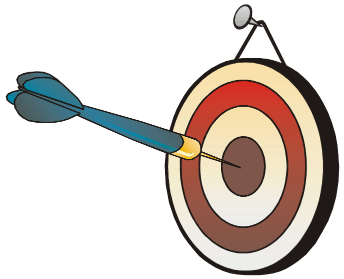
DUT Intro for Specman Elite Training

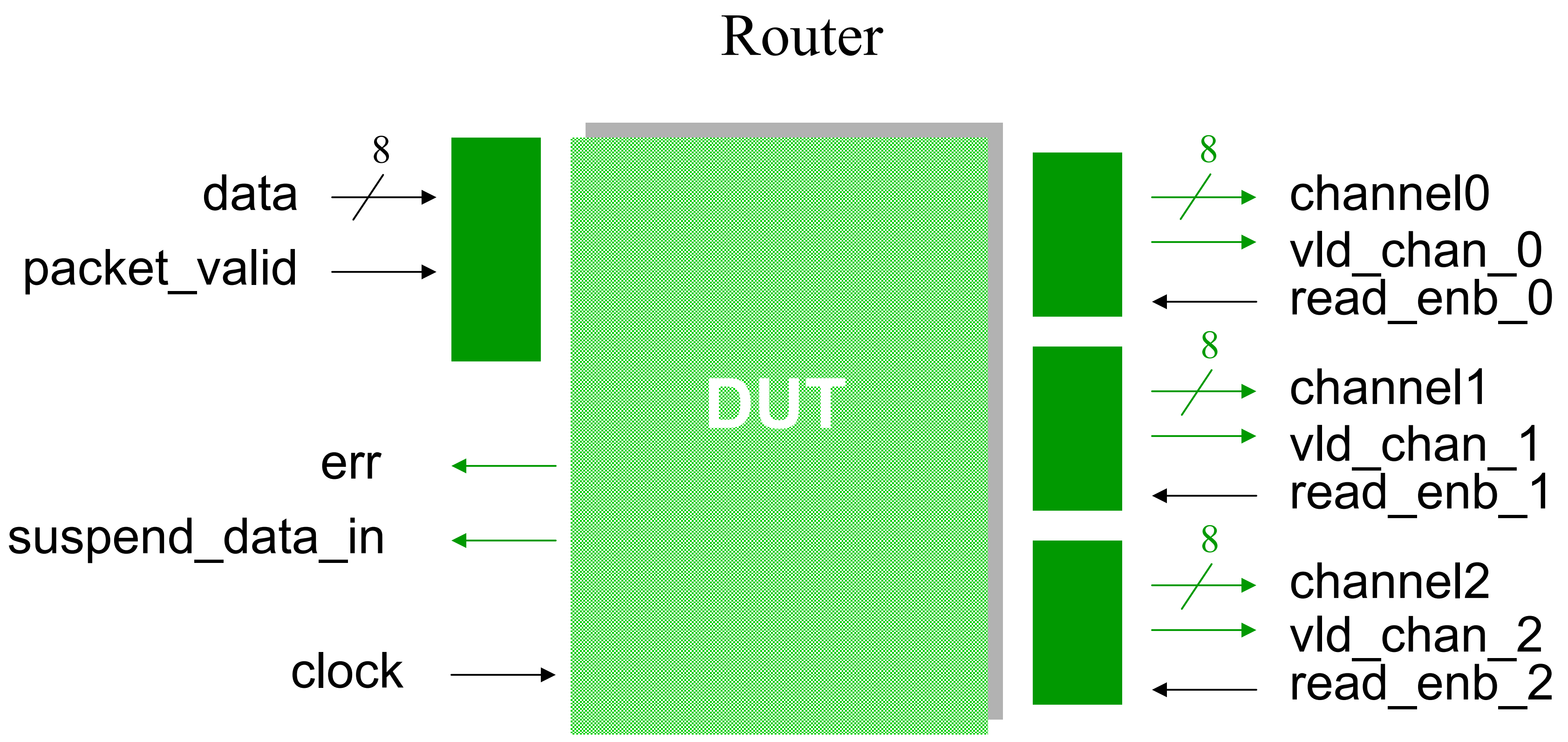
Objectives

- By the end of this intro you will be familiar with the router DUT:
  - Input Protocol
  - Output Protocol



Contents

A bird's view of the DUT

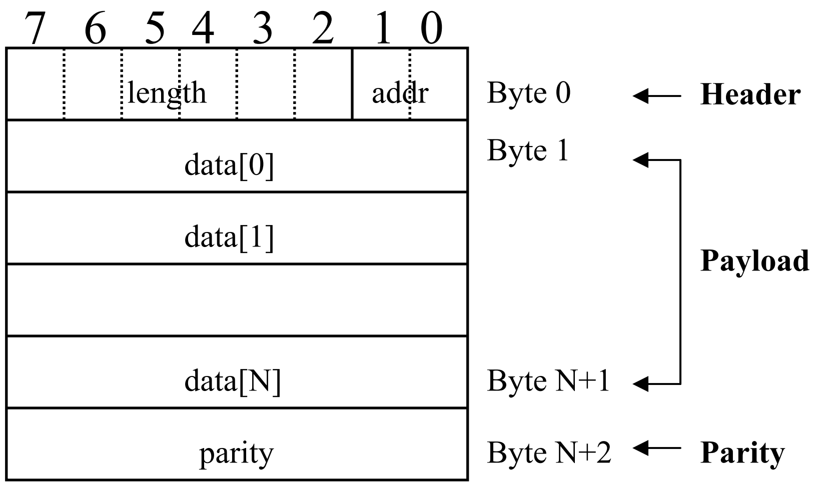


DUT Specification

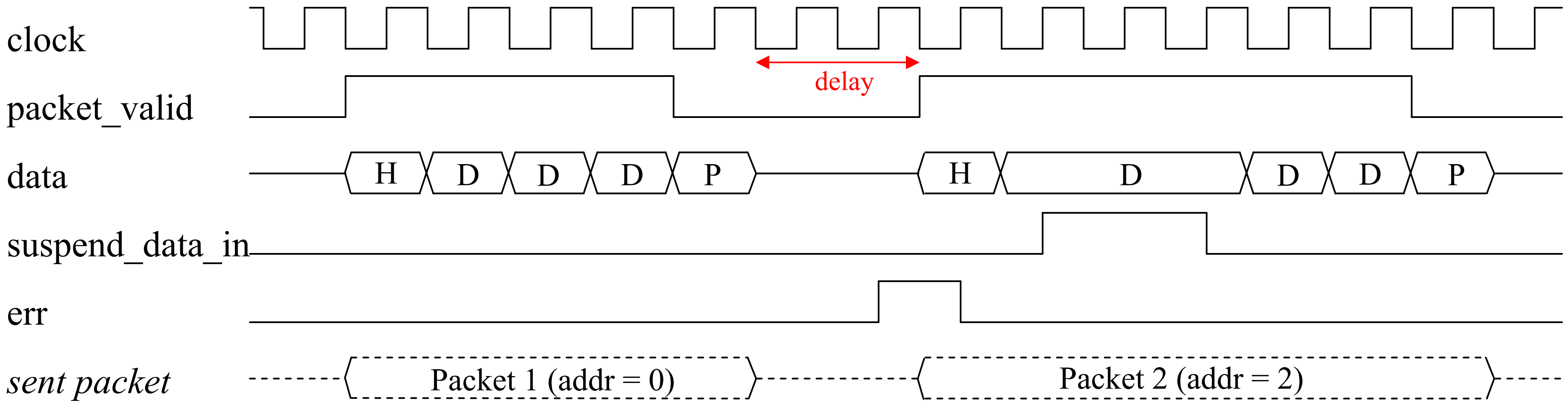
The router accepts data packets on a single input port, data, and routes the packets to one of three output channels: channel0, channel1, or channel2.

A packet is a sequence of bytes with the first byte containing a header, the next variable set of bytes containing data, and the last byte containing parity. The header consists of a 2-bit address field and a 6-bit length field. The address field is used to determine to which output channel the packet should be routed to, with address "3" being illegal. The length field specifies the number of data bytes (payload). A packet can have a minimum data size of 1 byte and a maximum data size of 63 bytes. The parity should be a byte of even, bitwise parity, calculated over the header and data bytes of the packet. The format of a packet is shown in the figure.

Packet Structure



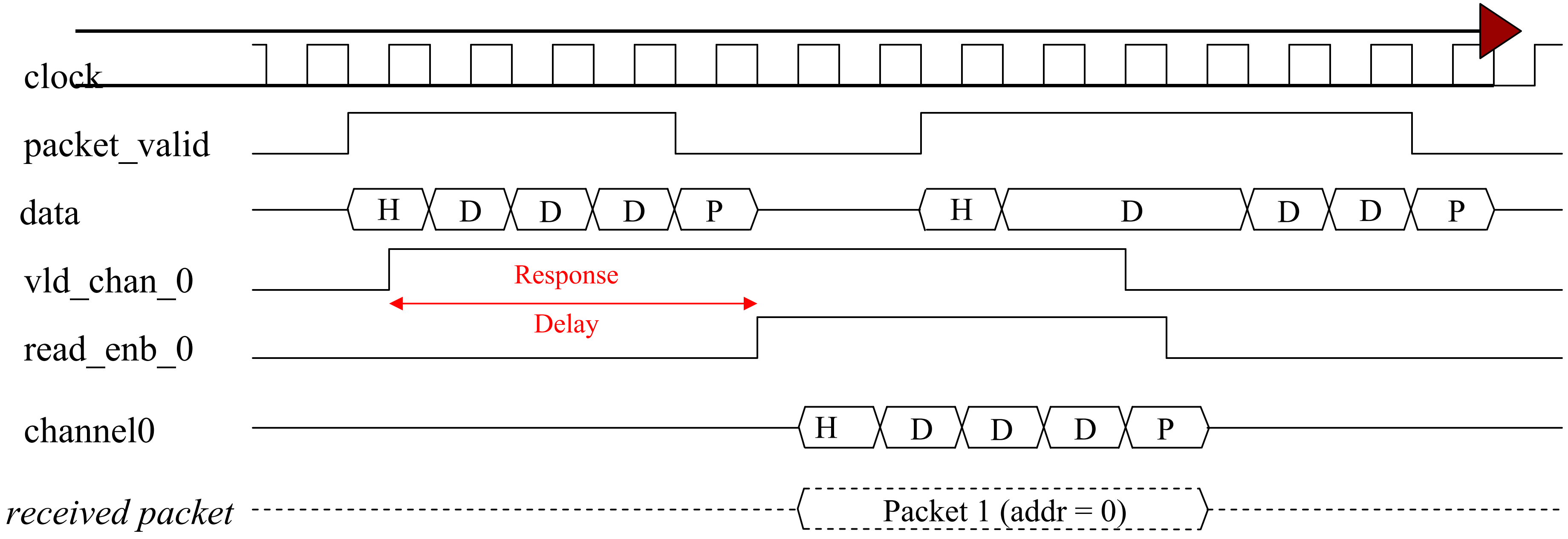
Input Protocol



H = Header, D = Data, P = Parity

All **input** signals are active high and are synchronized to the **falling** edge of the clock. The packet_valid signal has to be asserted on the same clock when the first byte of a packet (the header byte), is driven onto the data bus. Since the header byte contains the address, this tells the router to which output channel the packet should be routed. Each subsequent byte of data should be driven on the data bus with each new falling clock. After the last payload byte has been driven, on the next falling clock, the packet_valid signal must be deasserted (before the parity byte is driven). The packet parity byte should be driven on the next falling clock edge. The input data cannot change while suspend_data_in signal is active (indicating a FIFO overflow). The err signal asserts when a packet with bad parity is detected.

Output Protocol - (Example: port 0)



