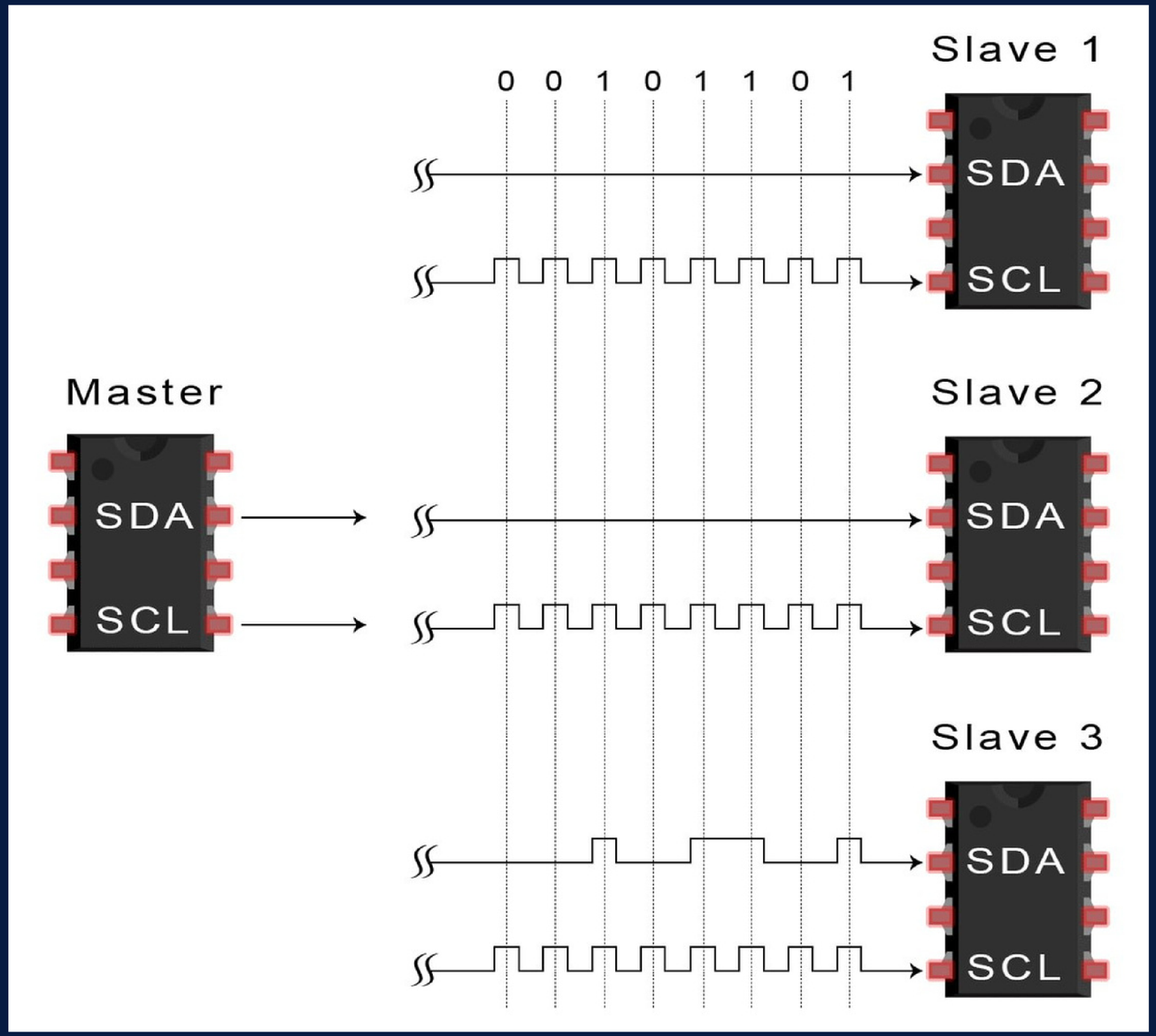
I2C

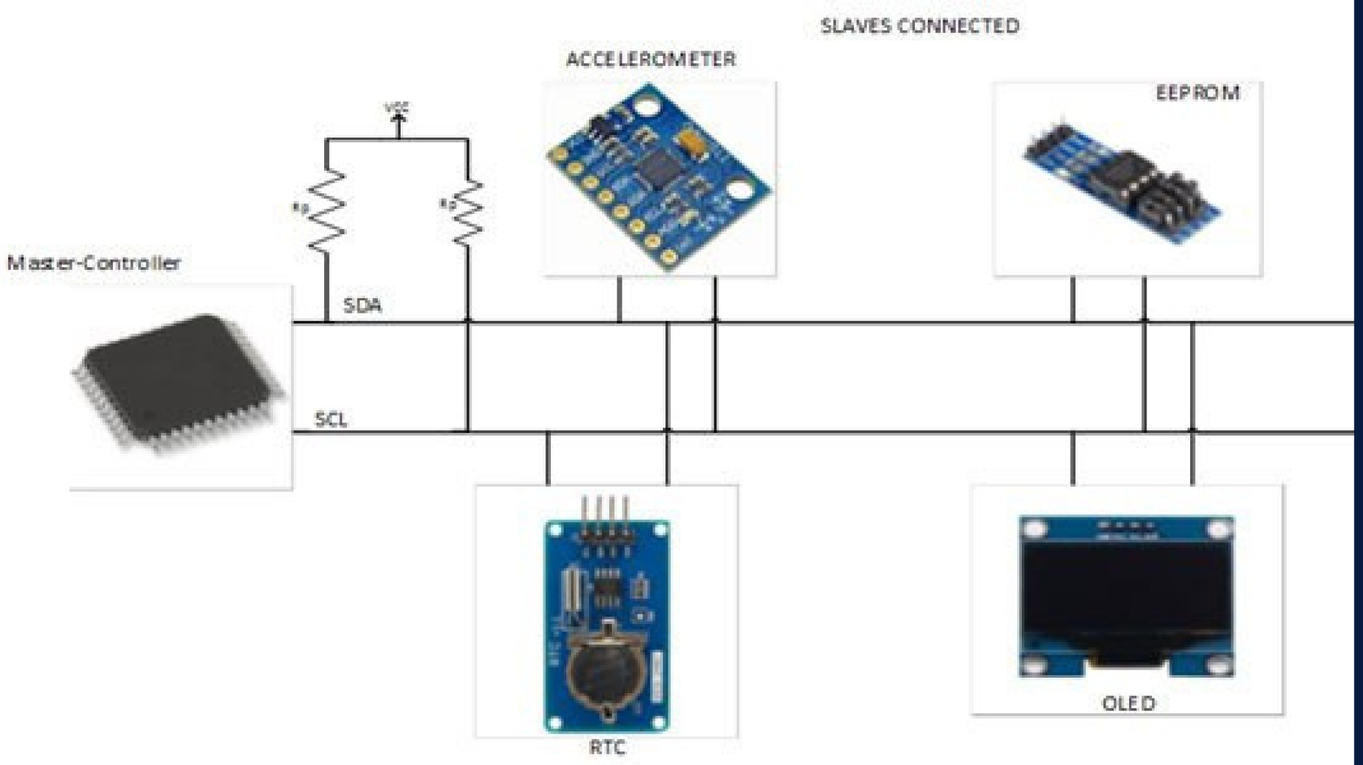
I2C (Inter-Integrated Circuit) stands as a highly prevalent and efficient serial communication protocol, enabling seamless data exchange between integrated circuits through a simple two-wire interface. Originally developed by Philips Semiconductors, now NXP Semiconductors, in the early 1980s, I2C has since established itself as the go-to standard for inter-chip communication across the electronics industry.

