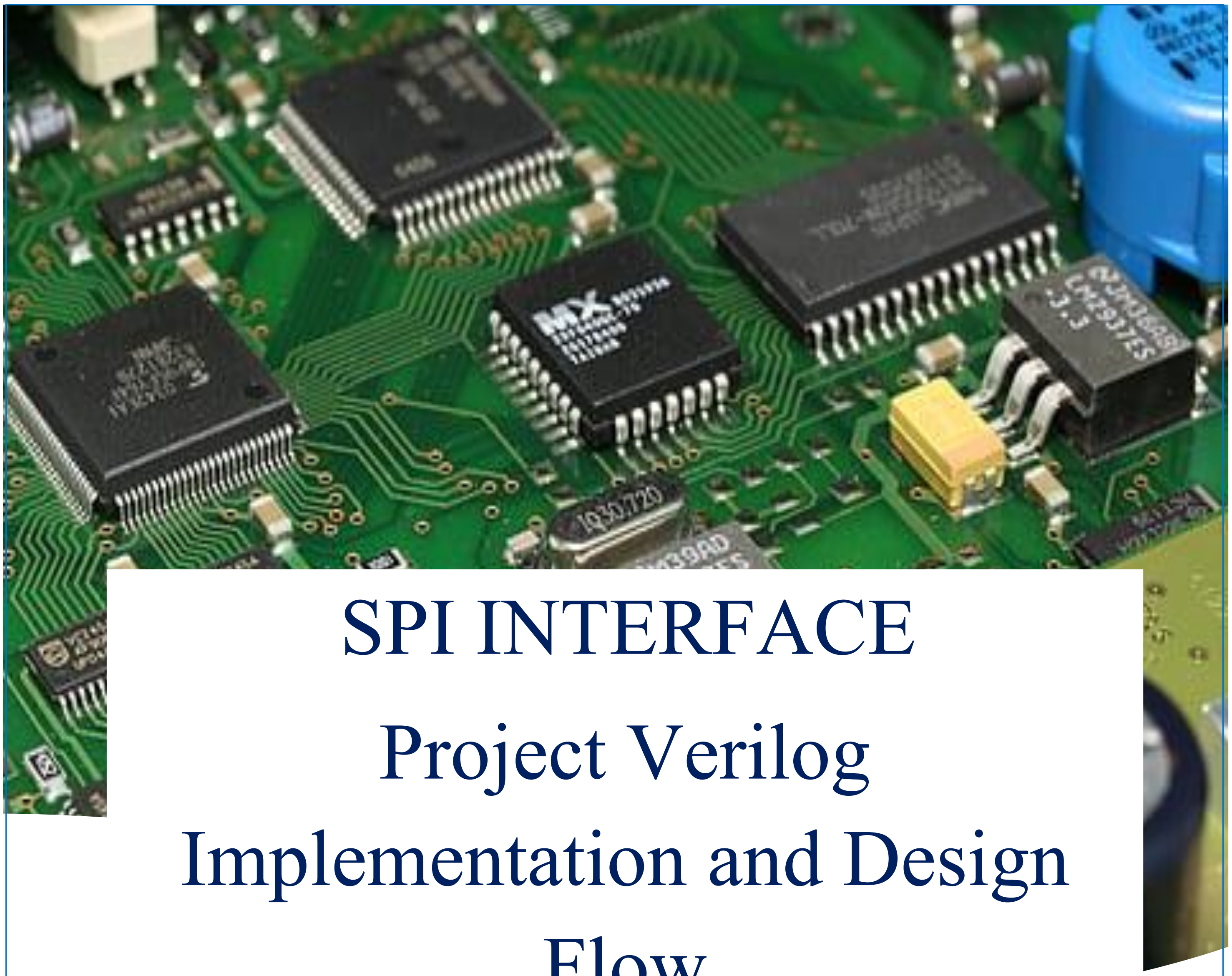


SPI INTERFACE

Project Verilog Implementation and Design Flow

Abdelrahman Khaled

Introduction

The **Serial Peripheral Interface (SPI)** is a high-speed, full-duplex, synchronous communication protocol commonly used in embedded systems for short-distance communication between microcontrollers and peripheral devices such as sensors, memory modules, and displays. SPI is favored for its simplicity, flexibility, and efficiency in transferring data with minimal overhead.

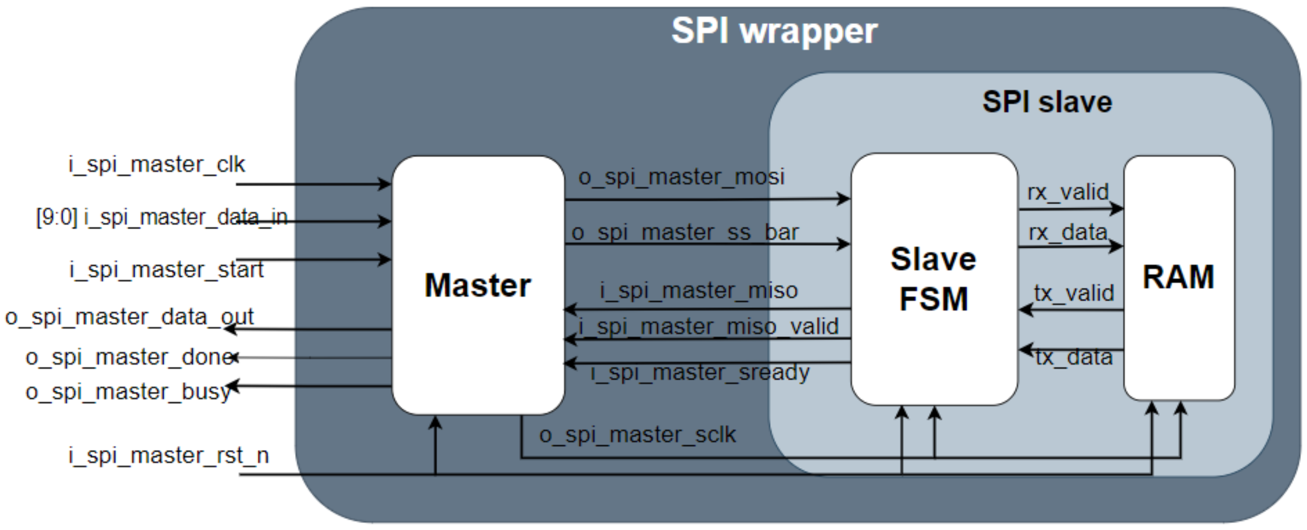
SPI operates using a **master-slave architecture** and utilizes four primary signals:

- **MOSI (Master Out, Slave In):** Transmits data from the master to the slave.
- **MISO (Master In, Slave Out):** Transmits data from the slave to the master.
- **SCLK (Serial Clock):** Clock signal generated by the master to synchronize data transmission.
- **SS_n (Slave Select, Active Low):** Selects the active slave device for communication.

