**RAM VERIFICATION**

**Test Plan:**

1-Work smart not hard

2-Use the golden model you have

3-Check if they output is the same

**Code Coverage Report:**

=== Instance: /ramtb/dut

# === Design Unit: work.ram

# =================================================================================

# Toggle Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# Toggles 44 42 2 95.45%

#

# ================================Toggle Details================================

#

# Toggle Coverage for instance /ramtb/dut --

#

# Node 1H->0L 0L->1H "Coverage"

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

# clk 1 1 100.00

# din[0-9] 1 1 100.00

# dout[7-0] 1 1 100.00

# rst\_n 1 1 100.00

# rx\_valid 0 0 0.00

# tx\_valid 1 1 100.00

#

# Total Node Count = 22

# Toggled Node Count = 21

# Untoggled Node Count = 1

#

# Toggle Coverage = 95.45% (42 of 44 bins)

#

# =================================================================================

# === Instance: /ramtb

# === Design Unit: work.ramtb

# =================================================================================

#

# Assertion Coverage:

# Assertions 2 2 0 100.00%

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

# Name File(Line) Failure Pass

# Count Count

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

# /ramtb/#ublk#7898274#50/immed\_\_51

# ramtb.sv(51) 0 1

# /ramtb/a ramtb.sv(77) 0 1

#

# Directive Coverage:

# Directives 1 1 0 100.00%

#

# DIRECTIVE COVERAGE:

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

# Name Design Design Lang File(Line) Hits Status

# Unit UnitType

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

# /ramtb/cover\_\_check ramtb Verilog SVA ramtb.sv(78) 980 Covered

# Statement Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# Statements 25 25 0 100.00%

#

# ================================Statement Details================================

#

# Statement Coverage for instance /ramtb --

#

# Line Item Count Source

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

# File ramtb.sv

# 1 module ramtb();

#

# 2

#

# 3 class ramrand;

#

# 4 rand bit [9:0]X,Y,Z,W;

#

# 5 rand bit reset;

#

# 6 constraint rst{

#

# 7 reset dist{1:=98,0:=2};

#

# 8 }

#

# 9 constraint ram{

#

# 10 // write address

#

# 11 X[9]==0;

#

# 12 X[8]==0;

#

# 13 // write data

#

# 14 Y[9]==0;

#

# 15 Y[8]==1;

#

# 16 // read address

#

# 17 Z[9]==1;

#

# 18 Z[8]==0;

#

# 19 // read data

#

# 20 W[9]==1;

#

# 21 W[8]==1;

#

# 22 }

#

# 23

#

# 24

#

# 25 endclass

#

# 26 parameter MEM\_DEPTH = 256;

#

# 27 parameter ADDR\_SIZE = 8;

#

# 28

#

# 29 logic [9:0] din;

#

# 30 logic clk , rst\_n,rx\_valid;

#

# 31 logic [7:0] dout;

#

# 32 logic tx\_valid;

#

# 33 1 1 ramrand values=new;

#

# 34 logic [9:0] address;

#

# 35 logic [7:0] checka;

#

# 36 int i;

#

# 37 ram dut(din,clk,rst\_n,rx\_valid,dout,tx\_valid);

#

# 38 initial begin

#

# 39 1 1 clk=0;

#

# 40 1 1 forever

#

# 41 1 8000 #1 clk=~clk;

#

# 41 2 7999

# 42 end

#

# 43 initial begin

#

# 44 1 1 @(posedge clk)

#

# 45 1 1 rst\_n = 0;

#

# 46 1 1 #2

#

# 47 1 1 rst\_n = 1;

#

# 48 end

#

# 49 initial begin

#

# 50 1 1 for (i=0;i<1000;i++)begin

#

# 50 2 1000

# 51 assert(values.randomize());

#

# 52 // write

#

# 53 1 1000 rst\_n=values.reset;

#

# 54 1 1000 rx\_valid=1;

#

# 55 1 1000 din=values.X;

#

# 56 1 1000 address=din;

#

# 57 1 1000 address[9]=1;

#

# 58 1 1000 #2

#

# 59

#

# 60 1 1000 din=values.Y;

#

# 61 1 1000 checka=din;

#

# 62 1 1000 #2

#

# 63 //read

#

# 64 //rx\_valid=0;

#

# 65 1 1000 din=address;

#

# 66 1 1000 #2

#

# 67 1 1000 din=values.W;

#

# 68 1 1000 #2;

#

# 69 end

#

# 70 1 1 $stop;

#

#

# Toggle Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# Toggles 144 93 51 64.58%

#

# ================================Toggle Details================================

#

# Toggle Coverage for instance /ramtb --

#

# Node 1H->0L 0L->1H "Coverage"

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

# address[9-8] 0 0 0.00

# address[7-0] 1 1 100.00

# checka[7-0] 1 1 100.00

# clk 1 1 100.00

# din[9-0] 1 1 100.00

# dout[7-0] 1 1 100.00

# i[31-10] 0 0 0.00

# i[9] 0 1 50.00

# i[8-0] 1 1 100.00

# rst\_n 1 1 100.00

# rx\_valid 0 0 0.00

# tx\_valid 1 1 100.00

#

# Total Node Count = 72

# Toggled Node Count = 46

# Untoggled Node Count = 26

#

# Toggle Coverage = 64.58% (93 of 144 bins)

#

#

# DIRECTIVE COVERAGE:

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

# Name Design Design Lang File(Line) Hits Status

# Unit UnitType

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

