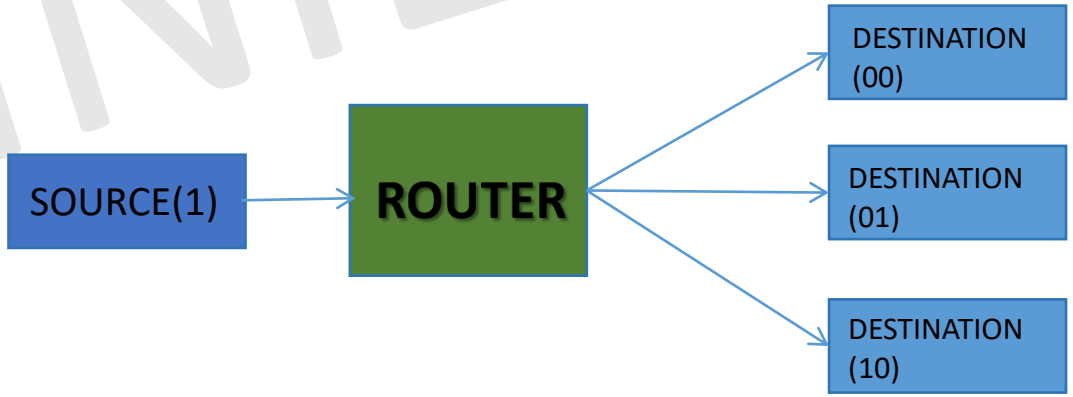
ROUTER 1x3(packet based design)

What is router

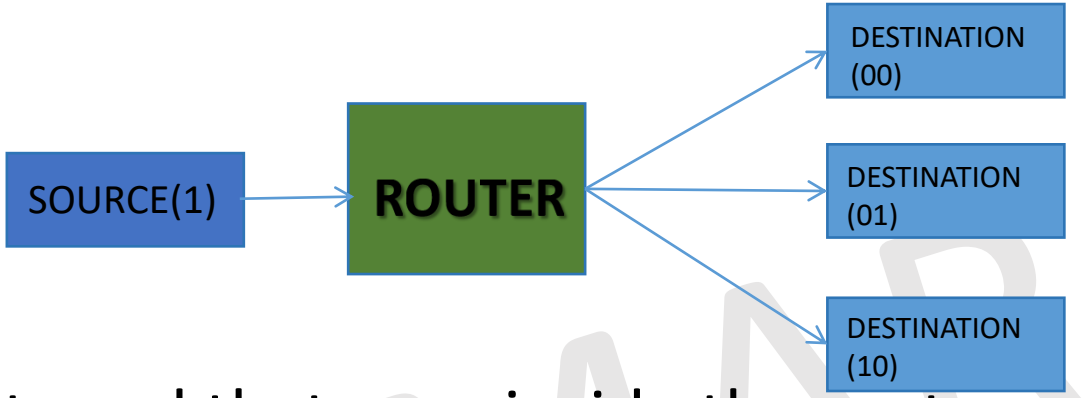
- It’s a device that helps transmit the data from source to destination.
- for detecting devices we have ip addresses.
- Data is transferred in form of packets.
- Generally sources and destinations are LANs of computer.

what is 1x3

- Here 1 is no. of sources.
- Here 3 is no. of destinations.
- Router is between Source and Destination.
- In general Router acts like a bridge.



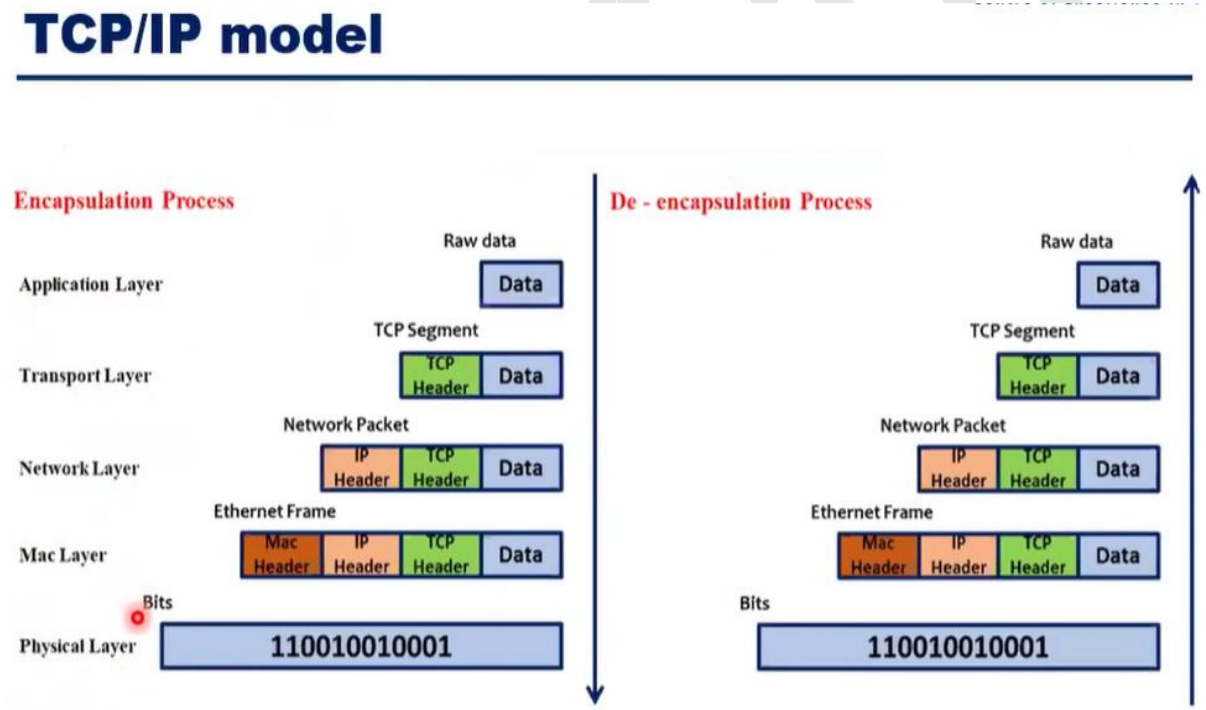
WORKING



- First source generates data in packets and that goes inside the router and stored.(this means router requires memory)
- Then Router reads the packets and decodes how many bits of data is there, Which destination to go depending on it.
- Now destination is sent a signal, Now destination will generate a read signal.
- Now destination reads the data stored in router that it got from source.