This communication is fully synchronous, with data bits typically shifted out on one clock edge and sampled on the opposite edge, depending on the selected SPI mode (Mode 0–3).

This project demonstrates a complete **SPI-based communication system** implemented in Verilog, integrating an SPI Master and Slave with a RAM module. The design facilitates **bidirectional data transfer** over SPI, enabling reliable **read/write access** to RAM using a low-pin-count interface—ideal for resource-constrained embedded systems.

Architecture of SPI Communication System



Main Components of SPI Architecture

1. Master Device

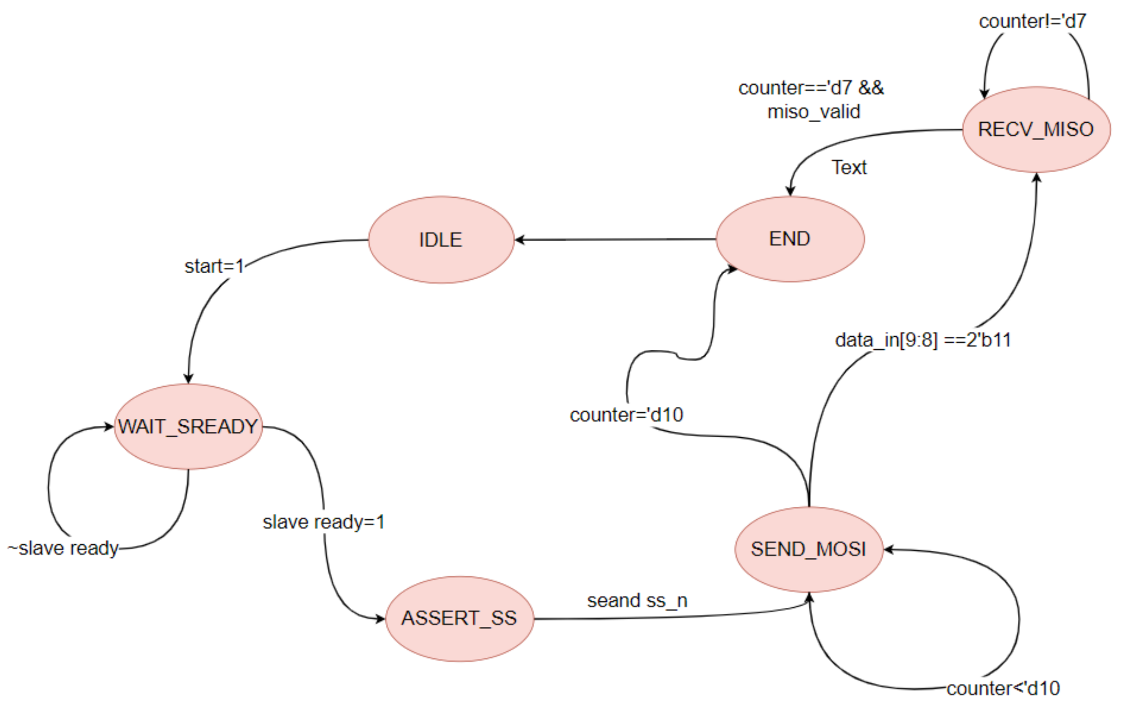
The master initiates and controls all SPI communication. It generates the clock signal and selects which slave device to communicate with.

- **Responsibilities:**
  - ➢ Generates the **serial clock (SCLK)**.
  - ➢ Sends data through **MOSI**.
  - ➢ Receives data from **MISO**.
  - ➢ Controls slave devices using **SS_n**.

- **Port Specification**

Port Name	Direction	Function
data_in	Input	This data is serialized and transmitted to the slave via the **MOSI** line.
start	Input	Initiates an SPI transaction.
busy	Output	Indicates an ongoing SPI transfer.
data_out	Output	Parallel output data received from the slave via MISO.
done	Output	the SPI transaction (read/write) is completed.
ss_n	Output	When asserted low, the selected slave becomes active.
MOSI	Output	Carries the serialized data_in from master to slave, starting with the MSB.
sready	Input	Indicates that the slave is ready for the next transaction.
valid_MISO	Input	Asserted by the slave when valid data is available on the **MISO** line.
MISO	Input	Carries serialized data from the slave to the master.

- **Master FSM**



2. Slave Device

Each slave responds to the master's commands. A slave becomes active when its corresponding **Slave Select (SS_n)** line is pulled low.