# /ramtb/cover\_\_check ramtb Verilog SVA ramtb.sv(78) 980 Covered

#

# TOTAL DIRECTIVE COVERAGE: 100.00% COVERS: 1

#

# ASSERTION RESULTS:

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

# Name File(Line) Failure Pass

# Count Count

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

# /ramtb/#ublk#7898274#50/immed\_\_51

# ramtb.sv(51) 0 1

# /ramtb/a ramtb.sv(77) 0 1

#

# Total Coverage By Instance (filtered view): 92.95%

**Assertion:**

For the following assertion I used a queue array and popped the right value of dout into a variable named ‘checka’

|  |  |
| --- | --- |
| **Feature** | **Assertion** |
| Checks that after the tx valid is 1, dout is right | @(posedge clk) $rose(tx\_valid)|->(dout==checka); |

Graphical user interface, text, application, email

Description automatically generated

**Bug Report:**

No bugs were found in the RAM. Both read and write functions work fine with all the right flags.

**SPI WRAPPER VERIFICATION**

**Code Coverage:**

=================================================================================

# === Instance: /spislavetb/test\_golden

# === Design Unit: work.SPISLAVE

# =================================================================================

# Branch Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# Branches 50 48 2 96.00%

#

# ================================Branch Details================================

#

# Branch Coverage for instance /spislavetb/test\_golden

#

# Line Item Count Source

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

# File SPISLAVE.v

# ------------------------------------CASE Branch------------------------------------

# 17 10568 Count coming in to CASE

# 19 1 2129 idle: begin

#

# 30 1 1581 chk\_cmd: begin

#

# 47 1 3422 write: begin

#

# 59 1 1864 read\_addr: begin

#

# 69 1 1571 read\_data: begin

#

# 79 1 1 default: ns=idle;

#

# Branch totals: 6 hits of 6 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 21 2129 Count coming in to IF

# 21 1 1065 if(SS\_n) begin

#

# 24 1 1064 else begin

#

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 31 1581 Count coming in to IF

# 31 1 664 if(SS\_n==0&&MOSI==0 && !flag) begin

#

# 34 1 516 else if(SS\_n==0&&MOSI==1&& !flag) begin

#

# 37 1 266 else if(SS\_n==0&&MOSI==1&&flag) begin

#

# 40 1 \*\*\*0\*\*\* else if(SS\_n==1)

#

# 42 1 135 else begin

#

# Branch totals: 4 hits of 5 branches = 80.00%

#

# ------------------------------------IF Branch------------------------------------

# 49 3422 Count coming in to IF

# 49 1 532 if(SS\_n==1) begin

#

# 52 1 2890 else begin

#

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 61 1864 Count coming in to IF

# 61 1 266 if(SS\_n==1) begin

#

# 64 1 1598 else begin

#

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 71 1571 Count coming in to IF

# 71 1 266 if(SS\_n==1) begin

#

# 74 1 1305 else begin

#

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 88 16504 Count coming in to IF

# 88 1 2 if(!rst\_n) begin

#

# 98 1 16502 else begin

#

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------CASE Branch------------------------------------

# 108 16503 Count coming in to CASE

# 109 1 5596 write: begin

#

# 129 1 2926 read\_addr:begin

#

# 148 1 4788 read\_data:begin

#

# 3193 All False Count

# Branch totals: 4 hits of 4 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 110 5596 Count coming in to IF

# 110 1 5088 if(count<10)

#

# 508 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 113 5596 Count coming in to IF

# 113 1 508 if(count==9) begin

#

# 5088 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 117 5596 Count coming in to IF

# 117 1 508 if(rx\_valid==1)begin

#

# 5088 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 118 508 Count coming in to IF

# 118 1 142 if(rx\_data[8]==0&&rx\_data[9]==0)begin

#

# 122 1 123 else if(rx\_data[8]==1 && rx\_data[9]==0)begin

#

# 243 All False Count

# Branch totals: 3 hits of 3 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 130 2926 Count coming in to IF

# 130 1 2660 if(count<10)

#

# 266 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 133 2926 Count coming in to IF

# 133 1 266 if(count==9) begin

#

# 2660 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 137 2926 Count coming in to IF

# 137 1 266 if(rx\_valid==1)begin

#

# 2660 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 138 266 Count coming in to IF

# 138 1 \*\*\*0\*\*\* if(rx\_data[8]==0&&rx\_data[9]==1)begin //stores the address in internal signal if most significant bits are 10