TCP/IP protocol

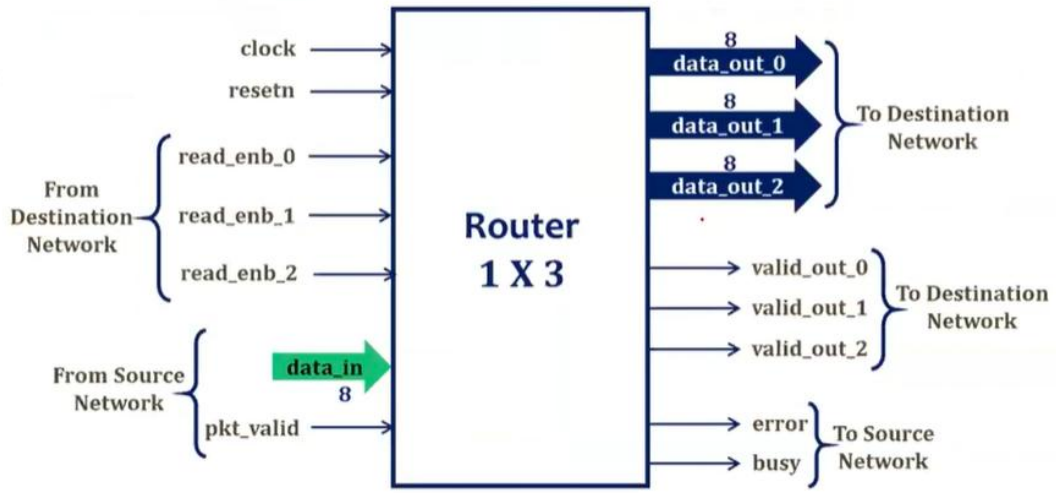
- This protocol is used in building Router.
- Transport layer adds TCP header
- Netwrok layer add IP address.
- MAC layer adds MAC Header.
- PHYSICAL LAYER is ethernet cable
- At destination all the above mentioned things are verified if they match with destination's and data is transmitted.



ROUTER DESIGN

- Clock(posedge) and resetn(active low) are basic pins.
- data_in(8bit) will be from source network, depending on datapacket destination will be selected and transmission is done.
- pkt_valid is set to high to start new packet, pkt_valid is set to low to end the packet.
- read_enb_x is sent from destination to source to indicate destination is ready/free.

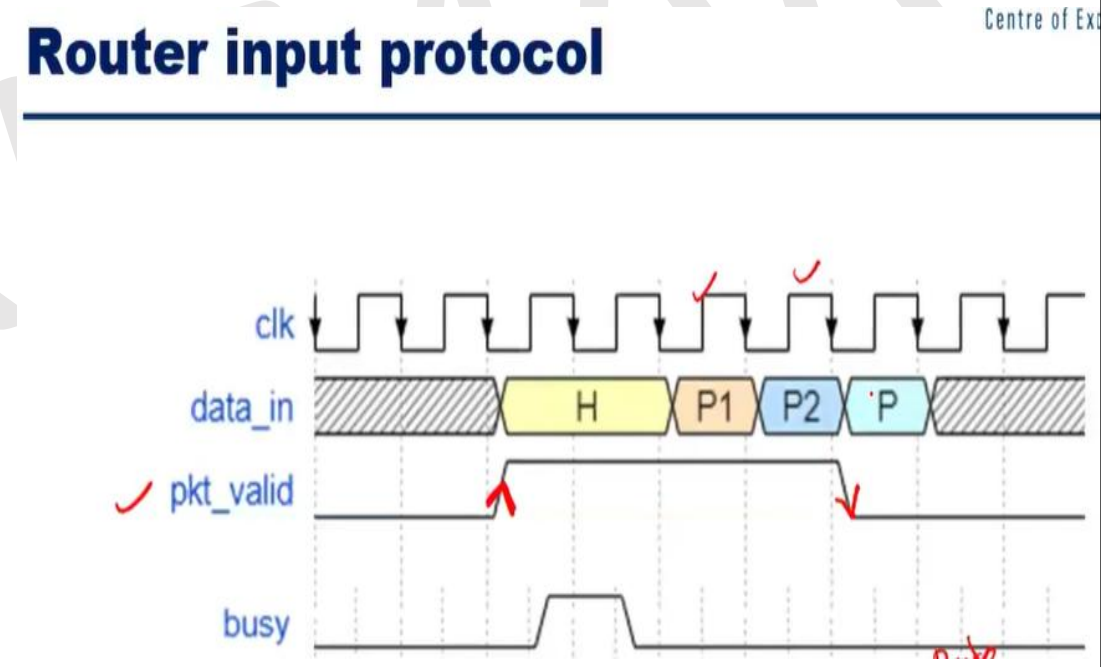
- Top Block



ROUTER DESIGN(Contd..)

- Here pkt_valid will be high till the data_payload (P1,P2) completely transmitted, pkt_valid goes low for parity bit.
- busy is held high till Header is read.
- The valid_out_x is input to destination.
- The valid_out_x is used to indicate that there is a valid info available please read to destrination.
- If valid_out_x is high the destination will generate read_end_x to source.
- error and busy are acknoledgement signals.
- error indicates error in data reception.
- busy indicates if destination is free to connect. After Header busy should go low to accept P1,P2.

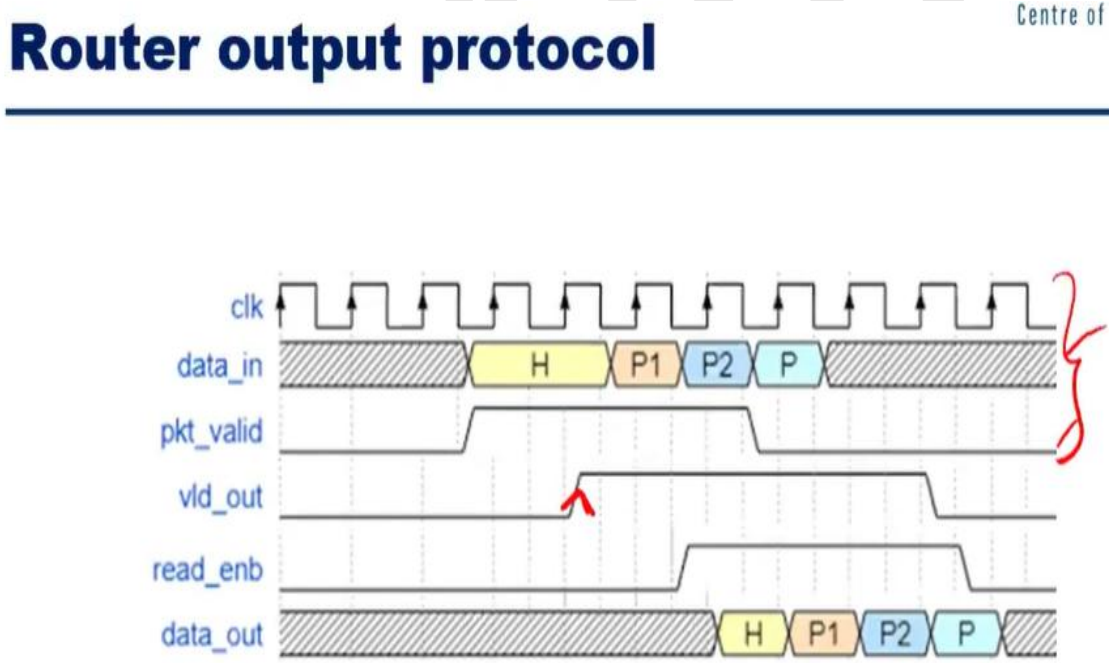
- Top Block



ROUTER DESIGN(Contd..)

- After vld_out goes high, within 30 clk cycles read_enb goes high.
- After read_enb goes high, @1st_posedge data_out will start to destination.