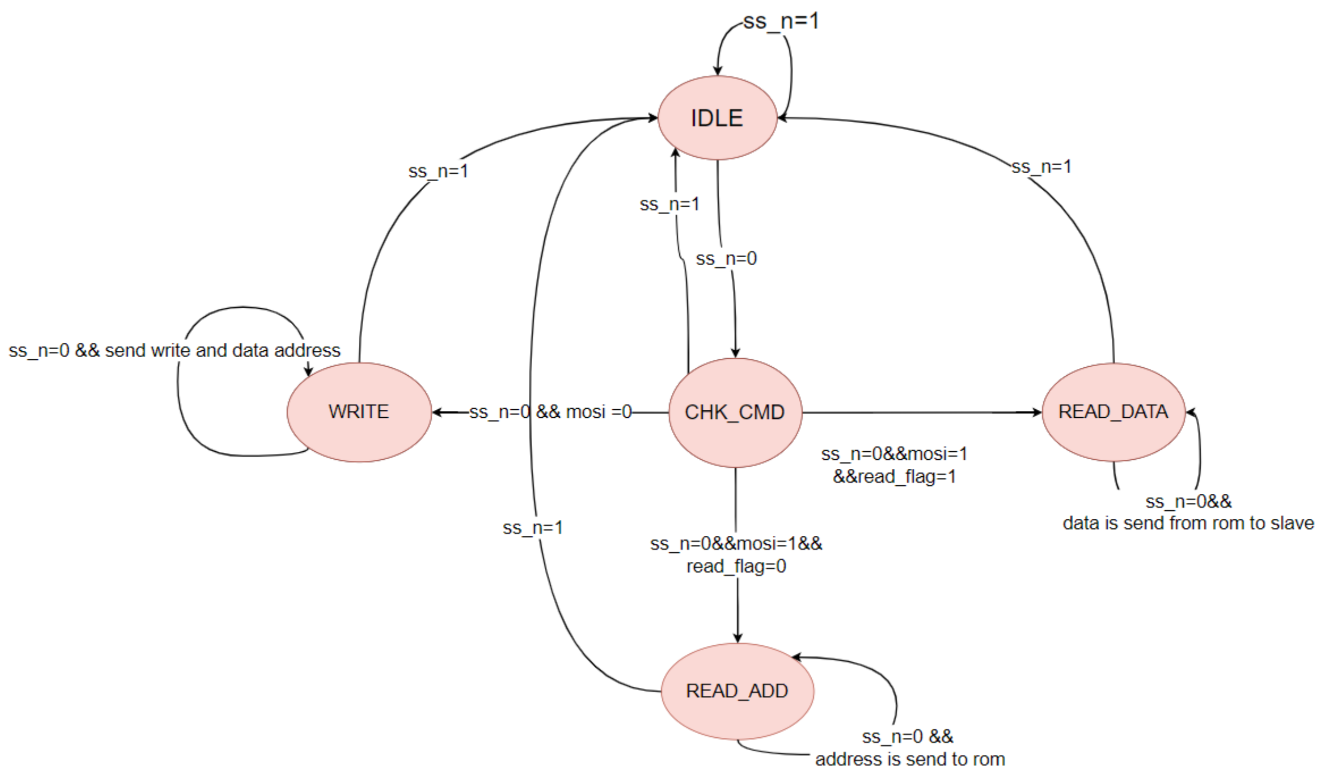
- **Responsibilities:**

  - Receives data from **MOSI**.
  - Sends data through **MISO**.
  - Operates using the clock received from **SCLK**.
  - Must monitor its **SS_n** line to detect selection.

- **Port Specification**

Port Name	Direction	Function
sclk	Input	Serial clock signal provided by the master.
ss_n	Input	When low, the slave is activated.
MOSI	Input	Carries data from the master to the slave, typically MSB first.
sready	Output	Asserted after the current operation is completed.
valid_MISO	Output	Asserted when the slave has valid data on the MISO.
ISO	Output	Transmits data from the slave to the master, synchronized with sclk.

- **slave FSM**



3. RAM

- **Port Specification**

Port Name	Direction	Function
rx_valid	Input	Asserted when a full 10-bit command has been received.
rx_data	Input	10-bit vector from MOSI.
tx_valid	Output	Asserted during a read operation.
tx_data	Output	8-bit data word read from RAM memory.

The SPI slave becomes active when the master asserts the `ss_n` signal low. Upon activation, the slave begins capturing a 10-bit serial input from the `MOSI` line, with data shifted in starting from the most significant bit (MSB). Once the complete 10-bit command is received, it is assembled and forwarded to the internal RAM module for interpretation and action.

- **Command Structure**

The 10-bit command is structured as follows:

- [9:8] – Control bits: Determine the operation type.
- [7:0] – Address or data, depending on the control bits.

- **Port Specification**

Control Bits	Operation	Description
2'b00	Write Address	Stores the lower 8 bits as the address to be used in a subsequent write.
2'b01	Write Data	Writes the 8-bit data to the previously stored address in RAM.
2'b10	Read Address	Stores the lower 8 bits as the address for a future read operation.
2'b11	Read Data	Reads data from the stored address.

SPI Wrapper Functional Verification

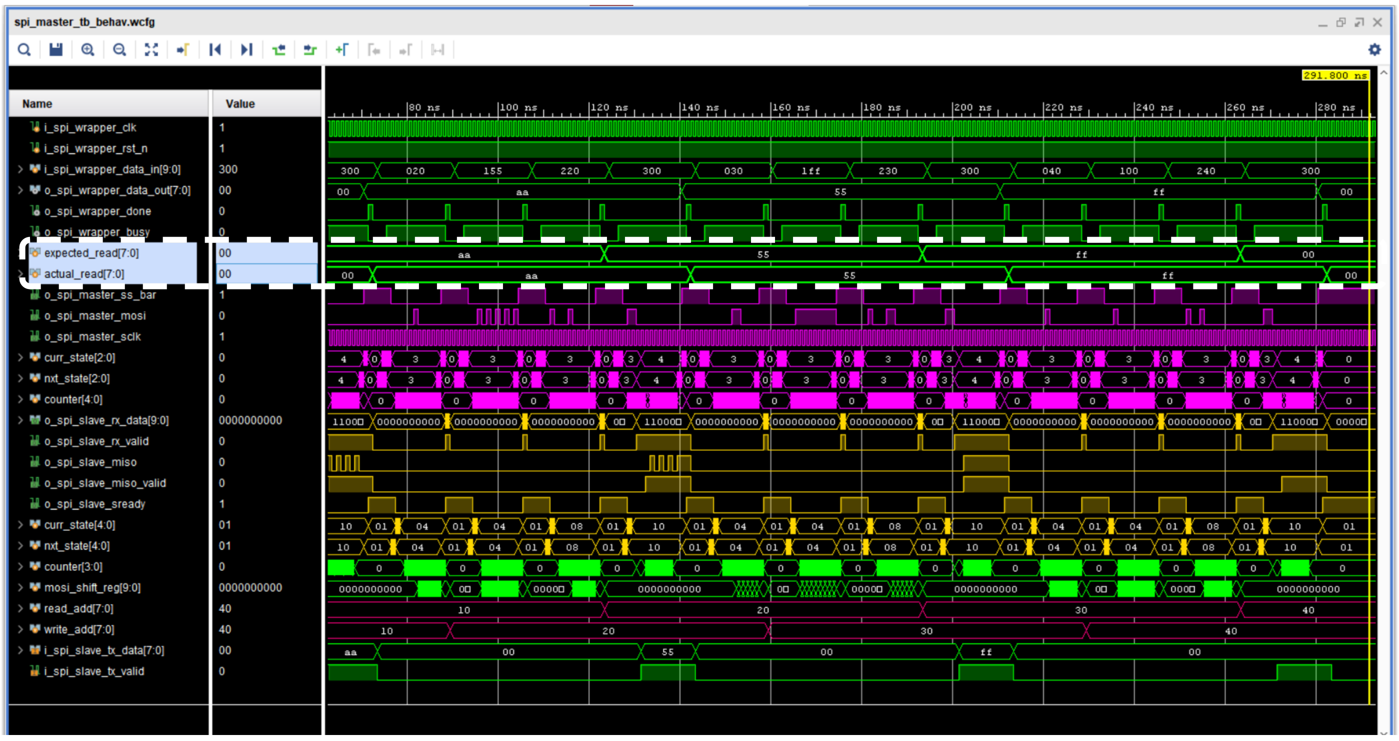
Testbench Case Table:

Case	Operation	Command Code	Input Value	Expected Behavior	Observed Output
1	Write Address	2'b00	(0x10)	Sets internal write address in slave	done asserted after operation
2	Write Data	2'b01	(0xAA)	Stores data at previously set write address	done asserted after operation
3	Read Address	2'b10	(0x10)	Sets internal read address in slave	done asserted after operation
4	Read Data	2'b11	(0x00)	Reads back data from the address set above	Output matches written data

Repeated Over:

Iteration	Address	Data
0	0x10	0xAA
1	0x20	0x55
2	0x30	0xFF
3	0x40	0x00

Each of the 4 cases is run for all 4 address/data pairs above, ensuring full functional coverage.

