Verilog HDL
Mini Project
"Router 1x3 Design"

www.maven-silicon.com

Table of Contents

Project Schedule

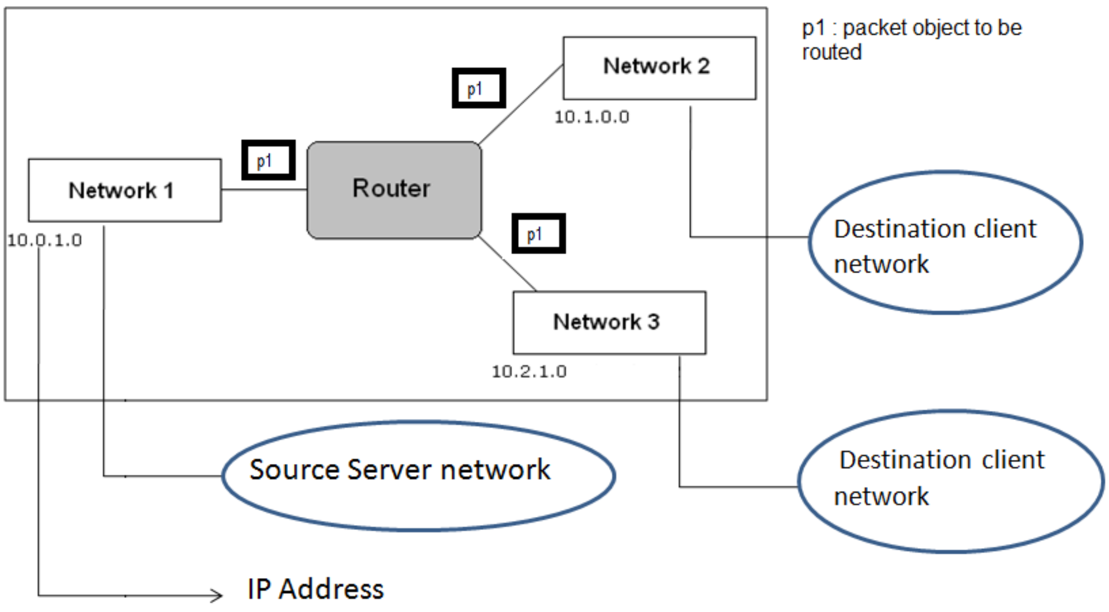
- Specification Analysis of Router protocol
- FIFO
- SYNCHRONIZER
- CONTROLLER
- REGISTER
- ROUTER TOP

Dates of RTL sign-off

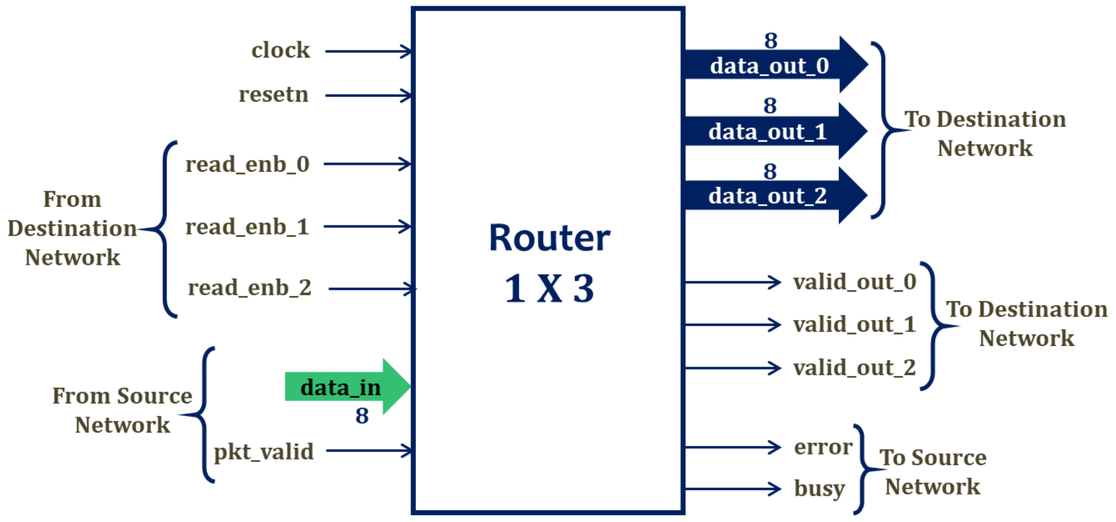
S. No	Module	Date of release	Status	Remarks
1	FIFO			
2	Router Synchronizer			
3	Router Controller			
4	Router Register			
5	Router Top			

Introduction

ROUTER is a device that forwards data packets between computer networks. It is an OSI layer 3 routing device. It drives an incoming packet to an output channel based on the address field contained in the packet header.



