ELEN 613

Due : 6/15/2019 11:59PM
Final Project

1. Objective

In this assignment, you will be verifying a 10G Ethernet MAC core

Note: This project is intended to be individually. I don't have a problem if you wish to discuss the project with other students; however, your final work must be your own.

2. Functional Specification

You will be given the RTL code for a 10G Ethernet MAC core, and you must write the testbench in SystemVerilog (Layered testbench methodology) or SystemVerilog UVM. Your testbench must check for correct functionality (received packet = transmitted packet)

3. Feature Testing

The 10G Ethernet MAC Core Specification has been uploaded to Camino. A System Verilog covergroup has been written for you, and you will need to add addition coverpoints to it.
The covergroup includes the following

- Configuration Register 0
- Interrupt Pending Register ← any kinds of exceptions that happen
    i. RX fragment Error
    ii. RX CRC error
    iii. RX Pause Frame ← start to transmit frame but never completed
    iv. Remote fault
    v. Local fault
    vi. RX Data Fifo Underflow
    vii. RX Data fifo overflow
    viii. TX Data fifo underflow
    ix. TX Data fifo overflow
- Interrupt Mask Register
- Statistics Registers 4 registers
- Packet length between 64 and 1522 Create Cover Point w/ bins of 64, 1522, and something in middle
- Number of packets received/transmitted

4. Interfaces

The 10G Ethernet MAC Core contains 3 different interfaces

- Wishbone ← how to write & read registers
- Packet receive / transmit
- XGMII receive / transmit
  ↑ physical layer (we don't touch this)

You will be provided the following tasks

- Wishbone_Read
- TxPacket
- RxPacket

You will need to modify these tasks so that they read/write through the clocking block of the interface (because rn, they are directly connected to IO)
You are not required to write the tasks that access the XGMII interface.

5. Sample Covergroup

transmit enable
↑
```
covergroup ethernet_cg  @(posedge tb.clk_xgmii_rx) ;
  TX_ENABLE :   coverpoint tb.dut.wishbone_if0.cpureg_config0
  {
    bins  enable = {1} ;
    ignore_bins  unused = {0};
  }
//   INT_PENDING :   coverpoint tb.dut.wishbone_if0.cpureg_int_pending[8:0] ;
  STATUS_CRC_ERROR :   coverpoint tb.dut.wishbone_if0.status_crc_error
  {
    bins  enable = {1} ;
    ignore_bins  unused = {0};
  }
  STATUS_FRAGMENT_ERROR :   coverpoint tb.dut.wishbone_if0.status_fragment_error
  {
    bins  enable = {1} ;
    ignore_bins  unused = {0};
  }
  STATUS_TXDFIFO_OVFLOW :   coverpoint tb.dut.wishbone_if0.status_txdfifo_ovflow
  {
    bins  enable = {1} ;
    ignore_bins  unused = {0};
  }
  STATUS_TXDFIFO_UDFLOW :   coverpoint tb.dut.wishbone_if0.status_txdfifo_udflow
  {
    bins  enable = {1} ;
    ignore_bins  unused = {0};
```

```
}
STATUS_RXDFIFO_OVFLOW :   coverpoint tb.dut.wishbone_if0.status_rxdfifo_ovflow
{
  bins  enable = {1} ;
  ignore_bins  unused = {0};
}
STATUS_RXDFIFO_UDFLOW :   coverpoint tb.dut.wishbone_if0.status_rxdfifo_udflow
{
  bins  enable = {1} ;
  ignore_bins  unused = {0};
}
STATUS_PAUSE_FRAME_RX :   coverpoint tb.dut.wishbone_if0.status_pause_frame_rx
{
  bins  enable = {1} ;
  ignore_bins  unused = {0};
}
STATUS_LOCAL_FAULT :   coverpoint tb.dut.wishbone_if0.status_local_fault
{
  bins  enable = {1} ;
  ignore_bins  unused = {0};
}
STATUS_REMOTE_FAULT :   coverpoint tb.dut.wishbone_if0.status_remote_fault
{
  bins  enable = {1} ;
  ignore_bins  unused = {0};
}

STATUS_LENGTH_ERROR :   coverpoint tb.dut.wishbone_if0.status_lenght_error
{
  bins  enable = {1} ;
  ignore_bins  unused = {0};
}

INT_MASK :    coverpoint tb.dut.wishbone_if0.cpureg_int_mask[8:0] ;
TX_OCTETS :   coverpoint tb.dut.stats0.stats_tx_octets[31:0]
{
  bins  ten = { [1:10] } ;
  bins  twenty = { [11:20] } ;
  bins  thirty = { [21:30] } ;
  bins  forty = { [31:40] } ;
  bins  fifty = { [41:50] } ;
  bins  hundred = {[51:100]} ;
  bins  thousand = {[101:1000]} ;
  bins  tenthousand = {[1001:10000]} ;
  bins  hundredthousand = {[100001:1000000]} ;
  bins  million = {[1000000:9999999]} ;
  bins  huge = {[10000000:100000000]} ;
}
```