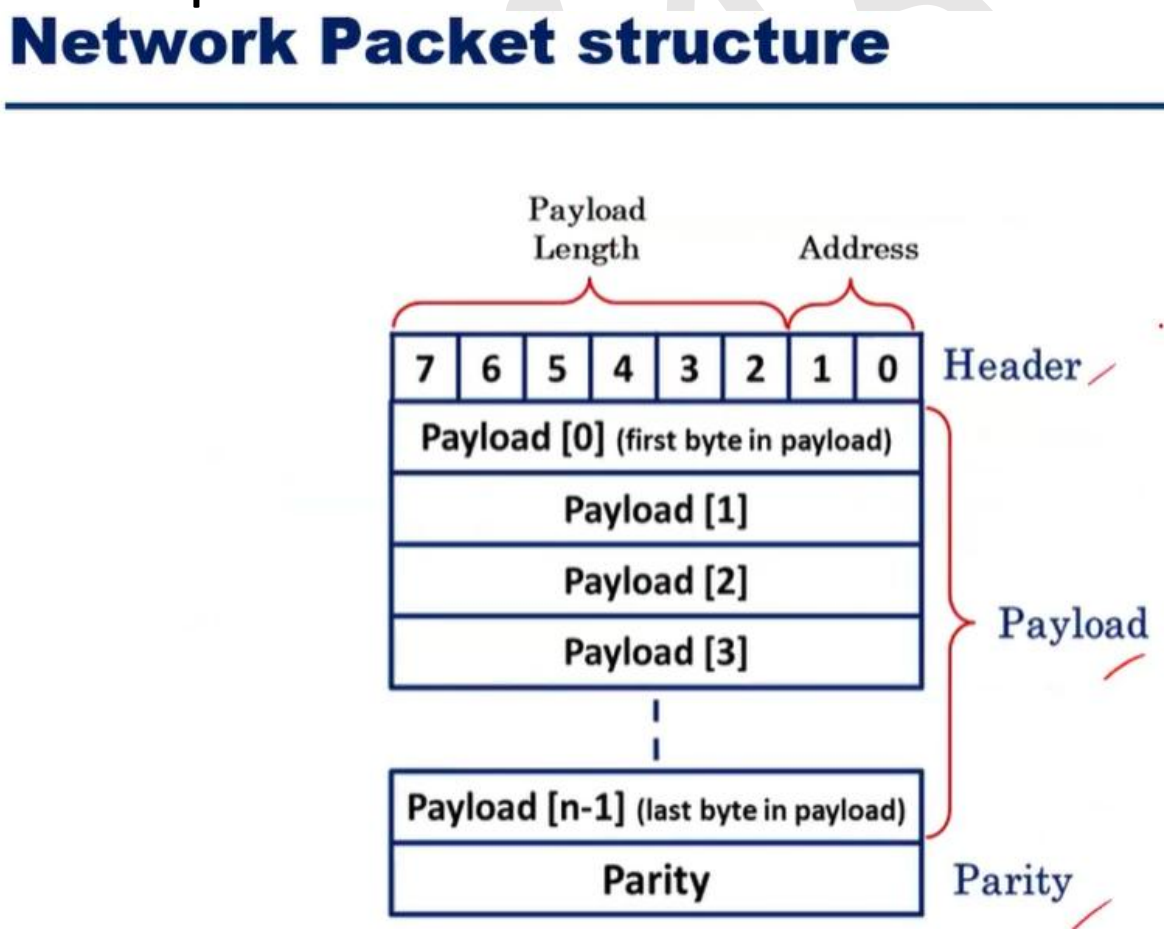
- Top Block



ROUTER DESIGN(Contd..)

- The first important thing is header.
- [1:0]Header is address of destination(00,01,10)
- [7:2]Header is no. of Data_payloads.
- Each Data_payload consists of 1byte(8bits).

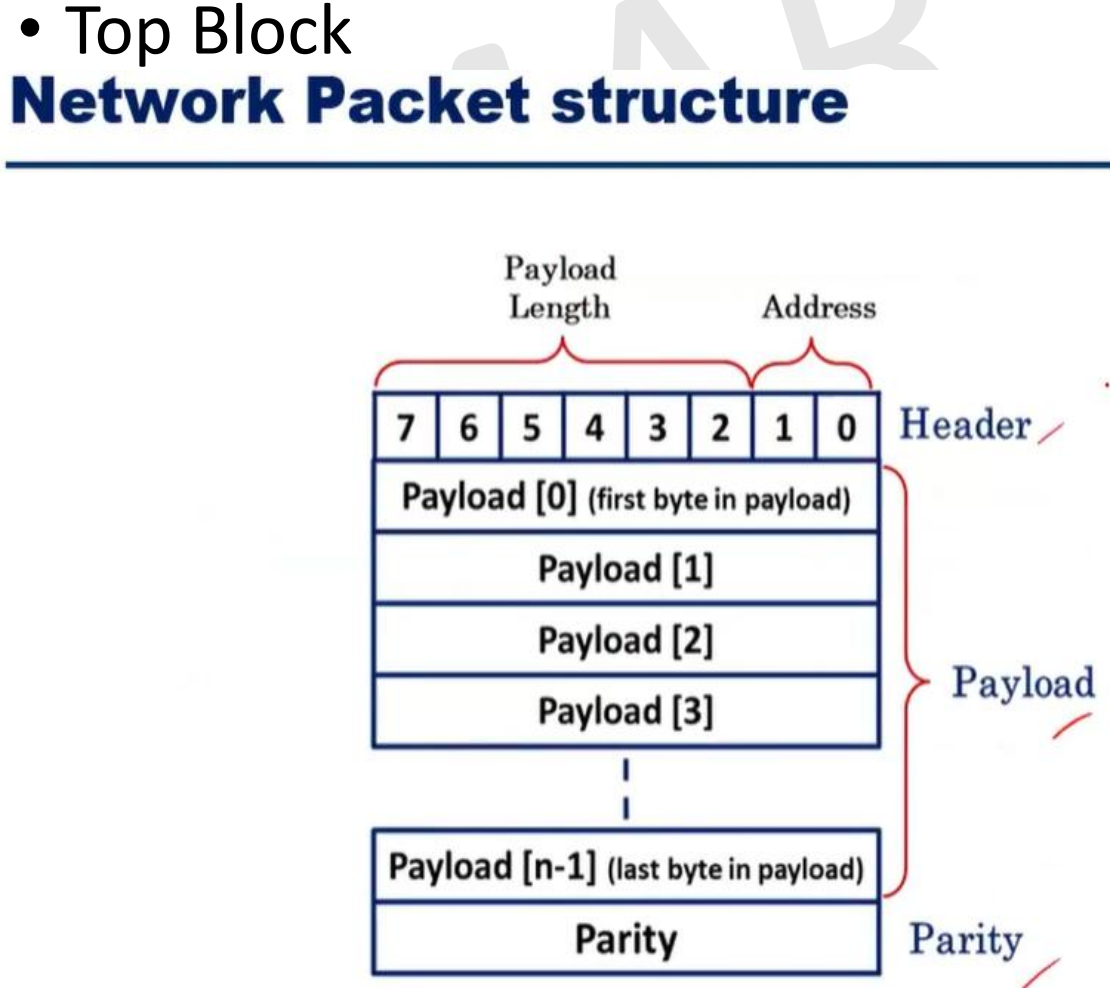
- Top Block



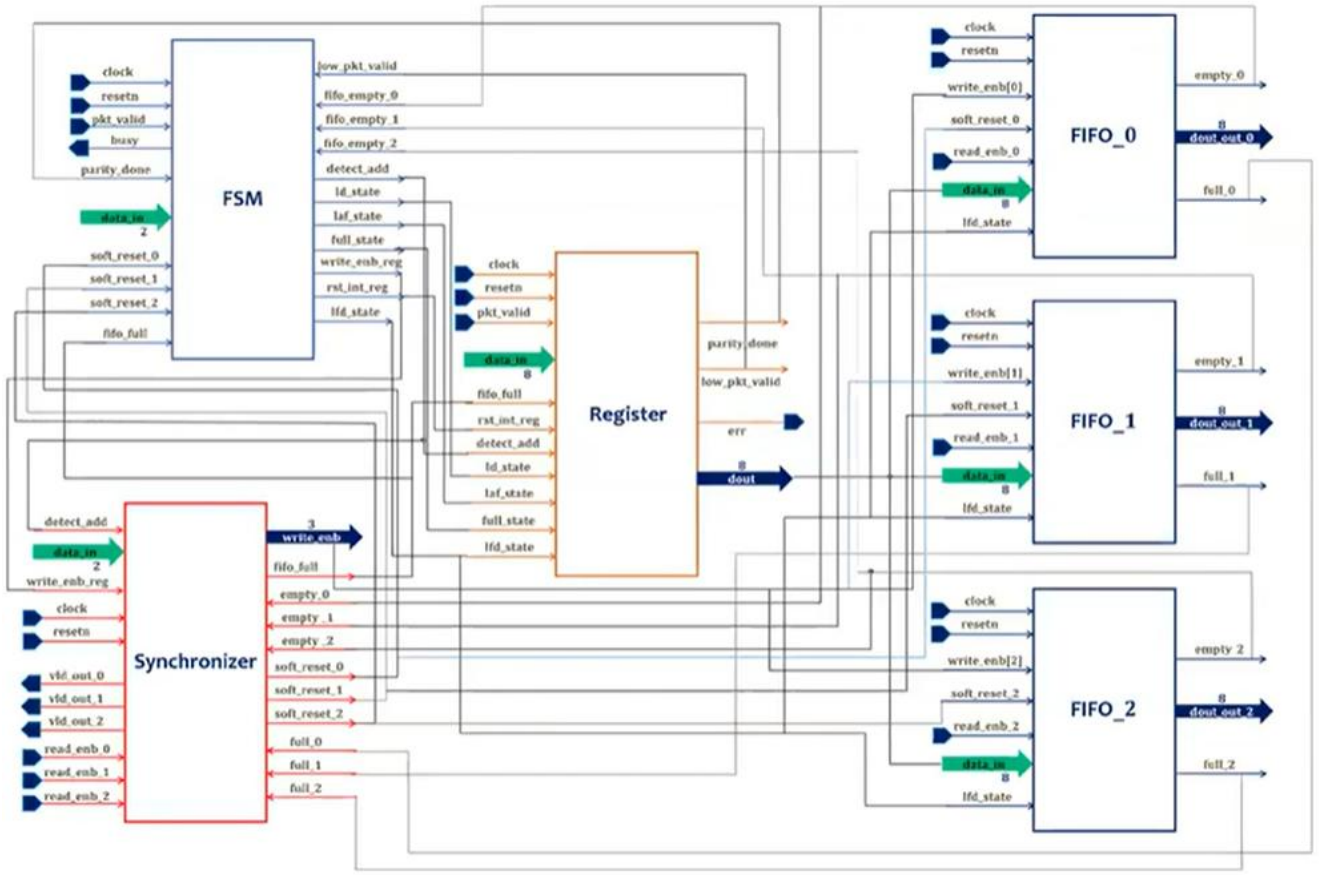
ROUTER DESIGN(Contd..)

- Setting Parity is done in steps.
- When Header is received, We do bitwise XOR for 8'b0 with Header bits.(lets store this as OP1)
- Now Payload[0] is bitwise XORed with OP1. (lets store this as OP2)
- Now Payload[1] is bitwise XORed with OP2.
- This is continued till last Payload[n-1].
- The final output will be parity byte.
- While receiving Bytes we calculate parity simulatneously and store it as Internal Parity.
- At end of recpetion, We calculate parity again and store it as final parity.
- Now we compare Internal Parity with final parity and make error=1 if they are not equal.

- Top Block



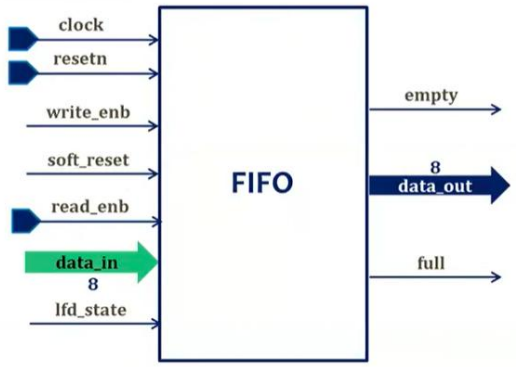
SUB-BLOCK DESIGN


FSM
clock
resetn
pkt_valid
busy
parity_done
data_in
2
soft_reset_0
soft_reset_1
soft_reset_2
fifo_full
low_pkt_valid
fifo_empty_0
fifo_empty_1
fifo_empty_2
detect_add
ld_state
laf_state
full_state
write_enb_reg
rst_int_reg
lfd_state
Register
clock
resetn
pkt_valid
data_in
8
fifo_full
rst_int_reg
detect_add
ld_state
laf_state
full_state
lfd_state
parity_done
low_pkt_valid
err
dout
8
Synchronizer
detect_add
data_in
2
write_enb_reg
clock
resetn
vld_out_0
vld_out_1
vld_out_2
read_enb_0
read_enb_1
read_enb_2
3
write_enb
fifo_full
empty_0
empty_1
empty_2
soft_reset_0
soft_reset_1
soft_reset_2
full_0
full_1
full_2
FIFO_0
clock
resetn
write_enb[0]
soft_reset_0
read_enb_0
data_in
8
lfd_state
empty_0
8
dout_out_0
full_0
FIFO_1
clock
resetn
write_enb[1]
soft_reset_1
read_enb_1
data_in
8
lfd_state
empty_1
8
dout_out_1
full_1
FIFO_2
clock
resetn
write_enb[2]
soft_reset_2
read_enb_2
data_in
8
lfd_state
empty_2
8
dout_out_2
full_2

FUNCTIONALITIES OF SUB BLOCKS

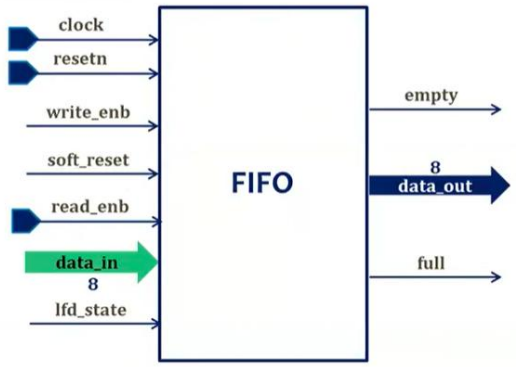
1. FIFO memories are used to store the data coming into Router.
2. Syncroniser generates timeout signals and write enable signal.
3. FSM generates inputs for other blocks
4. Register helps load the packet coming from the source and process them depending on control signals.

FIFO



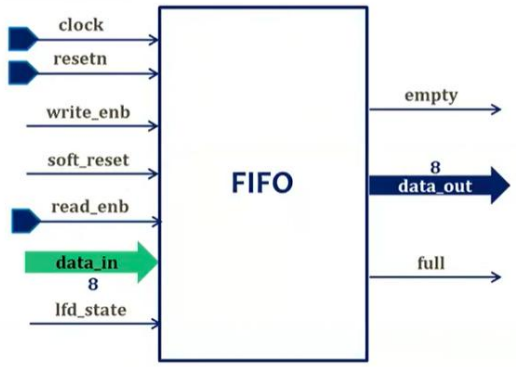
- FIFO we require is (16x9).
- Packets coming from the source will be stored in the memory.
- Packet pattern is {HEADER,PAYLOAD,PARITY}.

FIFO(Contd..)



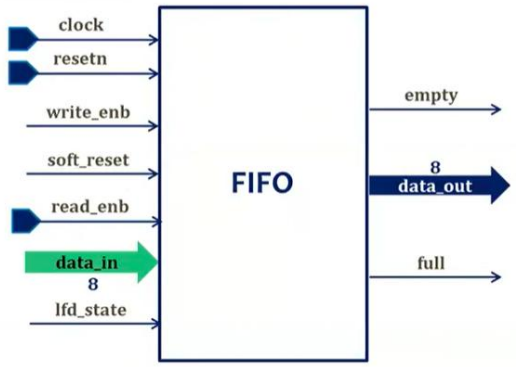
- Conditions to perform write operation:
  1- Posedge of CLK
  2- RESET is deactivated
  3- (WE==1 && FULL==0)
- For FIFO, [7:0] is data_in{HEADER,PAYLOAD,PARITY} and [8] is lfd_state.
- FIFO_MEM[wr_ptr]={lfd_state,data_in}.

FIFO(Contd..)



- Conditions to perform read operation:
  1- Posedge of CLK
  2- RESET is deactivated
  3- (RE==1 && EMPTY==0)
- For reading we read only [7:0], Don’t need to read LFD state.

FIFO(Contd..)

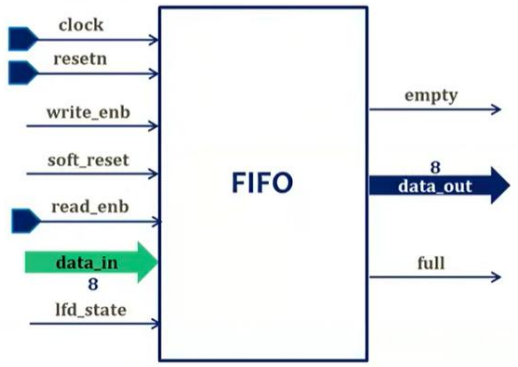


- lfd_state is required to denote the place of Header_byte.

LFD=1	0	0	0	1	0	0	1	0
LFD=0	1	1	1	1	1	1	1	0
LFD=0	1	1	1	1	1	1	0	0
LFD=0	1	1	1	1	1	0	0	0
LFD=0	1	1	1	1	0	0	0	0
LFD=0	parity	parity	parity	parity	parity	parity	parity	parity

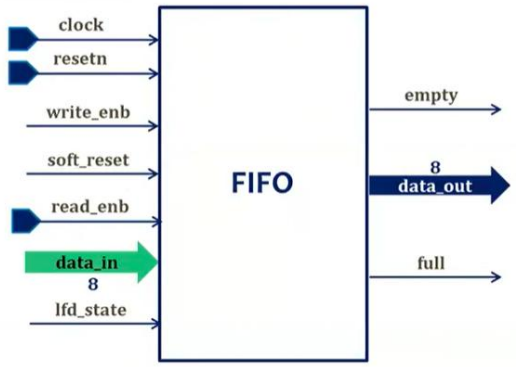
- In above table. When LFD=1. The bits highlighted in green are address. The bits highlighted in yellow are no. of data packets.
- In above table, When LFD=0, The row is completely dedicated to data.

FIFO(Contd..)



- After valid goes high, read_enb should be sent by destination within 30 clk cycles.
- If this doesn't occur, Then soft_reset occurs and data_packets inside FIFO_memory will be erased.
- This is the function of soft reset.
- The priority in operations will be-
  1- resetn(active_low)
  2- soft_reset(active_high)
  3- read/write
- When soft_reset occurs or when a packet is completely read by destination, The data_out pin should go to hi-z.

FIFO(Contd..)

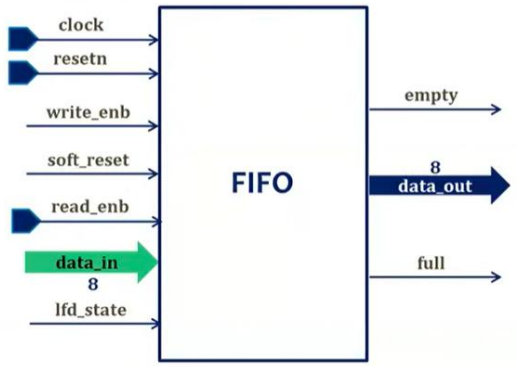


- To see if all packets are read, We can use a condition said below.

LFD=1	0	0	0	1	0	0	1	0
LFD=0	1	1	1	1	1	1	1	0
LFD=0	1	1	1	1	1	1	0	0
LFD=0	1	1	1	1	1	0	0	0
LFD=0	1	1	1	1	0	0	0	0
LFD=0	parity	parity	parity	parity	parity	parity	parity	parity

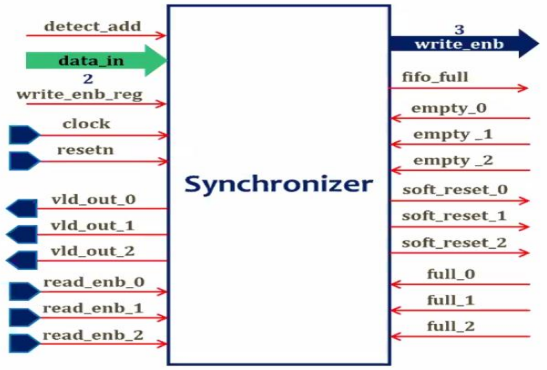
- total dat_bytes=data_payload+2.

FIFO(Contd..)



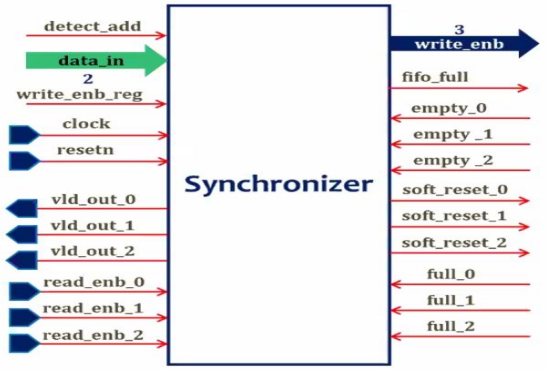
- After valid goes high, read_enb should be sent by destination within 30 clk cycles.
- If this doesn't occur, Then soft_reset occurs and data_packets inside FIFO_memory will be erased.
- This is the function of soft reset.
- The priority in operations will be-
  1- resetn(active_low)
  2- soft_reset(active_high)
  3- read/write
- When soft_reset occurs or when a packet is completely read by destination, The data_out pin should go to hi-z.

Synchronizer



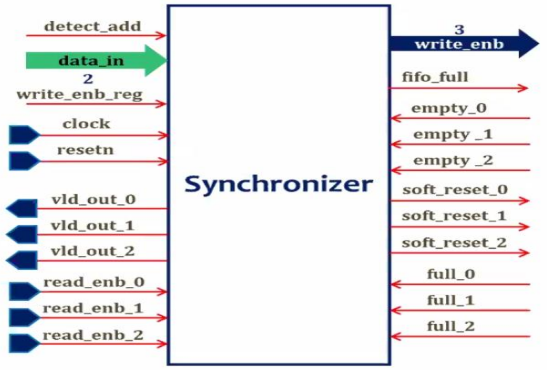
- This block is used to implement the timeout signal for soft_reset of FIFO.
- This also generates write_enable for FIFO.
- write_enb is one hot encoding.
- First job of Synchronizer is to capture the address.
- The Synchronizer also recieves FULL/EMPTY signals from FIFO.
- assign valid_out=~(empty)
- data_in is lsb of header(i.e.., address). This will get read when detect_add==1.

Synchronizer(Contd..)



- write_enb is a 3bit output which is one hot encoded for 3 FIFOs
  write_enb=001(select FIFO-0)
  write_enb=010(select FIFO-1)
  write_enb=100(select FIFO-2)
- write_enb will generate address when write_enb_reg=1;
- fifo_full will be assigned depending on address.
  data_in=00; assign fifo_full=full_0;
  data_in=01; assign fifo_full=full_1;
  data_in=10; assign fifo_full=full_2;
- if(empty_0==0)assign vld_out_0=1;
  if(empty_1==0)assign vld_out_1=1;
  if(empty_2==0)assign vld_out_2=1;

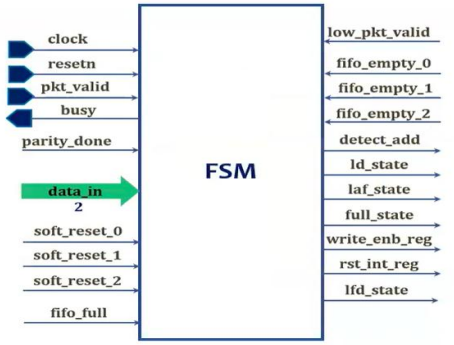
Synchronizer(Contd..)



- If vld_out_x goes high, Within 30 clock cycles read_enb_x should go high. If this doesn't happen soft_reset_x should go high.

```
assign vld_out_0=~(empty_0);
integer temp=0;
integer i;
always @(clock)
begin
if(vld_out_0==1 && read_enb_0==0 && temp<30)
begin
temp=temp+1;
end
```

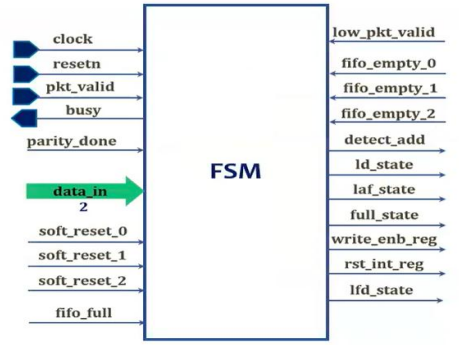
FSM

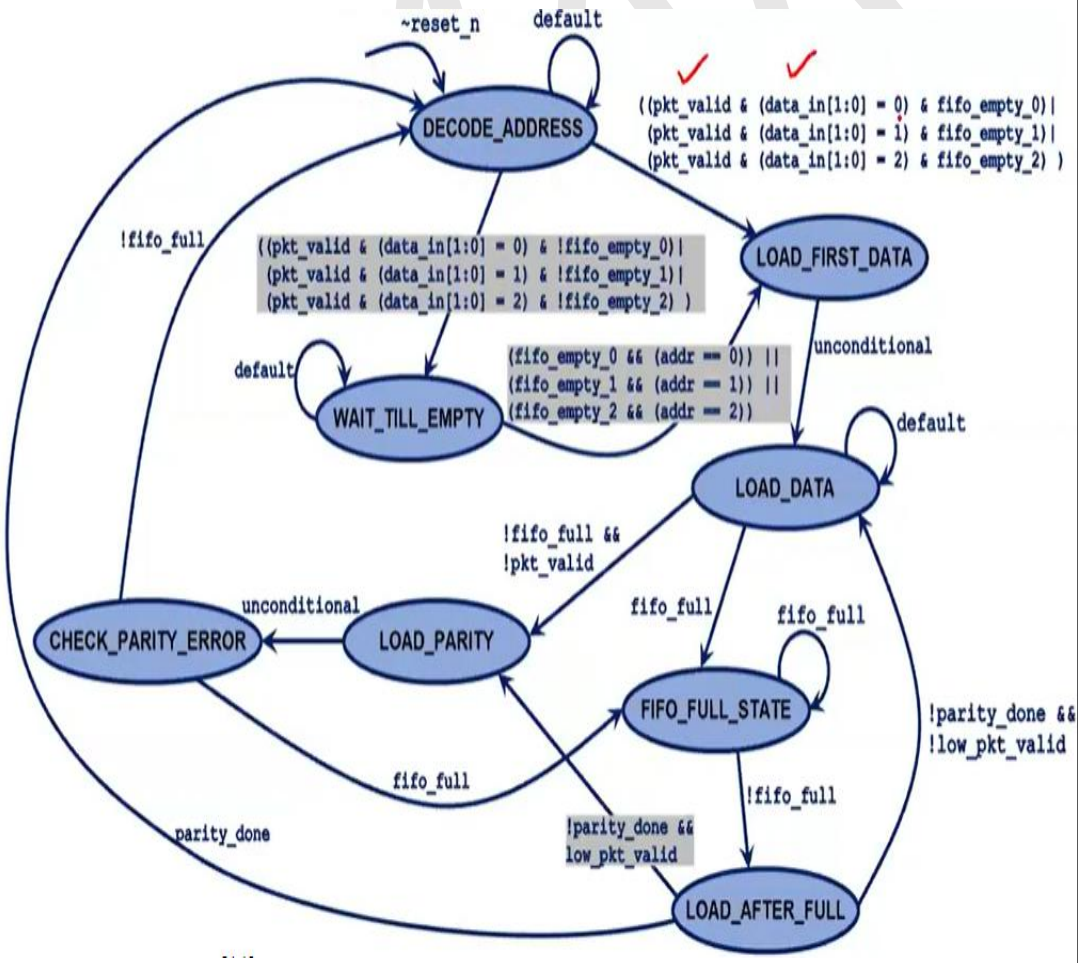


- Clock and resetn are default inputs.
- pkt_valid=1 indicates start of new packet
  pkt_valid=0 indicates end of current packet
- parity_done=1 indicates router has sent parity byte(i.e.., packet is over).
- data_in[1:0] is address.
- soft_reset_x=1, FSM goes to idle state.
- low_pkt_valid=~pkt_valid, high indicates termination of current packet.

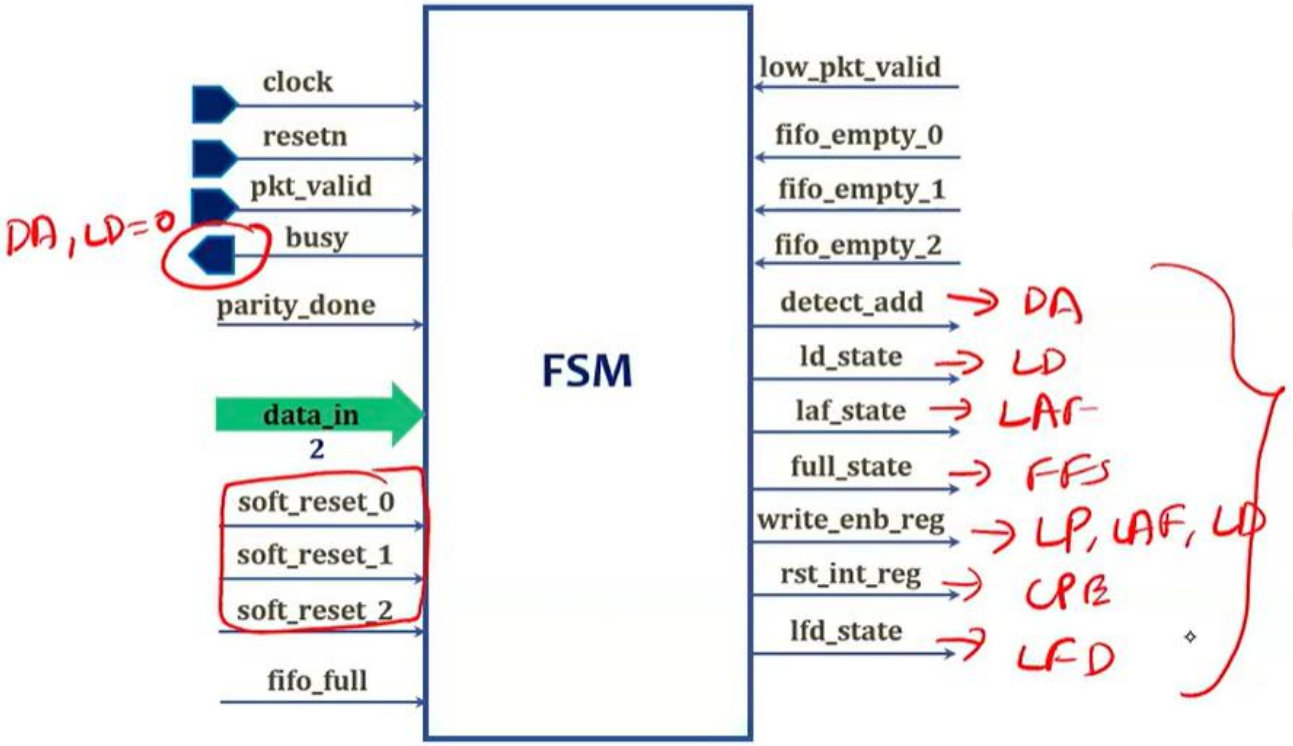
FSM(Contd..)

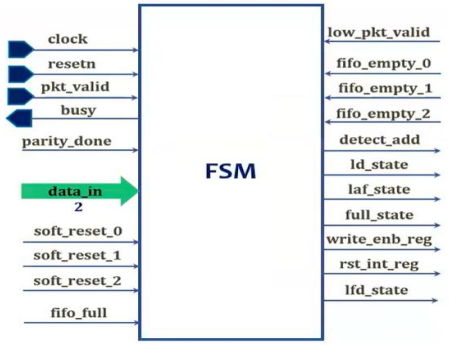
- The state transition is occured as shown in figure.
- The FIFO depth is 16. If no. of packets is greater than 14, From LOAD_DATA it'll go to FIFO_FULL_STATE and remaining bytes will start coming after we start reading the existing data.

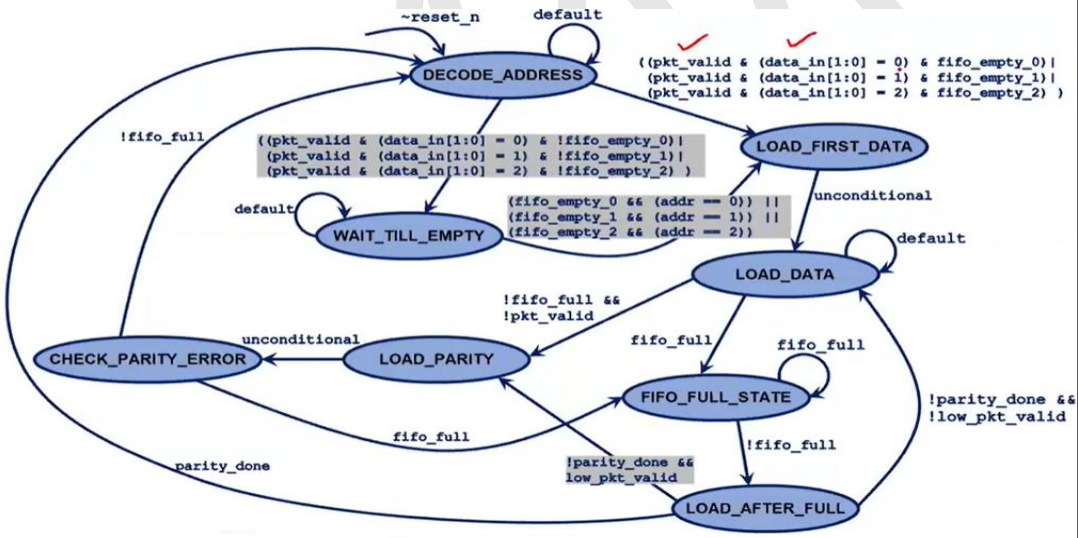


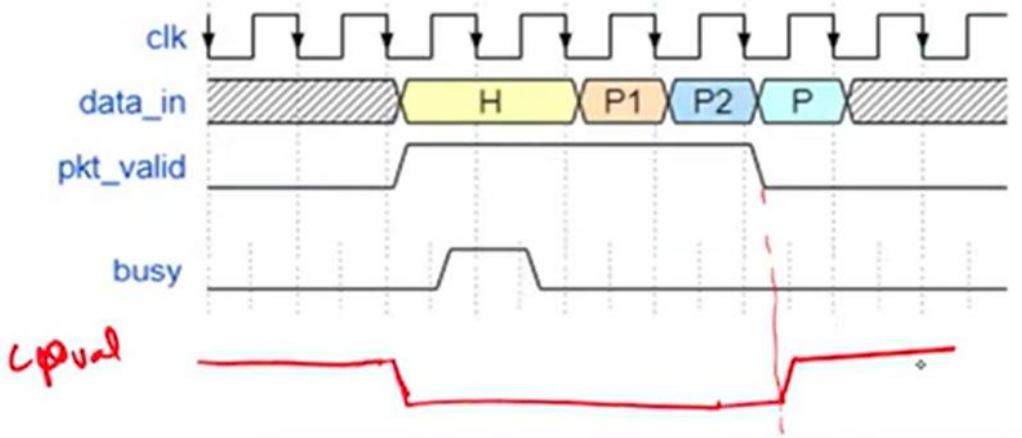


FSM(Contd..)