#

# 266 All False Count

# Branch totals: 1 hit of 2 branches = 50.00%

#

# ------------------------------------IF Branch------------------------------------

# 149 4788 Count coming in to IF

# 149 1 3192 if(count<10) begin

#

# 1596 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 153 4788 Count coming in to IF

# 153 1 266 if(count==9)begin

#

# 4522 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 157 4788 Count coming in to IF

# 157 1 2128 if(count2>=0 && tx\_valid)begin

#

# 2660 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

# ------------------------------------IF Branch------------------------------------

# 161 2128 Count coming in to IF

# 161 1 266 if(count2==0)begin

#

# 1862 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

#

# Condition Coverage:

# Enabled Coverage Bins Covered Misses Coverage

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

# Conditions 24 17 7 70.83%

#

# ================================Condition Details================================

#

# Condition Coverage for instance /spislavetb/test\_golden --

#

# File SPISLAVE.v

# ----------------Focused Condition View-------------------

# Line 31 Item 1 ((SS\_n ~| MOSI) && ~flag)

# Condition totals: 2 of 3 input terms covered = 66.66%

#

# Input Term Covered Reason for no coverage Hint

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

# SS\_n N '\_1' not hit Hit '\_1'

# MOSI Y

# flag Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 SS\_n\_0 (~flag && ~MOSI)

# Row 2: \*\*\*0\*\*\* SS\_n\_1 ~MOSI

# Row 3: 1 MOSI\_0 (~flag && ~SS\_n)

# Row 4: 1 MOSI\_1 ~SS\_n

# Row 5: 1 flag\_0 (SS\_n ~| MOSI)

# Row 6: 1 flag\_1 (SS\_n ~| MOSI)

#

# ----------------Focused Condition View-------------------

# Line 34 Item 1 ((~SS\_n && MOSI) && ~flag)

# Condition totals: 2 of 3 input terms covered = 66.66%

#

# Input Term Covered Reason for no coverage Hint

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

# SS\_n N '\_1' not hit Hit '\_1'

# MOSI Y

# flag Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 SS\_n\_0 (~flag && MOSI)

# Row 2: \*\*\*0\*\*\* SS\_n\_1 -

# Row 3: 1 MOSI\_0 ~SS\_n

# Row 4: 1 MOSI\_1 (~flag && ~SS\_n)

# Row 5: 1 flag\_0 (~SS\_n && MOSI)

# Row 6: 1 flag\_1 (~SS\_n && MOSI)

#

# ----------------Focused Condition View-------------------

# Line 37 Item 1 ((~SS\_n && MOSI) && flag)

# Condition totals: 1 of 3 input terms covered = 33.33%

#

# Input Term Covered Reason for no coverage Hint

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

# SS\_n N '\_1' not hit Hit '\_1'

# MOSI Y

# flag N '\_0' not hit Hit '\_0'

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 SS\_n\_0 (flag && MOSI)