The I2C protocol is known for its simplicity and versatility, making it suitable for various applications ranging from simple sensor networks to complex system-on-chip (SoC) designs. It enables efficient data transmission between devices by utilizing a master-slave architecture. In this setup, a master device initiates and controls the communication, while the slave devices respond to the master's commands and provide the requested data.

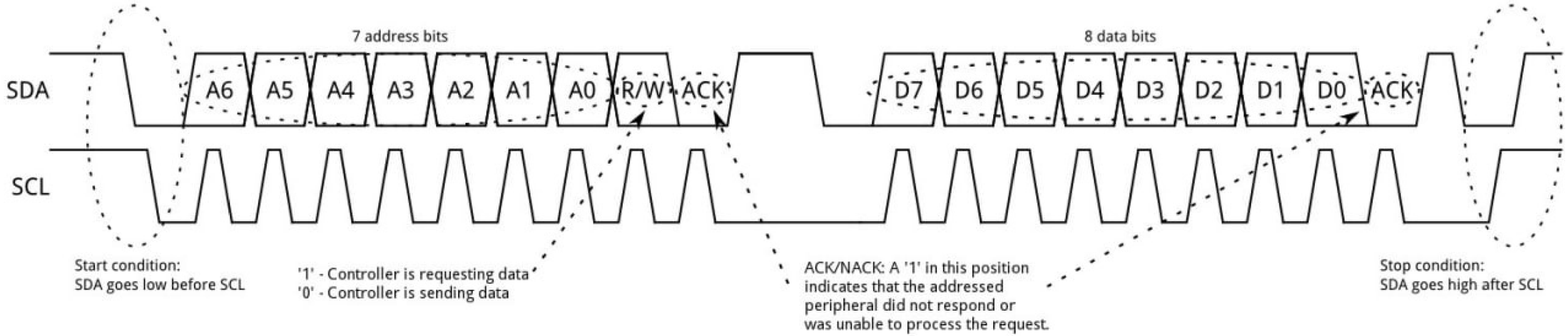
I2C supports various data transfer modes, including standard mode (up to 100 kbit/s), fast mode (up to 400 kbit/s), fast mode plus (up to 1 Mbit/s), and high-speed mode (up to 3.4 Mbit/s). The choice of data transfer mode depends on the specific requirements of the application, such as the desired data rate and the maximum capacitance and distance of the bus.

The two-wire interface in I2C consists of a serial data line (SDA) and a serial clock line (SCL). These lines are bidirectional, meaning that both the master and the slave devices can send and receive data over them. The SDA line carries the actual data, while the SCL line controls the timing of the communication.





I2C is a versatile and widely adopted communication protocol that provides a simple and efficient way for integrated circuits to communicate with each other. Its widespread use in the electronics industry has led to the availability of a wide range of I2C-compatible devices, making it an integral part of many electronic systems.



I2C Data Frame

What has been implemented here?

The Design

I have developed the design of an I2C master controller using SystemVerilog

UVM Verification Testbench

"I created the UVM testbench for an I2C master controller with a scoreboard, monitor, sequencer, and environment..etc.

Introduction

In the design of the I2C master, I assumed we are operating in fast mode with a speed of 400 kbits/sec. To transmit a single bit, I divided the pulse into four regions: data 1, data 2, data 3, and data 4. Additionally, the clock remains high while the data is stable. To achieve the desired division, I calculated the divisor as 5 MHz the frequency of the clock signal divided by (4 * 400 kbits/sec).

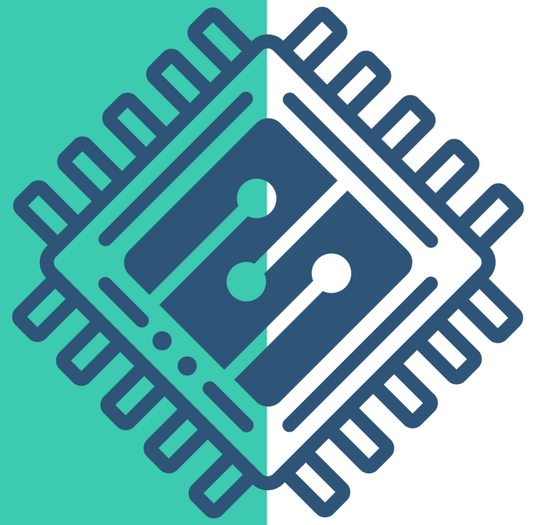
The features that have not been achieved in our design