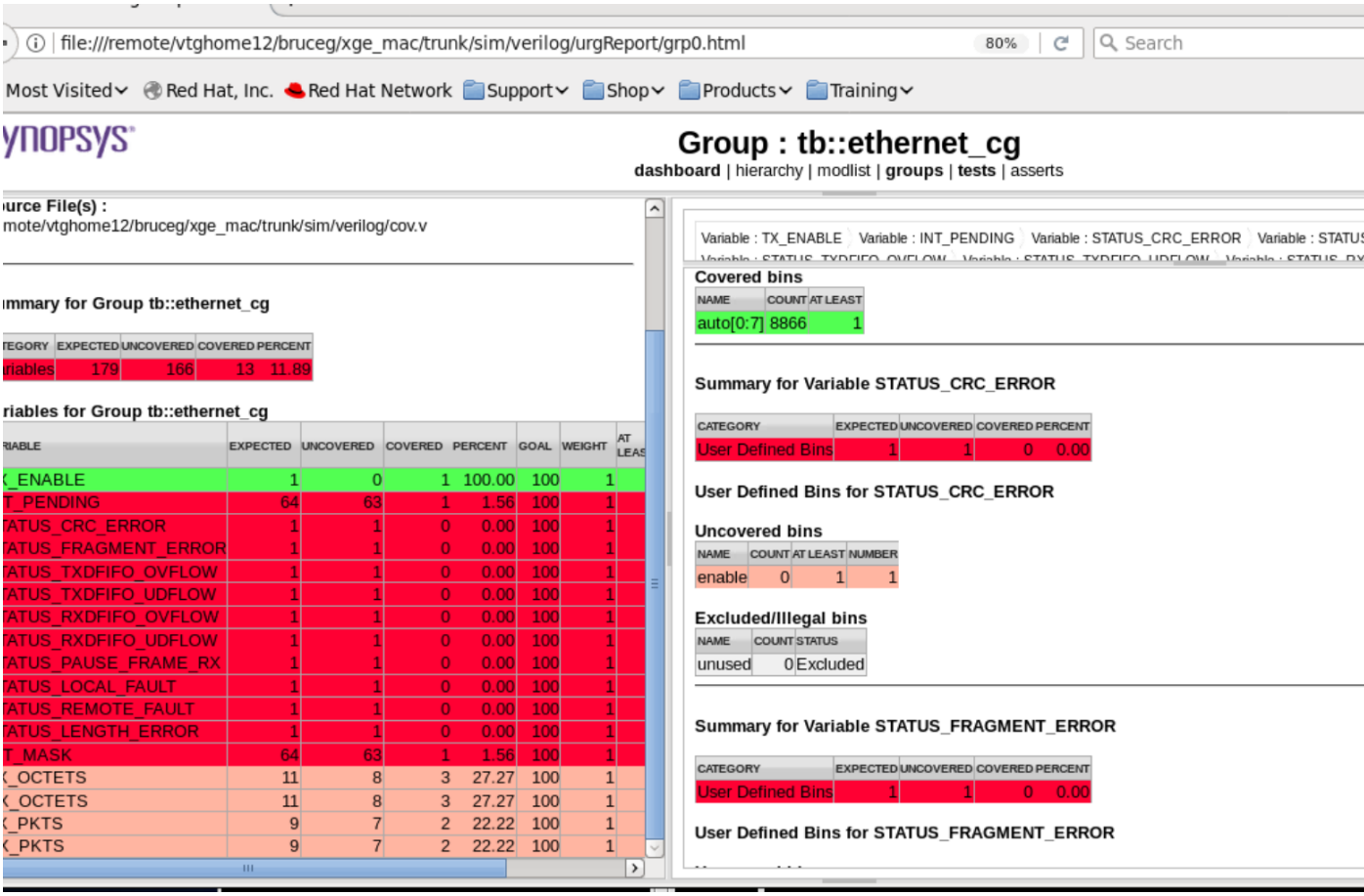
```
RX_OCTETS :   coverpoint tb.dut.stats0.stats_rx_octets[31:0]
{
   bins  ten = { [1:10] } ;
   bins  twenty = { [11:20] } ;
   bins  thirty = { [21:30] } ;
   bins  forty = { [31:40] } ;
   bins  fifty = { [41:50] } ;
   bins  hundred = {[51:100]} ;
   bins  thousand = {[101:1000]} ;
   bins  tenthousand = {[1001:10000]} ;
   bins  hundredthousand = {[100001:1000000]} ;
   bins  million = {[1000000:9999999]} ;
   bins  huge = {[10000000:100000000]} ;
}
TX_PKTS :     coverpoint tb.dut.stats0.stats_tx_pkts[31:0]
{
   bins  ten = { [1:10] } ;
   bins  twenty = { [11:20] } ;
   bins  thirty = { [21:30] } ;
   bins  forty = { [31:40] } ;
   bins  fifty = { [41:50] } ;
   bins  hundred = {[51:100]} ;
   bins  thousand = {[101:1000]} ;
   bins  tenthousand = {[1001:10000]} ;
   bins  huge = {[10001:1000000]} ;
}
```