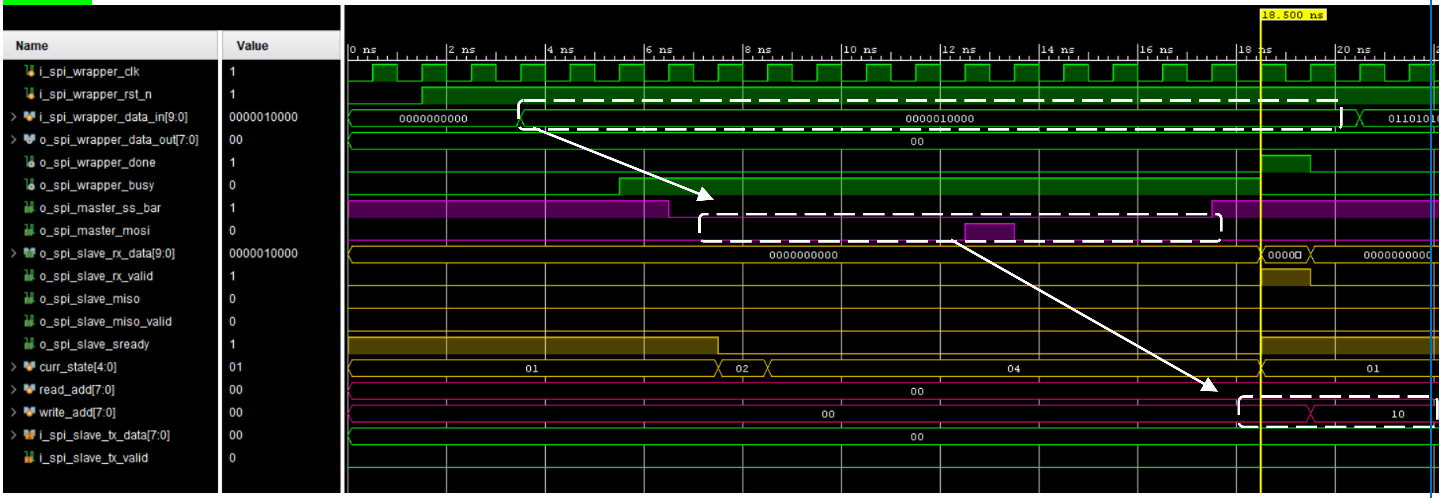


The comparison between the expected output and the actual output shows perfect alignment, confirming the correctness and functionality.

Iteration 0:

Case	Operation	Command (10-bit)	Description	Expected Output	Observed Output	Status
1	Write Address	00_00010000	Store address 0x10 in RAM	Done = 1	Done asserted	Passed
2	Write Data	01_10101010	Write data 0xAA to address 0x10	Done = 1	Done asserted	Passed
3	Read Address	10_00010000	Set read address to 0x10	Done = 1	Done asserted	Passed
4	Read Data	11_00000000	Read data from address 0x10	0xAA	0xAA	Passed

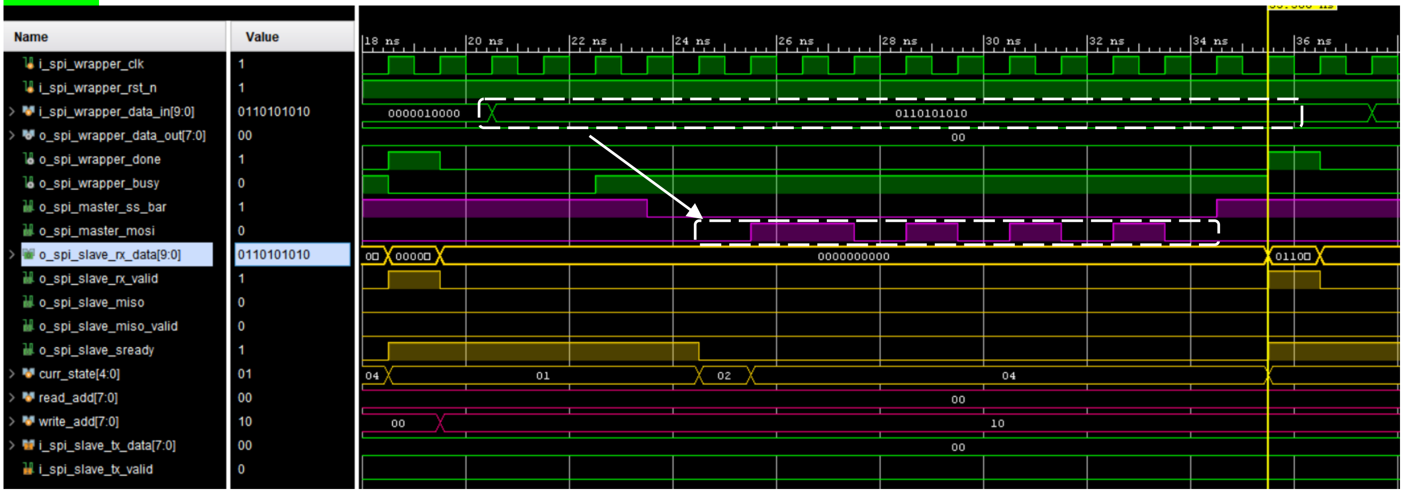
Case1



Name	Value
i_spi_wrapper_clk	1
i_spi_wrapper_rst_n	1
i_spi_wrapper_data_in[9:0]	0000010000
o_spi_wrapper_data_out[7:0]	00
o_spi_wrapper_done	1
o_spi_wrapper_busy	0
o_spi_master_ss_bar	1
o_spi_master_mosi	0
o_spi_slave_rx_data[9:0]	0000010000
o_spi_slave_rx_valid	1
o_spi_slave_miso	0
o_spi_slave_miso_valid	0
o_spi_slave_sready	1
curr_state[4:0]	01
read_add[7:0]	00
write_add[7:0]	00
i_spi_slave_tx_data[7:0]	00
i_spi_slave_tx_valid	0

- Command Sent (10-bit): 00_00010000 (Control = 00, Address = 0x10)
- The SPI master sends a command to store the address 0x10 into the slave's internal register. This address will later be used as the target for writing data.