feature 1

Our design does not support multi-master

feature 2

Our design does not support multiple slaves; it only supports one slave



Part of The SystemVerilog Code

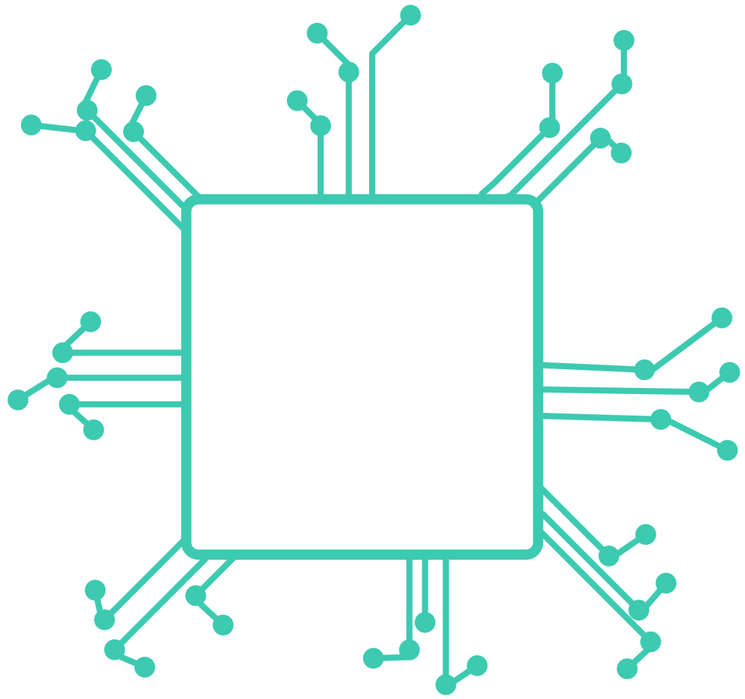
```
1
2
3
4
5   module I2C_Master#(
6                                   parameter Start_CMD = 3'b000,
7                                   parameter Stop_CMD = 3'b001,
8                                   parameter Read_CMD = 3'b010,
9                                   parameter Write_CMD = 3'b011,
10                                  parameter Restart_CMD = 3'b100
11                                                                  )
12
13                      (
14                                  input logic clk,
15                                  input logic reset,
16                                  input logic wr_i2c,
17                                  input logic[7:0]  data_in,
18                                  input logic[2:0] cmd,
19                                  input logic[15:0] dvsr,
20                                  output logic[7:0] data_out,
21                                  output logic ready,
22                                  output logic done_tick,
23                                  output logic ACK,
24                                  output tri SCL,
25                                  inout tri SDA
26                                                              );
27
28
29
30
31
32     // FSM state data type
33      typedef enum {idle, hold, start1, start2, data1, data2, data3, data4,
34          data_end, restart, stop1, stop2   } State_Type;
35
36
```

```
39
40      //Intermediate signals declaration
41
42      State_Type State_reg, State_next;
43
44      logic[8:0] TX_reg, TX_next; // 8 bit data + ack bit
45
46      logic[8:0] RX_reg, RX_next; // 8 bit data + ack bit
47
48      logic[2:0] CMD_reg, CMD_next;
49
50      logic[15:0] Count_reg, Count_next;
51
52      logic[3:0] NumBits_reg, NumBits_next; //number of bits transmitted or received
53
54      logic[15:0] Quarter, Half; //quarter cycle counts,half cycle counts
55
56      logic sda_t,sda_reg,scl_t,scl_reg,data_proc;
57
58      logic link,NACK; //write into bus , negative acknowledgement
59
60      logic done_tick_reg,done_tick_next, ready_t;
61
62
63
64
65      // SCL and SDA are pulled up resistors so the high by default
66
67      assign SCL = scl_reg ? 1'b1 : 1'b0 ;  // master is the only device that drives clock signal
68
69
70
71      /*      if master read and the num bits less than 8
72         or write and the current bit is the 8th
73         then connect the sda line to the master */
74
75      assign link = (data_proc && CMD_reg==Write_CMD && NumBits_reg==8) ||
```

