# CSCE 616 - Hardware Design Verification

# Lab – 4 HTAX Verification Plan

--SAUMIL GOGRI

## Objective – Verification Plan for HTAX Design

- Overview of HTAX Design
- TX/RX Interface
- About Virtual Channel
- Pillars of Verification
- Stimulus
- Coverage
- Assertion

# Overview of HTAX Design

The HTAX Design used in our project consists of 4 bi-directional ports as shown in figure.

Crossbar can be thought like a switch matrix

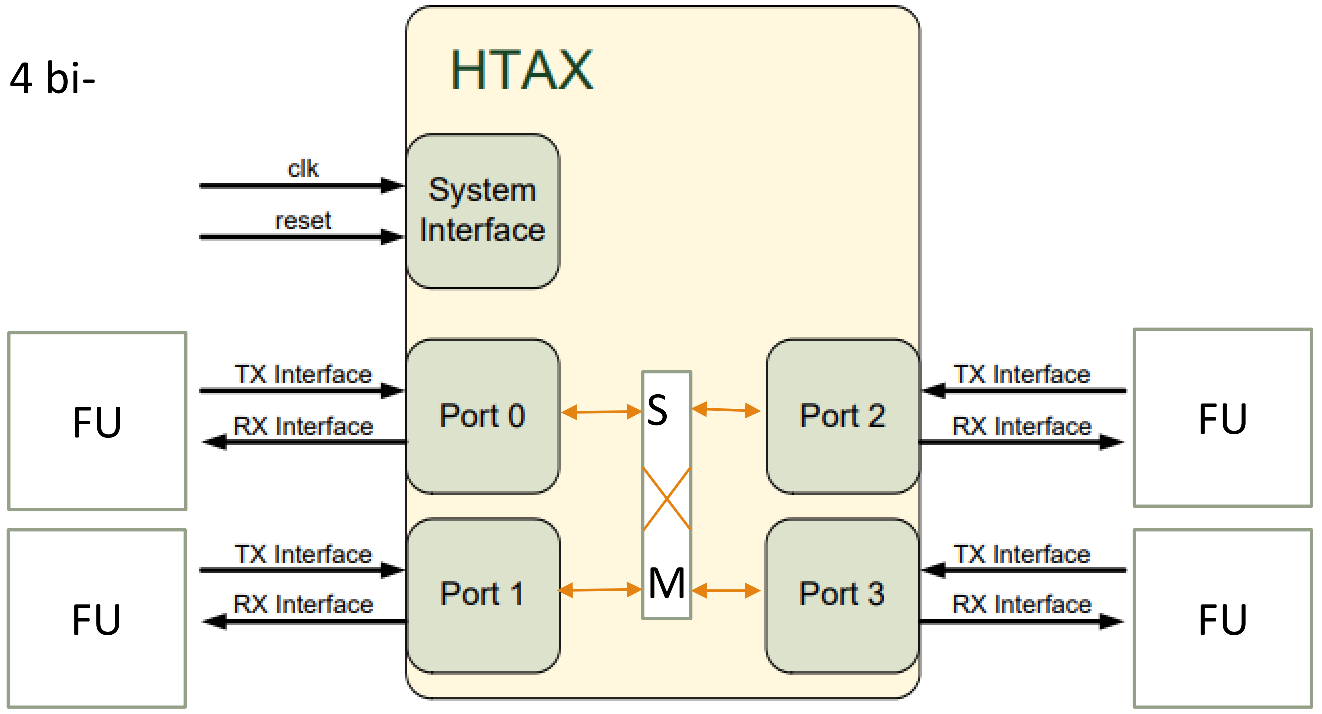


*Figure 1:HTAX block diagram*

# TX/RX Interface -- Protocol

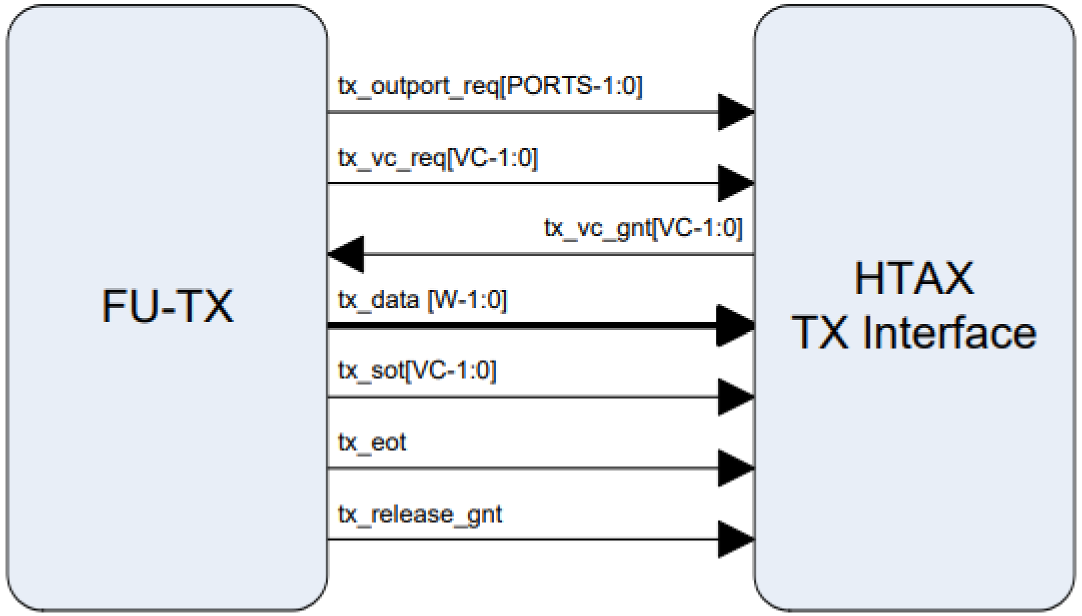


*Figure 2:HTAX TX Interface*

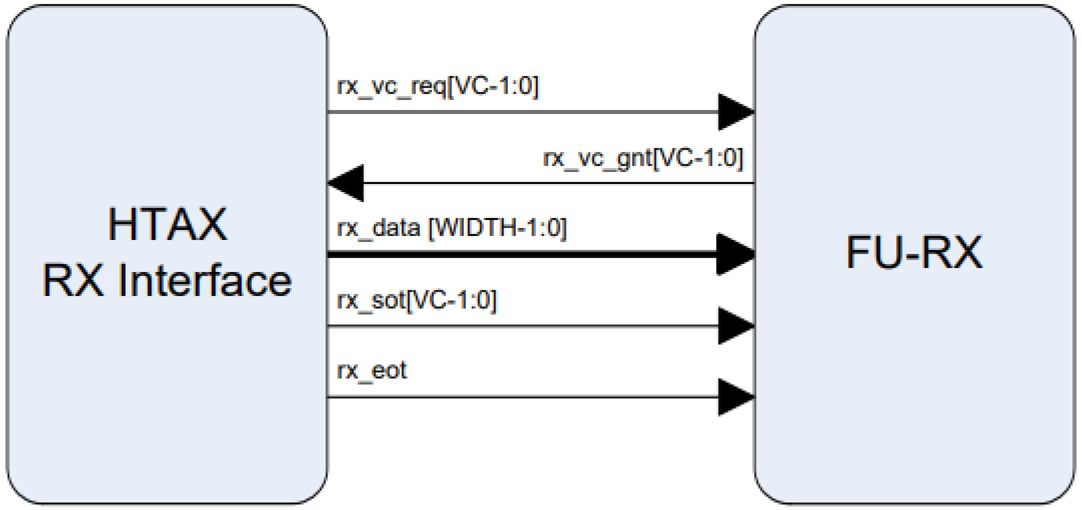
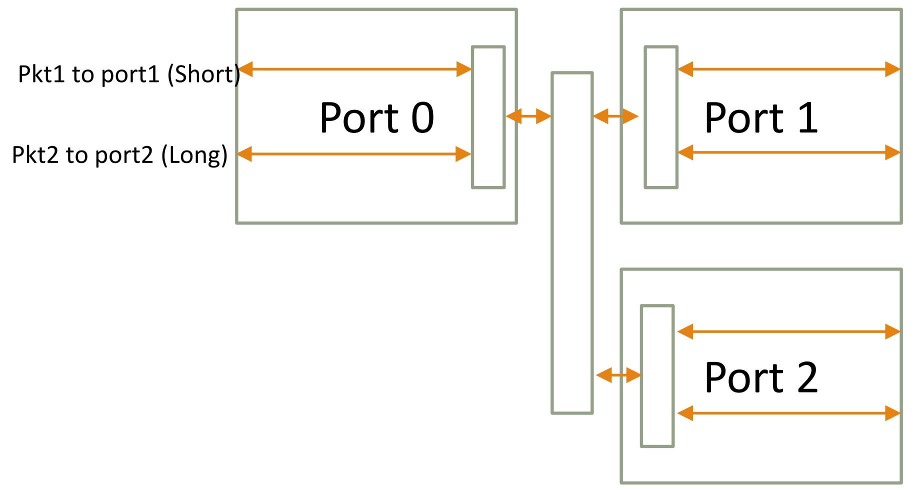


