1x3 Router Project Documentation

1. Architecture Overview

The 1x3 Router project consists of several interconnected modules designed to route incoming data packets to one of three output FIFOs. The main components include a Finite State Machine (FSM), a Register module, a Synchronizer, and three First-In, First-Out (FIFO) buffers (FIFO_0, FIFO_1, FIFO_2).

2. Module Descriptions

2.1 Finite State Machine (FSM)

The FSM is the central control unit of the router. It manages the overall flow of data based on the state of the system and incoming packet validity and parity. It generates control signals for other modules, such as write enables for the register and FIFOs, and reset signals for the FIFOs.

Inputs:

- `clock` : System clock.
- `resetn` : Asynchronous reset (active low).
- `pkt_valid` : Indicates a valid incoming packet.
- `busy` : Indicates if the router is currently busy processing a packet.
- `parity_done` : Indicates that parity checking is complete.
- `data_in` (2-bit): Input data, likely for address or control information.
- `soft_reset_0` , `soft_reset_1` , `soft_reset_2` : Soft reset signals for individual FIFOs.
- `fifo_full` : Indicates if any of the FIFOs are full.

Outputs:

- `low_pkt_valid` : Low pulse indicating a valid packet.
- `fifo_empty_0` , `fifo_empty_1` , `fifo_empty_2` : Indicate if respective FIFOs are empty.
- `detect_add` : Signal to detect address.
- `ld_state` : Load state signal.
- `laf_state` : Look-ahead FIFO state.
- `full_state` : Indicates a full state.
- `write_enb_reg` : Write enable for the Register module.

- `rst_int_reg` : Reset internal register.
- `lfd_state` : Load FIFO data state.

2.2 Register

The Register module is responsible for temporarily storing incoming data and performing parity checking. It receives data from the input and provides an 8-bit output (`dout`).

Inputs:

- `clock` : System clock.
- `resetn` : Asynchronous reset (active low).
- `pkt_valid` : Valid packet signal.
- `data_in` (8-bit): Incoming data to be registered.
- `fifo_full` : Indicates if any FIFO is full.
- `rst_int_reg` : Reset internal register signal from FSM.
- `detect_add` : Detect address signal from FSM.
- `ld_state` : Load state signal from FSM.
- `laf_state` : Look-ahead FIFO state from FSM.
- `full_state` : Full state signal from FSM.
- `lfd_state` : Load FIFO data state from FSM.

Outputs:

- `parity_done` : Indicates completion of parity check.
- `low_pkt_valid` : Low pulse indicating a valid packet.
- `err` : Error signal, likely for parity errors.
- `dout` (8-bit): Output data from the register.

2.3 Synchronizer

The Synchronizer module appears to manage the flow of data between the main router logic and the FIFOs, handling write and read enables, and providing status signals for the FIFOs.

Inputs:

- `clock` : System clock.
- `resetn` : Asynchronous reset (active low).
- `detect_add` : Detect address signal from FSM.

- `write_enb` (3-bit): Write enable signals for the three FIFOs.
- `data_in` (2-bit): Input data, possibly for address or control.
- `fifo_full` : Indicates if any FIFO is full.
- `vld_out_0` , `vld_out_1` , `vld_out_2` : Valid output signals from the FIFOs.
- `read_enb_0` , `read_enb_1` , `read_enb_2` : Read enable signals for the three FIFOs.

Outputs:

- `empty_0` , `empty_1` , `empty_2` : Indicate if respective FIFOs are empty.
- `soft_reset_0` , `soft_reset_1` , `soft_reset_2` : Soft reset signals for individual FIFOs.
- `full_0` , `full_1` , `full_2` : Indicate if respective FIFOs are full.

2.4 FIFO (FIFO_0, FIFO_1, FIFO_2)

There are three identical FIFO (First-In, First-Out) buffers, each responsible for buffering data packets before they are sent out. Each FIFO has standard FIFO interfaces for writing, reading, and status indication.