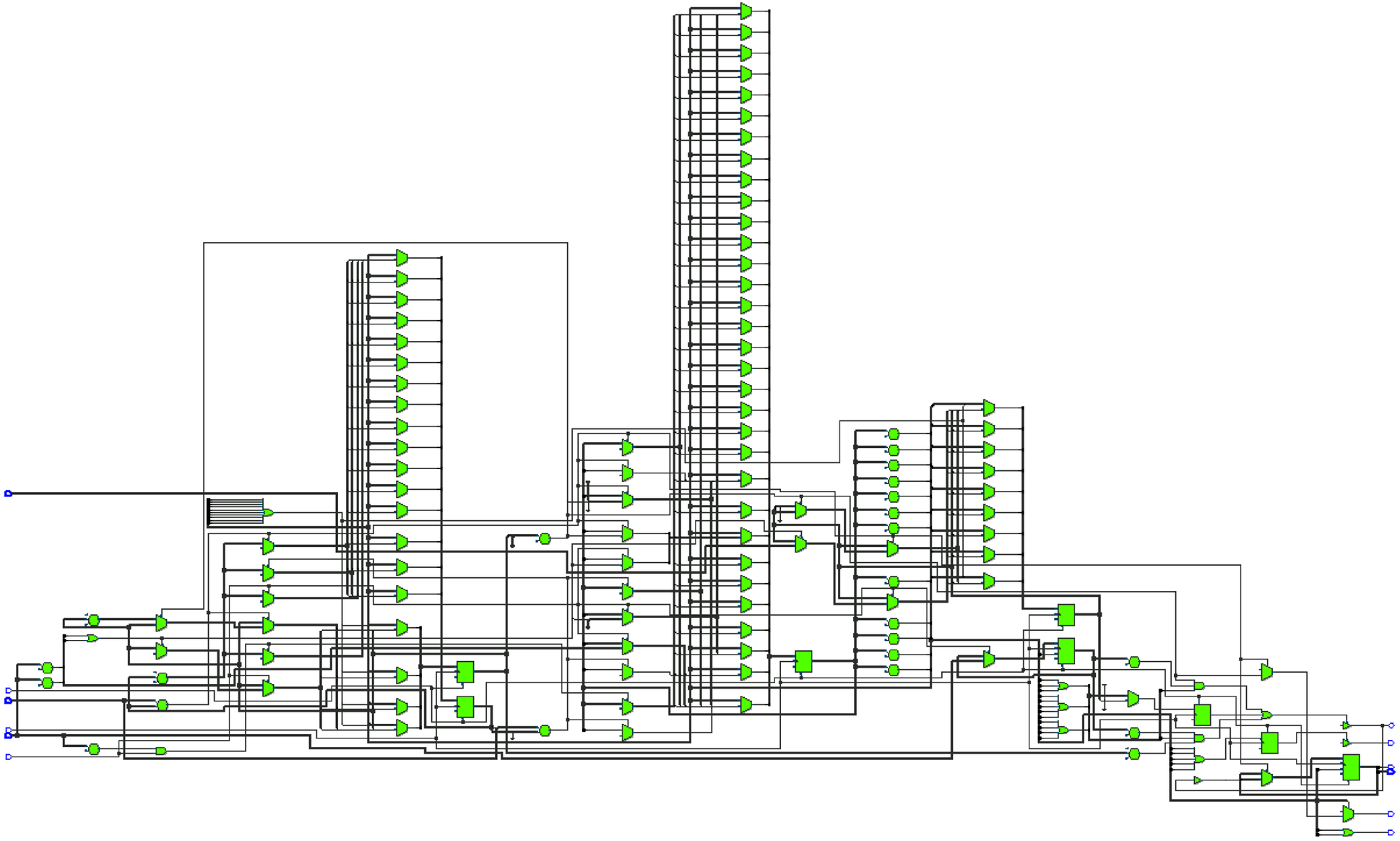
```
79
80    assign SDA = (link || sda_reg) ? 1'b1 : 1'b0 ;
81
82
83    assign Quarter = dvsr;
84
85
86    assign Half = dvsr * 2;
87
88
89
90
91
92    always_ff@(posedge clk, posedge reset) begin
93
94
95
96       if(reset) begin
97
98                scl_reg <= 1;
99                     sda_reg <= 1;
100
101                 end
102
103
104           else begin
105
106               scl_reg <= scl_t;
107                    sda_reg <= sda_t;
108
109
110                 end
111
112
113
114   end
115
```

```
119
120
121
122
123
124     always_ff@(posedge clk, posedge reset) begin
125
126
127      if(reset) begin
128
129                 State_reg = idle;
130                 TX_reg <= 0;
131                 RX_reg <= 0;
132                 CMD_reg <= 0;
133                 Count_reg <= 0;
134                 NumBits_reg <= 0;
135                 done_tick_reg <= 0;
136
137                   end
138
139
140             else begin
141
142                 State_reg = State_next;
143                 TX_reg <= TX_next;
144                 RX_reg <= RX_next;
145                 CMD_reg <= CMD_next;