Router- Top Level Block



Router- Interface

clock	Active high clocking event
pkt_valid	pkt_valid is an active high input signal that detects an arrival of a new packet from a source network
resetn	Active low synchronous reset
data_in	8 bit input data bus that transmits the packet from source network to router
read_enb_0	Active high input signal for reading the packet through output data bus data_out_0
read_enb_1	Active high input signal for reading the packet through output data bus data_out_1
read_enb_2	Active high input signal for reading the packet through output data bus data_out_2
data_out_0	8 bit output data bus that transmits the packet from the router to destination client network 1
data_out_1	8 bit output data bus that transmits the packet from the router to destination client network 2
data_out_2	8 bit output data bus that transmits the packet from the router to destination client network 3
vld_out_0	Active high signal that detects that a valid byte is available for destination client network 1
vld_out_1	Active high signal that detects that a valid byte is available for destination client network 2
vld_out_2	Active high signal that detects that a valid byte is available for destination client network 3
busy	Active high signal that detects a busy state for the router that stops accepting any new byte
error	Active high signal that detects the mismatch between packet parity and internal parity

Router top - Overview

The Router 1x3 design follows packet based protocol and it receives the network packet from a source LAN using **data_in** on a byte by byte basis on active posedge of the **clock**. **resetn** is an active low synchronous reset.

The start of a new packet is indicated by asserting **pkt_valid** and end of the current packet is indicated by de-asserting **pkt_valid**. The design stores the incoming packet inside a FIFO as per the address of the packet. The design has got 3 FIFO's for respective destination LANs.

During packet read operation, the destination LANs monitors **vld_out_x** (x can be 0, 1, or 2), and then asserts **read_enb_x** (x can be 0, 1, or 2). The packet is read by the destination LAN's using the channels **data_out_x** (x can be 0, 1, or 2).

Sometimes, router can enter into busy state which is indicated by the signal **busy.** The busy signal is sent back to the source LAN so that the source has to wait to send the next byte of the packet.

To confirm the correctness of the packet, received by the router, we have implemented an error detection mechanism i.e parity check. If there is a mismatch in the parity byte sent by the source LAN and the internal parity calculated by the router, then the **error** signal is asserted. This error signal is sent back to the source LAN so that by monitoring the same, the source LAN can resend the packet.

This design can receive only 1 packet at a time, but 3 packets can be read simultaneously.

Router- 1x3 Features

- **Packet Routing :**The packet is driven from the input port and is routed to any one output port, based on the address of the destination network.
- **Parity Checking :** An error detection technique that tests the integrity of digital data being transmitted between Server & Client. This technique ensures that the data transmitted by the Server network is received by the Client network without getting corrupted.
- **Reset :** It is an active low synchronous input that resets the router. Under reset condition, the router FIFOs are made empty and the valid out signals goes low indicating that no valid packet is detected on the output data bus.
- **Sending Packet :** Refer Router input protocol
- **Reading Packet :** Refer Router output protocol

Router- Packet

- **Packet Format :** The Packet consists of 3 parts i.e Header, payload and parity such that each id of 8 bits width and the length of the payload can be extended between 1 byte to 63 bytes.

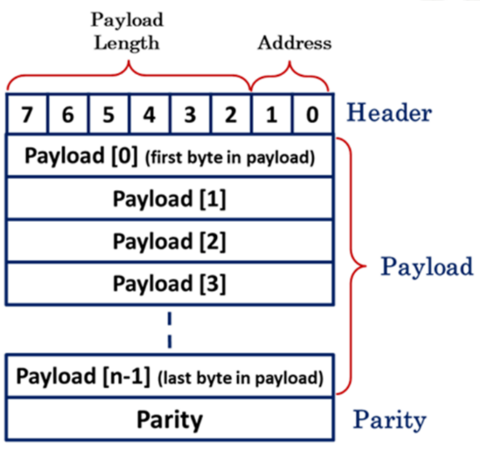
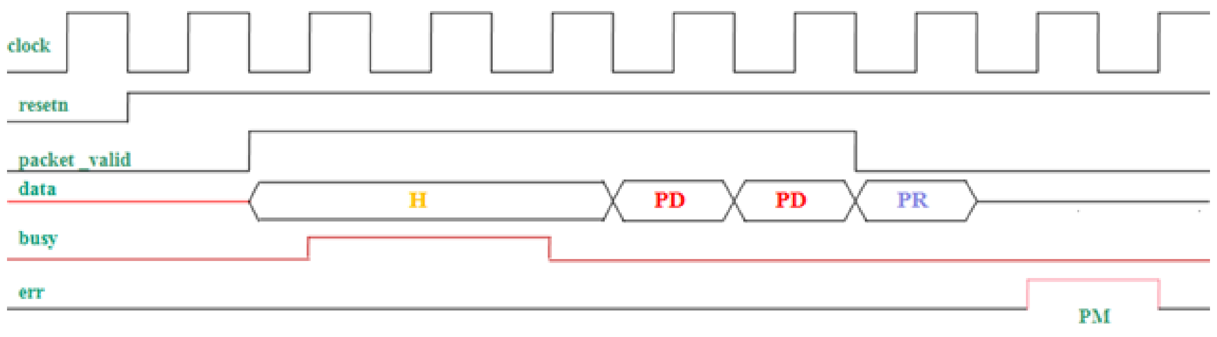