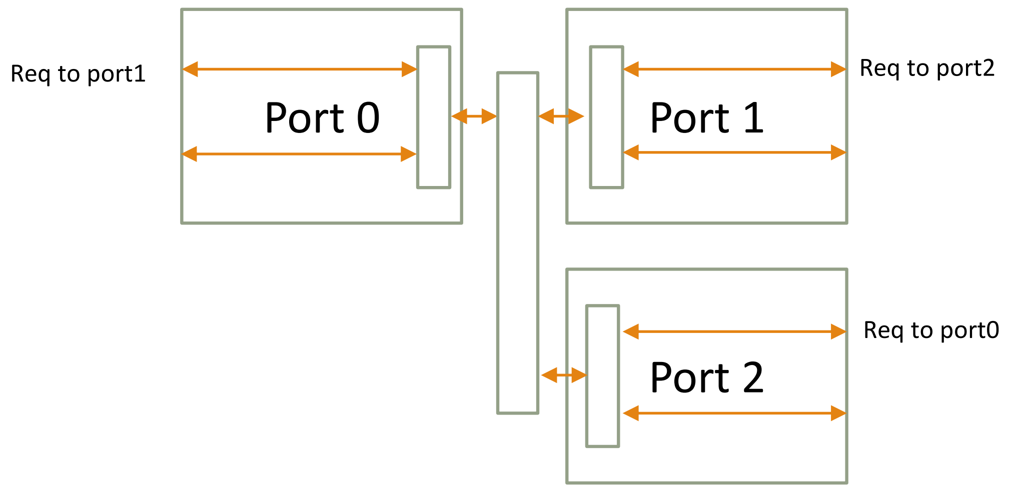
*Figure 6:HTAX RX Interface*

# About Virtual Channel

“The HTAX supports an arbitrary number of virtual channels for Quality of Service (QoS) and deadlock avoidance.” HTAX Specification – How?