Inputs:

- `clock` : System clock.
- `resetn` : Asynchronous reset (active low).
- `write_enb[X]` : Write enable for the specific FIFO (where X is 0, 1, or 2).
- `soft_reset_X` : Soft reset for the specific FIFO.
- `read_enb_X` : Read enable for the specific FIFO.
- `data_in` (8-bit): Data to be written into the FIFO.
- `lfd_state` : Load FIFO data state (from FSM/Register).

Outputs:

- `empty_X` : Indicates if the FIFO is empty.
- `dout_out_X` (8-bit): Output data from the FIFO.
- `full_X` : Indicates if the FIFO is full.

3. Interconnections

The modules are interconnected to facilitate the routing of data. Key interconnections include:

- The FSM controls the `write_enb_reg` for the Register and provides `soft_reset_X` signals to the FIFOs and Synchronizer.

- The Register provides `parity_done`, `low_pkt_valid`, `err`, and `dout` to other modules, including the FIFOs.
- The Synchronizer receives `detect_add` from the FSM and generates `soft_reset_X` and `full_X` signals for the FSM.
- The FIFOs receive `data_in` (8-bit) from the Register and provide `dout_out_X`, `empty_X`, and `full_X` status to the Synchronizer and FSM.

Further details on the exact data paths and control logic will be derived from the provided code.

2.5 FIFO Module (`router_fifo.v`)

This module implements a 16-entry deep FIFO buffer with an 8-bit data width. It includes read and write pointers, and logic for managing full and empty conditions.

Inputs:

- `clock` : System clock.
- `resetn` : Asynchronous active-low reset.
- `write_enb` : Write enable signal. Data is written when high and FIFO is not full.
- `soft_reset` : Synchronous soft reset. Resets write and read pointers and clears FIFO contents.
- `read_enb` : Read enable signal. Data is read when high and FIFO is not empty.
- `lfd_state` : Load FIFO data state. This 1-bit signal is stored along with the 8-bit `data_in` into the FIFO, making each entry 9 bits wide internally.
- `data_in` [7:0]: 8-bit input data to be written into the FIFO.

Outputs:

- `data_out` [7:0]: 8-bit output data read from the FIFO.
- `empty` : Indicates if the FIFO is empty (high when empty).
- `full` : Indicates if the FIFO is full (high when full).

Internal Logic:

- `fifo[15:0]` : A 16-entry register array, each entry storing 9 bits (`{temp_lfd, data_in}`).
- `r_ptr`, `w_ptr` : 4-bit read and write pointers, respectively, used to index the `fifo` array.
- `count` : An 8-bit register used for an internal counter, potentially related to packet length or burst transfers, as it's updated based on `fifo[r_ptr[3:0]][8]` (the `lfd_state` bit) and `fifo[r_ptr[3:0]][7:2]` .
- `temp_lfd` : A register that stores the `lfd_state` input synchronously.

Functionality:

- **Reset:** On `resetn` low, `w_ptr` and `r_ptr` are reset to 0, and all FIFO entries are cleared. On `soft_reset` high, `w_ptr` and `r_ptr` are reset to 0, and FIFO entries are cleared.
- **Write Operation:** When `write_enb` is high and the FIFO is not `full`, the concatenation of `temp_lfd` and `data_in` is written to the location pointed by `w_ptr`, and `w_ptr` increments.
- **Read Operation:** When `read_enb` is high and the FIFO is not `empty`, `data_out` is assigned the 8-bit data from the location pointed by `r_ptr`, and `r_ptr` increments.
- **Full Condition:** `full` is asserted when `w_ptr` is 15 and `r_ptr` is 0, indicating a wrap-around full condition.
- **Empty Condition:** `empty` is asserted when `w_ptr` equals `r_ptr`.
- **Count Logic:** The `count` register's behavior suggests it might be tracking the number of valid data words or the length of a packet. If the `lfd_state` bit (MSB of the stored FIFO entry) is 1, `count` is loaded with `fifo[r_ptr[3:0]][7:2]` plus 1. Otherwise, if `count` is not zero, it decrements. This implies `lfd_state` might signal the start of a new packet, and the subsequent bits (7:2) might indicate the packet length.