Figure - Packet Format

- **Header :** Packet header contains two fields DA and length.
    - DA: Destination address of the packet is of 2 bits. The router drives the packet to respective ports based on this destination address of the packets. Each output port has 2-bit unique port address. If the destination address of the packet matches the port address, then router drives the packet to the output port. The address “3” is invalid.
    - Length: Length of the data is of 6 bits. It specifies the number of data bytes. A packet can have a minimum data size of 1 byte and a maximum size of 63 bytes.
      If Length = 1, it means data length is 1 byte
      If Length = 63, it means data length is 63 bytes
- **Payload :** Payload is the data information. Data should be in terms of bytes.
- **Parity :** This field contains the security check of the packet. It is calculated as bitwise parity over the header and payload bytes of the packet as mentioned below.

Router- Input Protocol



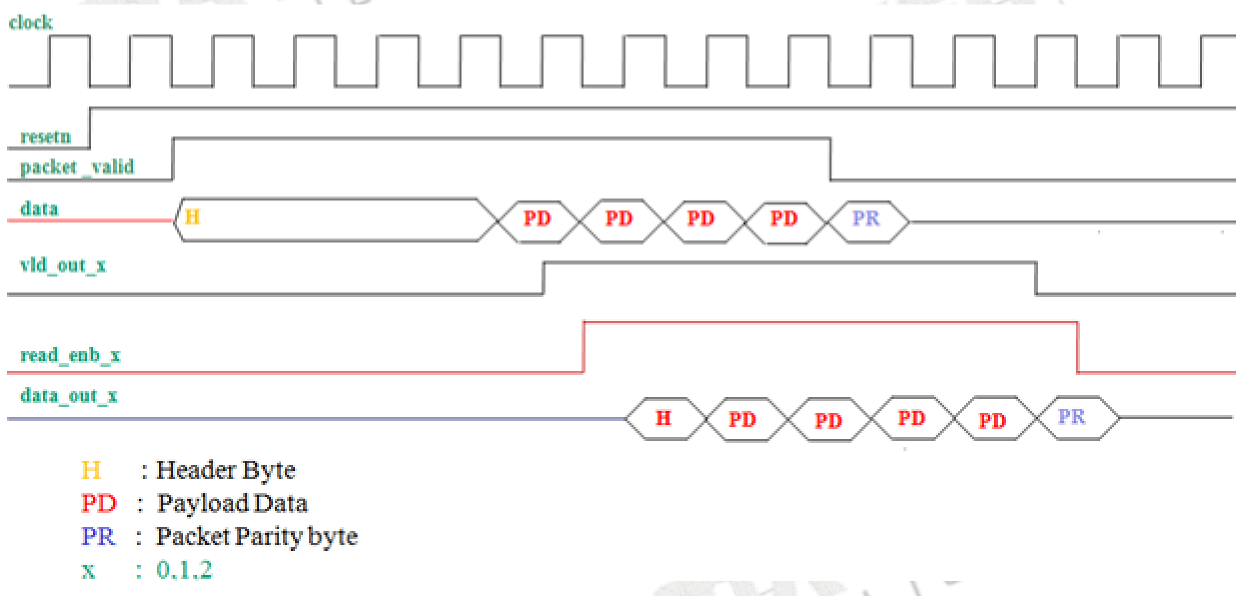
H : Header Byte
PD : Payload Data
PR : Packet Parity byte
PM : Parity Mismatch

The characteristics of the DUT input protocol are as follows:

