FSM for Router

Decode Address

↳ initial reset state

↳ signal detect_add is asserted, which is used to detect an incoming packet. (~ used to latch first byte as header byte)

Load_First_Data

↳ signal lfd_state is asserted, which is used to Load First data byte to FIFO.

↳ Signal busy is asserted

↳ It changes to state Load_data. unconditionally

Load Data

↳ signal ld_state is asserted, which is used to load payload data to FIFO

↳ signal busy is deasserted

↳ signal write_en_reg is asserted, which is used to write Packet Information (Header + Payload + Parity) to the selected FIFO

↳ state changes to Load_Parity when pkt_valid is low && !fifo_full & to FIFO_full_state when FIFO is full.

Load_Parity

↳ In this state Last Byte is latched (parity Byte).

↳ state changes to Check_Parity_Error unconditionally

↳ signal busy is asserted, so that router doesn't accept new data. & write_en_reg is also asserted. for latching the parity byte to

FIFO full state

↳ signal busy is made high & write_enb_reg is made low & full_state is also asserted

Load After Full

↳ signal laf_state is asserted, which is used to latch the data after FIFO_full_state.

↳ signal busy & write_enb_reg is asserted

↳ If signal parity done is high, then goes to decode address.

↳ else look for low_packet_valid, if high then goes to Load Parity otherwise goes to Load Data state

Wait till Empty

↳ Signal Busy is made High & write_enb_reg is made low

Check Parity Error

↳ Busy is asserted

↳ checks for FIFO full and goes to Decode address & FIFO_full_state accordingly.

↳ signal rst_int_reg is generated, which is used to reset low_packet_valid signal

module router_fsm (input clk, resetn,
packet valid,
input [1:0] data_in,
input fifo_full, fifo_empty 0/1/2
input soft reset 0/1/2, parity done,
low packet valid

output write enb_reg, detect_add,
ld_state, laf_state, lfd_state,
full_state, rst_int_reg, busy);

parameter decode address = 4'b0001
wait till empty = 4'b0010
load first data = 4'b0011
load data = 4'b0100
load parity = 4'b0101
fifo_full_state = 4'b0110
load after full = 4'b0111
check parity_error = 4'b1000;

parameterized all states.

```
reg [3:0] present state, next state;
reg [1:0] temp;
```

to synchronize data_in.

```
always @ (posedge clk)
    !resetn : temp <= 2'b00;
else if  detect add : temp <= datain.
```

```
always @ (posedge clk)
    ! reset : present_state <= decode address
```

on hard reset, back to initial state

```
else if ( soft rst 0 && temp = 2'b00 ||
          soft rst 1 && temp = 2'b01 ||
          soft rst 2 && temp = 2'b10 )

    present_state <= decode address

else

    present_state <= next state;
```

// state machine Logic

case (present state)

decode address

load_first_data

wait till empty.

load data.

fifo full state

load after full

load parity.

check parity error

↓ As per FSM

assign busy = ((present state == load first data

|| == load parity — == fifo full state

= load after full , — == wait till empty.

— == check parity error) ? 1 : 0 ;

assign detect_add = at decode address

d_fd state = at load first data

ld state = at load data.

write_enb_reg = at load data, load after full, load parity.

full state = at fifo full state.
laf state = at load after full
rst_int reg = at check parity error.