Reusable Verification Environment: Case Study Spi

Ripalkumar Patel[1] Kavankumar Pandya2

[1, 2] Electronics And Communication Department,

[1, 2] Ganpat University Kherva – 384012 , Dist : Mehsana North Gujarat (INDIA)

Abstract--- This project paper introduce with uniform verification environment using system Verilog as well as gives appropriate direction about reusability of verification environment's components. This project paper divides with five sections. First section contains overview of SPI interface protocol. Second section is about physical interface for SPI protocol. Third section leads towards verification environment using system Verilog and over view about verification components. Fourth section is about Master VIP using reusability of environment's components of SPI protocol with simulation results and coverage report using VCS. Fifth section is about Slave VIP using reusability of environment's components of SPI protocol with simulation results and coverage report using VCS.

I. OVERVIEW OF SPI INTERFACING PROTOCOL

The SPI is essentially a "three-wire plus slave selects" serial bus for eight or sixteen bit data transfer applications. The three wires carry information between devices connected to the bus. Each device on the bus acts simultaneously as a transmitter and receiver. Two of the three lines transfer data (one line for each direction) and the third is a serial clock. Some devices may be only transmitters while others only receivers. Generally, a device that transmits usually possesses the capability to receive data also. An SPI display is an example of a receive-only device while EEPROM is a receiver and transmit device. The devices connected to the SPI bus may be classified as Master or Slave devices. A master device initiates an information transfer on the bus and generates clock and control signals. A slave device is controlled by the master through a slave select (chip enable) line and is active only when selected. Generally, a dedicated select line is required for each slave device. The same device can possess the functionality of a master and a slave but at any point of time, only one master can control the bus in a multi-master mode configuration. Any slave device that is not selected must release (make it high impedance) the slave output line. The SPI bus employs a simple shift register data transfer scheme: Data is clocked out of and into the active devices in a first-in, first-out fashion. It is in this manner that SPI devices transmit and receive in full duplex mode. All lines on the SPI bus are unidirectional: The signal on the clock line (SCLK) is generated by the master and is primarily used to synchronize data transfer. The master-out, slave-in (MOSI) line carries data from the master to the slave and the master-in, slave-out (MISO) line carries data from the slave to the master. Each slave device is selected by the master via individual select lines. Information on the SPI bus can be transferred at a rate of near zero bits per second to 1 Mbits per second. Data transfer is usually performed in eight/sixteen bit blocks. All data transfer is synchronized by the serial clock (SCLK). One bit of data is transferred for each clock cycle. Four clock modes are defined for the SPI bus by the value of the clock polarity and the clock phase bits. The clock polarity determines the level of the clock idle state and the clock phase determines which clock edge places new data on the bus. Any hardware device capable of operation in more than one mode will have some method of selecting the value of these bits.

The SPI bus specifies four logic signals:

- SPI_CLK: Serial Clock.
- SPI_MOSI: Master Output, Slave Input .
- SPI_MISO: Master Input, Slave Output .
- SPI_SS: Slave Select.

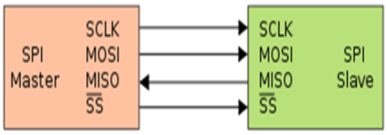


Fig. 1: Spi Interface

II. PHYSICAL INTERFACE OF SPI PROTOCOL

Before developing of SPI VIP we need to analyse actual SPI communication between to ICs. In this paper implemented actual data transmission according to SPI protocol employs two PSoC kits. One of them is working as a master and another one is working as a slave.

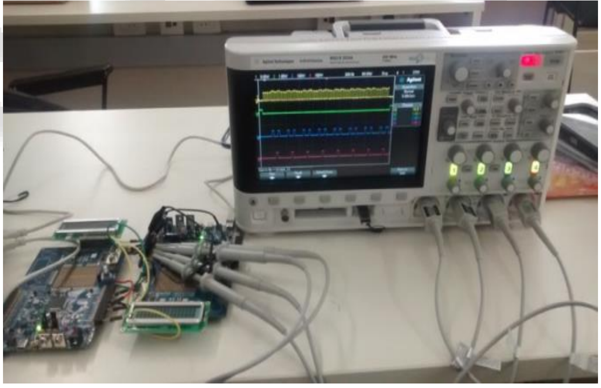


Fig. 2: physical interpretation of SPI protocol

The figure shows there are two PSoc kits connected with each other and transfer data in full duplex mode. We can analyse this data transfer using DSO which is shown in figure 2.

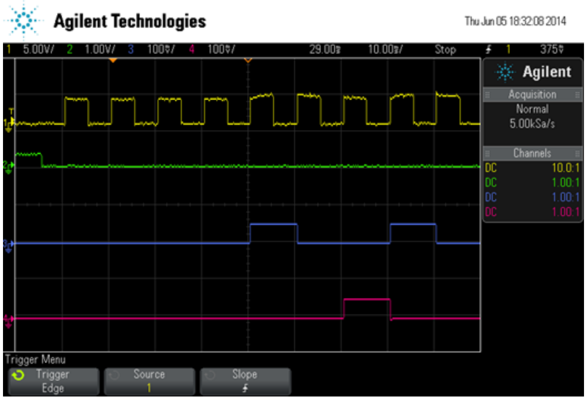


Fig. 3 Starting of Transmission

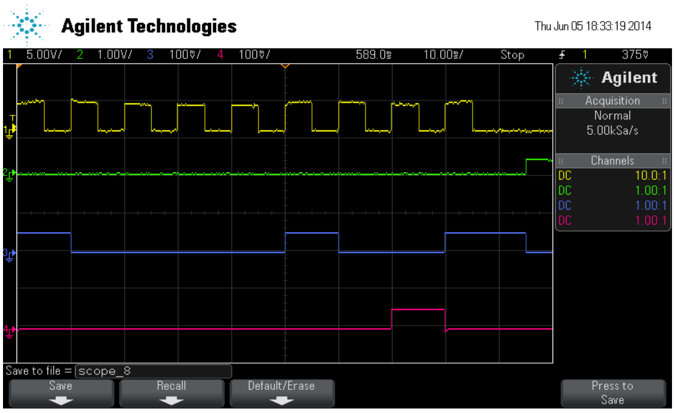


Fig. 4 End of Transmission

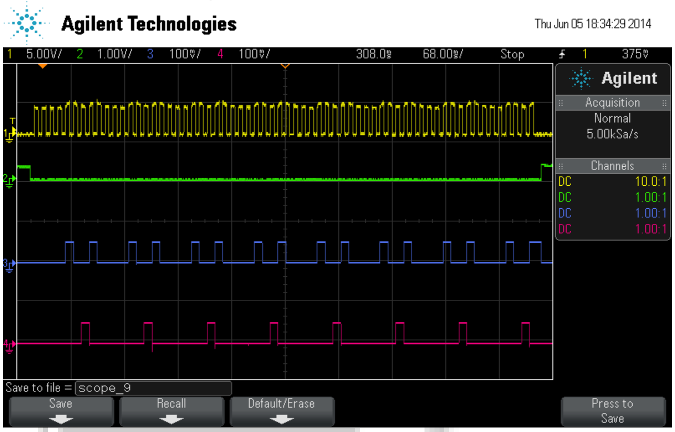


Fig. 5 Simulation Result of SPI Protocol

According to these simulation results we can say that this data transfer is employed SPI communication protocol. Physical implementation of SPI data transmission protocol is useful for understanding of Data transmission fashion in SPI protocol Overview about Verification Environment using System Verilog

Verification is technique which is verify DUT and find out bugs in that DUT if there. DUT is connected with components of verification environment and using these components entire DUT can verify.

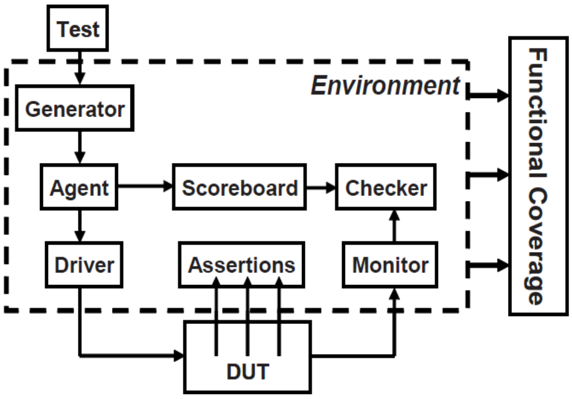


Fig. 6: End of Transmission

In this verification environment, generator generates stimulus for DUT which is drive by driver component. Driver components drives these generated stimulus in to DUT via interface. Interface is a medium between DUT and verification environment. All data transection in to DUT are done by this interface. Now generated stimulus available in to DUT and according to functionality of DUT, provides output which is taken by monitor. Monitor component is monitor data using interface. This data is transferred by monitor to scoreboard. In earlier generator puts data in to scoreboard and now monitor puts data in to scoreboard. There are two data available with Scoreboard component. Scoreboard compares these two data. If both data matched we can say that DUT is working with proper functionality.

III. Master Vip

Master VIP contains Slave DUT and master verification environment. This VIP designed for single master and single slave. But this is a reusable verification environment so this environment is also employable for multi master multi slave design. There is no need to do change in environment's components.

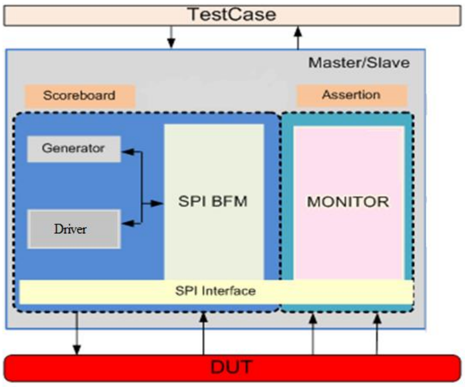
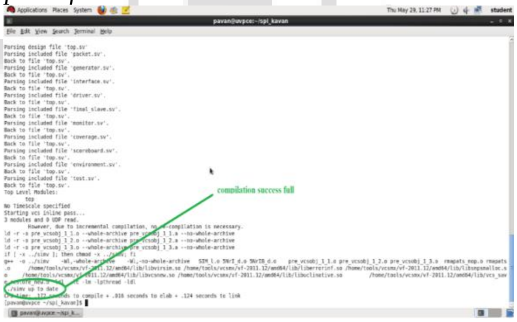


Fig. 7: Reusable SPI Verification Environment

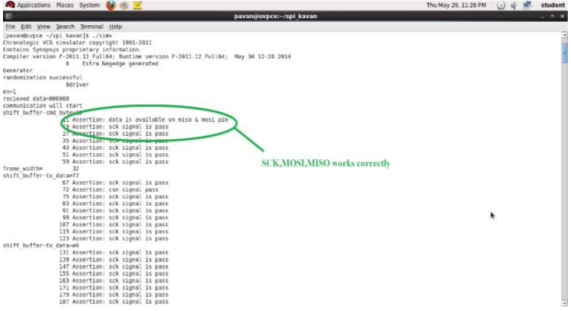
This environment we used for VIP development purpose. This is assertion based verification.

A. *Report of Master VIP*



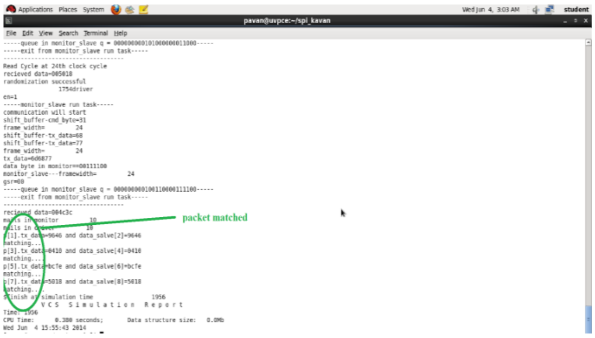
```
Parsing design file 'top.sv'
Parsing included file 'packet.sv'.
Back to file 'top.sv'.
Parsing included file 'generator.sv'.
Back to file 'top.sv'.
Parsing included file 'interface.sv'.
Back to file 'top.sv'.
Parsing included file 'driver.sv'.
Back to file 'top.sv'.
Parsing included file 'final_slave.sv'.
Back to file 'top.sv'.
Parsing included file 'monitor.sv'.
Back to file 'top.sv'.
Parsing included file 'coverage.sv'.
Back to file 'top.sv'.
Parsing included file 'scoreboard.sv'.
Back to file 'top.sv'.
Parsing included file 'environment.sv'.
Back to file 'top.sv'.
Parsing included file 'test.sv'.
Back to file 'top.sv'.
Top Level Modules:
       top
No TimeScale specified
Starting vcs inline pass...
3 modules and 0 UDP read.
       However, due to incremental compilation, no re-compilation is necessary.
ld -r -o pre_vcsobj_1_1.o --whole-archive pre_vcsobj_1_1.a --no-whole-archive
ld -r -o pre_vcsobj_1_2.o --whole-archive pre_vcsobj_1_2.a --no-whole-archive
ld -r -o pre_vcsobj_1_3.o --whole-archive pre_vcsobj_1_3.a --no-whole-archive
if [ -x ../simv ]; then chmod -x ../simv; fi
g++  -o ../simv    -Wl,-whole-archive    -Wl,-no-whole-archive   SIM_l.o 5NrI_d.o 5NrIB_d.o    pre_vcsobj_1_1.o pre_vcsobj_1_2.o pre_vcsobj_1_3.o  rmapats_mop.o rmapats
.o     /home/tools/vcsmx/vf-2011.12/amd64/lib/libvirsim.so /home/tools/vcsmx/vf-2011.12/amd64/lib/liberrorinf.so /home/tools/vcsmx/vf-2011.12/amd64/lib/libsnpsmalloc.s
o    /home/tools/vcsmx/vf-2011.12/amd64/lib/libvcsnew.so /home/tools/vcsmx/vf-2011.12/amd64/lib/libuclinative.so        /home/tools/vcsmx/vf-2011.12/amd64/lib/vcs_sav
e_restore_new.o -ldl    -lc -lm -lpthread -ldl
../simv up to date
CPU time: .172 seconds to compile + .016 seconds to elab + .124 seconds to link
[pavan@uvpce ~/spi_kavan]$ 
```

Fig. 8: Compilation of Master VIP


Applications Places System
Thu May 29, 11:28 PM
student
pavan@uvpce:~/spi_kavan
File Edit View Search Terminal Help
SCK,MOSI,MISO works correctly

```
[pavan@uvpce ~/spi_kavan]$ ./simv
Chronologic VCS simulator copyright 1991-2011
Contains Synopsys proprietary information.
Compiler version F-2011.12_Full64; Runtime version F-2011.12_Full64;  May 30 12:20 2014
                   0    Extra Negedge generated
Generator
randomization successful
                   0driver
en=1
recieved data=000000
communication will start
shift_buffer-cmd_byte=
                  11 Assertion: data is available on miso & mosi pin
                  19 Assertion: sck signal is pass
                  27 Assertion: sck signal is pass
                  35 Assertion: sck signal is pass
                  43 Assertion: sck signal is pass
                  51 Assertion: sck signal is pass
                  59 Assertion: sck signal is pass
frame_width=         32
shift_buffer-tx_data=f7
                  67 Assertion: sck signal is pass
                  72 Assertion: csn signal pass
                  75 Assertion: sck signal is pass
                  83 Assertion: sck signal is pass
                  91 Assertion: sck signal is pass
                  99 Assertion: sck signal is pass
                 107 Assertion: sck signal is pass
                 115 Assertion: sck signal is pass
                 123 Assertion: sck signal is pass
shift_buffer-tx_data=e6
                 131 Assertion: sck signal is pass
                 139 Assertion: sck signal is pass
                 147 Assertion: sck signal is pass
                 155 Assertion: sck signal is pass
                 163 Assertion: sck signal is pass
                 171 Assertion: sck signal is pass
                 179 Assertion: sck signal is pass
                 187 Assertion: sck signal is pass
```

Fig. 9: Assertion Which Checks All Pins Works Correctly

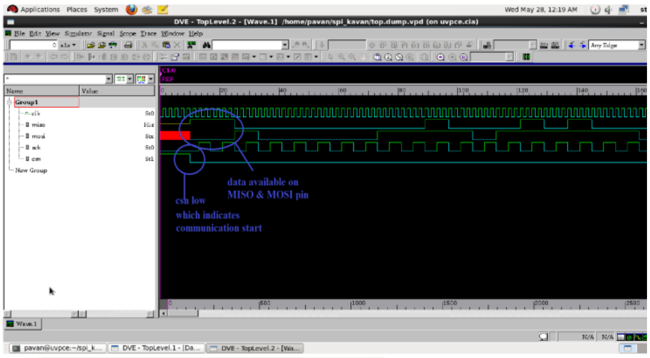


```
-----queue in monitor_slave q = 0000000001010000000011000-----
-----exit from monitor_slave run task-----
----------------------------------
Read Cycle at 24th clock cycle
recieved data=005018
randomization successful
                1754driver
en=1
-----monitor_slave run task-----
communication will start
shift_buffer-cmd_byte=31
frame_width=          24
shift_buffer-tx_data=68
shift_buffer-tx_data=77
frame_width=          24
tx_data=6d6877
data byte in monitor==00111100
monitor_slave---framewidth=          24
gsr=00
-----queue in monitor_slave q = 0000000001001100000111100-----
-----exit from monitor_slave run task-----
----------------------------------
recieved data=004c3c
mails in monitor          10
mails in driver           10
p[1].tx_data=9646 and data_salve[2]=9646
matching...
p[3].tx_data=0410 and data_salve[4]=0410
matching...
p[5].tx_data=bcfe and data_salve[6]=bcfe
matching...
p[7].tx_data=5018 and data_salve[8]=5018
matching...
$finish at simulation time                  1956
           V C S   S i m u l a t i o n   R e p o r t
Time: 1956
CPU Time:      0.380 seconds;       Data structure size:   0.0Mb
Wed Jun  4 15:55:43 2014
```

Applications Places System
Wed Jun 4, 3:03 AM student
pavan@uvpce:~/spi_kavan
File Edit View Search Terminal Help
packet matched

Fig. 10: Matching of Packets

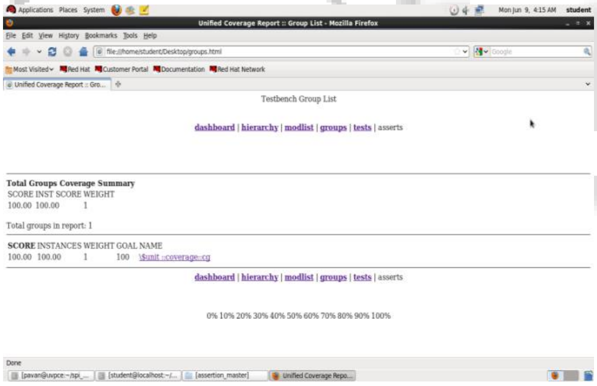
B. Simulation result of master VIP



DVE - TopLevel.2 - [Wave.1] /home/pavan/spi_kavan/top.dump.vpd (on uvpce.cia)
Name
Value
Group1
clk
miso
mosi
sck
csn
New Group
data available on MISO & MOSI pin
csn low which indicates communication start
Wave.1

Fig. 11: simulation Result of Master VIP

C. Functional coverage report of master VIP



Testbench Group List

dashboard | hierarchy | modlist | groups | tests | asserts

Total Groups Coverage Summary

```
SCORE INST SCORE WEIGHT
100.00 100.00      1

Total groups in report: 1

SCORE INSTANCES WEIGHT GOAL NAME
100.00 100.00      1      100   \$unit ::coverage::cg
```

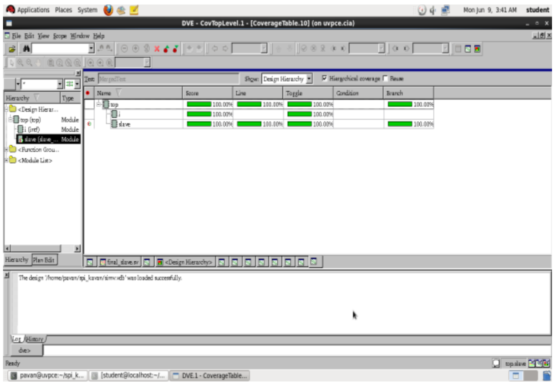
dashboard | hierarchy | modlist | groups | tests | asserts

0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

Unified Coverage Report :: Group List - Mozilla Firefox
file:///home/student/Desktop/groups.html
Most Visited Red Hat Customer Portal Documentation Red Hat Network
Done

Fig. 12: Coverage Report of Master VIP

D. Code coverage report of master VIP



Name	Score	Line	Toggle	Condition	Branch
top	100.00%	100.00%	100.00%		100.00%
i	100.00%		100.00%		
slave	100.00%	100.00%	100.00%		100.00%

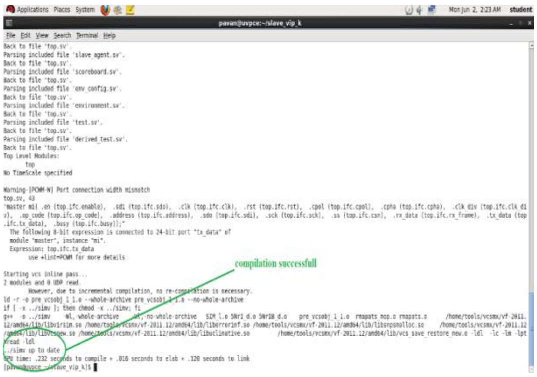
DVE - CovTopLevel.1 - [CoverageTable.10] (on uvpce.cia)
Test: MergedTest
Show: Design Hierarchy
☑ Hierarchical coverage □ Reuse
Hierarchy
Type
Module
Module
Module

Fig. 13: code coverage of Master VIP

IV. SLAVE VIP

SLAVE VIP of SPI contains master DUT and Slave environment. This environment is used to verify master DUT. If we got desirable output in scoreboard class or we can say if packets matched in to scoreboard than functionality of master DUT is up to date.

A. Report of slave VIP



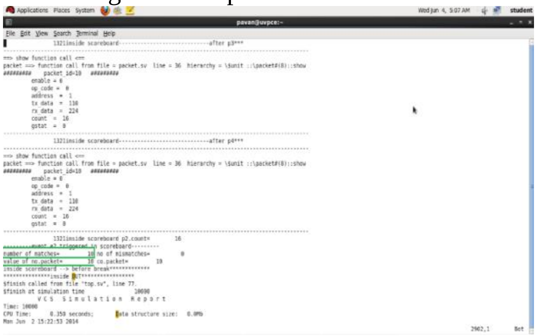
```
Back to file 'top.sv'.
Parsing included file 'slave_agent.sv'.
Back to file 'top.sv'.
Parsing included file 'scoreboard.sv'.
Back to file 'top.sv'.
Parsing included file 'env_config.sv'.
Back to file 'top.sv'.
Parsing included file 'environment.sv'.
Back to file 'top.sv'.
Parsing included file 'test.sv'.
Back to file 'top.sv'.
Parsing included file 'derived_test.sv'.
Back to file 'top.sv'.
Top Level Modules:
       top
No TimeScale specified

Warning-[PCWM-W] Port connection width mismatch
top.sv, 43
"master mi( .en (top.ifc.enable),  .sdi (top.ifc.sdo),  .clk (top.ifc.clk),  .rst (top.ifc.rst),  .cpol (top.ifc.cpol),  .cpha (top.ifc.cpha),  .clk_div (top.ifc.clk_di
v),  .op_code (top.ifc.op_code),  .address (top.ifc.address),  .sdo (top.ifc.sdi),  .sck (top.ifc.sck),  .ss (top.ifc.csn),  .rx_data (top.ifc.rx_frame),  .tx_data (top
.ifc.tx_data),  .busy (top.ifc.busy));"
  The following 8-bit expression is connected to 24-bit port "tx_data" of
  module "master", instance "mi".
  Expression: top.ifc.tx_data
       use +lint=PCWM for more details

Starting vcs inline pass...
2 modules and 0 UDP read.
       However, due to incremental compilation, no re-compilation is necessary.
ld -r -o pre_vcsobj_1_1.o --whole-archive pre_vcsobj_1_1.a --no-whole-archive
if [ -x ../simv ]; then chmod -x ../simv; fi
g++  -o ../simv    -Wl,-whole-archive    -Wl,-no-whole-archive   5IM_l.o 5NrI_d.o 5NrIB_d.o    pre_vcsobj_1_1.o  rmapats_mop.o rmapats.o      /home/tools/vcsmx/vf-2011.
12/amd64/lib/libvirsim.so /home/tools/vcsmx/vf-2011.12/amd64/lib/liberrorinf.so /home/tools/vcsmx/vf-2011.12/amd64/lib/libsnpsmalloc.so     /home/tools/vcsmx/vf-2011.12
/amd64/lib/libvcsnew.so /home/tools/vcsmx/vf-2011.12/amd64/lib/libuclinative.so        /home/tools/vcsmx/vf-2011.12/amd64/lib/vcs_save_restore_new.o -ldl  -lc -lm -lpt
hread -ldl
../simv up to date
CPU time: .232 seconds to compile + .016 seconds to elab + .120 seconds to link
[pavan@uvpce ~/slave_vip_k]$
```

Applications Places System
Mon Jun 2, 2:23 AM student
pavan@uvpce:~/slave_vip_k
File Edit View Search Terminal Help
compilation successfull

Fig. 14: Compilation of Slave VIP

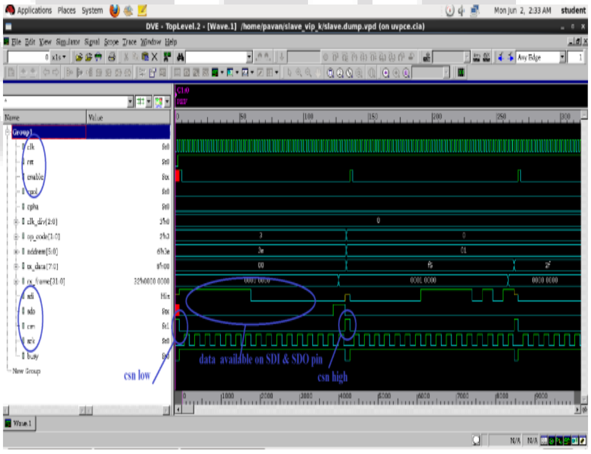


```
              1321inside scoreboard---------------------------after p3***
-----------------------------------------------------------------------------
==> show function call <==
packet ==> function call from file = packet.sv  line = 36  hierarchy = \$unit ::\packet#(8)::show
#########    packet_id=10    #########
        enable = 0
        op_code =  0
        address  =  1
        tx_data  =  110
        rx_data  =  224
        count  =  16
        gstat  =  0
-----------------------------------------------------------------------------
              1321inside scoreboard---------------------------after p4***
-----------------------------------------------------------------------------
==> show function call <==
packet ==> function call from file = packet.sv  line = 36  hierarchy = \$unit ::\packet#(8)::show
#########    packet_id=10    #########
        enable = 0
        op_code =  0
        address  =  1
        tx_data  =  110
        rx_data  =  224
        count  =  16
        gstat  =  0
-----------------------------------------------------------------------------
              1321inside scoreboard p2.count=        16
----------event e2 triggered in scoreboard---------
number of matches=        10 no of mismatches=          0
value of no.packet=       10 co.packet=        10
inside scoreboard --> before break*************
***************inside DUT*****************
$finish called from file "top.sv", line 77.
$finish at simulation time              10000
           V C S   S i m u l a t i o n   R e p o r t
Time: 10000
CPU Time:      0.350 seconds;       Data structure size:   0.0Mb
Mon Jun  2 15:22:53 2014
                                                                2962,1         Bot
```

Applications Places System
Wed Jun 4, 5:07 AM student
pavan@uvpce:~
File Edit View Search Terminal Help

Fig. 15: Packet Matching

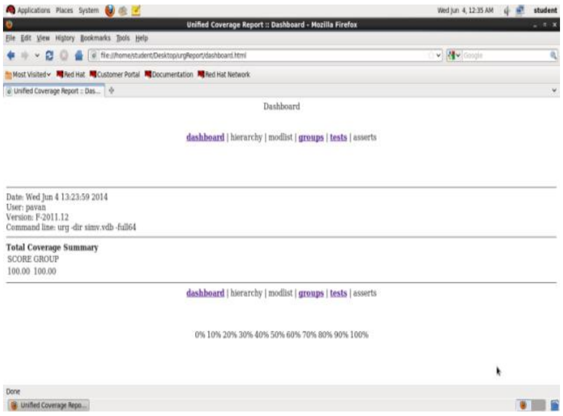
B. Simulation result of slave VIP



Applications Places System
Mon Jun 2, 2:33 AM student
DVE - TopLevel.2 - [Wave.1] /home/pavan/slave_vip_k/slave.dump.vpd (on uvpce.cia)
Name
Value
Group1
clk
rst
enable
cpol
cpha
clk_div[2:0]
op_code[1:0]
address[5:0]
tx_data[7:0]
rx_frame[31:0]
sdi
sdo
csn
sck
busy
New Group
data available on SDI & SDO pin
csn low
csn high
Wave.1

Fig. 16: Simulation Result of Slave VIP

C. Coverage report of slave VIP



Dashboard

dashboard | hierarchy | modlist | groups | tests | asserts

Date: Wed Jun 4 13:23:59 2014
User: pavan
Version: F-2011.12
Command line: urg -dir simv.vdb -full64

Total Coverage Summary

```
SCORE GROUP
100.00 100.00
```

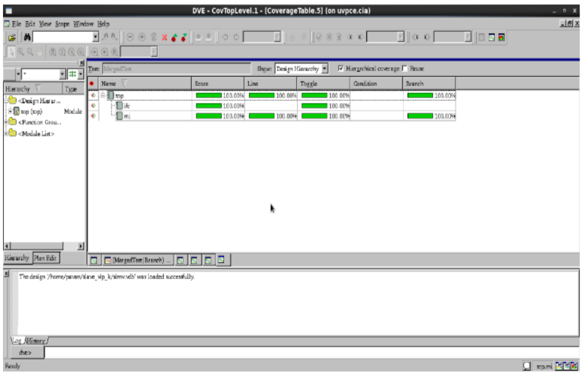
dashboard | hierarchy | modlist | groups | tests | asserts

0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

Applications Places System
Wed Jun 4, 12:35 AM student
Unified Coverage Report :: Dashboard - Mozilla Firefox
file:///home/student/Desktop/urgReport/dashboard.html
Most Visited Red Hat Customer Portal Documentation Red Hat Network
Done

Fig. 17 Coverage Report of Slave VIP

D. Code coverage of slave VIP



Name	Score	Line	Toggle	Condition	Branch
top	100.00%	100.00%	100.00%		100.00%
ifc	100.00%		100.00%		
mi	100.00%	100.00%	100.00%		100.00%

The design '/home/pavan/slave_vip_k/simv.vdb' was loaded successfully.

Fig. 18: code Coverage of slave VIP

V. CONCLUSIONS

In the project the verification environment of SPI Interface was developed using System Verilog, The idea was to achieve the properties like re-usability and robustness, the results shown here are limited to only one master and slave but if used for multiple slave the components are designed such that they can be reused easily which can be checked with the architectural diagram of the environment. We have designed parameterized components so that can be configured as per requirement. Analysis of results shows that that reusability can be achieved using the flow described in the report.

REFERENCES

[1] An introduction to i2c and spi protocols ,by Frédéric Leens, IEEE Instrumentation & Measurement Magazine February 2009.

[2] 2. Design and Test of General-Purpose SPI Master/Slave IPs on OPB Bus ,by A.K. Oudjida, M.L. Berrandjia, A. Liacha, R. Tiar, K. Tahraoui & Y.N. Alhoumays , 2010 7th International Multi-Conference on Systems, Signals and Devices

[3] IP Design of Universal Multiple Devices SPI Interface , by Tianxiang Liu , Yunfeng Wang.

[4] Development of Verification Envioronment for SPI Master Interface Using System Verilog, by Zhili Zhou, Zheng Xie, Xin'an Wang and Teng Wang 2012 IEEE.

[5] Solutions for SPI protocol testing and debugging in embedded system , Byte Paradigm White Paper.

[6] Designing a Serial Peripheral Interface , Application Note by Eric Lin.

[7] 7. TN0897 Technical note ST SPI protocol , October 2012 Doc ID 023176 Rev 1

[8] AN10216-01 I2C MANUAL, Philips Semiconductors March 24, 2003.

[9] Design and Verification of Serial Peripheral Interface, by M.Sandya, K.Rajasekhar, International Journal of Engineering Trends and Technology- Volume3Issue4-2012