H = **H**eader, **D** = **D**ata, **P** = **P**arity

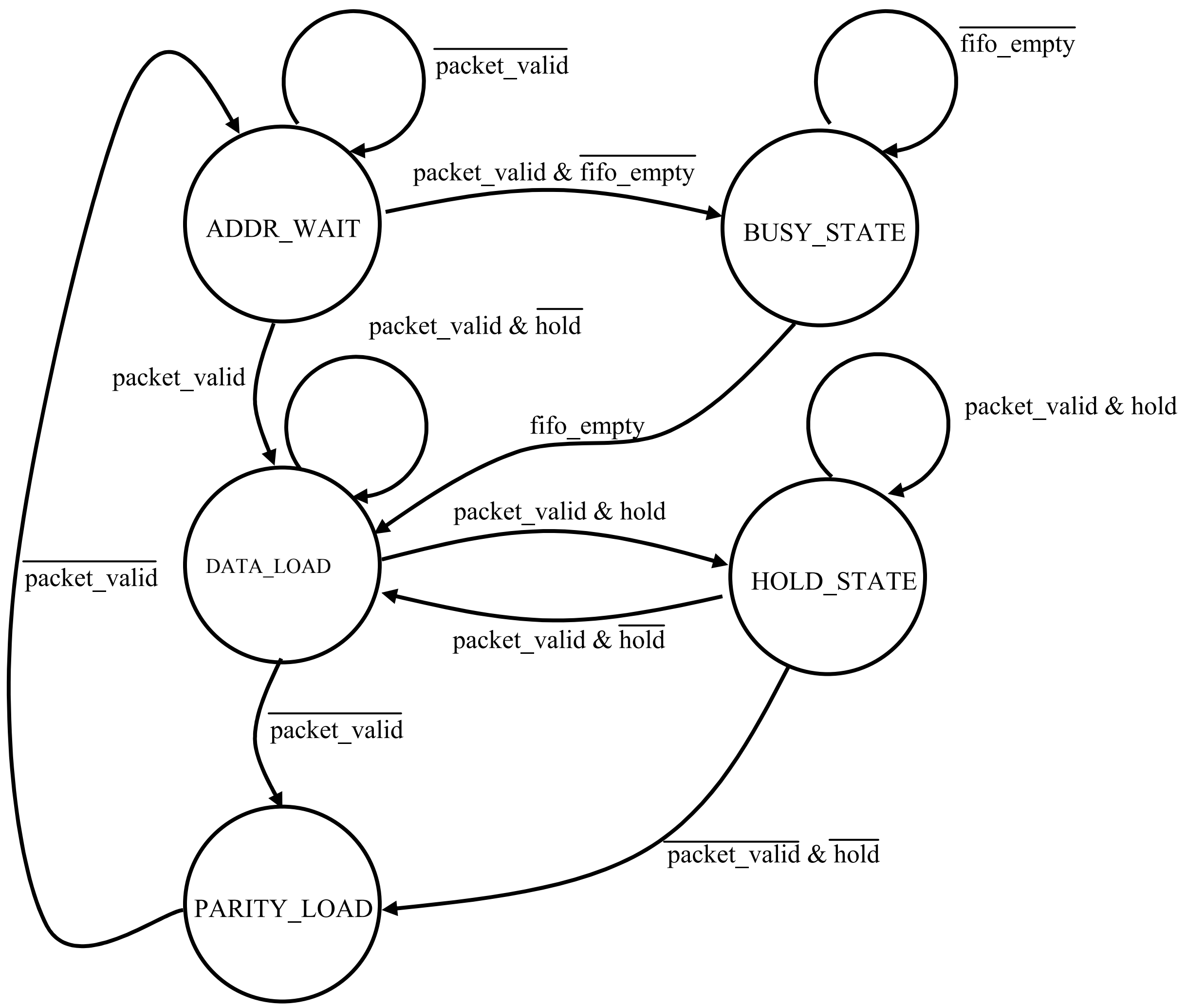
All **output** signals are active high and are synchronized to the **falling** edge of the clock. Each output port is internally buffered by a FIFO of depth 16 and a width of 1 byte. The router asserts the vld_chan_x signal when valid data appears on the channelx output bus. The read_enb_x input signal must then be asserted on the falling clock edge in which data is read from the channelx bus. Read_enb_x must be asserted within 30 cycles. As long the read_enb_x signal remains active, the channelx bus drives a valid packet byte on each rising clock edge.

Signal Name Appendix

Signal Name	Function
top/data	Input bus – 8 bits
top/packet_valid	Input – 1 bit
top/clock	Input – 1 bit
top/read_enb_0	Input – 1 bit
top/read_enb_1	Input – 1 bit
top/read_enb_2	Input – 1 bit
top/err	Output – 1 bit
top/suspend_data_in	Output – 1 bit
top/channel0	Output – 8 bits
top/vld_chan_0	Output – 1 bit
top/channel1	Output – 8 bits
top/vld_chan_1	Output – 1 bit
top/channel2	Output – 8 bits
top/vld_chan_2	Output – 1 bit