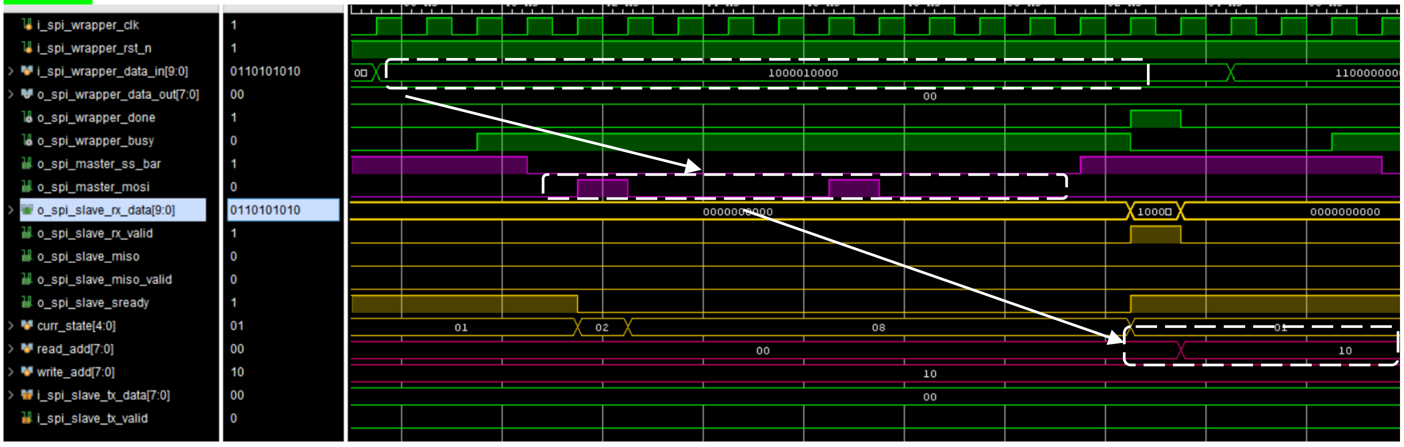
Case2



Name	Value
i_spi_wrapper_clk	1
i_spi_wrapper_rst_n	1
i_spi_wrapper_data_in[9:0]	0110101010
o_spi_wrapper_data_out[7:0]	00
o_spi_wrapper_done	1
o_spi_wrapper_busy	0
o_spi_master_ss_bar	1
o_spi_master_mosi	0
o_spi_slave_rx_data[9:0]	0110101010
o_spi_slave_rx_valid	1
o_spi_slave_miso	0
o_spi_slave_miso_valid	0
o_spi_slave_sready	1
curr_state[4:0]	01
read_add[7:0]	00
write_add[7:0]	10
i_spi_slave_tx_data[7:0]	00
i_spi_slave_tx_valid	0

- Command Sent (10-bit): 01_10101010 (Control = 01, Data = 0xAA)
- The SPI master sends the data byte 0xAA to be written at the previously stored address 0x10. This completes the write operation to RAM.

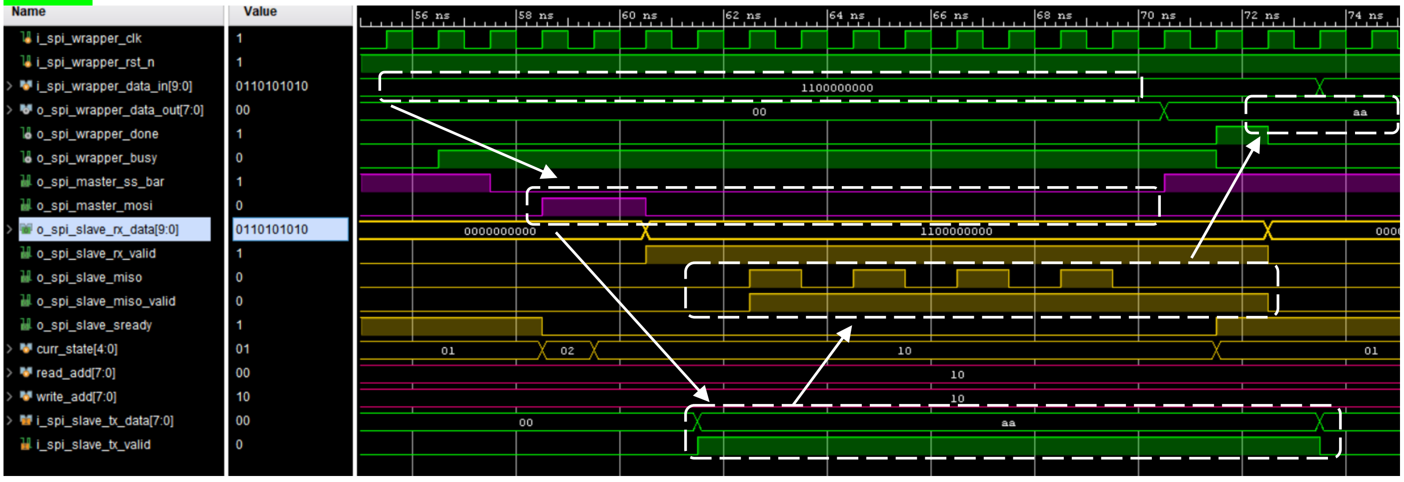
Case3



i_spi_wrapper_clk	1
i_spi_wrapper_rst_n	1
i_spi_wrapper_data_in[9:0]	0110101010
o_spi_wrapper_data_out[7:0]	00
o_spi_wrapper_done	1
o_spi_wrapper_busy	0
o_spi_master_ss_bar	1
o_spi_master_mosi	0
o_spi_slave_rx_data[9:0]	0110101010
o_spi_slave_rx_valid	1
o_spi_slave_miso	0
o_spi_slave_miso_valid	0
o_spi_slave_sready	1
curr_state[4:0]	01
read_add[7:0]	00
write_add[7:0]	10
i_spi_slave_tx_data[7:0]	00
i_spi_slave_tx_valid	0

- Command Sent (10-bit): 10_00010000 (Control = 10, Address = 0x10)
- The SPI master sends a read address command to access the memory location 0x10. This sets up the RAM to prepare the data at that address for retrieval.