- **TestBench Note :** All input signals are active high except active low reset and are synchronized to the falling edge of the clock. This is because the DUT router is sensitive to the rising edge of the clock. Therefore, in the testbench, driving input signals on the falling edge ensures adequate setup and hold time. But in the SystemVerilog/UVM based testbench, clocking block can be used to drive the signals on the positive edge of the clock itself and thus avoids metastability.
- The packet_valid signal is asserted on the same clock edge when the header byte is driven onto the input data bus.
- Since the header byte contains the address, this tells the router to which output channel the packet should be routed to (data_out_0, data_out_1, or data_out_2).
- Each subsequent byte of payload after header byte should be driven on the input data bus for every new falling edge of clock.
- After the last payload byte has been driven, on the next falling clock, the packet_valid signal must be deasserted, and the packet parity byte should be driven. This signals completion of the packet.

- The testbench shouldn’t drive any bytes when busy signal is detected instead it should hold the last driven value.
- The “busy” signal when asserted drops any incoming byte of data.
- The “err” signal is asserted when a packet parity mismatch is detected.

Router- Output Protocol

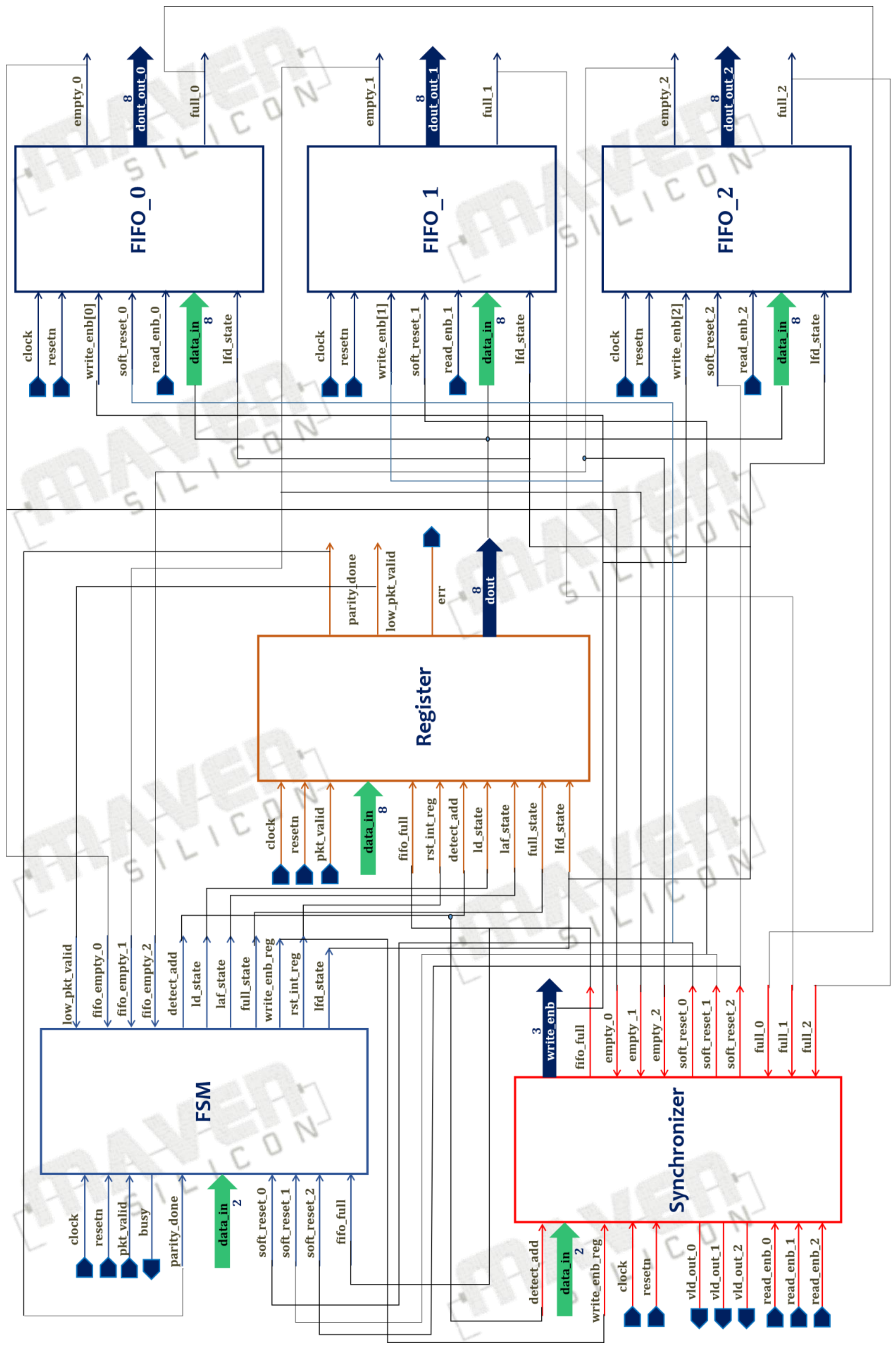


The characteristics of the output protocol are as follows:

- **TestBench Note :** All output signals are active high and are synchronized to the rising edge of the clock.
- Each output port data_out_X (data_out_0, data_out_1, data_out_2) is internally buffered by a FIFO of size 16X9.
- The router asserts the vld_out_X (vld_out_0, vld_out_1 or vld_out_2) signal when valid data appears on the vld_out_X (data_out_0, data_out_1 or data_out_2) output bus. This is a signal to the receiver's client which indicates that data is available on a particular output data bus.
- The packet receiver will then wait until it has enough space to hold the bytes of the packet and then respond with the assertion of the read_enb_X (read_enb_0, read_enb1 or read_enb_2) signal.
- The read_enb_X (read_enb_0, read_enb_1 or read_enb_2) input signal can be asserted on the falling clock edge in which data are read from the data_out_X (data_out_0, data_out_1 or data_out_2) bus.

- The read_enb_X (read_enb_0, read_enb_1 or read_enb_2) must be asserted within 30 clock cycles of vld_out_X (vld_out_0, vld_out_1 or vld_out_2) being asserted else time-out occurs, which resets the FIFO.
- The data_out_X bus will be tri-stated during a scenario when a packet's byte is lost due to time-out condition.

Router top- Block level architecture

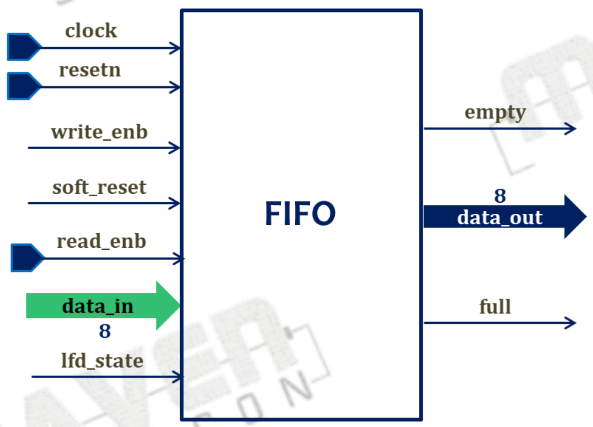


Router top- Sub-blocks

The top level block module consists of 6 sub blocks as follows :

- 3 FIFOs
- Synchronizer
- Register
- Finite state Machine

Router- FIFO



Functionality :

There are 3 FIFOs used in the router design. Each FIFO is of 9 bits width and of 16 locations as depth.The FIFO works on the system clock and is reset with a synchronous active low reset. The FIFO is also internally reset by an internal reset signal **soft_reset** .**soft_reset** is an active high signal which is generated by the SYNCHRONIZER block during the time out state of the ROUTER.

If **resetn** is low then full =0, empty = 1 and data_out = 0.

The FIFO memory size is 16X 9. The extra bit in the data width is appended in order to detect the header byte. The **lfd_state** detects the header byte of a packet.The 9th bit is 1 for header byte and 0 for remaining bytes.

Write operation :

- Signal **data_in** is sampled at the rising edge of the clock when **write_enb** is high.
- Write operation only takes place when FIFO is not full in order to avoid over_run condition.

Read Operation:

- The data is read from **data_out** at rising edge of the clock, when **read_enb** is high.
- Read operation only takes place when FIFO is not empty in order to avoid under_run condition.
- During the read operation when a header byte is read, an internal counter is loaded with the payload length of the packet plus "1" (Parity byte) and starts decrementing every clock cycle till it reaches 0. The counter holds 0 till it is reloaded back with a new packet payload length.
- During the time out state, full = 0, empty = 1.
- **data_out** is driven to HIGH impedance state under 2 scenarios :
  - When the FIFO memory is read completely (Header+Payload+Parity).
  - Under the time out state of the ROUTER.

full – FIFO status which indicates that all the locations inside FIFO have been written.

empty – FIFO status which indicates that all the locations of FIFO have been read and made empty.

Read and Write operation can be done simultaneously.

<u>RTL Design Procedure :</u>

[1]Naming Convention :

[A] Modules & Files

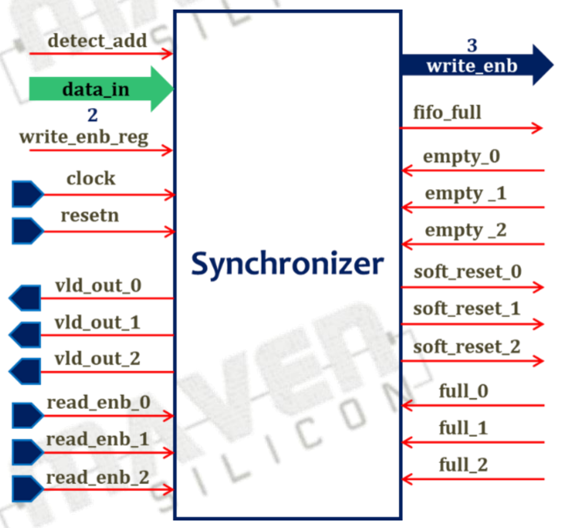
Module name: router_fifo	File Name: router_fifo.v
Testbench name: router_fifo_tb	File Name: router_fifo_tb.v

[B] Name the ports as per the block

[2]Verify the functionality of the RTL

[3]Synthesize the RTL

Router- Synchronizer



Functionality :

This module provides synchronization between router FSM and router FIFO modules. It provides faithful communication between the single input port and three output ports.

- **detect_add** and **data_in** signals are used to select a FIFO till a packet routing is over for the selected FIFO.
- Signal **fifo_full** signal is asserted based on full status of FIFO_0 or FIFO_1 or FIFO_2.
- If **data_in** = 2'b00 then **fifo_full** = **full_0**
- If **data_in** = 2'b01 then **fifo_full** = **full_1**
- If **data_in** = 2'b10 then **fifo_full** = **full_2 else fifo_full = 0**
- The signal **vld_out_x** signal is generated based on empty status of the FIFO as shown below
    - **vld_out_0 = ~empty_0**
    - **vld_out_1 = ~empty_1**
    - **vld_out_2 = ~empty_2**

- The **write_enb_reg** signal is used to generate **write_enb** signal for the write operation of the selected FIFO.
- There are 3 internal reset signals **(soft_reset_0,soft_reset_1,soft_reset_2)** for each of the FIFO respectively. The respective internal reset signals goes high if read_enb_X (read_enb_0, read_enb_1 or read_enb_2) is not asserted within 30 clock cycles of the vld_out_X (vld_out_0, vld_out_1 or vld_out_2) being asserted respectively.

<u>RTL Design Procedure :</u>

[1]Naming Convention :

[A] Modules & Files

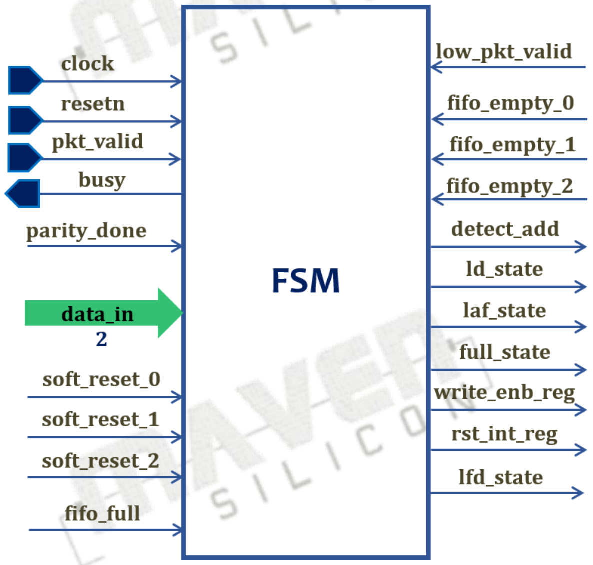
Module name: router_sync	File Name: router_sync.v
Testbench name: router_sync_tb	File Name: router_sync_tb.v

[B] Name the ports as per the block

[2]Verify the functionality of the RTL

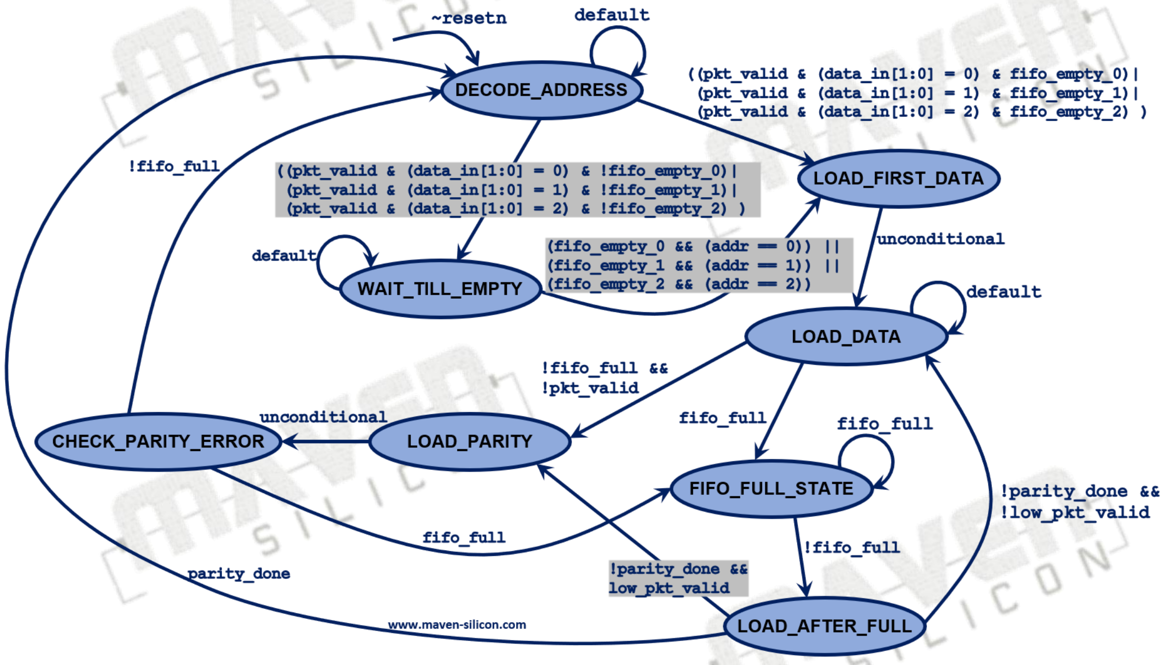
[3]Synthesize the RTL

Router- Controller



Functionality:

The FSM module is the controller circuit for the ROUTER. This module generates all the control signals when a new packet is received by the ROUTER. These control signals are used by other design components in order to transfer the packet to the output port.


~resetn
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
default
WAIT_TILL_EMPTY
(fifo_empty_0 && (addr == 0)) ||
(fifo_empty_1 && (addr == 1)) ||
(fifo_empty_2 && (addr == 2))
unconditional
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
www.maven-silicon.com
LOAD_AFTER_FULL

STATE – DECODE_ADDRESS

- This is the initial reset state.
- Signal **detect_add** is asserted in this state which is used to detect an incoming packet. It is also used to latch the first byte as a header byte.

STATE - LOAD_FIRST_DATA

- Signal **lfd_state** is asserted in this state which is used to load the first data byte to the FIFO.
- Signal **busy** is also asserted in this state so that header byte that is already latched doesn't update to a new value for the current packet.
- This state is changed to **LOAD_DATA** state unconditionally in the next clock cycle.

STATE - LOAD_DATA

- In this state the signal **ld_state** is asserted which is used to load the payload data to the FIFO.
- Signal **busy** is deasserted in this state, so that ROUTER can receive new data from input source every clock cycle.
- Signal **write_enb_reg** is asserted in this state in order to write the the Packet information(Header+Payload+Parity)to the selected FIFO.
- This state transits to LOAD_PARITY state when pkt_valid goes low and to FIFO_FULL_STATE when FIFO is full.

STATE – LOAD_PARITY

- In this state the last byte is latched which is the parity byte.
- It goes unconditionally to the state CHECK_PARITY_ERROR..
- Signal **busy** is asserted so that ROUTER doesn't accepts any new data.
- **write_enb_reg** is made high for latching the parity byte to FIFO.

STATE – FIFO_FULL_STATE

- **busy** signal is made high and **write_enb_reg** signal is made low.
- Signal **full_state** is asserted which detects the FIFO full state.

STATE – LOAD_AFTER_FULL

- In this state **laf_state** signal is asserted which is used to latch the data after FIFO_FULL_STATE.
- Signal **busy** & **write_enb_reg** is asserted.
- It checks for **parity_done** signal and if it is high ,shows that LOAD_PARITY state has been detected and it goes to the state DECODE_ADDRESS.
- If **low_pkt_valid** is high it goes to LOAD_PARITY state otherwise it goes back to the LOAD_DATA state.

STATE – WAIT_TILL_EMPTY

- **busy** signal is made high and **write_enb_reg** signal is made low.

STATE – CHECK_PARITY_ERROR

- In this state **rst_int_reg** signal is generated, which is used to reset **low_pkt_valid** signal.
- This state changes to **DECODE_ADDRESS** when **FIFO** is not full and to **FIFO_FULL_STATE** when **FIFO** is full.
- **busy** is asserted in this state.

P.S: The Soft-reset signals should be used in the FSM in such a way that the current state should change back to **"DECODE_ADDRESS"** state only for the timeout situation of the current transmitted packet.

RTL Design Procedure :

[1]Naming Convention :

[A] Modules & Files

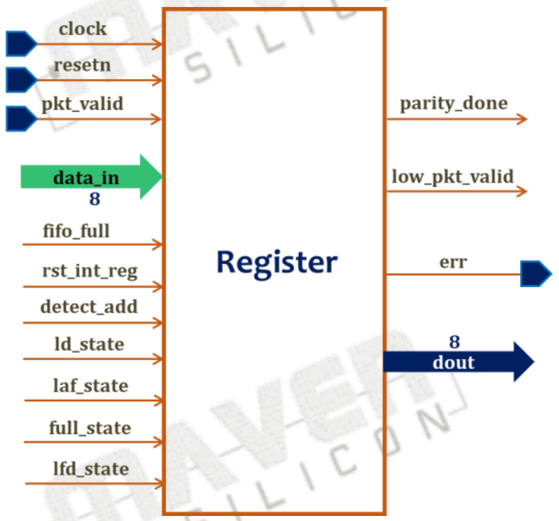
Module name: router_fsm	File Name: router_fsm.v
Testbench name: router_fsm_tb	File Name: router_fsm_tb.v

[B] Name the ports as per the block

[2]Verify the functionality of the RTL

[3]Synthesize the RTL

Router- Register



Functionality :

This module implements 4 internal registers in order to hold header byte, FIFO full state byte ,internal parity and packet parity byte.

All the registers in this module are latched on the rising edge of the **clock**.

- If **resetn** is low then the signals (**dout, err, parity_done and low_pkt_valid**) are made low.
- The signal **parity_done** is high under the following conditions
  - When signal **ld_state** is high and signals (**fifo_full and pkt_valid)** are low.
  - When signals **laf_state** and **low_pkt_valid** both are high and the previous value of **parity_done** is low.
- **rst_int_reg** signal is used to reset **low_pkt_valid** signal.
- **detect_add** signal is used to reset **parity_done** signal.
- **Signal low_pkt_valid** is high when **ld_state** is high and **pkt_valid** is low. **low_pkt_valid** shows that pkt_valid for current packet has been deasserted.

- First data byte i.e., header is latched inside an internal register when **detect_add** and **pkt_valid** signals are high. This data is latched to the output **dout** when **lfd_state** signal goes high.
- Then signal **data_in** i.e payload is latched to **dout** if **ld_state** signal is high and **fifo_full** is low.
- Signal **data_in** is latched to an internal register when **ld_state** and **fifo_full** are high.This data is latched to output **dout** when **laf_state** goes high.
- **full_state** is used to calculate internal parity.
- Another internal register is used to store internal parity for parity matching .Internal parity is calculated using the bit-wise xor operation between header byte,payload byte and previous parity values as shown below :

  parity_reg = parity_reg_previous ^ header_byte ---- t1 clock cycle

  parity_reg = parity_reg_previous ^ payload_byte1 --- t2 clock cycle

  parity_reg = parity_reg_previous ^ payload_byte 2--- t3 clock cycle

  .

  .

  .

  parity_reg = parity_reg_previous ^ payload_byte n---tn clock cycle

  last payload byte

- The **err** is calculated only after packet parity is loaded and goes high if the packet parity doesn't match with the internal parity.

RTL Design Procedure :

[1]Naming Convention :

[A] Modules & Files

Module name: router_reg	File Name: router_reg.v
Testbench name: router_reg_tb	File Name: router_reg_tb.v

[B] Name the ports as per the block

[2]Verify the functionality of the RTL

[3]Synthesize the RTL

Router- Top level module

RTL Design Procedure :

[1] Naming Convention :

[A] Modules & Files

Module name: router_top	File Name: router_top.v
Testbench name: router_top_tb	File Name: router_top_tb.v

[B] Name the ports as per the block

[2] Instantiate all the lower level modules and implement the top module as per the architecture

[3] Verify the functionality of the RTL

[4] Synthesize the RTL

Synthesis Report:

Max. Clock frequency

FFs used

LUTs used

RAMs inferred

Latches inferred