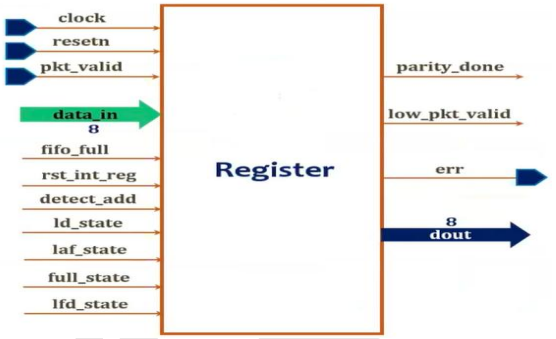

clock
resetn
pkt_valid
busy
parity_done
data_in
2
soft_reset_0
soft_reset_1
soft_reset_2
fifo_full
FSM
low_pkt_valid
fifo_empty_0
fifo_empty_1
fifo_empty_2
detect_add
ld_state
laf_state
full_state
write_enb_reg
rst_int_reg
lfd_state
DA, LD=0
DA
LD
LAF
FFS
LP, LAF, LD
RST
LFD


clock
resetn
pkt_valid
busy
parity_done
data_in
2
soft_reset_0
soft_reset_1
soft_reset_2
fifo_full
FSM
low_pkt_valid
fifo_empty_0
fifo_empty_1
fifo_empty_2
detect_add
ld_state
laf_state
full_state
write_enb_reg
rst_int_reg
lfd_state


~reset_n
default
DECODE_ADDRESS
((pkt_valid & (data_in[1:0] = 0) & fifo_empty_0)|
(pkt_valid & (data_in[1:0] = 1) & fifo_empty_1)|
(pkt_valid & (data_in[1:0] = 2) & fifo_empty_2))
!fifo_full
((pkt_valid & (data_in[1:0] = 0) & !fifo_empty_0)|
(pkt_valid & (data_in[1:0] = 1) & !fifo_empty_1)|
(pkt_valid & (data_in[1:0] = 2) & !fifo_empty_2))
LOAD_FIRST_DATA
unconditional
default
WAIT_TILL_EMPTY
(fifo_empty_0 && (addr == 0)) ||
(fifo_empty_1 && (addr == 1)) ||
(fifo_empty_2 && (addr == 2))
default
LOAD_DATA
!fifo_full &&
!pkt_valid
unconditional
CHECK_PARITY_ERROR
LOAD_PARITY
fifo_full
fifo_full
FIFO_FULL_STATE
!parity_done &&
!low_pkt_valid
fifo_full
!fifo_full
parity_done
!parity_done &&
low_pkt_valid
LOAD_AFTER_FULL

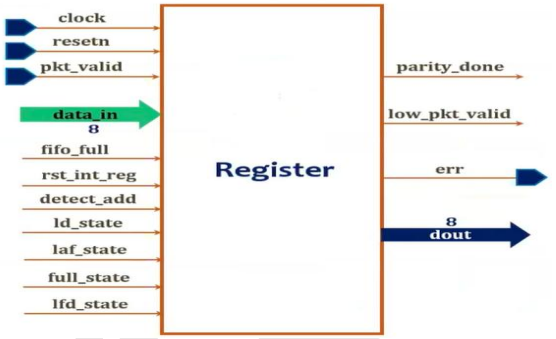

clk
data_in
H
P1
P2
P
pkt_valid
busy
LPvalid

REGISTER



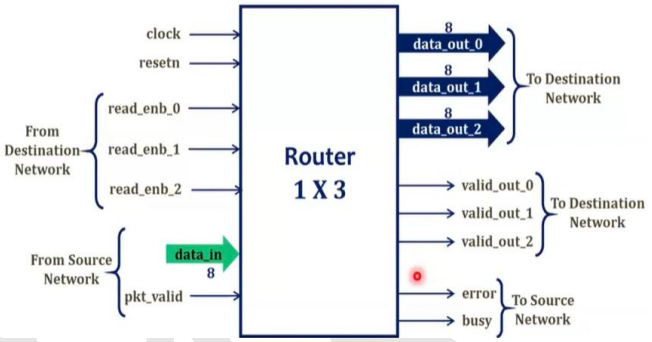
- It acts as Loadable Register. i.e.., to load the packet driven from source by using control signals from FSM.
- From source, data is going to data_in and goes to FIFO through dout.
- We have four internal variables,
  1-Header Byte
  2-FIFO Full state Byte
  3-Internal Parity
  4-Packet Parity
- Header byte is used to store the Header.
- FIFO Full state byte is used to temporarily store the data_in byte incase of next byte arrival.
- Internal Parity byte store the XOR of successive bytes coming in.
- Packet Parity byte is used to store the parity coming from source directly. When this gets stored, parity_done=1.

REGISTER(Contd..)



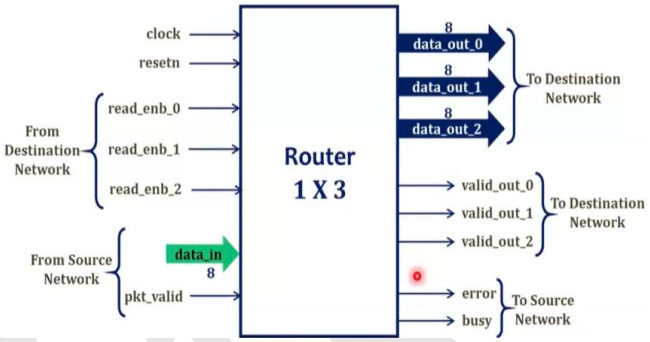
- When detect_add=1, The contents of data_in will go to Header_byte.
- When lfd_state=1, The byte inside Header_byte variable will go to dout. From dout it'll go to one of the FIFO referring the address.
- When ld_state=1, The data_in will be directly driven to dout.
- When fifo_full=1, The data_in will be stored inside FIFO Full state Byte.
- When laf=1, The contents of FIFO Full state Byte register will go to dout.
- When rst_int_reg=1, It'll reset parity registers.
- When full_state=1, internal parity register calculation is stopped.
- When packet_parity byte is received parity_done=1.
- low_pkt_valid=~pkt_valid.
- If contents of internal parity and packet parity doesn't match, err=1.
- dout corresponds to FIFOs depending upon address.

TOP_BLOCK



- We write the code in such way we include all the subblocks.
- We need
  FSM(1 No.s)
  Syncroniser(1 No.s)
  Register(1 No.s)
  FIFO(3 No.s)
- We need to verify packets of lengths of 12,14,16.

TOP_BLOCK(Contd..)



- The task in beside code is ised to write TB code.
- We can write task for 14packets, 12packets & 16packets.
- We call each task and do read_enb=1 for readback.