Signal Name	Function
top/router1/in_port/state	State Machine – enum type
top/router1/queue_0/full	Ch 0 FIFO Full flag – bit
top/router1/queue_1/full	Ch 1 FIFO Full flag – bit,
top/router1/queue_2/full	Ch 2 FIFO Full flag – bit
top/router1/queue_0/read_ptr	Ch 0 read addr pointer – 4 bits
top/router1/queue_1/read_ptr	Ch 1 read addr pointer – 4 bits
top/router1/queue_2/read_ptr	Ch 2 read addr pointer – 4 bits
top/router1/queue_0/write_ptr	Ch 0 read addr pointer – 4 bits
top/router1/queue_1/write_ptr	Ch 1 read addr pointer – 4 bits
top/router1/queue_2/write_ptr	Ch 2 read addr pointer – 4 bits

State Machine


$\overline{\text{packet\_valid}}$
$\overline{\text{fifo\_empty}}$
ADDR_WAIT
packet_valid & $\overline{\text{fifo\_empty}}$
BUSY_STATE
packet_valid & $\overline{\text{hold}}$
packet_valid
fifo_empty
packet_valid & hold
$\overline{\text{packet\_valid}}$
DATA_LOAD
packet_valid & hold
HOLD_STATE
packet_valid & $\overline{\text{hold}}$
$\overline{\text{packet\_valid}}$
$\overline{\text{packet\_valid}}$ & $\overline{\text{hold}}$
PARITY_LOAD

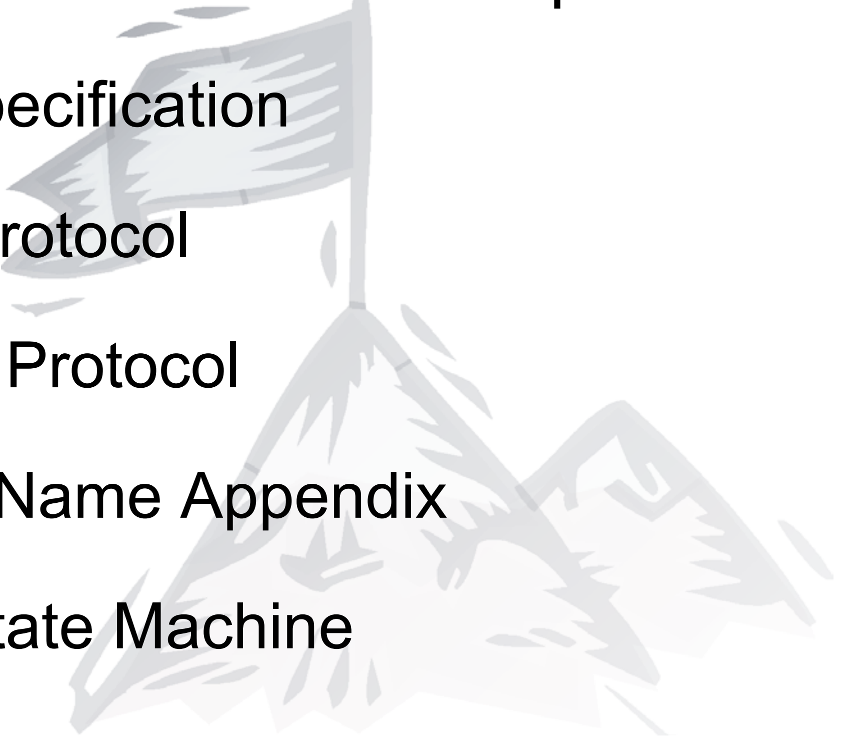
State Machine (cont.)

LEGAL STATE TRANSITIONS

Current State	Next State
ADDR_WAIT = 0x0	ADDR_WAIT, DATA_LOAD, BUSY_STATE
DATA_LOAD = 0x1	DATA_LOAD, PARITY_LOAD, HOLD_STATE
PARITY_LOAD = 0x2	ADDR_WAIT
HOLD_STATE = 0x3	HOLD_STATE, DATA_LOAD, PARITY_LOAD
BUSY_STATE = 0x4	BUSY_STATE, DATA_LOAD

Summary

Now you know the course sample DUT:



- ✓ DUT Specification
- ✓ Input Protocol
- ✓ Output Protocol
- ✓ Signal Name Appendix
- ✓ DUT State Machine