# Row 2: \*\*\*0\*\*\* SS\_n\_1 -

# Row 3: 1 MOSI\_0 ~SS\_n

# Row 4: 1 MOSI\_1 (flag && ~SS\_n)

# Row 5: \*\*\*0\*\*\* flag\_0 (~SS\_n && MOSI)

# Row 6: 1 flag\_1 (~SS\_n && MOSI)

#

# ----------------Focused Condition View-------------------

# Line 110 Item 1 (count < 10)

# Condition totals: 1 of 1 input term covered = 100.00%

#

# Input Term Covered Reason for no coverage Hint

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

# (count < 10) Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 (count < 10)\_0 -

# Row 2: 1 (count < 10)\_1 -

#

# ----------------Focused Condition View-------------------

# Line 113 Item 1 (count == 9)

# Condition totals: 1 of 1 input term covered = 100.00%

#

# Input Term Covered Reason for no coverage Hint

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

# (count == 9) Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 (count == 9)\_0 -

# Row 2: 1 (count == 9)\_1 -

#

# ----------------Focused Condition View-------------------

# Line 118 Item 1 (rx\_data[8] ~| rx\_data[9])

# Condition totals: 2 of 2 input terms covered = 100.00%

#

# Input Term Covered Reason for no coverage Hint

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

# rx\_data[8] Y

# rx\_data[9] Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 rx\_data[8]\_0 ~rx\_data[9]

# Row 2: 1 rx\_data[8]\_1 ~rx\_data[9]

# Row 3: 1 rx\_data[9]\_0 ~rx\_data[8]

# Row 4: 1 rx\_data[9]\_1 ~rx\_data[8]

#

# ----------------Focused Condition View-------------------

# Line 122 Item 1 (rx\_data[8] && ~rx\_data[9])

# Condition totals: 2 of 2 input terms covered = 100.00%

#

# Input Term Covered Reason for no coverage Hint

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

# rx\_data[8] Y

# rx\_data[9] Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 rx\_data[8]\_0 -

# Row 2: 1 rx\_data[8]\_1 ~rx\_data[9]

# Row 3: 1 rx\_data[9]\_0 rx\_data[8]

# Row 4: 1 rx\_data[9]\_1 rx\_data[8]

#

# ----------------Focused Condition View-------------------

# Line 130 Item 1 (count < 10)

# Condition totals: 1 of 1 input term covered = 100.00%

#

# Input Term Covered Reason for no coverage Hint

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

# (count < 10) Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 (count < 10)\_0 -

# Row 2: 1 (count < 10)\_1 -

#

# ----------------Focused Condition View-------------------

# Line 133 Item 1 (count == 9)

# Condition totals: 1 of 1 input term covered = 100.00%

#

# Input Term Covered Reason for no coverage Hint

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

# (count == 9) Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 (count == 9)\_0 -

# Row 2: 1 (count == 9)\_1 -

#

# ----------------Focused Condition View-------------------

# Line 138 Item 1 (~rx\_data[8] && rx\_data[9])

# Condition totals: 0 of 2 input terms covered = 0.00%

#

# Input Term Covered Reason for no coverage Hint

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

# rx\_data[8] N '\_0' not hit Hit '\_0'

# rx\_data[9] N '\_1' not hit Hit '\_1'

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: \*\*\*0\*\*\* rx\_data[8]\_0 rx\_data[9]