146                 Count_reg <= Count_next;
147                 NumBits_reg <= NumBits_next;
148                 done_tick_reg <= done_tick_next;
149
150                   end
151
152
153     end
154
155
```

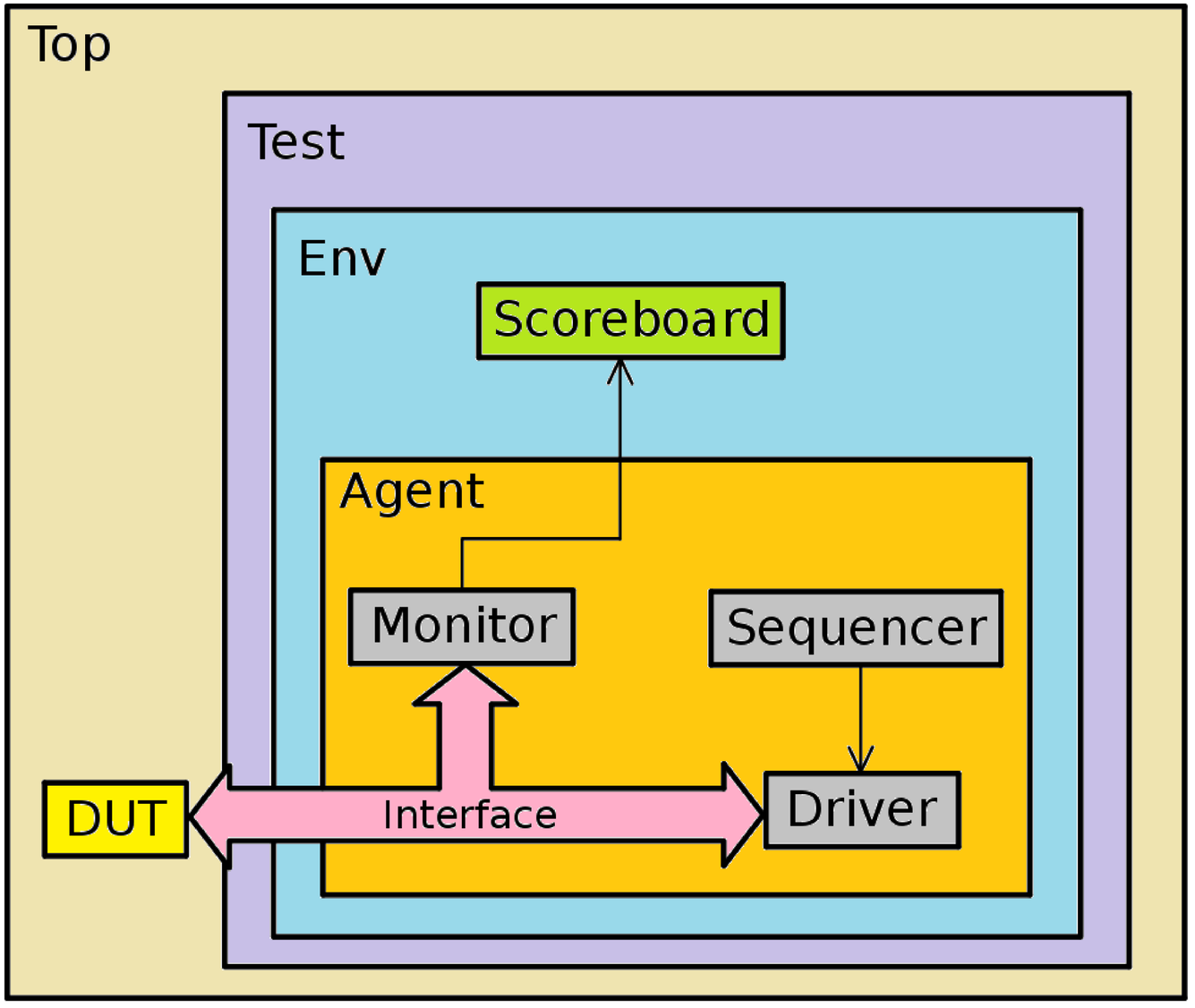
```
159
160
161
162     always_comb begin
163
164
165           State_next = State_reg;
166           TX_next = TX_reg;
167           RX_next = RX_reg;
168           CMD_next = CMD_reg;
169           Count_next = Count_reg +1 ;
170           NumBits_next = NumBits_reg;
171           sda_t = 1'b1;
172           scl_t = 1'b1;
173           done_tick_next = done_tick_reg;
174           ready_t = 1'b0;
175           data_proc = 1'b0;
176
177
178                     case (State_reg)
179
180
181
182
183                        idle: begin
184
185
186                                    ready_t = 1'b1;
187
188                                    if(wr_i2c && cmd==Start_CMD) begin
189
190                                       State_next = start1;
191                                       Count_next = 0;
192
193                                    end
194
```



