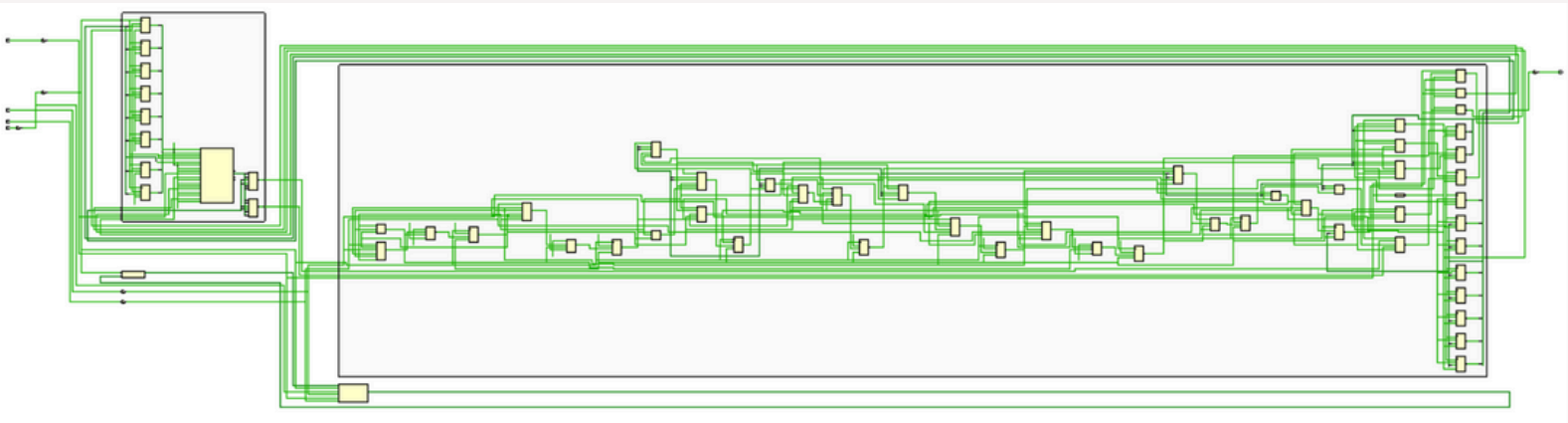
Report

```
create_debug_core u_ila_0 ila
set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
set_property port_width 1 [get_debug_ports u_ila_0/clk]
connect_debug_port u_ila_0/clk [get_nets [list clk_IBUF_BUFG]]
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe0]
set_property port_width 1 [get_debug_ports u_ila_0/probe0]
connect_debug_port u_ila_0/probe0 [get_nets [list clk_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe1]
set_property port_width 1 [get_debug_ports u_ila_0/probe1]
connect_debug_port u_ila_0/probe1 [get_nets [list MOSI_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe2]
set_property port_width 1 [get_debug_ports u_ila_0/probe2]
connect_debug_port u_ila_0/probe2 [get_nets [list rst_n_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe3]
set_property port_width 1 [get_debug_ports u_ila_0/probe3]
connect_debug_port u_ila_0/probe3 [get_nets [list SS_n_IBUF]]
set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
connect_debug_port dbg_hub/clk [get_nets clk_IBUF_BUFG]
```

Synthesis



Synthesis_Internal



Vivado

Debug Core

Report Timing Summary after synthesis

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.445 ns	Worst Hold Slack (WHS):	0.152 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:	74	Total Number of Endpoints:	74	Total Number of Endpoints:	31

All user specified timing constraints are met.

Report Utilization after synthesis

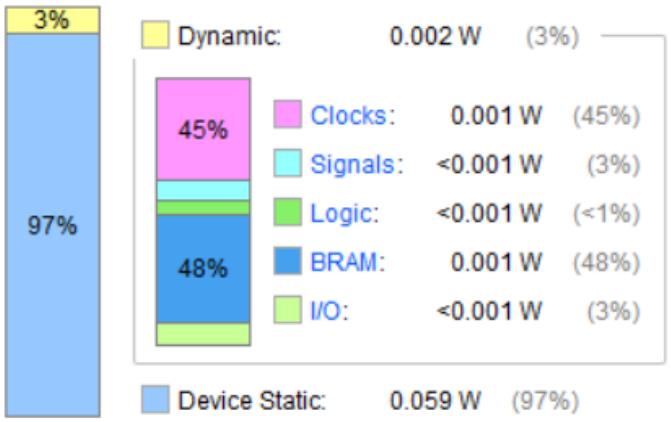
Name	Slice LUTs (20800)	Slice Registers (41600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)
SPI_Wrapper	26	31	0.5	5	1
Block_1 (SPI_Slave)	24	23	0	0	0
Block_2 (spr_sync)	2	8	0.5	0	0
dbg_hub (dbg_hub_CV)	0	0	0	0	0
u_ila_0 (u_ila_0_CV)	0	0	0	0	0

Report Power after synthesis

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.06 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	74.7°C (14.9 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

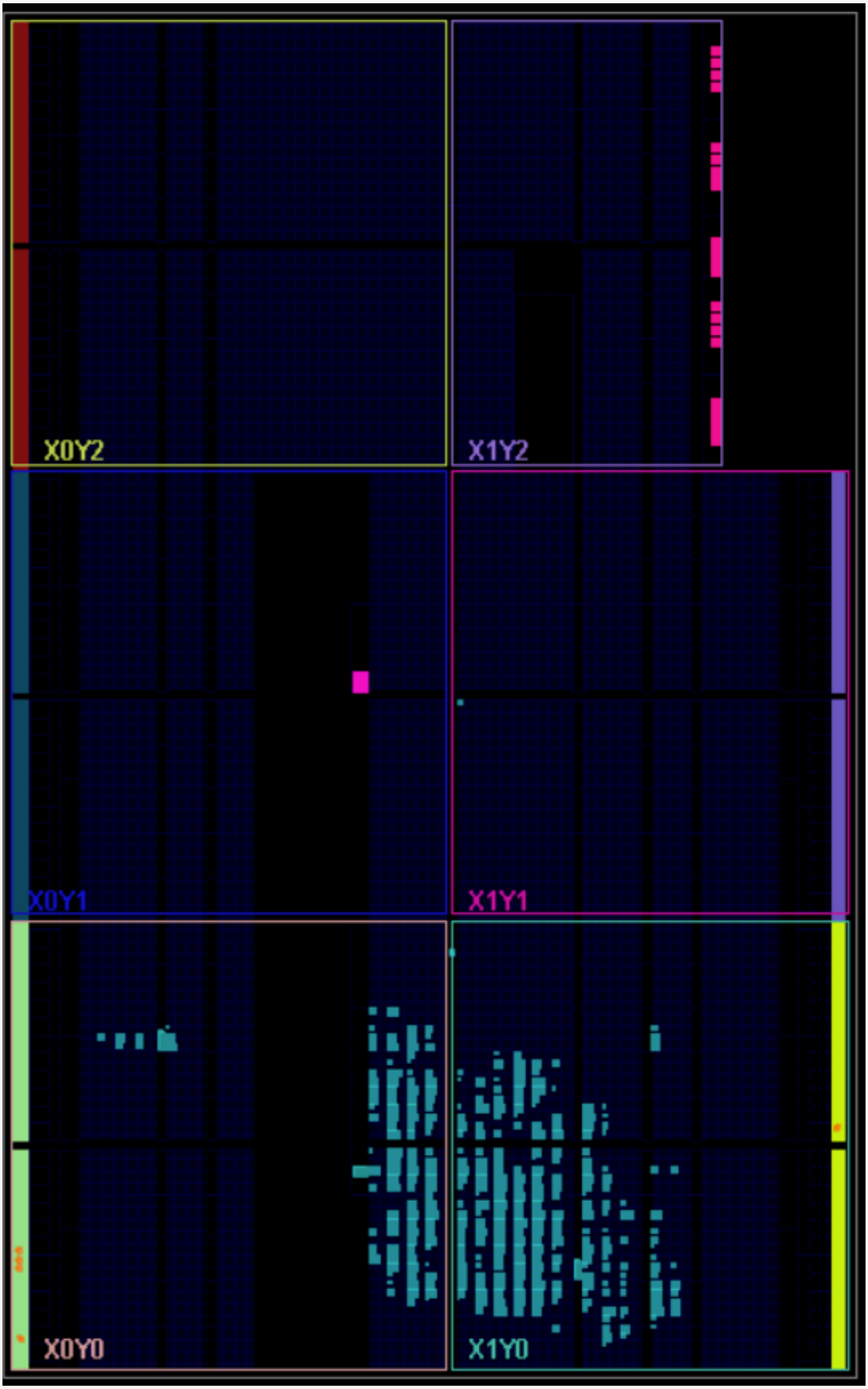
On-Chip Power



Vivado

Debug Core

IMPLEMENTATION



Vivado

Debug Core

Report Timing Summary after implementation

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.166 ns	Worst Hold Slack (WHS):	0.045 ns	Worst Pulse Width Slack (WPWS):	3.750 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:	3764	Total Number of Endpoints:	3748	Total Number of Endpoints:	2067

All user specified timing constraints are met.

Report Utilization after implementation

Hierarchy

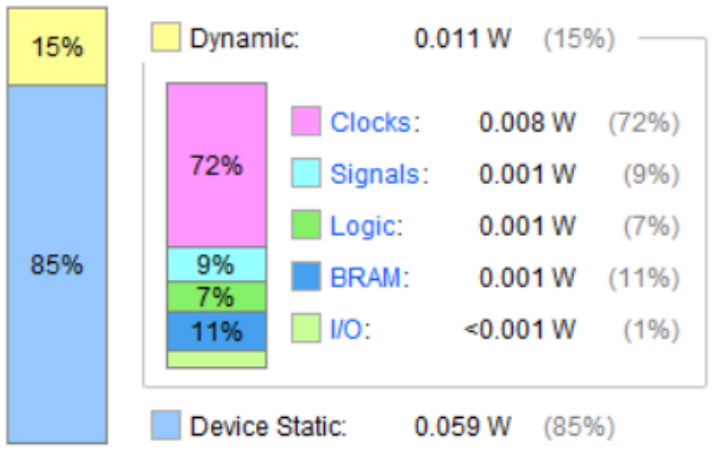
Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)	BSCANE2 (4)
SPI_Wrapper	1215	1893	8	592	1112	103	711	1	5	2	1
Block_1 (SPI_Slave)	24	23	0	12	24	0	10	0	0	0	0
Block_2 (spr_sync)	2	8	0	3	2	0	0	0.5	0	0	0
dbg_hub (dbg_hub)	475	727	0	223	451	24	314	0	0	1	1
u_ila_0 (u_ila_0)	714	1135	8	359	635	79	387	0.5	0	0	0

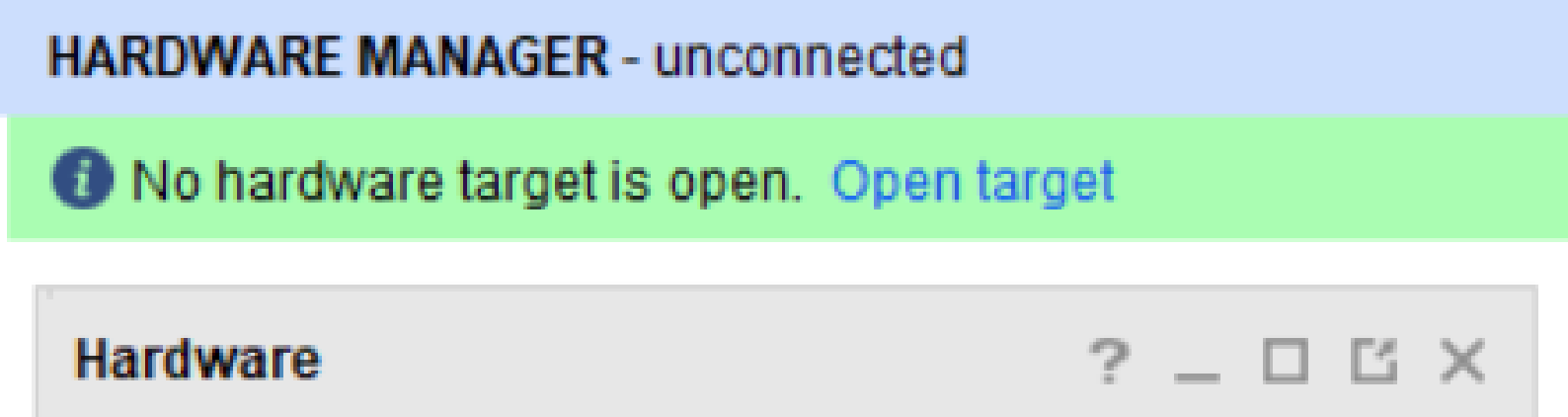
Report Power after implementation

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.069 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	74.7°C (14.8 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

On-Chip Power



Vivado

Bitstream Generation