# Row 2: 1 rx\_data[8]\_1 -

# Row 3: 1 rx\_data[9]\_0 ~rx\_data[8]

# Row 4: \*\*\*0\*\*\* rx\_data[9]\_1 ~rx\_data[8]

#

# ----------------Focused Condition View-------------------

# Line 149 Item 1 (count < 10)

# Condition totals: 1 of 1 input term covered = 100.00%

#

# Input Term Covered Reason for no coverage Hint

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

# (count < 10) Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 (count < 10)\_0 -

# Row 2: 1 (count < 10)\_1 -

#

# ----------------Focused Condition View-------------------

# Line 153 Item 1 (count == 9)

# Condition totals: 1 of 1 input term covered = 100.00%

#

# Input Term Covered Reason for no coverage Hint

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

# (count == 9) Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 (count == 9)\_0 -

# Row 2: 1 (count == 9)\_1 -

#

# ----------------Focused Condition View-------------------

# Line 157 Item 1 ((count2 >= 0) && tx\_valid)

# Condition totals: 1 of 2 input terms covered = 50.00%

#

# Input Term Covered Reason for no coverage Hint

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

# (count2 >= 0) N '\_0' not hit Hit '\_0'

# tx\_valid Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: \*\*\*0\*\*\* (count2 >= 0)\_0 -

# Row 2: 1 (count2 >= 0)\_1 tx\_valid

# Row 3: 1 tx\_valid\_0 (count2 >= 0)

# Row 4: 1 tx\_valid\_1 (count2 >= 0)

#

# ----------------Focused Condition View-------------------

# Line 161 Item 1 (count2 == 0)

# Condition totals: 1 of 1 input term covered = 100.00%

#

# Input Term Covered Reason for no coverage Hint

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

# (count2 == 0) Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 (count2 == 0)\_0 -

# Row 2: 1 (count2 == 0)\_1 -

#

#

# FSM Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# FSM States 5 5 0 100.00%

# FSM Transitions 8 7 1 87.50%

#

# ================================FSM Details================================

#

# FSM Coverage for instance /spislavetb/test\_golden --

#

# FSM\_ID: cs

# Current State Object : cs

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

# State Value MapInfo :

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

# Line State Name Value

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

# 19 idle 0

# 30 chk\_cmd 1

# 69 read\_data 4

# 59 read\_addr 3

# 47 write 2

# Covered States :

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

# State Hit\_count

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

# idle 2130

# chk\_cmd 1064

# read\_data 4788

# read\_addr 2926

# write 5596

# Covered Transitions :

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

# Line Trans\_ID Hit\_count Transition

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

# 25 0 1064 idle -> chk\_cmd

# 38 2 266 chk\_cmd -> read\_data

# 35 3 266 chk\_cmd -> read\_addr

# 32 4 532 chk\_cmd -> write

# 72 5 266 read\_data -> idle

# 62 6 266 read\_addr -> idle

# 50 7 532 write -> idle

# Uncovered Transitions :

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

# Line Trans\_ID Transition

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

# 41 1 chk\_cmd -> idle

#

#

# Summary Bins Hits Misses Coverage

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

# FSM States 5 5 0 100.00%

# FSM Transitions 8 7 1 87.50%

# Statement Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# Statements 52 50 2 96.15%

#

# ================================Statement Details================================

#

# Statement Coverage for instance /spislavetb/test\_golden --

#

# Line Item Count Source

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

# File SPISLAVE.v

# 1 module SPISLAVE(MOSI,MISO,SS\_n,clk,rst\_n);

#

# 2 //rx\_data=din tx\_data=dout

#

# 3 input MOSI,SS\_n,clk,rst\_n;

#

# 4 output reg MISO;

#

# 5 reg [9:0] rx\_data;

#

# 6 reg [7:0] tx\_data;

#

# 7 reg rx\_valid,tx\_valid,flag; //flag to diffrentiate betweeen read address and read data

#

# 8 reg [7:0] mem[255:0];

#

# 9 reg [2:0] cs,ns;

#

# 10 reg [3:0] count,count2; //internal signal for the MOSI and MISO count

#

# 11 reg [7:0] wr\_add,rd\_add; //internal signal for the address to be written/read

#

# 12

#

# 13 parameter idle=0,chk\_cmd=1,write=2,read\_addr=3,read\_data=4;

#

# 14

#

# 15 1 10568 always@(cs or SS\_n or MOSI) begin

#

# 16

#

# 17 case(cs)

#

# 18

#

# 19 idle: begin

#

# 20

#

# 21 if(SS\_n) begin

#

# 22 1 1065 ns=idle;

#

# 23 end

#

# 24 else begin

#

# 25 1 1064 ns=chk\_cmd;

#

# 26 end

#

# 27

#

# 28 end

#

# 29

#

# 30 chk\_cmd: begin

#

# 31 if(SS\_n==0&&MOSI==0 && !flag) begin

#

# 32 1 664 ns=write;

#

# 33 end

#

# 34 else if(SS\_n==0&&MOSI==1&& !flag) begin

#

# 35 1 516 ns=read\_addr;

#

# 36 end

#

# 37 else if(SS\_n==0&&MOSI==1&&flag) begin

#

# 38 1 266 ns=read\_data;

#

# 39 end

#

# 40 else if(SS\_n==1)

#

# 41 1 \*\*\*0\*\*\* ns=idle;

#

# 42 else begin

#

# 43 1 135 ns=chk\_cmd;

#

# 44 end

#

# 45 end

#

# 46

#

# 47 write: begin

#

# 48

#

# 49 if(SS\_n==1) begin

#

# 50 1 532 ns=idle;

#

# 51 end

#

# 52 else begin

#

# 53 1 2890 ns=write;

#

# 54 end

#

# 55

#

# 56

#

# 57 end

#

# 58

#

# 59 read\_addr: begin

#

# 60

#

# 61 if(SS\_n==1) begin

#

# 62 1 266 ns=idle;

#

# 63 end

#

# 64 else begin

#

# 65 1 1598 ns=read\_addr;

#

# 66 end

#

# 67 end

#

# 68

#

# 69 read\_data: begin

#

# 70

#

# 71 if(SS\_n==1) begin

#

# 72 1 266 ns=idle;

#

# 73 end

#

# 74 else begin

#

# 75 1 1305 ns=read\_data;

#

# 76 end

#

# 77 end

#

# 78

#

# 79 1 1 default: ns=idle;

#

# 80 endcase

#

# 81

#

# 82

#

# 83 end

#

# 84

#

# 85

#

# 86 1 16504 always@(posedge clk or negedge rst\_n) begin

#

# 87

#

# 88 if(!rst\_n) begin

#

# 89 1 2 cs<=idle;

#

# 90 1 2 count<=0;

#

# 91 1 2 count2<=7;

#

# 92 1 2 rx\_valid<=0;

#

# 93 1 2 tx\_valid<=0;

#

# 94 1 2 flag<=0;

#

# 95 1 2 rx\_data<=0;

#

# 96 1 2 tx\_data<=0;

#

# 97 end

#

# 98 else begin

#

# 99 1 16502 cs<=ns;

#

# 100

#

# 101 end

#

# 102 end

#

# 103

#

# 104

#

# 105

#

# 106

#

# 107 1 16503 always@(posedge clk) begin

#

# 108 case(cs)

#

# 109 write: begin

#

# 110 if(count<10)

#

# 111 1 5088 rx\_data<={rx\_data[8:0],MOSI}; //takes the MOSI and shifts register if count is less than 10

#

# 112 1 5596 count<=count+1;

#

# 113 if(count==9) begin

#

# 114 1 508 rx\_valid<=1; //rx valid to inform memory that it will receive data or the write addr to receive address

#

# 115 end

#

# 116

#

# 117 if(rx\_valid==1)begin

#

# 118 if(rx\_data[8]==0&&rx\_data[9]==0)begin

#

# 119 1 142 wr\_add<=rx\_data[7:0]; //stores the address in internal signal if most significant bits are 00

#

# 120

#

# 121 end

#

# 122 else if(rx\_data[8]==1 && rx\_data[9]==0)begin

#

# 123 1 123 mem[wr\_add]<=rx\_data[7:0]; //stores the data in memory address previously saved in wr\_add if most significant bits are 01