The Synthesis Results

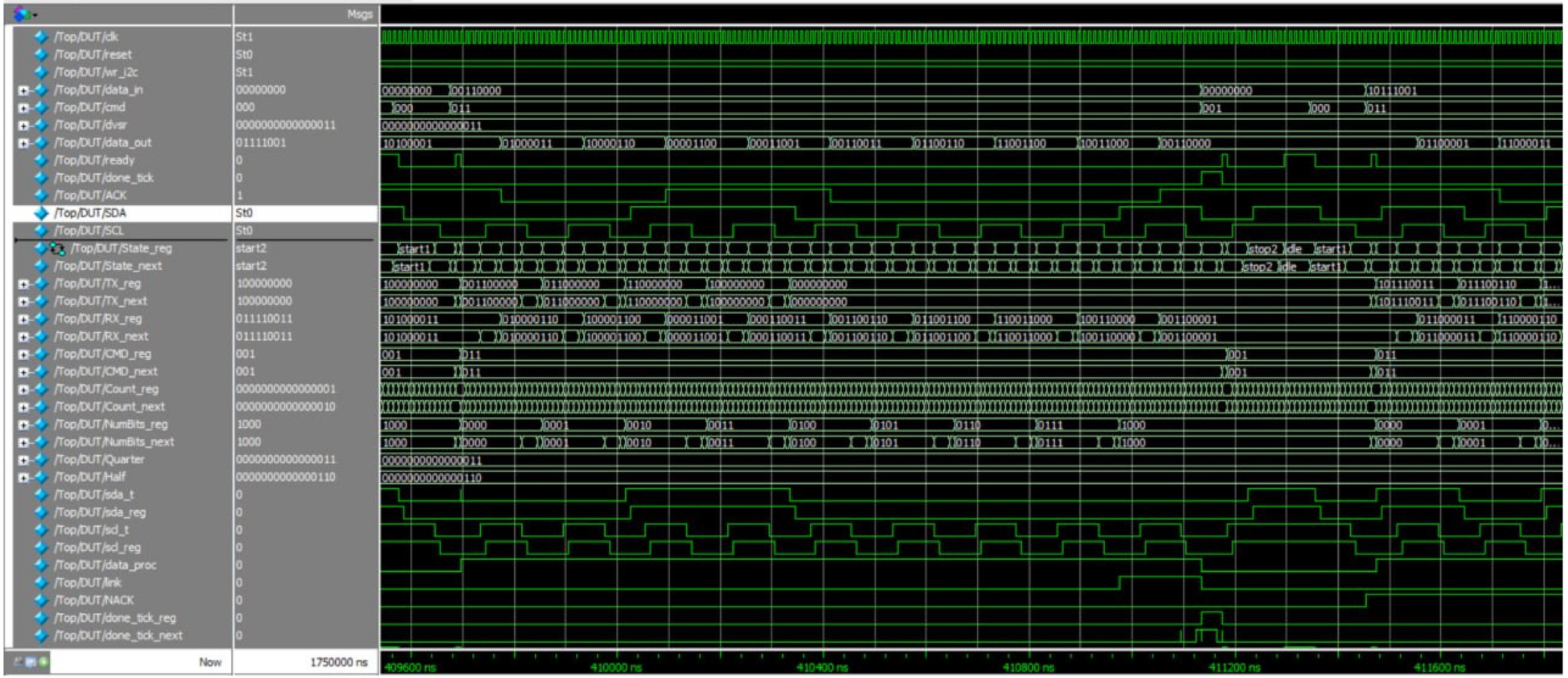


The UVM Architecture


Top
Test
Env
Scoreboard
Agent
Monitor
Sequencer
DUT
Interface
Driver

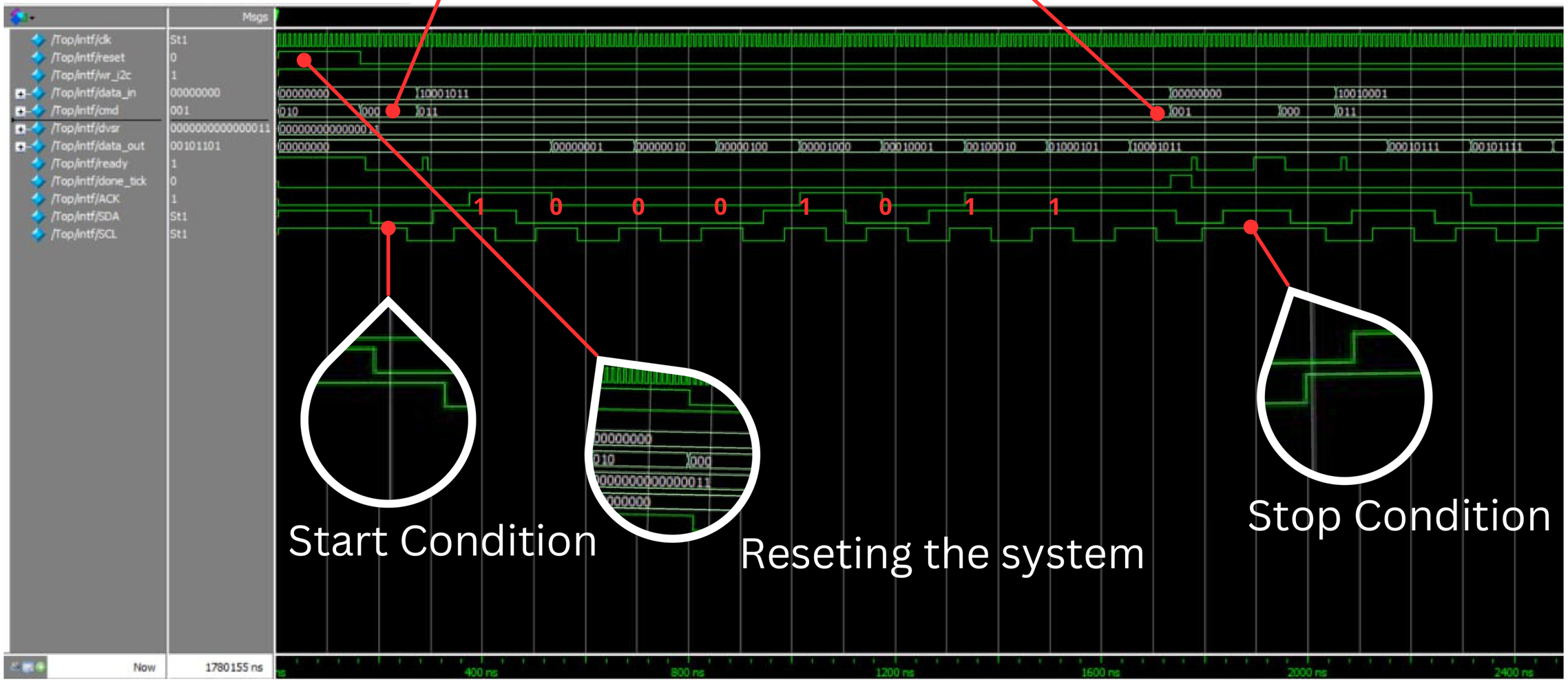
You can find all codes details on the GitHub repository

The Simulation Results


Msgs
/Top/DUT/clk
/Top/DUT/reset
/Top/DUT/wr_i2c
/Top/DUT/data_in
/Top/DUT/cmd
/Top/DUT/dvsr
/Top/DUT/data_out
/Top/DUT/ready
/Top/DUT/done_tick
/Top/DUT/ACK
/Top/DUT/SDA
/Top/DUT/SCL
/Top/DUT/State_reg
/Top/DUT/State_next
/Top/DUT/TX_reg
/Top/DUT/TX_next
/Top/DUT/RX_reg
/Top/DUT/RX_next
/Top/DUT/CMD_reg
/Top/DUT/CMD_next
/Top/DUT/Count_reg
/Top/DUT/Count_next
/Top/DUT/NumBits_reg
/Top/DUT/NumBits_next
/Top/DUT/Quarter
/Top/DUT/Half
/Top/DUT/sda_t
/Top/DUT/sda_reg
/Top/DUT/scl_t
/Top/DUT/scl_reg
/Top/DUT/data_proc
/Top/DUT/link
/Top/DUT/NACK
/Top/DUT/done_tick_reg
/Top/DUT/done_tick_next
Now
1750000 ns
409600 ns
410000 ns
410400 ns
410800 ns
411200 ns
411600 ns

Start Condition State

Stop Condition State



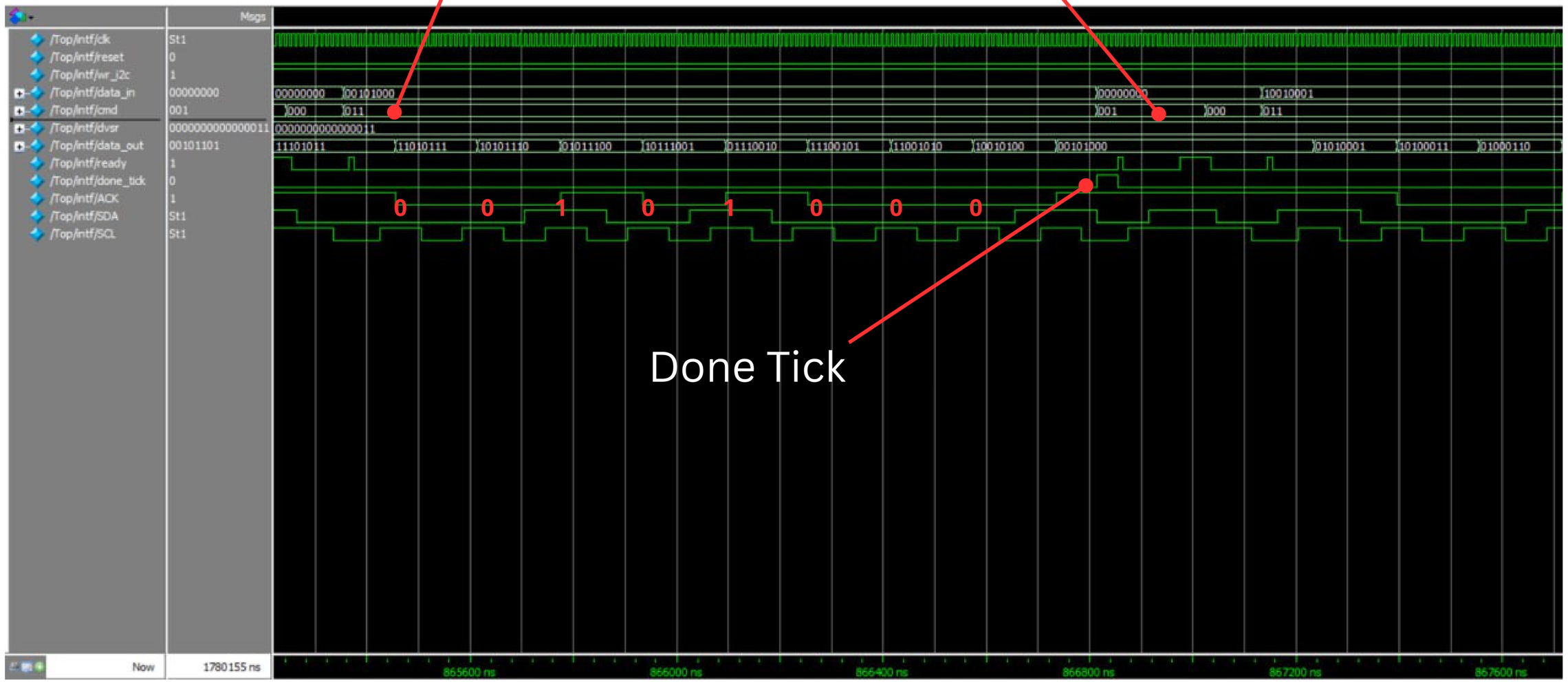
Start Condition

Reseting the system

Stop Condition

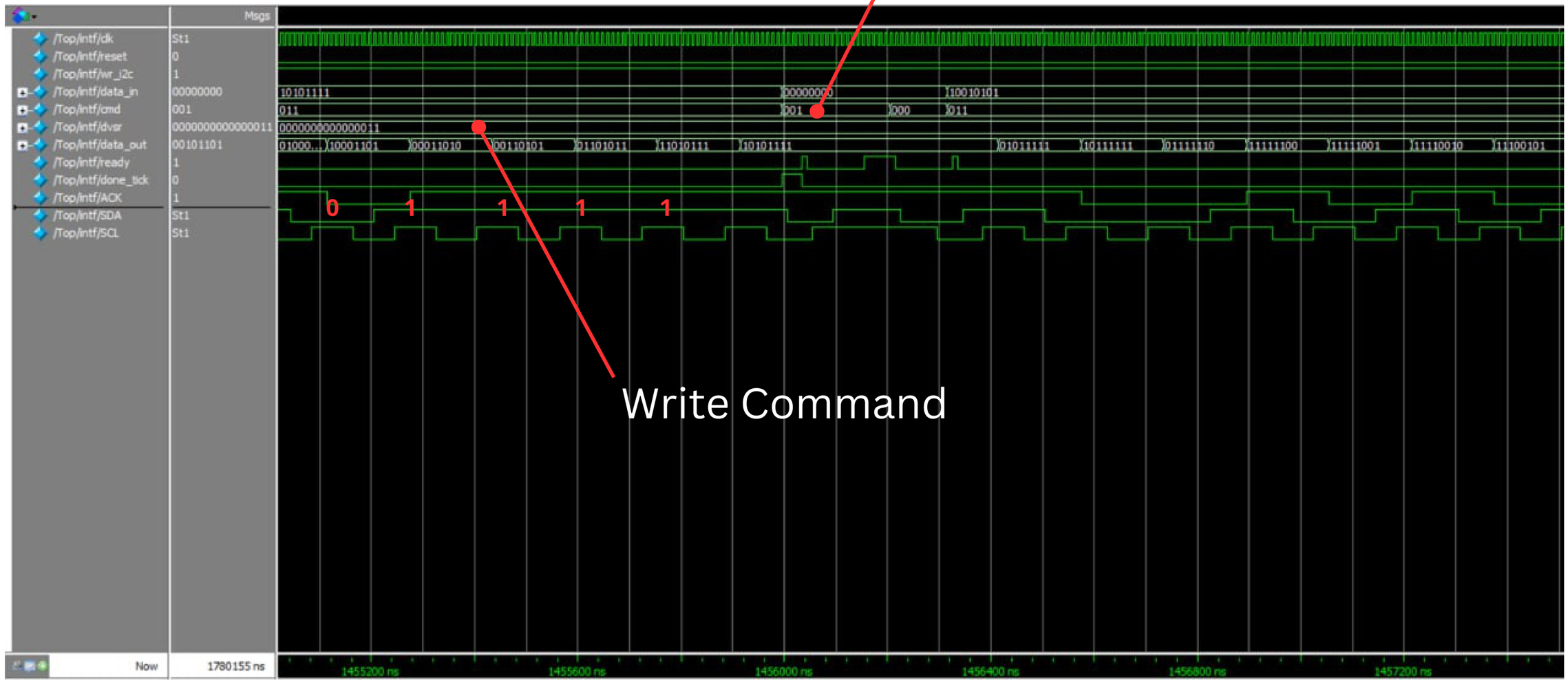
Start Condition State

Stop Condition State

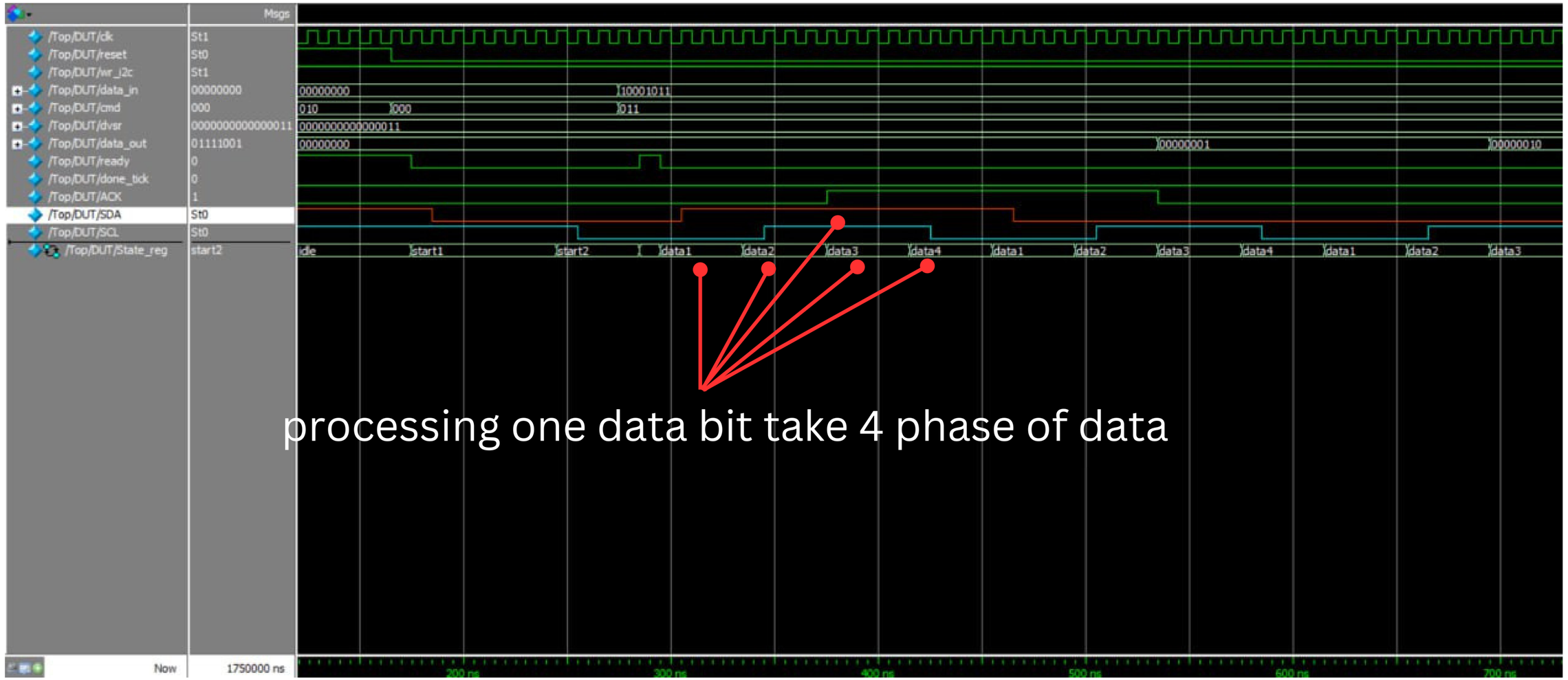


Done Tick

Stop Condition



Write Command



processing one data bit take 4 phase of data

The Simulation Log Results

```
# UVM_INFO I2C_Test.sv(33) @ 0: uvm_test_top [I2C_Test] Inside constructor of I2C Test Class
# UVM_INFO @ 0: reporter [RNTST] Running test I2C_Test...
# UVM_INFO I2C_Test.sv(52) @ 0: uvm_test_top [I2C_Test] Inside build phase of I2C Test Class
# UVM_INFO Environment.sv(22) @ 0: uvm_test_top.I2C_environment [Environment] Inside constructor of Environment Class
# UVM_INFO Environment.sv(39) @ 0: uvm_test_top.I2C_environment [Environment] Inside build phase of Environment Class
# UVM_INFO Agent.sv(22) @ 0: uvm_test_top.I2C_environment.I2C_Agent [Agent] Inside constructor of Agent Class
# UVM_INFO Scoreboard.sv(30) @ 0: uvm_test_top.I2C_environment.I2C_Scoreboard [Scoreboard] Inside constructor of Scoreboard Class
# UVM_INFO Agent.sv(39) @ 0: uvm_test_top.I2C_environment.I2C_Agent [Agent] Inside build phase of Agent Class
# UVM_INFO Monitor.sv(31) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_monitor [Monitor] Inside constructor of Monitor Class
# UVM_INFO Driver.sv(25) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_driver [Driver] Inside constructor of Driver Class
# UVM_INFO Sequencer.sv(16) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_sequencer [Sequencer] Inside constructor of Sequencer Class
# UVM_INFO Driver.sv(42) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_driver [Driver] Inside build phase of Driver Class
# UVM_INFO Monitor.sv(50) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_monitor [Monitor] Inside build phase of Monitor Class
# UVM_INFO Sequencer.sv(33) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_sequencer [Sequencer] Inside build phase of Sequencer Class
# UVM_INFO Scoreboard.sv(48) @ 0: uvm_test_top.I2C_environment.I2C_Scoreboard [Scoreboard] Inside build phase of Scoreboard Class
# UVM_INFO Driver.sv(67) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_driver [Driver] Inside connect phase of Driver Class
# UVM_INFO Monitor.sv(78) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_monitor [Monitor] Inside connect phase of Monitor Class
# UVM_INFO Sequencer.sv(51) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_sequencer [Sequencer] Inside connect phase of Sequencer Class
# UVM_INFO Agent.sv(65) @ 0: uvm_test_top.I2C_environment.I2C_Agent [Agent] Inside connect phase of Agent Class
# UVM_INFO Scoreboard.sv(69) @ 0: uvm_test_top.I2C_environment.I2C_Scoreboard [Scoreboard] Inside connect phase of Scoreboard Class
# UVM_INFO Environment.sv(63) @ 0: uvm_test_top.I2C_environment [Environment] Inside connect phase of Environment Class
# UVM_INFO I2C_Test.sv(74) @ 0: uvm_test_top [I2C_Test] Inside connect phase of I2C Test Class
# UVM_INFO I2C_Test.sv(93) @ 0: uvm_test_top [I2C_Test] Inside run phase of I2C Test Class
# UVM_INFO Environment.sv(83) @ 0: uvm_test_top.I2C_environment [Environment] Inside run phase of Environment Class
# UVM_INFO Scoreboard.sv(86) @ 0: uvm_test_top.I2C_environment.I2C_Scoreboard [Scoreboard] Inside run phase of Scoreboard Class
# UVM_INFO Agent.sv(86) @ 0: uvm_test_top.I2C_environment.I2C_Agent [Agent] Inside run phase of Agent Class
# UVM_INFO Sequencer.sv(68) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_sequencer [Sequencer] Inside run phase of Sequencer Class
# UVM_INFO Monitor.sv(96) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_monitor [Monitor] Inside run phase of Monitor Class
# UVM_INFO Driver.sv(84) @ 0: uvm_test_top.I2C_environment.I2C_Agent.I2C_driver [Driver] Inside run phase of Driver Class
```

```
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   30
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Agent]     4
# [Driver]     4
# [Environment]     4
# [I2C_Test]     4
# [Monitor]     4
# [RNTST]     1
# [Scoreboard]     4
# [Sequencer]     4
# [TEST_DONE]     1
```

Thank You

Youssef Ahmed