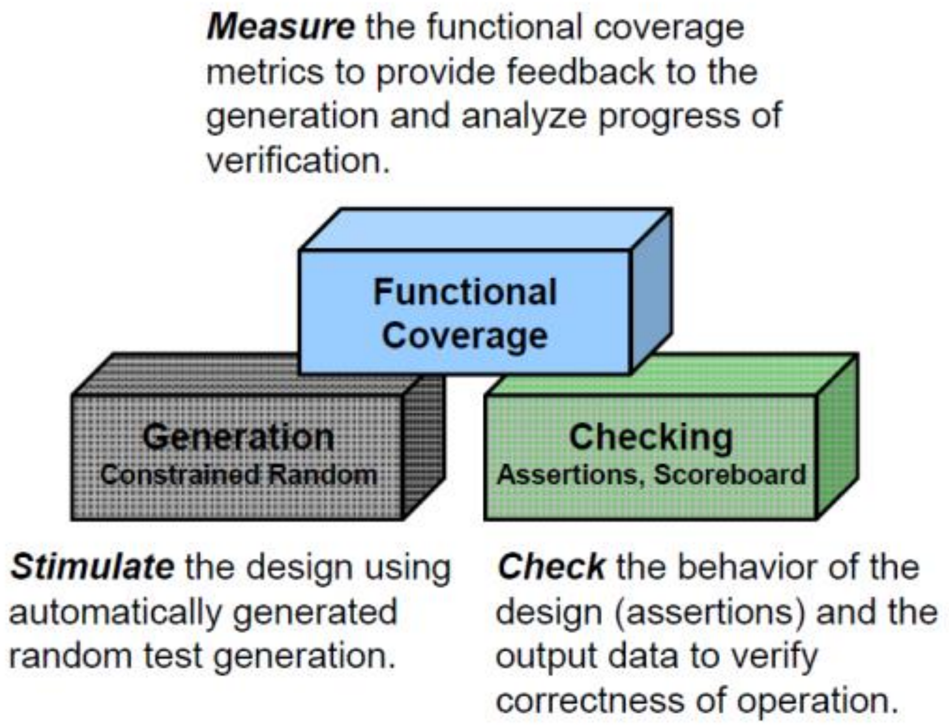


Increased Throughput



Deadlock Avoidance

## Three Pillars of Verification


***Measure*** the functional coverage metrics to provide feedback to the generation and analyze progress of verification.
Functional Coverage
Generation
Constrained Random
Checking
Assertions, Scoreboard
***Stimulate*** the design using automatically generated random test generation.
***Check*** the behavior of the design (assertions) and the output data to verify correctness of operation.

The verification plan which we'll draft will be based on 3 pillars

## Stimulus

"Test, Sequence, Transaction/Packet"

**For simple-mem/cache-mem design**

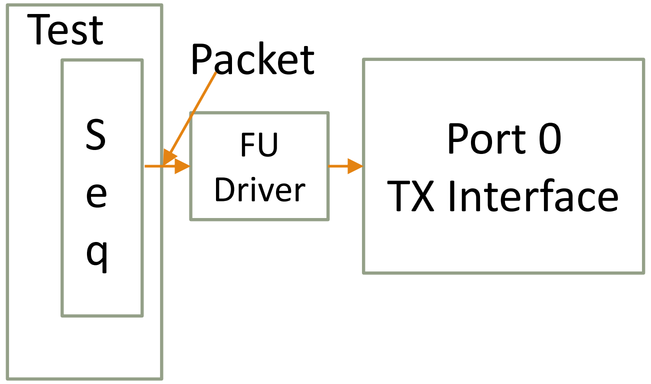
- The addr+data for write-req and addr for read-req constitute our stimulus
- Directed test (write i into address i) vs random test (write random data into random address)

**For HTAX Design**

- Familiar with our transaction/packet class. (Think of what different stimulus/test you can drive)
- E.g: 1. Test driven by fixed port fixed destination address (packets from port i to port j only)

2. Short packet test (length of data would be between (3-10)

3. Random Test (Any port to any port with random delay and random data length)



## Coverage

“How much of design we have verified”

**For simple-mem/cache-mem design**

Write on all address (0-31), Read on all address, Write all possible data (0-255)

**For HTAX Design**

Transmitted from all ports (0,1,2,3)

Received on all the ports (0,1,2,3)

Packets with all possible data-length (3-60)

**We’ll use the cross-product of the above coverage to increase coverage space

## Checking

“Rules/Protocol Checking” – Easiest trick is go through specification and mark lines which you think forms a rule for design

**For simple-mem/cache-mem design**

Write request only when wr_en is 1 and Read request when rd_en is 1. Can’t do both simultaneously.

**For HTAX Design**

“tx_outport_req[PORTS-1:0] - It is a one-hot encoded signal and its width depends on the number of outports connected to the HTAX.”

“tx_vc_req[VC-1:0] The signal width depends on the number of supported virtual channels. tx_vc_req has to be asserted and deasserted simultaneously with the tx_outport_req signal.”

## Thank you