#

# 124 end

#

# 125 1 508 count<=0; //resets the count and rx valid

#

# 126 1 508 rx\_valid<=0;

#

# 127 end

#

# 128 end

#

# 129 read\_addr:begin

#

# 130 if(count<10)

#

# 131 1 2660 rx\_data<={rx\_data[8:0],MOSI}; //takes the MOSI and shifts register if count is less than 10

#

# 132 1 2926 count<=count+1;

#

# 133 if(count==9) begin

#

# 134 1 266 rx\_valid<=1; //rx valid to inform rd\_add that it will receive address

#

# 135 end

#

# 136

#

# 137 if(rx\_valid==1)begin

#

# 138 if(rx\_data[8]==0&&rx\_data[9]==1)begin //stores the address in internal signal if most significant bits are 10

#

# 139 1 \*\*\*0\*\*\* rd\_add<=rx\_data[7:0];

#

# 140 end

#

# 141 1 266 count<=0; //resets the count and rx valid

#

# 142 1 266 rx\_valid<=0;

#

# 143 1 266 flag<=1; //flag==1 to go to read memory not address in the next case

#

# 144 end

#

# 145

#

# 146 end

#

# 147

#

# 148 read\_data:begin

#

# 149 if(count<10) begin

#

# 150 1 3192 rx\_data<={rx\_data[8:0],MOSI}; //takes the MOSI and shifts register if count is less than 10

#

# 151 end

#

# 152 1 4788 count<=count+1;

#

# 153 if(count==9)begin

#

# 154 1 266 tx\_valid<=1; //tx valid to inform the slave that it will load data out on MISO from tx data

#

# 155 1 266 tx\_data<=mem[rd\_add];

#

# 156 end

#

# 157 if(count2>=0 && tx\_valid)begin

#

# 158 1 2128 MISO<=tx\_data[count2]; //MISO takes data from tx data

#

# 159

#

# 160 1 2128 count2<=count2-1;

#

# 161 if(count2==0)begin

#

# 162 1 266 tx\_valid<=0;

#

# 163 1 266 tx\_data<=0;

#

# 164 1 266 count2<=7;

#

# 165 1 266 flag<=0; //flag=0 to go in the next time to read address not read data

#

# 166 1 266 count<=0;

#

# 167 1 266 rx\_data<=0;

#

#

# Toggle Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# Toggles 112 75 37 66.96%

#

# ================================Toggle Details================================

#

# Toggle Coverage for instance /spislavetb/test\_golden --

#

# Node 1H->0L 0L->1H "Coverage"

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

# MISO 0 0 0.00

# MOSI 1 1 100.00

# SS\_n 1 1 100.00

# clk 1 1 100.00

# count[3-0] 1 1 100.00

# count2[3] 0 0 0.00

# count2[2-0] 1 1 100.00

# cs[2-0] 1 1 100.00

# flag 1 1 100.00

# ns[2-0] 1 1 100.00

# rd\_add[7-0] 0 0 0.00

# rst\_n 0 1 50.00

# rx\_data[9-0] 1 1 100.00

# rx\_valid 1 1 100.00

# tx\_data[7-0] 0 0 0.00

# tx\_valid 1 1 100.00

# wr\_add[7-0] 1 1 100.00

#

# Total Node Count = 56

# Toggled Node Count = 37

# Untoggled Node Count = 19

#

# Toggle Coverage = 66.96% (75 of 112 bins)

#

# =================================================================================

# === Instance: /spislavetb/test

# === Design Unit: work.SPI\_Wrapper

# =================================================================================

# Toggle Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# Toggles 10 9 1 90.00%

#

# ================================Toggle Details================================

#

# Toggle Coverage for instance /spislavetb/test --

#

# Node 1H->0L 0L->1H "Coverage"

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

# MISO 1 1 100.00

# MOSI 1 1 100.00

# SS\_n 1 1 100.00

# clk 1 1 100.00

# rst\_n 0 1 50.00

#

# Total Node Count = 5

# Toggled Node Count = 4

# Untoggled Node Count = 1

#

# Toggle Coverage = 90.00% (9 of 10 bins)

#

# =================================================================================

# === Instance: /spislavetb

# === Design Unit: work.spislavetb

# =================================================================================

# Branch Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# Branches 2 2 0 100.00%

#

# ================================Branch Details================================

#

# Branch Coverage for instance /spislavetb

# NOTE: The modification timestamp for source file 'spislavetb.sv' has been altered since compilation.

#

# Line Item Count Source

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