2.6 FSM Module (`router_fsm.v`)

This module implements the Finite State Machine (FSM) that controls the overall operation of the 1x3 router. It manages the data flow based on packet validity, FIFO status, and parity checking.

Inputs:

- `clock` : System clock.
- `resetn` : Asynchronous active-low reset.
- `pkt_valid` : Indicates a valid incoming packet.
- `parity_done` : Indicates that parity checking is complete by the Register module.
- `soft_reset_0` , `soft_reset_1` , `soft_reset_2` : Soft reset signals for individual FIFOs, which also reset the FSM to `DECODE_ADDRESS` state.
- `fifo_full` : Indicates if any of the target FIFOs are full.
- `low_pkt_valid` : A low pulse indicating a valid packet (likely from the Register module).
- `fifo_empty_0` , `fifo_empty_1` , `fifo_empty_2` : Indicate if respective FIFOs are empty.
- `data_in` [1:0]: 2-bit input, likely representing the destination address for the packet (00, 01, or 10 for FIFO_0, FIFO_1, FIFO_2 respectively).

Outputs:

- `detect_add` : Asserted in `DECODE_ADDRESS` state to signal address detection.
- `ld_state` : Asserted in `LOAD_DATA` state to indicate data loading.
- `laf_state` : Asserted in `LOAD_AFTER_FULL` state.
- `full_state` : Asserted in `FIFO_FULL_STATE` .
- `write_enb_reg` : Write enable signal for the Register module.
- `rst_int_reg` : Reset internal register signal, asserted in `CHECK_PARITY_ERROR` state.
- `lfd_state` : Load first data state, asserted in `LOAD_FIRST_DATA` state.
- `busy` : Indicates if the FSM is in a busy state (not `DECODE_ADDRESS`).

Parameters (States):

- `DECODE_ADDRESS` (3'b000): Initial state, waiting for a valid packet and decoding its destination address.
- `LOAD_FIRST_DATA` (3'b001): Loads the first data word of a packet into the FIFO.
- `LOAD_DATA` (3'b010): Continuously loads data words into the FIFO.
- `FIFO_FULL_STATE` (3'b011): State entered when the target FIFO is full.
- `LOAD_AFTER_FULL` (3'b100): Continues loading data after the FIFO is no longer full.
- `LOAD_PARITY` (3'b101): Loads the parity byte of the packet.
- `CHECK_PARITY_ERROR` (3'b110): Checks for parity errors and resets internal registers if needed.
- `WAIT_TILL_EMPTY` (3'b111): Waits for the target FIFO to become empty before processing a new packet for that FIFO.

State Transitions:

- **Reset:** On `resetn` low or any `soft_reset_X` high, the FSM transitions to `DECODE_ADDRESS` .
- **DECODE_ADDRESS:**
  - If `pkt_valid` is high and the destination FIFO (determined by `data_in`) is empty, transitions to `LOAD_FIRST_DATA` .
  - If `pkt_valid` is high and the destination FIFO is not empty, transitions to `WAIT_TILL_EMPTY` .
  - Otherwise, remains in `DECODE_ADDRESS` .
- **LOAD_FIRST_DATA:** Always transitions to `LOAD_DATA` .
- **LOAD_DATA:**
  - If `fifo_full` is high, transitions to `FIFO_FULL_STATE` .

    - If `fifo_full` is low and `pkt_valid` is low (end of packet data), transitions to `LOAD_PARITY`.
    - Otherwise, remains in `LOAD_DATA`.
- **FIFO_FULL_STATE:**
    - If `fifo_full` becomes low, transitions to `LOAD_AFTER_FULL`.
    - Otherwise, remains in `FIFO_FULL_STATE`.
- **LOAD_AFTER_FULL:**
    - If `parity_done` is low and `low_pkt_valid` is high, transitions to `LOAD_PARITY`.
    - If `parity_done` is low and `low_pkt_valid` is low, transitions to `LOAD_DATA`.
    - If `parity_done` is high, transitions to `DECODE_ADDRESS`.
    - Otherwise, remains in `LOAD_AFTER_FULL`.
- **LOAD_PARITY:** Always transitions to `CHECK_PARITY_ERROR`.
- **CHECK_PARITY_ERROR:**
    - If `fifo_full` is high, transitions to `FIFO_FULL_STATE`.
    - Otherwise, transitions to `DECODE_ADDRESS`.
- **WAIT_TILL_EMPTY:**
    - If the selected FIFO becomes empty, transitions to `LOAD_FIRST_DATA`.
    - Otherwise, remains in `WAIT_TILL_EMPTY`.