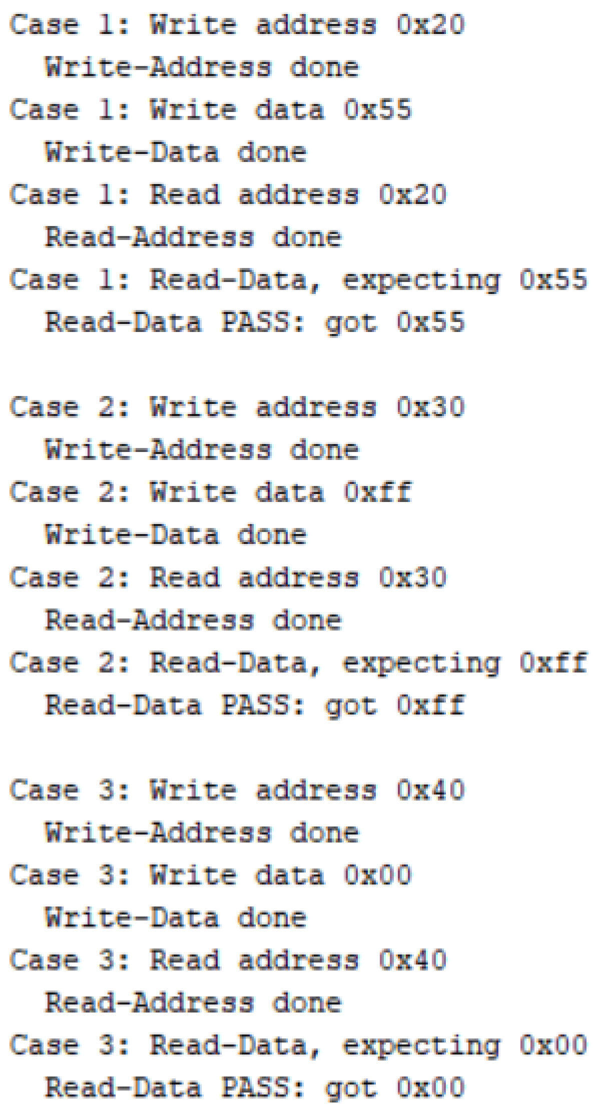
Case4



Name	Value
i_spi_wrapper_clk	1
i_spi_wrapper_rst_n	1
i_spi_wrapper_data_in[9:0]	0110101010
o_spi_wrapper_data_out[7:0]	00
o_spi_wrapper_done	1
o_spi_wrapper_busy	0
o_spi_master_ss_bar	1
o_spi_master_mosi	0
o_spi_slave_rx_data[9:0]	0110101010
o_spi_slave_rx_valid	1
o_spi_slave_miso	0
o_spi_slave_miso_valid	0
o_spi_slave_sready	1
curr_state[4:0]	01
read_add[7:0]	00
write_add[7:0]	10
i_spi_slave_tx_data[7:0]	00
i_spi_slave_tx_valid	0

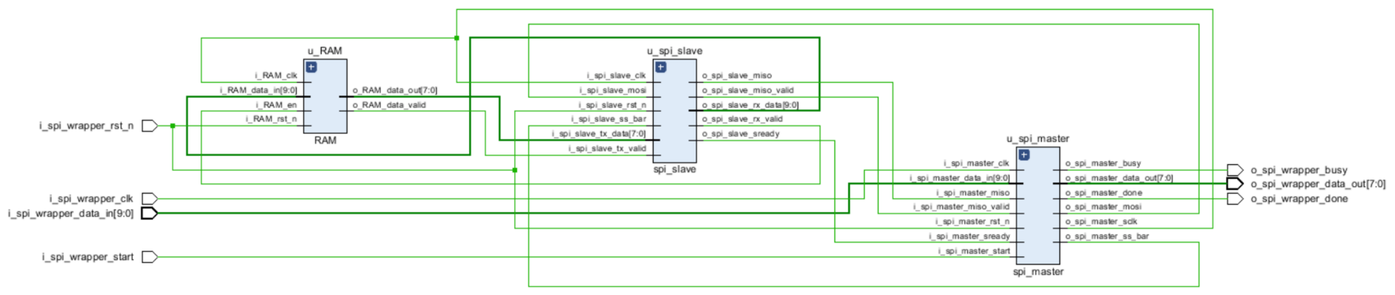
- Command Sent (10-bit): 11_00000000 (Control = 11, data bits = don’t care)
- This command triggers the SPI slave to output the data stored at the previously received read address (0x10).

correct output observed in iterations 0, 1, and 2



```
Case 1: Write address 0x20
  Write-Address done
Case 1: Write data 0x55
  Write-Data done
Case 1: Read address 0x20
  Read-Address done
Case 1: Read-Data, expecting 0x55
  Read-Data PASS: got 0x55

Case 2: Write address 0x30
  Write-Address done
Case 2: Write data 0xff
  Write-Data done
Case 2: Read address 0x30
  Read-Address done
Case 2: Read-Data, expecting 0xff
  Read-Data PASS: got 0xff

Case 3: Write address 0x40
  Write-Address done
Case 3: Write data 0x00
  Write-Data done
Case 3: Read address 0x40
  Read-Address done
Case 3: Read-Data, expecting 0x00
  Read-Data PASS: got 0x00
```

Elaborated Design

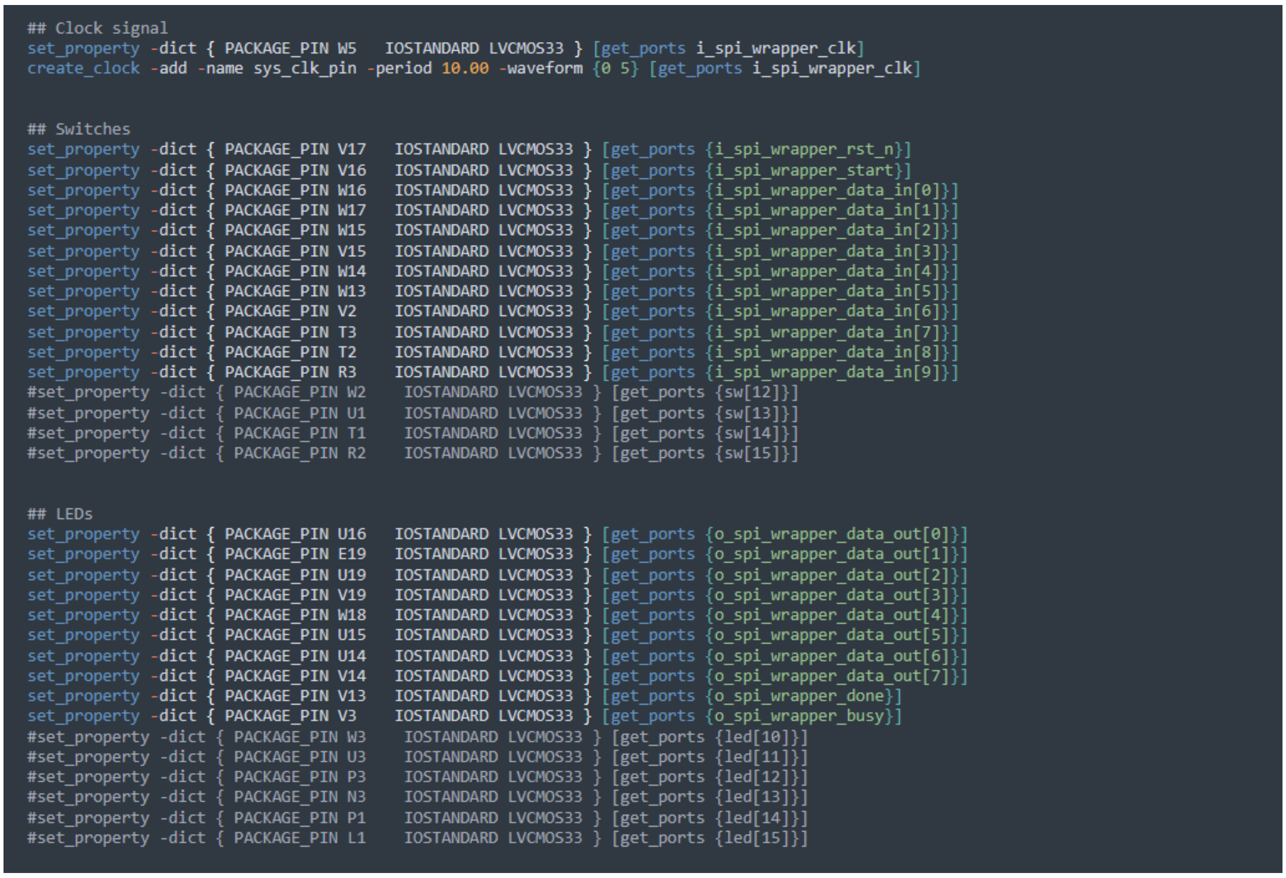


FPGA

Xilinx Artix-7 FPGA xc7a35ticpg236-1L

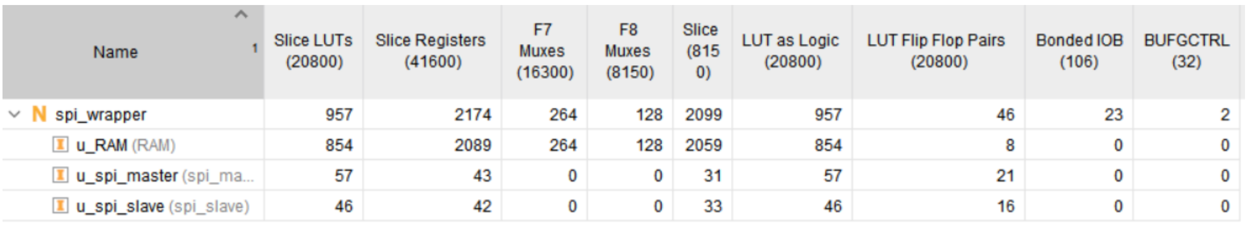
Parameter	Value
Family	Artix-7
Device	XC7A35T
Package	CPG236 (CSBGA 236-ball package)

Constraints



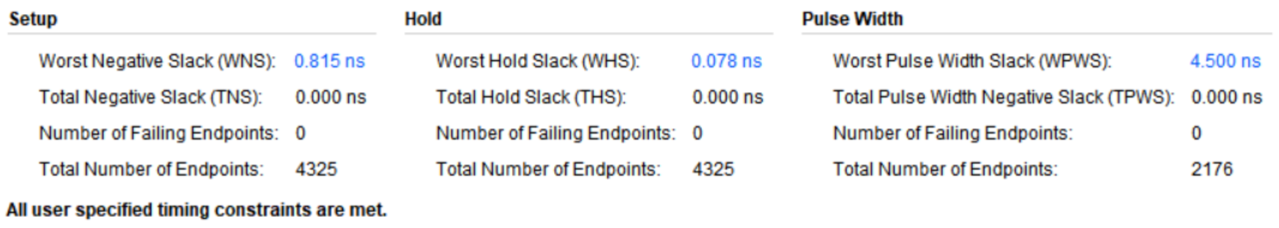
```
## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports i_spi_wrapper_clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports i_spi_wrapper_clk]


## Switches
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_rst_n}]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_start}]
set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_data_in[0]}]
set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_data_in[1]}]
set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_data_in[2]}]
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_data_in[3]}]
set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_data_in[4]}]
set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_data_in[5]}]
set_property -dict { PACKAGE_PIN V2    IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_data_in[6]}]
set_property -dict { PACKAGE_PIN T3    IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_data_in[7]}]
set_property -dict { PACKAGE_PIN T2    IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_data_in[8]}]
set_property -dict { PACKAGE_PIN R3    IOSTANDARD LVCMOS33 } [get_ports {i_spi_wrapper_data_in[9]}]
#set_property -dict { PACKAGE_PIN W2    IOSTANDARD LVCMOS33 } [get_ports {sw[12]}]
#set_property -dict { PACKAGE_PIN U1    IOSTANDARD LVCMOS33 } [get_ports {sw[13]}]
#set_property -dict { PACKAGE_PIN T1    IOSTANDARD LVCMOS33 } [get_ports {sw[14]}]
#set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {sw[15]}]


## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {o_spi_wrapper_data_out[0]}]
set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {o_spi_wrapper_data_out[1]}]
set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 } [get_ports {o_spi_wrapper_data_out[2]}]
set_property -dict { PACKAGE_PIN V19   IOSTANDARD LVCMOS33 } [get_ports {o_spi_wrapper_data_out[3]}]
set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 } [get_ports {o_spi_wrapper_data_out[4]}]
set_property -dict { PACKAGE_PIN U15   IOSTANDARD LVCMOS33 } [get_ports {o_spi_wrapper_data_out[5]}]
set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {o_spi_wrapper_data_out[6]}]
set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports {o_spi_wrapper_data_out[7]}]
set_property -dict { PACKAGE_PIN V13   IOSTANDARD LVCMOS33 } [get_ports {o_spi_wrapper_done}]
set_property -dict { PACKAGE_PIN V3    IOSTANDARD LVCMOS33 } [get_ports {o_spi_wrapper_busy}]
#set_property -dict { PACKAGE_PIN W3    IOSTANDARD LVCMOS33 } [get_ports {led[10]}]
#set_property -dict { PACKAGE_PIN U3    IOSTANDARD LVCMOS33 } [get_ports {led[11]}]
#set_property -dict { PACKAGE_PIN P3    IOSTANDARD LVCMOS33 } [get_ports {led[12]}]
#set_property -dict { PACKAGE_PIN N3    IOSTANDARD LVCMOS33 } [get_ports {led[13]}]
#set_property -dict { PACKAGE_PIN P1    IOSTANDARD LVCMOS33 } [get_ports {led[14]}]
#set_property -dict { PACKAGE_PIN L1    IOSTANDARD LVCMOS33 } [get_ports {led[15]}]
```