# File spislavetb.sv

# ------------------------------------IF Branch------------------------------------

# 156 2128 Count coming in to IF

# 156 1 2127 if(MISO!==MISO2) begin

#

# 1 All False Count

# Branch totals: 2 hits of 2 branches = 100.00%

#

#

# Condition Coverage:

# Enabled Coverage Bins Covered Misses Coverage

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

# Conditions 1 1 0 100.00%

#

# ================================Condition Details================================

#

# Condition Coverage for instance /spislavetb --

# NOTE: The modification timestamp for source file 'spislavetb.sv' has been altered since compilation.

#

# File spislavetb.sv

# ----------------Focused Condition View-------------------

# Line 156 Item 1 (MISO !== MISO2)

# Condition totals: 1 of 1 input term covered = 100.00%

#

# Input Term Covered Reason for no coverage Hint

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

# (MISO !== MISO2) Y

#

# Rows: Hits FEC Target Non-masking condition(s)

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

# Row 1: 1 (MISO !== MISO2)\_0 -

# Row 2: 1 (MISO !== MISO2)\_1 -

#

#

# Statement Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# Statements 127 127 0 100.00%

#

# ================================Statement Details================================

#

# Statement Coverage for instance /spislavetb --

# NOTE: The modification timestamp for source file 'spislavetb.sv' has been altered since compilation.

#

# Line Item Count Source

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

# File spislavetb.sv

# 1 module spislavetb();

#

# 2

#

# 3 logic MOSI,MISO,clk,rst\_n,SS\_n;

#

# 4 integer i,j,index=0;

#

# 5 bit [7:0]m=0,n=0;

#

# 6 logic MISO2;

#

# 7 initial begin

#

# 8 1 1 clk=0;

#

# 9 1 1 forever begin

#

# 10 1 33006 #1 ;

#

# 11 1 33005 clk=~clk;

#

# 12 end

#

# 13 end

#

# 14

#

# 15

#

# 16 SPISLAVE test\_golden(MOSI,MISO,SS\_n,clk,rst\_n); //golden model

#

# 17 SPI\_Wrapper test(MOSI,MISO2,SS\_n,clk,rst\_n); //device to be tested

#

# 18

#

# 19

#

# 20 initial begin

#

# 21 1 1 rst\_n=0;

#

# 22 1 1 SS\_n=1;

#

# 23 1 1 #2

#

# 24 1 1 rst\_n=1;

#

# 25 1 1 for(i=0;i<256;i=i+1) begin

#

# 25 2 256

# 26 //fill the whole memory

#

# 27 1 256 write\_addr();

#

# 28 1 256 write\_data();

#

# 29

#

# 30 end

#

# 31

#

# 32 1 1 for(i=0;i<256;i=i+1) begin

#

# 32 2 256

# 33 //read the whole memory

#

# 34 1 256 read\_addr();

#

# 35 1 256 read\_data();

#

# 36 end

#

# 37

#

# 38 1 1 for(i=0;i<10;i=i+1) begin

#

# 38 2 10

# 39 //read memory randomly

#

# 40 1 10 read\_addr\_random();

#

# 41 1 10 read\_data();

#

# 42 end

#

# 43 1 1 for(i=0;i<10;i=i+1) begin

#

# 43 2 10

# 44 //overwrite random places in the memory

#

# 45 1 10 write\_addr\_random();

#

# 46 1 10 write\_data();

#

# 47 end

#

# 48

#

# 49 1 1 $stop;

#

# 50

#

# 51 end

#

# 52

#

# 53

#

# 54 task write\_addr();

#

# 55 1 256 SS\_n=0;

#

# 56 1 256 #2

#

# 57 1 256 MOSI=0;

#

# 58 1 256 #2

#

# 59 1 256 MOSI=0;

#

# 60 #2

#

# 61 MOSI=0;

#

# 62 1 256 for(j=7;j>=1;j=j-1) begin

#

# 62 2 1792

# 63 1 1792 #2

#

# 64 1 1792 MOSI=m[j];

#

# 65 end

#

# 66 1 256 #2

#

# 67 1 256 MOSI=m[0];

#

# 68 1 256 m++;

#

# 69 1 256 #2

#

# 70 1 256 SS\_n=1;

#

# 71 1 256 #4;

#

# 72 endtask

#

# 73 task