Receive Packets →

```
RX_PKTS :     coverpoint tb.dut.stats0.stats_rx_pkts[31:0]
{
   bins  ten = { [1:10] } ;
   bins  twenty = { [11:20] } ;
   bins  thirty = { [21:30] } ;
   bins  forty = { [31:40] } ;
   bins  fifty = { [41:50] } ;
   bins  hundred = {[51:100]} ;
   bins  thousand = {[101:1000]} ;
   bins  tenthousand = {[1001:10000]} ;
   bins  huge = {[10001:1000000]} ;
}

endgroup
```

Sample Functional Coverage Report



6. Grading

The project is worth 65% of your final grade, and is broken into several parts (each with it's own due date)

make it work w/ interfaces

5/22 : Basic infrastructure : interface/modport/clocking block/ top -level / tasks for wishbone read/write 10-points
5/27 : Covergroup model, which should include all features you are testing with their definitions : 5 points
5/27 : data class for randomization + driving task for ethernet frame + task for monitoring received ethernet frame : 10 points
6/15 : final project due, should be able to show > 95% coverage : 40 points

7. Extra Credit

Just like the UART assignment, there are several "Bugs" present in the Ethernet core, and 1-point extra credit will be given for every "Bug" found (if you find one, email me and I will give you a bug-fix-number)

/home/bgreene/ELEN613/Ethernet
tb.v → recode this
eth.v DUT
defines.v

random

directed

Total

cover

$\cup$ {Random, Directed}

11% 12% 13% 20%

Cannot add percentages