Output Logic:
Outputs are assigned based on the current state, indicating control signals and status flags for other modules. For example, `detect_add` is active only in `DECODE_ADDRESS`, `lfd_state` in `LOAD_FIRST_DATA`, and `write_enb_reg` is active during data and parity loading states.

2.7 Register Module (`router_reg.v`)

This module acts as a data buffer and performs parity checking for incoming packets. It stores various packet-related information and generates control signals for parity status and errors.

Inputs:

- `clock`: System clock.
- `resetn`: Asynchronous active-low reset.
- `pkt_valid`: Indicates a valid incoming packet (from external source).
- `fifo_full`: Indicates if the target FIFO is full (from FSM/Synchronizer).

- `rst_int_reg` : Reset internal register signal (from FSM).
- `detect_add` : Signal to detect address (from FSM).
- `ld_state` : Load state signal (from FSM).
- `laf_state` : Look-ahead FIFO state (from FSM).
- `full_state` : Indicates a full state (from FSM).
- `lfd_state` : Load first data state (from FSM).
- `data_in` [7:0]: 8-bit input data.

Outputs:

- `parity_done` : Indicates that parity calculation is complete.
- `low_pkt_valid` : A low pulse indicating a valid packet, used for timing or control.
- `err` : Indicates a parity error.
- `dout` [7:0]: 8-bit output data, which can be header, data, or full state byte.

Internal Registers:

- `full_state_byte` [7:0]: Stores the `data_in` when `full_state` is asserted.
- `header` [7:0]: Stores the initial `data_in` when `detect_add` and `pkt_valid` are asserted and `data_in[1:0]` is not 3 (likely indicating a valid address).
- `packet_parity` [7:0]: Stores the parity byte received as part of the packet.
- `internal_parity` [7:0]: Calculates the XOR sum of all data bytes (including header) received for a packet, used for comparison with `packet_parity` .

Functionality:

- **`dout` Logic:**
  - On reset, `dout` is 0.
  - If `detect_add` and `pkt_valid` are asserted and `data_in[1:0]` is not 3, `dout` retains its current value (likely waiting for `lfd_state`).
  - If `lfd_state` is asserted, `dout` outputs the stored `header` .
  - If `ld_state` is asserted and FIFO is not full, `dout` outputs `data_in` .
  - If `ld_state` is asserted and FIFO is full, `dout` retains its current value.
  - If `laf_state` is asserted, `dout` outputs `full_state_byte` .
- **`header` Register:** Captures the first 8-bit data (`data_in`) of a new packet when an address is detected and valid.
- **`internal_parity` Calculation:** This register accumulates the XOR sum of the `header` (when `lfd_state` is high) and subsequent `data_in` (when `pkt_valid` and `ld_state` are high

and not in `full_state`). This is a running XOR sum for parity checking.

- **`packet_parity` Storage:** Stores the `data_in` when `ld_state` is high and `pkt_valid` is low, indicating the end of data and the arrival of the parity byte.
- **`parity_done` Signal:** Asserted when `ld_state` is high, FIFO is not full, and `pkt_valid` is low (end of packet data), or when `laf_state` is high, `low_pkt_valid` is high, and `parity_done` is not yet asserted.
- **`low_pkt_valid` Signal:** This signal is primarily controlled by the FSM. It is reset on `rst_int_reg` or when `ld_state` is high and `pkt_valid` is low.
- **`err` Signal:** Asserted if `packet_parity` is non-zero and `internal_parity` is zero, indicating a mismatch and thus a parity error.
- **`full_state_byte` Register:** Stores the `data_in` when the FSM enters the `full_state`, likely to be output later when the FIFO is no longer full.