write\_addr\_random();

#

# 74 1 10 SS\_n=0;

#

# 75 1 10 #2

#

# 76 1 10 MOSI=0;

#

# 77 1 10 #2

#

# 78 1 10 MOSI=0;

#

# 79 1 10 #2

#

# 80 1 10 MOSI=0;

#

# 81 1 10 for(j=7;j>=1;j=j-1) begin

#

# 81 2 70

# 82 1 70 #2

#

# 83 1 70 MOSI=$random;

#

# 84 end

#

# 85 1 10 #2

#

# 86 1 10 MOSI=$random;

#

# 87 1 10 m++;

#

# 88 1 10 #2

#

# 89 1 10 SS\_n=1;

#

# 90 1 10 #4;

#

# 91 endtask

#

# 92 task write\_data();

#

# 93 1 266 SS\_n=0;

#

# 94 1 266 #2

#

# 95 1 266 MOSI=0;

#

# 96 1 266 #2

#

# 97 1 266 MOSI=0;

#

# 98 1 266 #2

#

# 99 1 266 MOSI=1;

#

# 100 1 266 for(j=0;j<7;j=j+1) begin

#

# 100 2 1862

# 101 1 1862 #2

#

# 102 1 1862 MOSI=$random;

#

# 103 end

#

# 104 1 266 #2

#

# 105 1 266 MOSI=$random;

#

# 106 1 266 #2

#

# 107 1 266 SS\_n=1;

#

# 108 1 266 #4;

#

# 109 endtask

#

# 110

#

# 111 task read\_addr();

#

# 112 1 256 SS\_n=0;

#

# 113 1 256 #2

#

# 114 1 256 MOSI=1;

#

# 115 1 256 #2

#

# 116 1 256 MOSI=1;

#

# 117 1 256 #2

#

# 118 1 256 MOSI=0;

#

# 119 1 256 for(j=7;j>=1;j=j-1) begin

#

# 119 2 1792

# 120 1 1792 #2

#

# 121 1 1792 MOSI=n[j];

#

# 122 //MOSI=$random;

#

# 123 end

#

# 124 1 256 #2

#

# 125 1 256 MOSI=n[0];

#

# 126 //MOSI=$random;

#

# 127 1 256 n++;

#

# 128 1 256 #2

#

# 129 1 256 SS\_n=1;

#

# 130 1 256 #4;

#

# 131 endtask

#

# 132 task read\_data();

#

# 133 1 266 SS\_n=0;

#

# 134 1 266 #2

#

# 135 1 266 MOSI=1;

#

# 136 1 266 #2

#

# 137 1 266 MOSI=1;

#

# 138 1 266 #2

#

# 139 1 266 MOSI=1;

#

# 140 1 266 for(j=0;j<7;j=j+1) begin

#

# 140 2 1862

# 141 1 1862 #2

#

# 142 1 1862 MOSI=$random;

#

# 143 end

#

# 144 1 266 #2

#

# 145 1 266 MOSI=$random;

#

# 146 1 266 for(j=0;j<8;j=j+1) begin

#

# 146 2 2128

# 147 1 2128 #2

#

# 148 1 2128 check();

#

# 149 end

#

# 150 1 266 SS\_n=1;

#

# 151 1 266 #4;

#

# 152 endtask

#

# 153

#

# 154 task check();

#

# 155

#

# 156 if(MISO!==MISO2) begin

#

# 157 1 2127 $display("MISO should be %0b instead it's %0b", MISO, MISO2);

#

# 158 end

#

# 159

#

# 160

#

# 161 endtask

#

# 162

#

# 163 task read\_addr\_random();

#

# 164 1 10 SS\_n=0;

#

# 165 1 10 #2

#

# 166 1 10 MOSI=1;

#

# 167 1 10 #2

#

# 168 1 10 MOSI=1;

#

# 169 1 10 #2

#

# 170 1 10 MOSI=0;

#

# 171 1 10 for(j=7;j>=1;j=j-1) begin

#

# 171 2 70

# 172 1 70 #2

#

# 173 //MOSI=n[j];

#

# 174 1 70 MOSI=$random;

#

# 175 end

#

# 176 1 10 #2

#

# 177 //MOSI=n[0];

#

# 178 1 10 MOSI=$random;

#

# 179 1 10 n++;

#

# 180 1 10 #2

#

# 181 1 10 SS\_n=1;

#

# 182 1 10 #4;

#

#

# Toggle Coverage:

# Enabled Coverage Bins Hits Misses Coverage

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

# Toggles 172 65 107 37.79%

#

# ================================Toggle Details================================

#

# Toggle Coverage for instance /spislavetb --

#

# Node 1H->0L 0L->1H "Coverage"

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

# MISO 0 0 0.00

# MISO2 1 1 100.00

# MOSI 1 1 100.00

# SS\_n 1 1 100.00

# clk 1 1 100.00

# i[0-7] 1 1 100.00

# i[8-31] 0 0 0.00

# j[0-3] 1 1 100.00

# j[4-31] 0 0 0.00

# m[0-7] 1 1 100.00

# n[0-7] 1 1 100.00

# rst\_n 0 1 50.00

#

# Total Node Count = 86

# Toggled Node Count = 32

# Untoggled Node Count = 54

#

# Toggle Coverage = 37.79% (65 of 172 bins)

#

#

# Total Coverage By Instance (filtered view): 84.20%

**Assertions:**

No assertions were used. Will explain further during the presentation.

**Bug Report:**

Fatal Error: Read doesn’t work properly, MISO outputs 10 bits, MISO doesn’t stop communication when SS\_n=1, but continues sending the 10 bits.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated