**Project2 – Distributed Network**

Weichun Xu

Florida Institute of Technology

**Abstract**

*Distributed Network is a distributed computing network system and a part of distributed computing architecture, which deliver the data to remote computers, a number of networks, processors and intermediary devices for processing, i.e., the data to be worked on are spread out over more than one computer when the computer operating. A distributed network is powered by network management tools, which manage data routing, combine and allocate network bandwidth and control access and other networking resources.*

*A distributed network is a type of computer network that is spread over different networks. This provides a single data communication network, which can be managed jointly or separately by each network. Besides shared communication within the network, a distributed network often also distributes processing.*

*Client/server computing architecture is a popular trend of a distributed network in which distributed computer networks consist of clients and servers connected in such a way that any client can potentially communicate with any server. The server is the producer of a resource and many interconnected remote users are the consumers who access the application from different networks. This means that an application may be hosted and executed from a single machine but accessed by many others.*

*A distributed environment has interesting characteristics. It takes advantage of client/server computing and architectures. It distributes processing to inexpensive systems and relieves servers of many tasks. Data may be accessed from a diversity of sites over wired or wireless networks. Data may be replicated to other systems to provide fault tolerance and place data close to users.*

*Prior to the emergence of low-cost desktop computer power, computing was generally centralized to one computer. Although such centers still exist, distribution networking applications and data operate more efficiently over a mix of desktop workstations, local area network servers, regional servers, Web servers, and other servers.*

*Nowadays the trend of processing data is from expensive centralized systems to low-cost distributed systems that can be installed in large numbers. In the late 1980s and early 1990s, distributed systems consisted of large numbers of desktop computers. Today, with the development of smartphone and tablet, the Internet and Web technologies have greatly expanded the concept of distributed systems.*

*One aspect of distributed computing is the technique of parallel processing, which is best described as multiprocessing that takes place on systems that are locally attached via a high-speed interface, like GPU parallel computing. There is also dedicated distributed computing which has the ability to run programs in parallel on multiple computers. It is best described as multiprocessing that takes place across computers connected via LANs or the Internet, like IBM and Oracle’s gridEngine. HPC (high performance computing) environments are large collections of workstations, often called "grid environments."*

*Distributed parallel processing across multiple computer systems requires an authoritative scheduling program that can decide where and when to execute parts of a program. Distribution of tasks may take place in real time or on a more relaxed schedule.*

*Companies that have grown in scale over the years and those that are continuing to grow are finding it extremely challenging to manage their distributed network in the traditional client/server computing model. The recent developments in the field of cloud computing has created new possibilities. Cloud-based networking vendors have started to sprout offering solutions for enterprise distributed networking needs, which may turn out to revolutionize the distributed networking space.*

1. **INTRODUCTION**
   1. **Introduction of Distributed Network**

Distributed Network is a distributed computing network system and a part of distributed computing architecture, which deliver the data to different remote computers, a number of networks, processors and intermediary devices for processing.

* 1. **Problem Statement**

When delivering the data to remote computers for processing, the delivery system will be the bottleneck if using CPU. Although multiple CPUs is common and easy to be available in today’s world, the number of parallel processing is still limited.

* 1. **Motivation**

When using CPUs or GPUs to do parallel computing, there is always limitation in the number of parallel threads which is constrained by the limiation of hardware. However FPGA can run all the processes in parallel and current technology make FPGA contain a large amount of resources.

* 1. **Proposed Solution**

FPGA platform is a suitable platform for parallel computing and FPGA board is getting cheaper and cheaper, so it’s eay to get a FPGA development platform. Based on the feature of parallel computing of FPGA, we will use FPGA to deliver the data in parallel, which will send the data to other FPGAs. In the future we will also use FPGA to compute the data in parallel to reduce the computation time.

1. **RELATED WORK**

Recent research focus on using a number of CPUs or GPUs to deliver and process data in parallel.

In the paper [1], the author solves the problem of training a deep network with billions of parameters using tens of thousands of CPU cores. The time in computing is reduce a lot by using multiple CPUs for parallel computing.

1. **PROJECT PROPOSAL**

FPGA is an original hardware parallel processing device, therefore it’s feasible and suitable to use FPGA in distributed network for delivering and processing data.

1. **Structure of the Design**

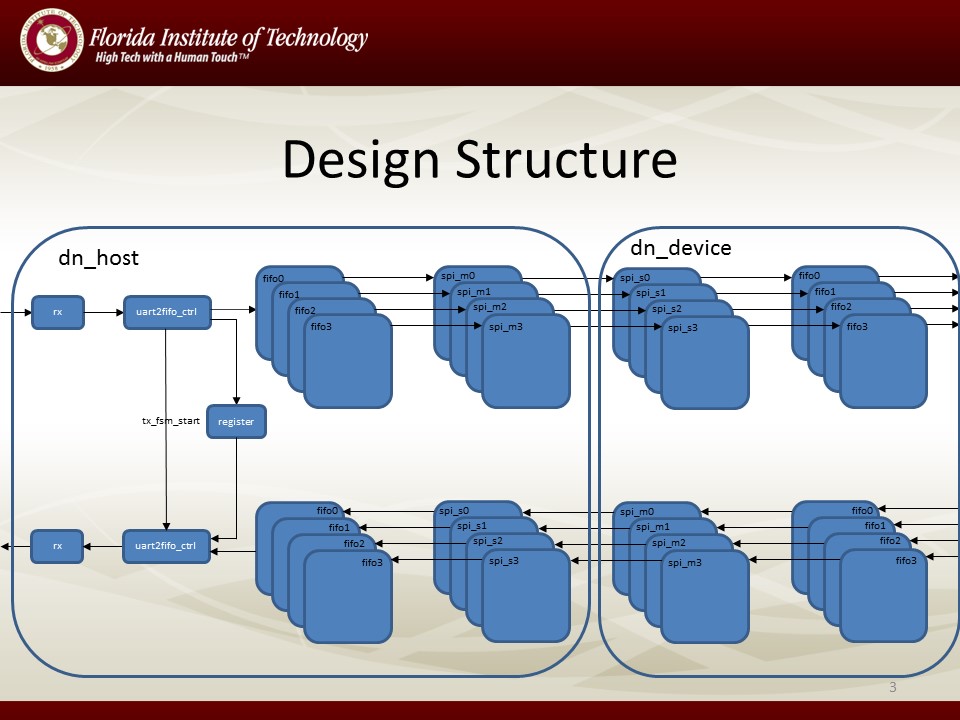


Figure design structure

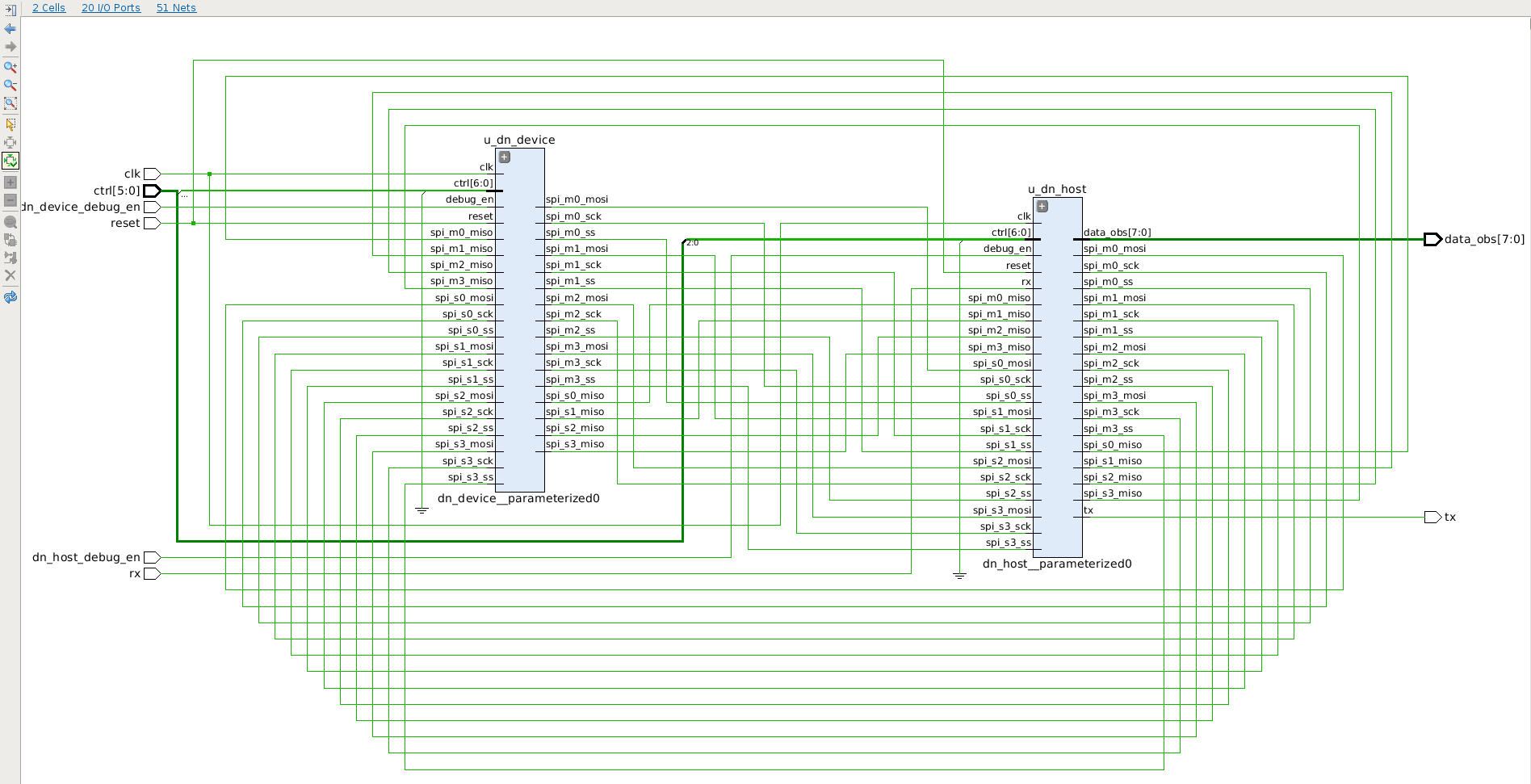


Figure schematic of virtual\_env

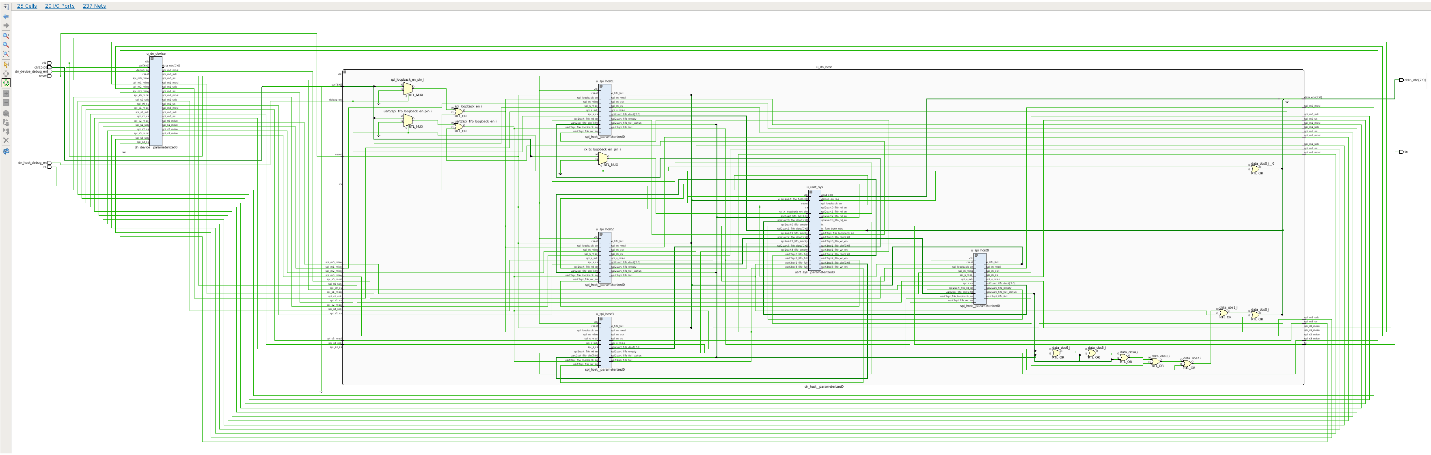


Figure schematic of dn\_host

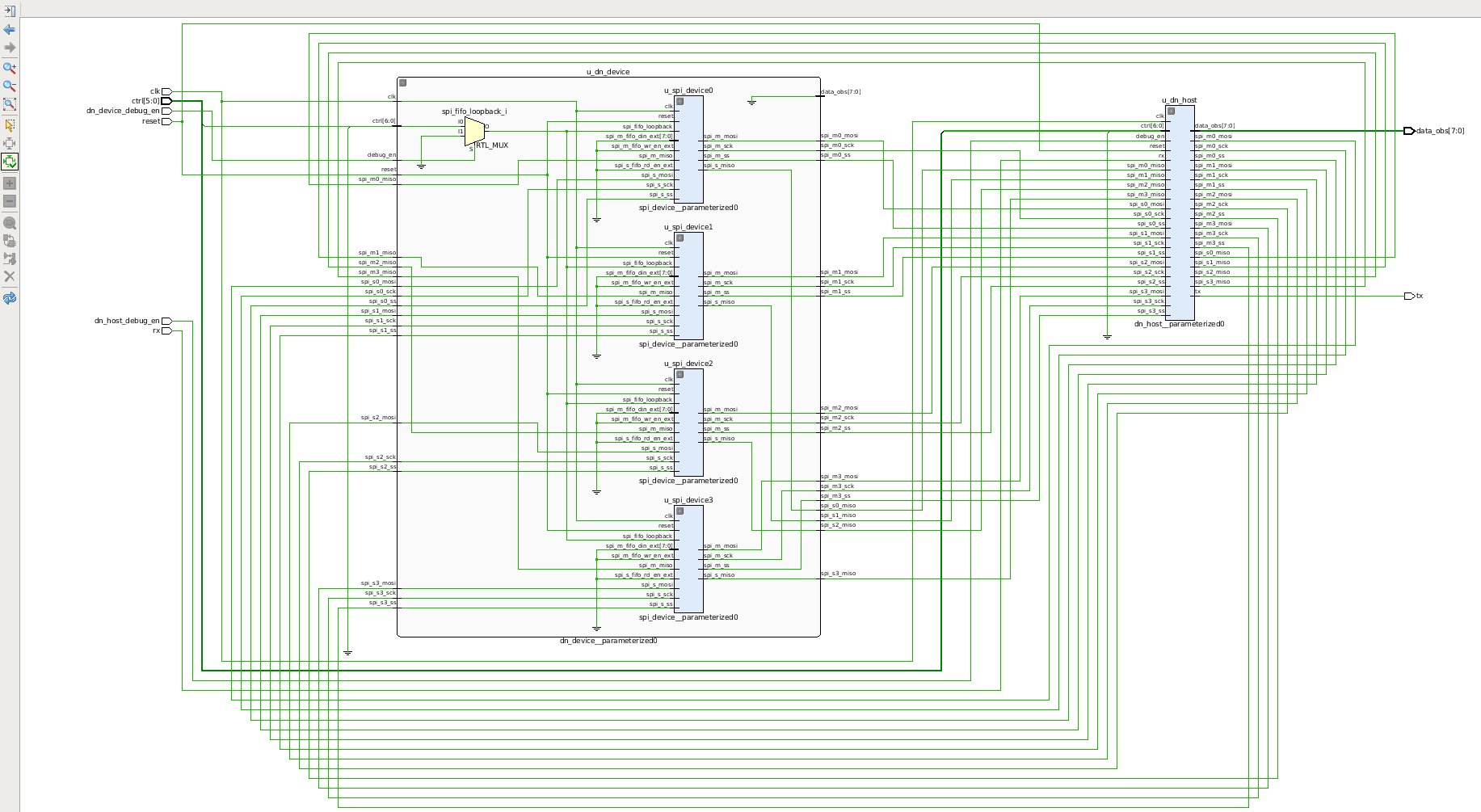


Figure schematic of dn\_device

1. **Command Format**
   1. Write Register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | x | 0 | addr[2] | addr[1] | addr[0] |
| data[7] | data[6] | data[5] | data[4] | data[3] | data[2] | data[1] | data[0] |

Table write register

* 1. Read register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | x | 0 | addr[2] | addr[1] | addr[0] |

Table read register

* 1. Write Fifo
     1. Unicast Mode

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 1 | x | x | x |
| length[7] | length[6] | length[5] | length[4] | length[3] | length[2] | length[1] | length[0] |
| data[0][7] | data[0][6] | data[0][5] | data[0][4] | data[0][3] | data[0][2] | data[0][1] | data[0][0] |
| …… | | | | | | | |
| data[length-1][7] | data[length-1][6] | data[length-1][5] | data[length-1][4] | data[length-1][3] | data[length-1][2] | data[length-1][1] | data[length-1][0] |

Table unicast write fifo

* + 1. Broadcast Mode

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 1 | 1 | x | x | x |
| length[7] | length[6] | length[5] | length[4] | length[3] | length[2] | length[1] | length[0] |
| data[0][7] | data[0][6] | data[0][5] | data[0][4] | data[0][3] | data[0][2] | data[0][1] | data[0][0] |
| …… | | | | | | | |
| data[length-1][7] | data[length-1][6] | data[length-1][5] | data[length-1][4] | data[length-1][3] | data[length-1][2] | data[length-1][1] | data[length-1][0] |

Table broadcast write fifo

* 1. Read Fifo

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | x | 1 | x | x | x |

Table read fifo

1. **Debug Mode**

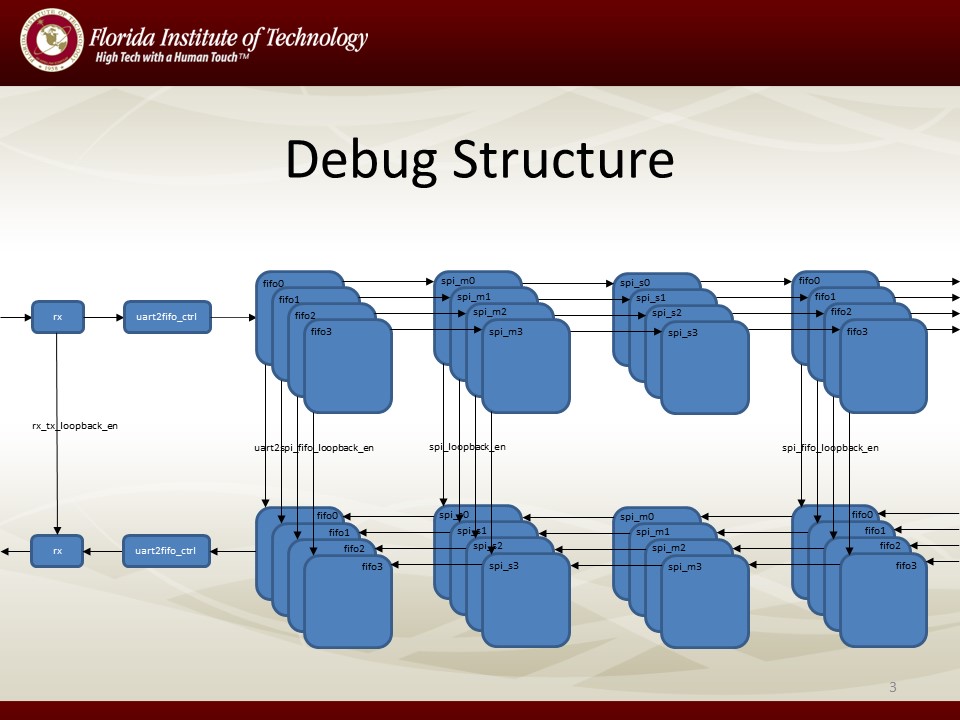


Figure loopback structure

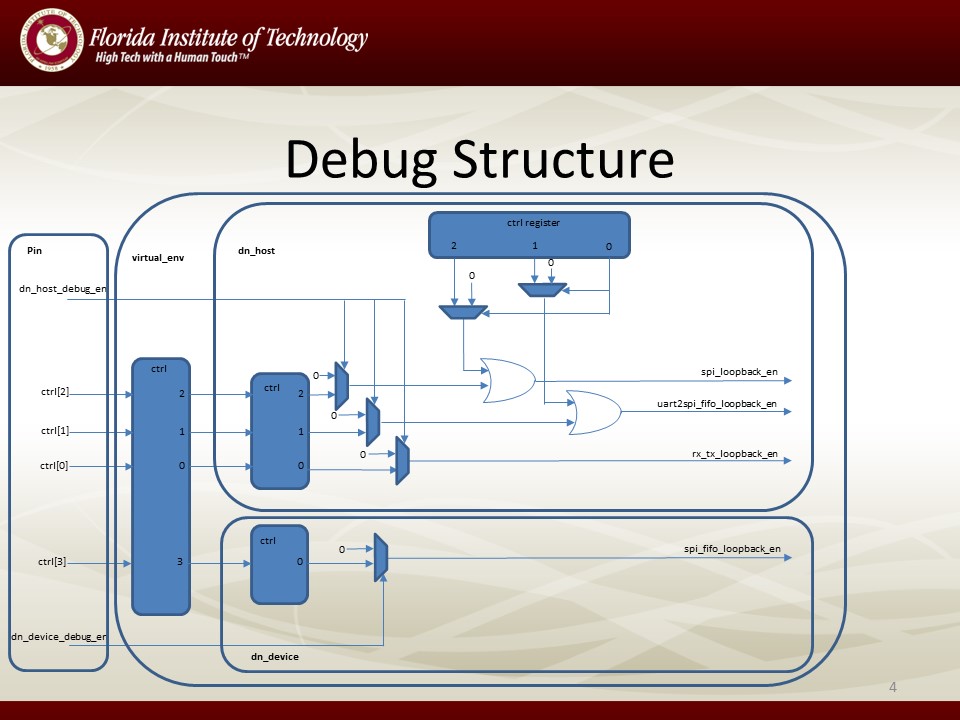


Figure debug control system

|  |  |
| --- | --- |
| Signal | Description |
| rx\_tx\_loopback\_en | Loop UART RX back to UART TX |
| uart2spi\_fifo\_loopback\_en | Loop data from SPI master fifo directly back to SPI slave fifo in dn\_host |
| spi\_loopback\_en | Loop data from SPI master back to SPI slave in dn\_host |
| spi\_fifo\_loopback\_en | Loop data from SPI slave fifo directly back to SPI master fifo in dn\_device |

Table Debug Signal Description

1. **Tables of Pin Assignment**

|  |  |  |  |
| --- | --- | --- | --- |
| Ports | Pin Location | Pin Function | Comment |
| clk | Y9 | GCLK | Clock Signal |
| reset | T18 | BTNU | Reset Signal |
| tx | W11 | JB2 | UART TX |
| rx | V10 | JB3 | UART RX |
| dn\_host\_debug\_en | M15 | SW7 | Enable dn\_host debug |
| dn\_device\_debug\_en | H17 | SW6 | Enable dn\_device debug |
| ctrl[5] | H18 | SW5 | NA |
| ctrl[4] | H19 | SW4 | NA |
| ctrl[3] | F21 | SW3 | dn\_device fifo loopback |
| ctrl[2] | H22 | SW2 | dn\_host SPI Master to Slave loopback |
| ctrl[1] | G22 | SW1 | dn\_host fifo loopback |
| ctrl[0] | F22 | SW0 | UART RX to TX loopback |
| data\_obs[7] | U14 | LD7 | If the machine is in debug mode |
| data\_obs[6] | U19 | LD6 | - |
| data\_obs[5] | W22 | LD5 | UART TX FSM busy signal |
| data\_obs[4] | V22 | LD4 | Fifo Full status of all the fifos between UART and SPI |
| data\_obs[3] | U21 | LD3 | - |
| data\_obs[2] | U22 | LD2 | Bit 2 of RX Received Command |
| data\_obs[1] | T21 | LD1 | Bit 1 of RX Received Command |
| data\_obs[0] | T22 | LD0 | Bit 0 of RX Received Command |

Table Pin Assignment

1. **Timing Report**

Copyright 1986-2014 Xilinx, Inc. All Rights Reserved.

--------------------------------------------------------------------------------------------------------------------------------------------------------------------

| Tool Version : Vivado v.2014.4 (lin64) Build 1071353 Tue Nov 18 16:47:07 MST 2014

| Date : Sat Apr 18 15:07:08 2015

| Host : localhost.localdomain running 64-bit CentOS Linux release 7.1.1503 (Core)

| Command : report\_timing\_summary -warn\_on\_violation -max\_paths 10 -file virtual\_top\_timing\_summary\_routed.rpt -rpx virtual\_top\_timing\_summary\_routed.rpx

| Design : virtual\_top

| Device : 7z020-clg484

| Speed File : -1 PRODUCTION 1.11 2014-09-11

| Temperature Grade : C

--------------------------------------------------------------------------------------------------------------------------------------------------------------------

Timing Summary Report

------------------------------------------------------------------------------------------------

| Timer Settings

| --------------

------------------------------------------------------------------------------------------------

Enable Multi Corner Analysis : Yes

Enable Pessimism Removal : Yes

Pessimism Removal Resolution : Nearest Common Node

Enable Input Delay Default Clock : No

Enable Preset / Clear Arcs : No

Disable Flight Delays : No

Corner Analyze Analyze

Name Max Paths Min Paths

------ --------- ---------

Slow Yes Yes

Fast Yes Yes

check\_timing report

Table of Contents

-----------------

1. checking no\_clock

2. checking constant\_clock

3. checking pulse\_width\_clock

4. checking unconstrained\_internal\_endpoints

5. checking no\_input\_delay

6. checking no\_output\_delay

7. checking multiple\_clock

8. checking generated\_clocks

9. checking loops

10. checking partial\_input\_delay

11. checking partial\_output\_delay

12. checking unexpandable\_clocks

13. checking latch\_loops

1. checking no\_clock

--------------------

There are 718 register/latch pins with no clock driven by root clock pin: clk (HIGH)

There are 184 register/latch pins with no clock driven by root clock pin: ctrl[2] (HIGH)

There are 184 register/latch pins with no clock driven by root clock pin: dn\_host\_debug\_en (HIGH)

There are 23 register/latch pins with no clock driven by root clock pin: u\_dn\_device/u\_spi\_device0/u\_spi\_m/u\_spi\_clk\_gen/r\_reg\_reg[1]/C (HIGH)

There are 23 register/latch pins with no clock driven by root clock pin: u\_dn\_device/u\_spi\_device1/u\_spi\_m/u\_spi\_clk\_gen/r\_reg\_reg[1]/C (HIGH)

There are 23 register/latch pins with no clock driven by root clock pin: u\_dn\_device/u\_spi\_device2/u\_spi\_m/u\_spi\_clk\_gen/r\_reg\_reg[1]/C (HIGH)

There are 23 register/latch pins with no clock driven by root clock pin: u\_dn\_device/u\_spi\_device3/u\_spi\_m/u\_spi\_clk\_gen/r\_reg\_reg[1]/C (HIGH)

There are 46 register/latch pins with no clock driven by root clock pin: u\_dn\_host/u\_spi\_host0/u\_spi\_m/u\_spi\_clk\_gen/r\_reg\_reg[1]/C (HIGH)

There are 46 register/latch pins with no clock driven by root clock pin: u\_dn\_host/u\_spi\_host1/u\_spi\_m/u\_spi\_clk\_gen/r\_reg\_reg[1]/C (HIGH)

There are 46 register/latch pins with no clock driven by root clock pin: u\_dn\_host/u\_spi\_host2/u\_spi\_m/u\_spi\_clk\_gen/r\_reg\_reg[1]/C (HIGH)

There are 46 register/latch pins with no clock driven by root clock pin: u\_dn\_host/u\_spi\_host3/u\_spi\_m/u\_spi\_clk\_gen/r\_reg\_reg[1]/C (HIGH)

There are 184 register/latch pins with no clock driven by root clock pin: u\_dn\_host/u\_uart\_sys/u\_dn\_reg/ctrl\_reg[0]/C (HIGH)

There are 184 register/latch pins with no clock driven by root clock pin: u\_dn\_host/u\_uart\_sys/u\_dn\_reg/ctrl\_reg[2]/C (HIGH)

2. checking constant\_clock

--------------------------

There are 0 register/latch pins with constant\_clock.

3. checking pulse\_width\_clock

-----------------------------

There are 0 register/latch pins which need pulse\_width check

4. checking unconstrained\_internal\_endpoints

--------------------------------------------

There are 1981 pins that are not constrained for maximum delay. (HIGH)

There are 0 pins that are not constrained for maximum delay due to constant clock.

5. checking no\_input\_delay

--------------------------

There are 9 input ports with no input delay specified. (HIGH)

There are 0 input ports with no input delay but user has a false path constraint.

6. checking no\_output\_delay

---------------------------

There are 7 ports with no output delay specified. (HIGH)

There are 0 ports with no output delay but user has a false path constraint

There are 0 ports with no output delay but with a timing clock defined on it or propagating through it

7. checking multiple\_clock

--------------------------

There are 0 register/latch pins with multiple clocks.

8. checking generated\_clocks

----------------------------

There are 0 generated clocks that are not connected to a clock source.

9. checking loops

-----------------

There are 0 combinational loops in the design.

10. checking partial\_input\_delay

--------------------------------

There are 0 input ports with partial input delay specified.

11. checking partial\_output\_delay

---------------------------------

There are 0 ports with partial output delay specified.

12. checking unexpandable\_clocks

--------------------------------

There are 0 unexpandable clock pairs.

13. checking latch\_loops

------------------------

There are 0 combinational latch loops in the design through latch input

------------------------------------------------------------------------------------------------

| Design Timing Summary

| ---------------------

------------------------------------------------------------------------------------------------

WNS(ns) TNS(ns) TNS Failing Endpoints TNS Total Endpoints WHS(ns) THS(ns) THS Failing Endpoints THS Total Endpoints WPWS(ns) TPWS(ns) TPWS Failing Endpoints TPWS Total Endpoints

------- ------- --------------------- ------------------- ------- ------- --------------------- ------------------- -------- -------- ---------------------- --------------------

NA NA NA NA NA NA NA NA NA NA NA NA

All user specified timing constraints are met.

------------------------------------------------------------------------------------------------

| Clock Summary

| -------------

------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------

| Intra Clock Table

| -----------------

------------------------------------------------------------------------------------------------

Clock WNS(ns) TNS(ns) TNS Failing Endpoints TNS Total Endpoints WHS(ns) THS(ns) THS Failing Endpoints THS Total Endpoints WPWS(ns) TPWS(ns) TPWS Failing Endpoints TPWS Total Endpoints

----- ------- ------- --------------------- ------------------- ------- ------- --------------------- ------------------- -------- -------- ---------------------- --------------------

------------------------------------------------------------------------------------------------

| Inter Clock Table

| -----------------

------------------------------------------------------------------------------------------------

From Clock To Clock WNS(ns) TNS(ns) TNS Failing Endpoints TNS Total Endpoints WHS(ns) THS(ns) THS Failing Endpoints THS Total Endpoints

---------- -------- ------- ------- --------------------- ------------------- ------- ------- --------------------- -------------------

------------------------------------------------------------------------------------------------

| Other Path Groups Table

| -----------------------

------------------------------------------------------------------------------------------------

Path Group From Clock To Clock WNS(ns) TNS(ns) TNS Failing Endpoints TNS Total Endpoints WHS(ns) THS(ns) THS Failing Endpoints THS Total Endpoints

---------- ---------- -------- ------- ------- --------------------- ------------------- ------- ------- --------------------- -------------------

------------------------------------------------------------------------------------------------

| Timing Details

| --------------

------------------------------------------------------------------------------------------------

1. **Verification Environment**

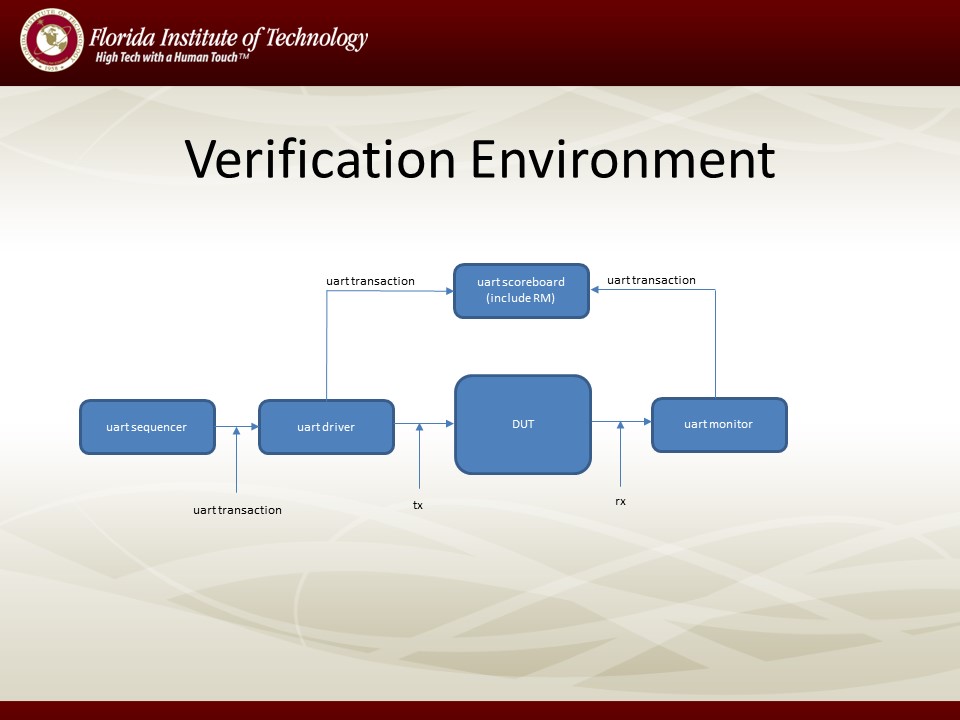


Figure verification environment

1. **Test List**

|  |  |
| --- | --- |
| Test Name | Description |
| uart\_multiple\_write\_reg\_test | Check register path |
| uart\_multiple\_write\_fifo\_unicast\_test | Check data path, verify unicast mode and write one byte |
| uart\_multiple\_write\_fifo\_burst\_unicast\_test | Check data path, verify unicast mode and write multiple bytes |
| uart\_multiple\_write\_fifo\_broadcast\_test | Check data path, verify broadcast mode and write one byte |
| uart\_multiple\_write\_fifo\_burst\_broadcast\_test | Check data path, verify broadcast mode and write multiple bytes |
| uart\_multiple\_write\_fifo\_randcast\_test | Check data path, write one byte, random unicast and broadcast |
| uart\_multiple\_write\_fifo\_burst\_randcast\_test | Check data path, write multiple bytes, random unicast and broadcast |
| uart\_rx\_tx\_loopback\_test | Loopback uart tx to uart tx |
| uart\_hard\_uart2spi\_fifo\_loopback\_test | Use pin to enable debug mode and enable SPI fifo loopback in dn\_host |
| uart\_soft\_uart2spi\_fifo\_loopback\_test | Use register to enable debug mode and enable SPI fifo loopback in dn\_host |
| uart\_hard\_spi\_loopback\_test | Use pin to enable debug mode and enable SPI loopback in dn\_host |
| uart\_soft\_spi\_loopback\_test | Use register to enable debug mode and enable SPI loopback in dn\_host |

Table Test List

1. **Simulations Snapshots of Results**

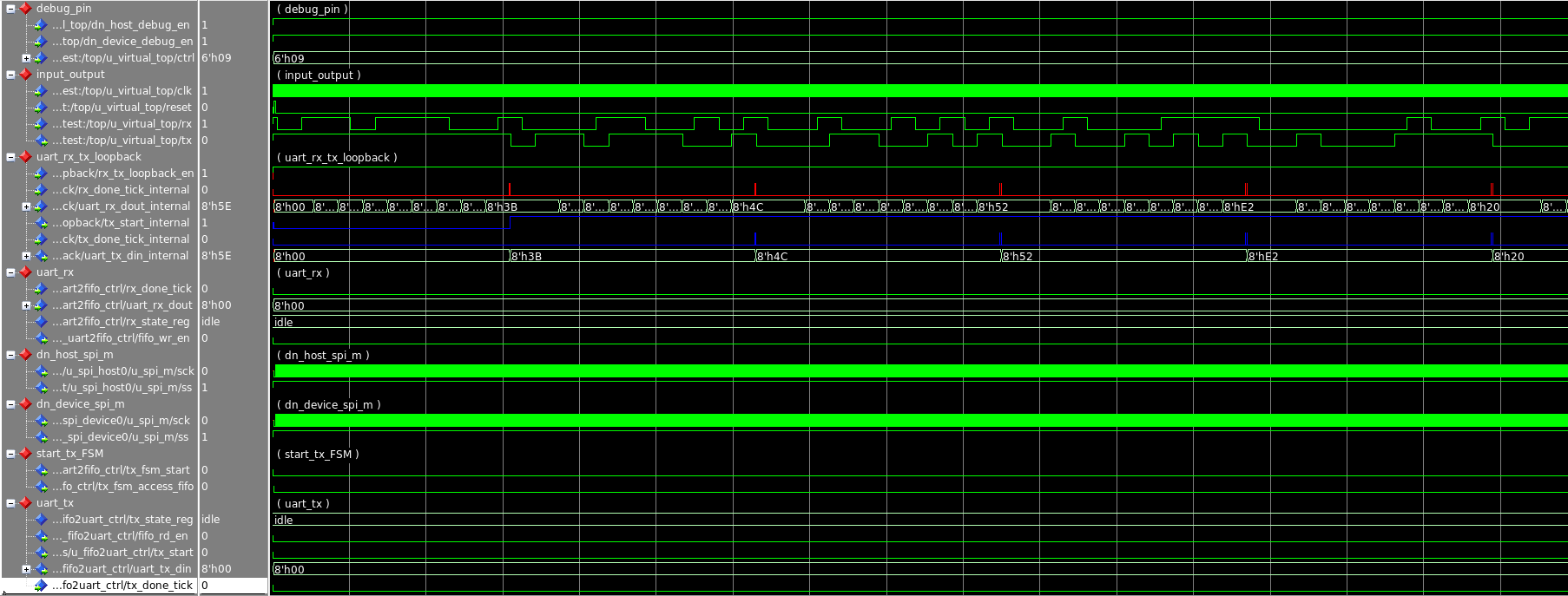


Figure loopback uart rx to uart tx

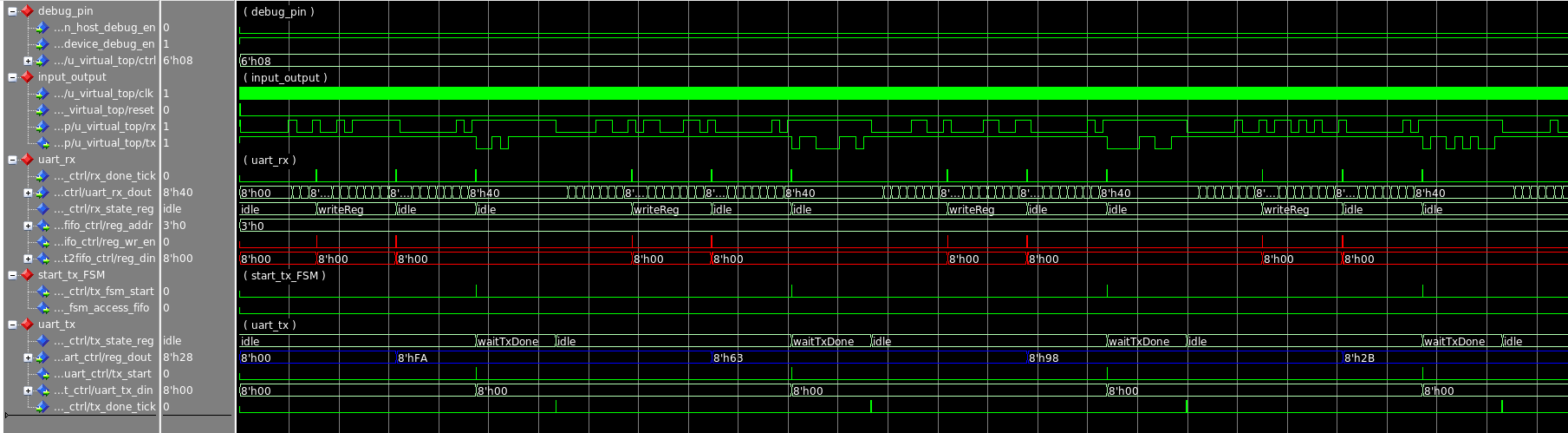


Figure write and read register

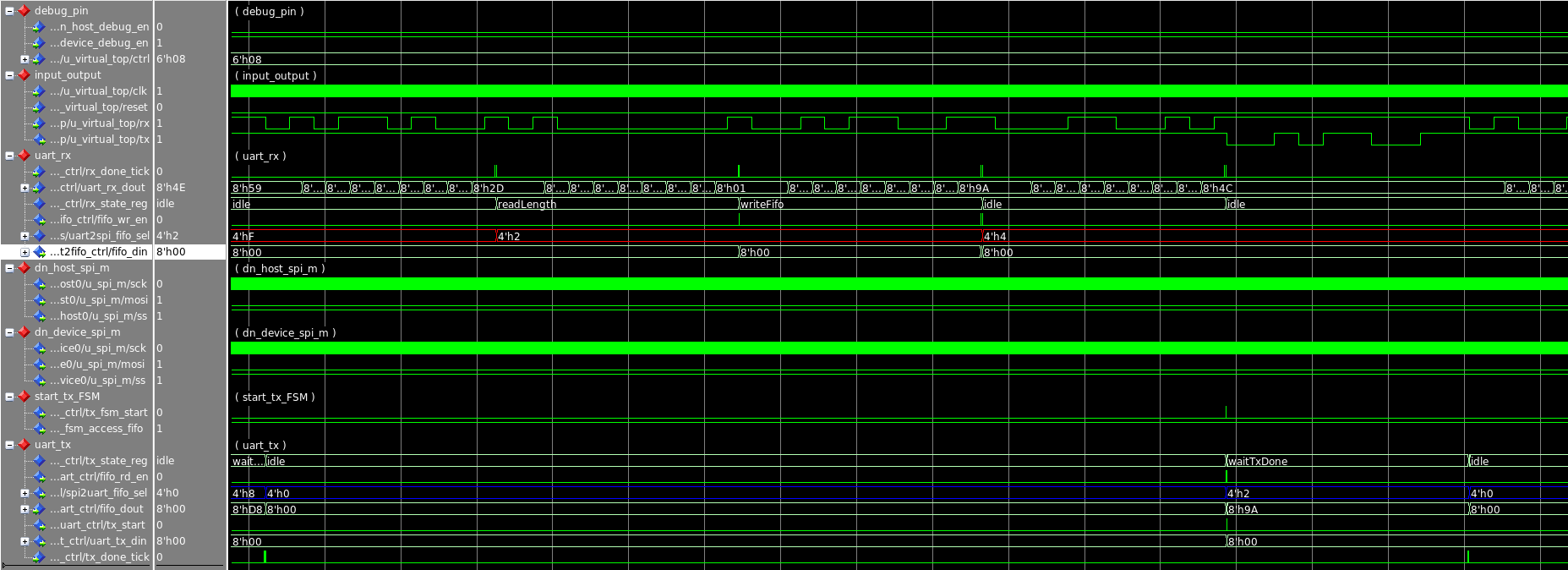
****

Figure write and read one byte to FPGA by using unicast

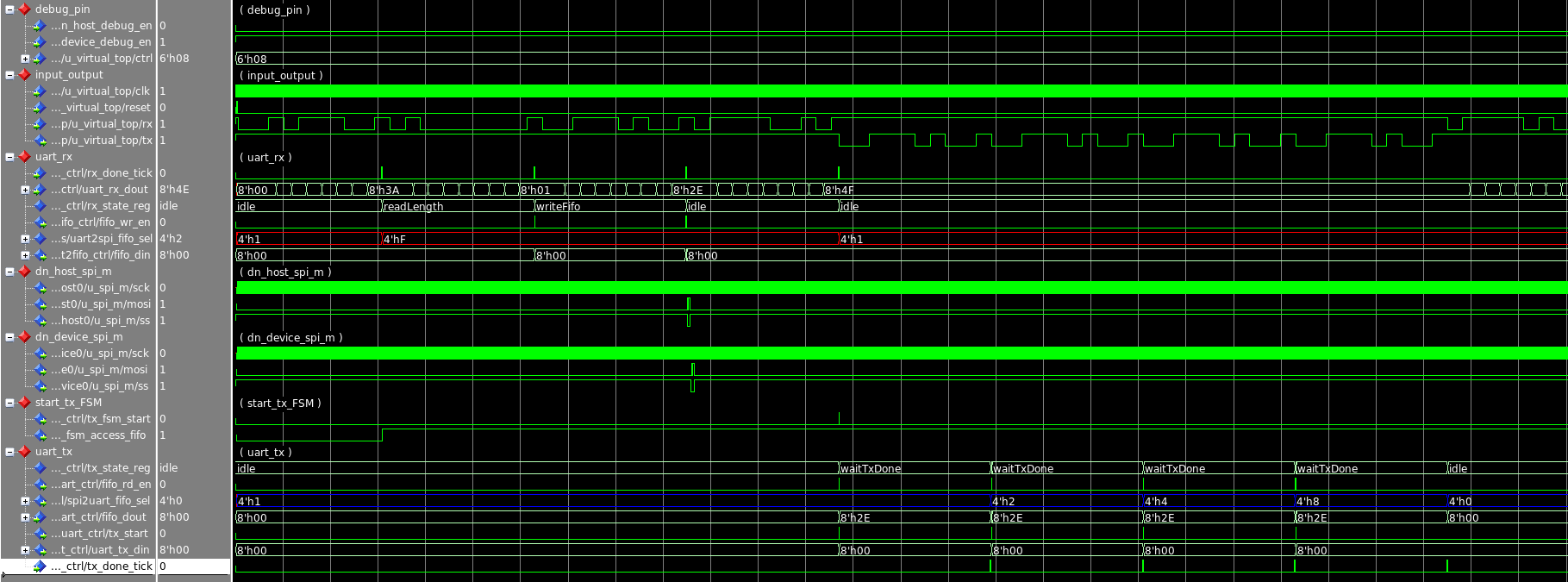
****

Figure write and read one byte to FPGA by using broadcast

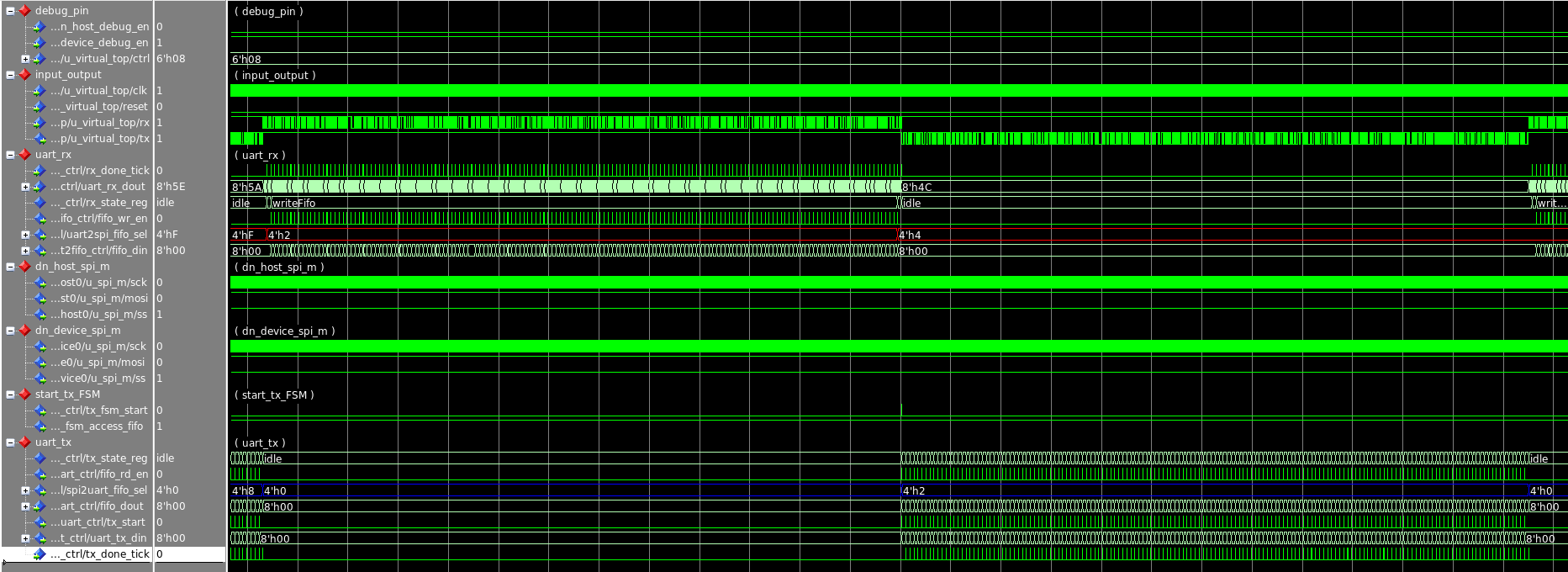
****

Figure write and read multiple bytes to FPGA by using unicast

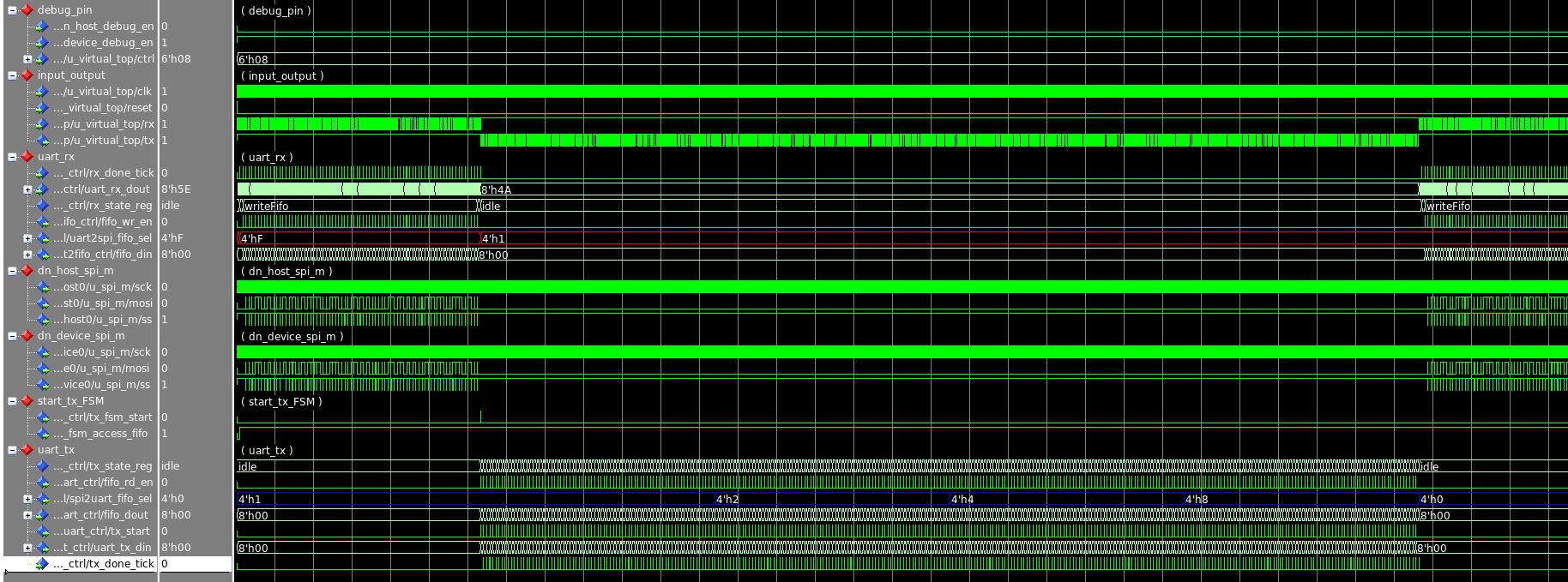
****

Figure write and read multiple bytes to FPGA by using broadcast

1. **Code Coverage**

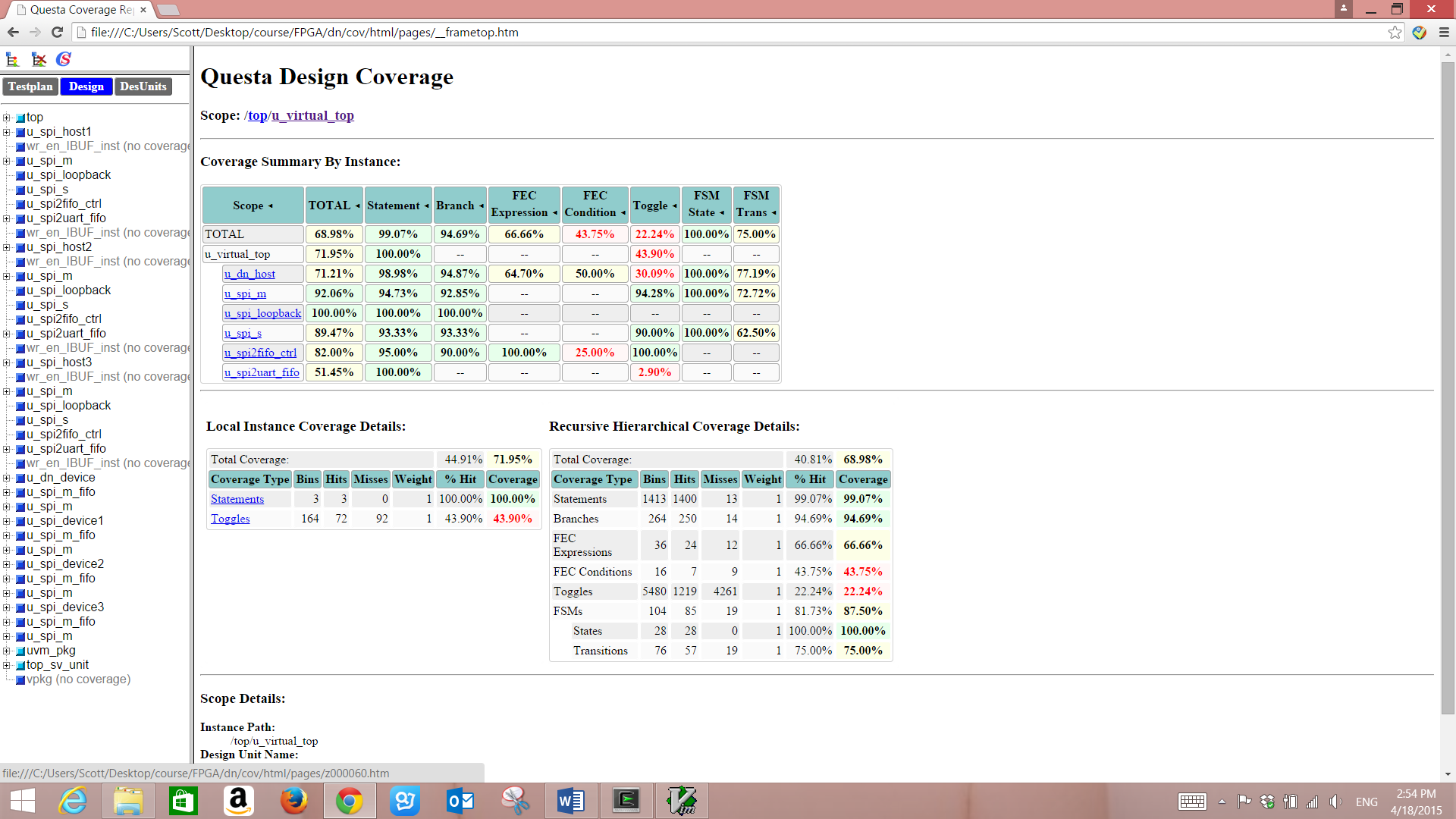


Figure code coverage

1. **CONCLUSIONS and FUTURE WORK**

In the model of distributed network based on FPGAs, the data is delivered more efficiently by using FPGA parallel computing ability. The performance of our distributed network is really good and fast in delivering the data to other FPGAs.

Current problem in our distributed network becomes that how to generate enough data in the host which our distributed system can send it to remote uses in parallel to reduce the idle time of our system.

In current design, as the limitation of FPGA development board, we can only use low speed interfaces like UART or SPI. In the future, high speed interface like SATA, etc. should be implemented to accelerate the speed of transmission.

**Appendix**

1. **Structure of the Directory**

* Pin Assignment Files

xdc/dn.xdc

* Design Files

rtl/file.lst

rtl/virtual\_top/virtual\_top.vhd

rtl/dn\_device/dn\_device.vhd

rtl/dn\_device/spi\_device.vhd

rtl/dn\_device/spi\_m\_fifo.vhd

rtl/dn\_device/spi\_s\_fifo.vhd

rtl/dn\_host/baud\_gen.vhd

rtl/dn\_host/dn\_host.vhd

rtl/dn\_host/dn\_reg.vhd

rtl/dn\_host/fifo2uart\_ctrl.vhd

rtl/dn\_host/flag\_buff.vhd

rtl/dn\_host/spi2uart\_fifo.vhd

rtl/dn\_host/spi\_host.vhd

rtl/dn\_host/uart.vhd

rtl/dn\_host/uart2fifo\_ctrl.vhd

rtl/dn\_host/uart2spi\_fifo.vhd

rtl/dn\_host/uart\_loopback.vhd

rtl/dn\_host/uart\_rx.vhd

rtl/dn\_host/uart\_sys.vhd

rtl/dn\_host/uart\_tx.vhd

rtl/common/fifo\_loopback.vhd

rtl/common/spi2fifo\_ctrl.vhd

rtl/common/spi\_clk\_gen.vhd

rtl/common/spi\_loopback.vhd

rtl/common/spi\_m.vhd

rtl/common/spi\_s.vhd

rtl/fifo\_model/spi2uart\_fifo\_sim.vhd

rtl/fifo\_model/uart2spi\_fifo\_sim.vhd

* TestBench Files

env/top.sv

env/env.sv

env/system\_virtual\_sequencer.sv

env/uart\_env/uart\_env.sv

env/uart\_env/uart\_scoreboard.sv

env/uart\_env/uart\_agent.sv

env/uart\_env/uart\_driver.sv

env/uart\_env/uart\_monitor.sv

env/uart\_env/uart\_sequencer.sv

env/uart\_env/uart\_transaction.sv

env/uart\_env/uart\_interface.svi

* Test Files

test/test.lst

test/test.sv

test/sequence/uart\_sequence/uart\_sequence.sv

test/sequence/uart\_sequence/uart\_base\_sequence.sv

* Scripts

script/run\_regression.pl

script/Makefile

script/do.tcl

* Timing Files

timing/virtual\_top\_timing\_summary\_routed.rpt

* Schematic Files

schematic/dn\_host\_and\_dn\_device.png

schematic/dn\_device.png

schematic/dn\_host.png

* Waveform Files

wave/reg\_write\_read.png

wave/data\_write\_read\_multiple\_bytes\_unicast.png

wave/data\_write\_read\_multiple\_bytes\_broadcast.png

wave/data\_write\_read\_one\_byte\_unicast.png

wave/data\_write\_read\_one\_byte\_broadcast.png

wave/uart\_rx\_tx\_loopback.png

* Simulation Logs

log/uart\_hard\_spi\_loopback\_test.run.log

log/uart\_hard\_spi\_loopback\_test.sv.compile.log

log/uart\_hard\_spi\_loopback\_test.vhdl.compile.log

log/uart\_hard\_uart2spi\_fifo\_loopback\_test.run.log

log/uart\_hard\_uart2spi\_fifo\_loopback\_test.sv.compile.log

log/uart\_hard\_uart2spi\_fifo\_loopback\_test.vhdl.compile.log

log/uart\_multiple\_write\_fifo\_broadcast\_test.run.log

log/uart\_multiple\_write\_fifo\_broadcast\_test.sv.compile.log

log/uart\_multiple\_write\_fifo\_broadcast\_test.vhdl.compile.log

log/uart\_multiple\_write\_fifo\_burst\_broadcast\_test.run.log

log/uart\_multiple\_write\_fifo\_burst\_broadcast\_test.sv.compile.log

log/uart\_multiple\_write\_fifo\_burst\_broadcast\_test.vhdl.compile.log

log/uart\_multiple\_write\_fifo\_burst\_randcast\_test.run.log

log/uart\_multiple\_write\_fifo\_burst\_randcast\_test.sv.compile.log

log/uart\_multiple\_write\_fifo\_burst\_randcast\_test.vhdl.compile.log

log/uart\_multiple\_write\_fifo\_burst\_unicast\_test.run.log

log/uart\_multiple\_write\_fifo\_burst\_unicast\_test.sv.compile.log

log/uart\_multiple\_write\_fifo\_burst\_unicast\_test.vhdl.compile.log

log/uart\_multiple\_write\_fifo\_randcast\_test.run.log

log/uart\_multiple\_write\_fifo\_randcast\_test.sv.compile.log

log/uart\_multiple\_write\_fifo\_randcast\_test.vhdl.compile.log

log/uart\_multiple\_write\_fifo\_unicast\_test.run.log

log/uart\_multiple\_write\_fifo\_unicast\_test.sv.compile.log

log/uart\_multiple\_write\_fifo\_unicast\_test.vhdl.compile.log

log/uart\_multiple\_write\_reg\_test.run.log

log/uart\_multiple\_write\_reg\_test.sv.compile.log

log/uart\_multiple\_write\_reg\_test.vhdl.compile.log

log/uart\_rx\_tx\_loopback\_test.run.log

log/uart\_rx\_tx\_loopback\_test.sv.compile.log

log/uart\_rx\_tx\_loopback\_test.vhdl.compile.log

log/uart\_soft\_spi\_loopback\_test.run.log

log/uart\_soft\_spi\_loopback\_test.sv.compile.log

log/uart\_soft\_spi\_loopback\_test.vhdl.compile.log

log/uart\_soft\_uart2spi\_fifo\_loopback\_test.run.log

log/uart\_soft\_uart2spi\_fifo\_loopback\_test.sv.compile.log

log/uart\_soft\_uart2spi\_fifo\_loopback\_test.vhdl.compile.log

* Code coverage

cov/html

cov/cov\_all.ucdb

cov/uart\_hard\_spi\_loopback\_test.ucdb

cov/uart\_hard\_uart2spi\_fifo\_loopback\_test.ucdb

cov/uart\_multiple\_write\_fifo\_broadcast\_test.ucdb

cov/uart\_multiple\_write\_fifo\_burst\_broadcast\_test.ucdb

cov/uart\_multiple\_write\_fifo\_burst\_randcast\_test.ucdb

cov/uart\_multiple\_write\_fifo\_burst\_unicast\_test.ucdb

cov/uart\_multiple\_write\_fifo\_randcast\_test.ucdb

cov/uart\_multiple\_write\_fifo\_unicast\_test.ucdb

cov/uart\_multiple\_write\_reg\_test.ucdb

cov/uart\_rx\_tx\_loopback\_test.ucdb

cov/uart\_soft\_spi\_loopback\_test.ucdb

cov/uart\_soft\_uart2spi\_fifo\_loopback\_test.ucdb

**References**

1. Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc’Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang and Andrew Y. Ng: Large Scale Distributed Deep Networks.
2. Erik Nygren, Ramesh K. Sitaraman and Jennifer Sun: The Akamai Network: A Platform for High-Performance Internet Applications.
3. Chee Shin Yeo, Rajkumar Buyya, Hossein Pourreza, Rasit Eskicioglu, Peter Graham and Frank Sommers: Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers.
4. David P. Anderson, Eric Korpela and Rom Walton: High-Performance Task Distribution for Volunteer Computing.