2.8 Synchronizer Module (`router_sync.v`)

This module acts as an interface between the FSM/Register and the FIFOs, primarily responsible for generating the correct write enable signals for the FIFOs based on the decoded address, and managing soft resets for the FIFOs after a certain number of cycles or when they become empty.

Inputs:

- `clock` : System clock.
- `resetn` : Asynchronous active-low reset.
- `detect_add` : Signal from FSM indicating address detection.
- `write_enb_reg` : Write enable signal from FSM for the Register module.
- `read_enb_0` , `read_enb_1` , `read_enb_2` : Read enable signals for the respective FIFOs.
- `full_0` , `full_1` , `full_2` : Full status signals from the respective FIFOs.
- `empty_0` , `empty_1` , `empty_2` : Empty status signals from the respective FIFOs.
- `data_in` [1:0]: 2-bit input, representing the destination address (00, 01, or 10).

Outputs:

- `fifo_full` : A combined signal indicating if the currently selected FIFO is full.
- `soft_reset_0` , `soft_reset_1` , `soft_reset_2` : Soft reset signals for the respective FIFOs.
- `vld_out_0` , `vld_out_1` , `vld_out_2` : Valid output signals for the respective FIFOs (active low, i.e., `~empty_X`).
- `write_enb` [2:0]: 3-bit one-hot encoded write enable signal for the three FIFOs.

Internal Registers:

- `temp` [1:0]: Stores the `data_in` (address) when `detect_add` is asserted.
- `count_0` , `count_1` , `count_2` [5:0]: Counters for each FIFO, used to generate soft reset signals.

Functionality:

- **Address Storage:** The `temp` register captures the 2-bit `data_in` (address) when `detect_add` is high. This stored address is then used to direct the write enable and full status signals.
- `write_enb` **Generation:** This is a combinational logic block. If `write_enb_reg` is asserted, it generates a one-hot `write_enb` signal based on the `temp` (address) value:
  - `2'b00` (FIFO_0): `write_enb` = `3'b001`
  - `2'b01` (FIFO_1): `write_enb` = `3'b010`
  - `2'b10` (FIFO_2): `write_enb` = `3'b100`
  - `2'b11` (Invalid/Default): `write_enb` = `3'b000`
    If `write_enb_reg` is not asserted, `write_enb` is `3'b000` .
- `fifo_full` **Aggregation:** This is also a combinational logic block. It selects the `full` status of the FIFO corresponding to the `temp` (address) value and assigns it to `fifo_full` .
- `vld_out` **Generation:** These are simple assignments: `vld_out_X` is the inverse of `empty_X` .
- `soft_reset` **Generation and Counters:** Each FIFO has its own counter (`count_0` , `count_1` , `count_2`) and `soft_reset` logic. These counters increment on each clock cycle if the corresponding FIFO is not being read (`read_enb_X` is low) and is not empty (`vld_out_X` is high). If a counter reaches `5'd30` (decimal 30), the corresponding `soft_reset_X` signal is asserted for one cycle, and the counter is reset to `5'b1` . This mechanism seems to implement a timeout or periodic reset for the FIFOs if they remain in a non-empty, non-reading state for 30 cycles. If a FIFO becomes empty (`!vld_out_X`) or is being read (`read_enb_X`), its counter is reset to `5'b1` and `soft_reset_X` is deasserted. On `resetn` low, all counters are reset to `5'b1` and `soft_reset_X` are deasserted.

2.9 Top-Level Module (`router_top.v`)

The `router_top` module instantiates and connects all the sub-modules (FSM, Synchronizer, Register, and three FIFOs) to form the complete 1x3 router system. It defines the top-level inputs and outputs of the router.

Inputs:

- `clock` : System clock.

- `resetn` : Asynchronous active-low reset.
- `read_enb_0` , `read_enb_1` , `read_enb_2` : Read enable signals for the respective output FIFOs.
- `pkt_valid` : Indicates a valid incoming packet.
- `data_in` [7:0]: 8-bit input data stream.

Outputs:

- `vld_out_0` , `vld_out_1` , `vld_out_2` : Valid output signals from the respective FIFOs.
- `error` : Indicates a parity error from the Register module.
- `busy` : Indicates if the FSM is in a busy state.
- `data_out_0` , `data_out_1` , `data_out_2` [7:0]: 8-bit output data streams from the respective FIFOs.

Internal Wires:

- `write_enb` [2:0]: 3-bit one-hot encoded write enable for FIFOs (from Synchronizer).
- `parity_done` : Parity calculation complete signal (from Register).
- `soft_reset_0` , `soft_reset_1` , `soft_reset_2` : Soft reset signals for FIFOs (from Synchronizer).
- `fifo_full` : Combined FIFO full status (from Synchronizer).
- `low_pkt_valid` : Low pulse valid packet signal (from Register).
- `fifo_empty_0` , `fifo_empty_1` , `fifo_empty_2` : Empty status signals from FIFOs.
- `detect_add` , `ld_state` , `laf_state` , `full_state` , `write_enb_reg` , `rst_int_reg` , `lfd_state` : Control signals from FSM.
- `empty_0` , `empty_1` , `empty_2` : Empty status signals from FIFOs (used internally).
- `full_0` , `full_1` , `full_2` : Full status signals from FIFOs (used internally).
- `dout` [7:0]: Data output from the Register module, fed into the FIFOs.

Module Instantiations and Connections:

- **`router_fsm` (FSM):**
  - Inputs: `clock` , `resetn` , `pkt_valid` , `parity_done` , `soft_reset_0/1/2` , `fifo_full` , `low_pkt_valid` , `fifo_empty_0/1/2` , `data_in[1:0]` (for address).
  - Outputs: `detect_add` , `ld_state` , `laf_state` , `full_state` , `write_enb_reg` , `rst_int_reg` , `lfd_state` , `busy` .
- **`router_sync` (SYNC):**
  - Inputs: `clock` , `resetn` , `detect_add` , `write_enb_reg` , `read_enb_0/1/2` , `full_0/1/2` , `empty_0/1/2` (connected to `fifo_empty_0/1/2` from FIFOs), `data_in[1:0]` .

    - Outputs: `fifo_full`, `soft_reset_0/1/2`, `vld_out_0/1/2`, `write_enb`.
- **`router_reg` (REG):**
    - Inputs: `clock`, `resetn`, `pkt_valid`, `fifo_full`, `rst_int_reg`, `detect_add`, `ld_state`, `laf_state`, `full_state`, `lfd_state`, `data_in`.
    - Outputs: `parity_done`, `low_pkt_valid`, `err` (connected to top-level `error`), `dout`.
- **`router_fifo` (FIFO1, FIFO2, FIFO3):** Three instances of the `router_fifo` module.
    - Inputs: `clock`, `resetn`, `write_enb[X]` (from SYNC), `soft_reset_X` (from SYNC), `read_enb_X` (from top-level), `lfd_state` (from FSM), `data_in` (connected to `dout` from REG).
    - Outputs: `data_out_X` (connected to top-level `data_out_X`), `empty_X` (connected to `fifo_empty_X` for FSM and SYNC), `full_X` (connected to `full_X` for SYNC).

This top-level module effectively orchestrates the data flow: incoming `data_in` is processed by the Register, controlled by the FSM, and then routed to the appropriate FIFO via the Synchronizer. Data is read out from the FIFOs based on external `read_enb` signals.

3. Conclusion

This document provides a comprehensive overview of the 1x3 Router project, detailing the functionality and interconnections of its core modules: the Finite State Machine (FSM), Register, Synchronizer, and three FIFO buffers. The design effectively manages incoming data packets, routes them to the appropriate output queues, and includes mechanisms for parity checking and handling FIFO full/empty conditions. The modular approach allows for clear separation of concerns and facilitates potential future enhancements or modifications to individual components.

Further analysis of the simulation results and synthesis reports would provide deeper insights into the performance, timing, and resource utilization of this router design.