Utilization report



Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Bonded IOB (106)	BUFGCTRL (32)
spi_wrapper	957	2174	264	128	2099	957	46	23	2
u_RAM (RAM)	854	2089	264	128	2059	854	8	0	0
u_spi_master (spi_ma...	57	43	0	0	31	57	21	0	0
u_spi_slave (spi_slave)	46	42	0	0	33	46	16	0	0

Time report



Setup

Worst Negative Slack (WNS):	0.815 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	4325

Hold

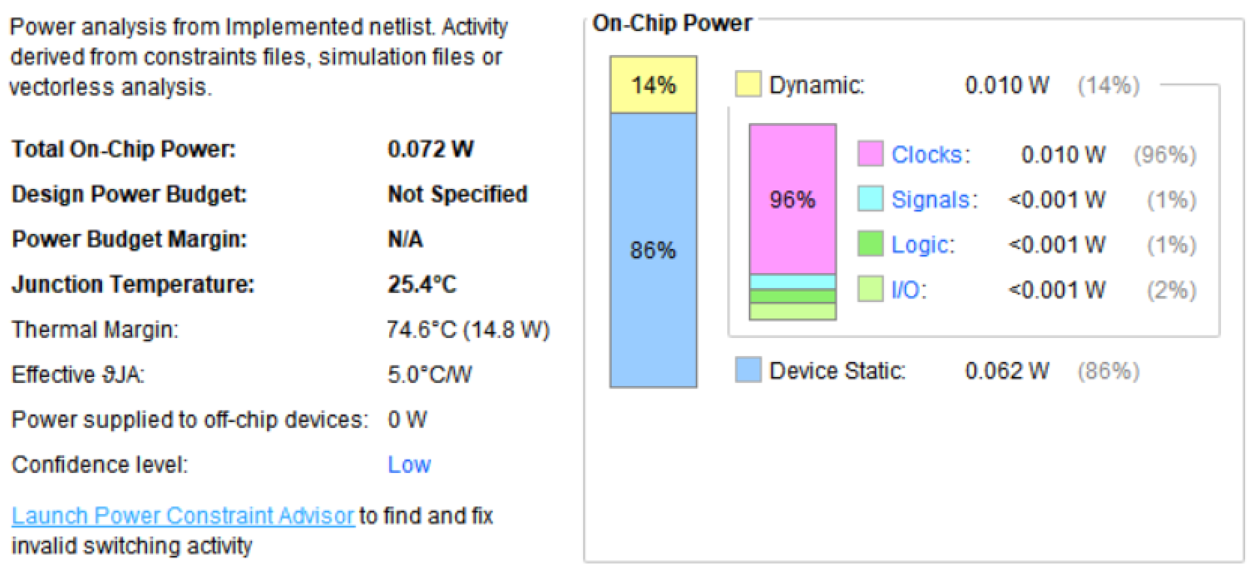
Worst Hold Slack (WHS):	0.078 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	4325

Pulse Width

Worst Pulse Width Slack (WPWS):	4.500 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	2176

All user specified timing constraints are met.

Power report



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

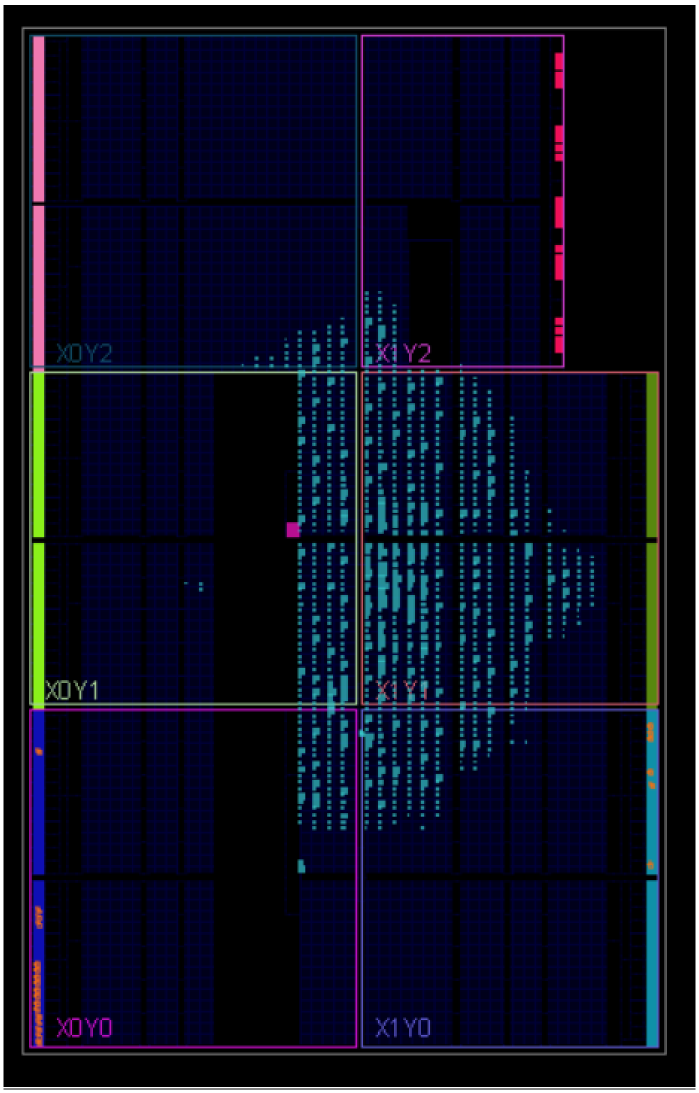
Total On-Chip Power:	**0.072 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.4°C**
Thermal Margin:	74.6°C (14.8 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

🟨	Dynamic:	0.010 W	(14%)
🟪	Clocks:	0.010 W	(96%)
🟦	Signals:	<0.001 W	(1%)
🟩	Logic:	<0.001 W	(1%)
🟩	I/O:	<0.001 W	(2%)
🟦	Device Static:	0.062 W	(86%)

Implementation


X0Y2
X1Y2
X0Y1
X1Y1
X0Y0
X1Y0