LP_TIMER IO Description:

- **S_AXI:** AXI Lite interface connected as a Microblaze peripheral. In order to access the timer value, you can simply issue a read transaction at any address offset to the base address assigned in address editor, and the timer value will appear on the RDATA line

- **lp_start:** start signal to start the timer, should be pulsed for only one clock cycle. If this signal is pulsed, the value in the timer resets to 0, and timing counters start.

- **lp_end:** end signal to end the timer, should be pulsed for only one clock cycle. If this signal is pulsed, the timer is stopped, and the S_AXI READY signals go high, indicating that there is valid timing data available to be read.

FIFO_READ_INTERFACE IO Description:

- **rst_busy:** when FIFO is being reset it takes ~60 cycles to clear the buffer and initialize FIFO, so FIFO_READ_INTERFACE needs to wait until that is done

- **fifo_full:** asserted when FIFO is full

- **num_cols (NOTE: this will be changed to num_elements):** the number of elements you want to read from memory

- **addr_offset:** the address to start reading from, needs to correspond to a memory mapped peripheral memory.
    - Example 1: if DDR is located at 0x8000_0000 in Address Editor, then to read the first 64 elements of DDR, you would set addr_offset = 0x8000_0000, and num_cols (i.e. num_elements) = 64.
    - Example 2: if tableau is located at 0x0100_0000 in DDR (and DDR is at 0x8000_000 in Address Editor), then you would set addr_offset = 0x8100_0000, num_cols = NUM COLS OF TABLEAU.
    - Example 3: if BRAM is located at 0x0ECE_0532 in Address Editor and you want to read an entire column, then set addr_offset = 0x0ECE_0532, num_cols (i.e. num_elements) = NUM ROWS OF TABLEAU

- **start:** start signal to begin reading from memory and writing results to FIFO

- **M_AXIS:** AXI Stream interface to write to FIFO

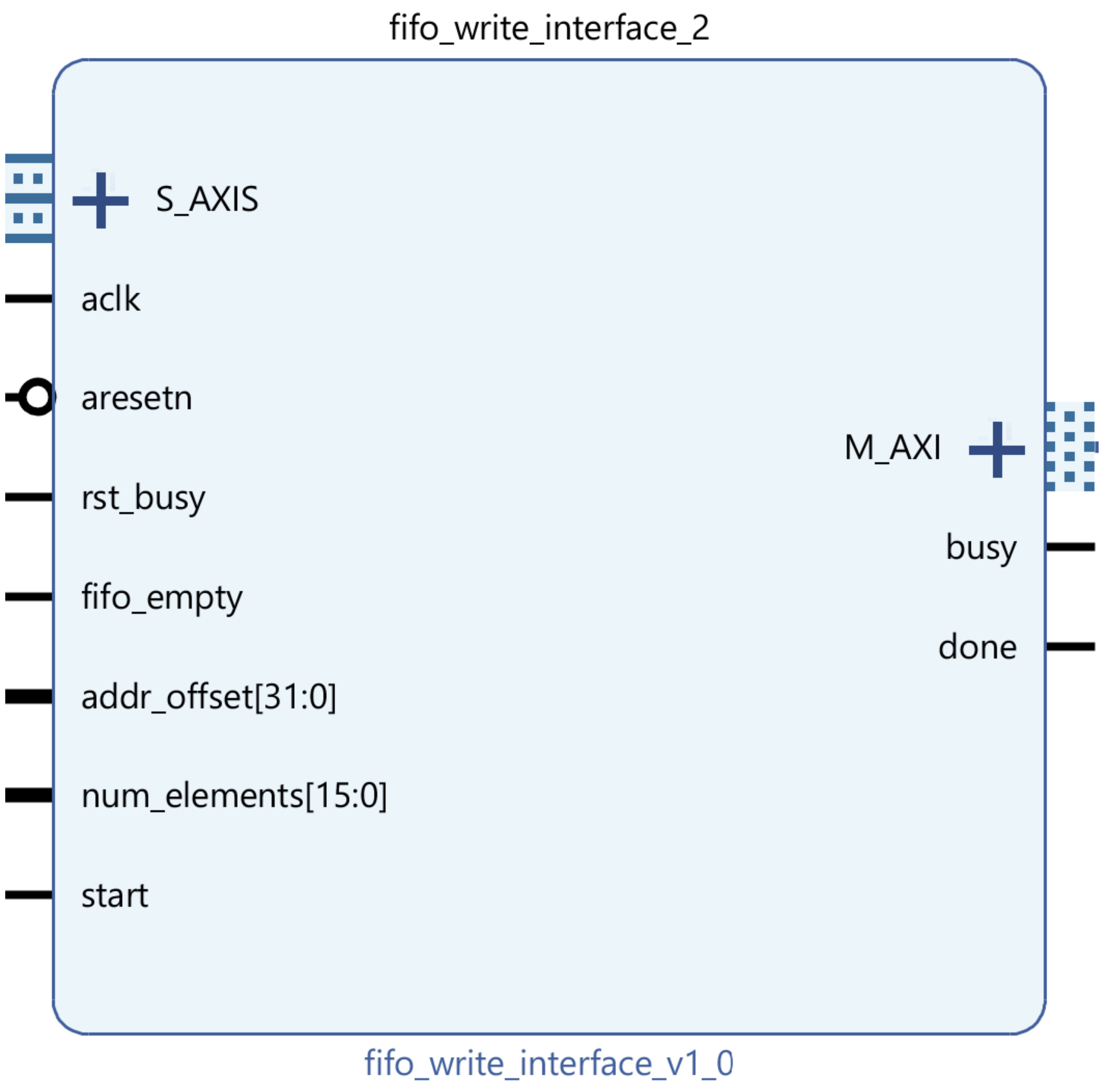
- **M_AXI:** Full AXI interface to read from memory (either DDR or BRAM)

- **done (much more akin to "busy" than done, will rename):** set to 1 if FIFO_READ_INTERFACE is chilling, set to 0 if FIFO_READ_INTERFACE if currently in the process of reading from memory and writing to FIFO

Each memory component, whether DDR or BRAM, should have a FIFO READ and FIFO WRITE Master interface to enable reading and writing. For example, for a design with 2 BRAM blocks and a DDR block:

```
                           ------------
FIFO_READ_BRAM_1     ---> |            | ---> BRAM Controller 1
FIFO_WRITE_BRAM_1    ---> |            | ---> BRAM Controller 2
FIFO_READ_BRAM_2     ---> |    AXI     | ---> DDR
FIFO_WRITE_BRAM_2    ---> |  INTRCNCT  |
FIFO_READ_DDR        ---> |            |
FIFO_WRITE_DDR       ---> |            |
                           ------------
```

FIFO_WRITE_INTERFACE IO Description:



- **Summary:** this block reads data from a FIFO and writes the data that it reads to memory.

- **rst_busy:** when FIFO is being reset it takes ~60 cycles to clear the buffer and initialize FIFO, so FIFO_READ_INTERFACE needs to wait until that is done (this rst comes from the **rd_rst_busy** signal from FIFO)

- **fifo_full:** comes from the FIFO, lets the block know if the FIFO is empty.

- **num_elements:** the number of elements you want to write to memory
    - **Note:** this is a parameterized signal. Double click on the fifo_write_interface to change it to your desired width.

- **addr_offset:** the address to start writing to, needs to correspond to a memory mapped peripheral memory.
    - Example 1: if DDR is located at 0x8000_0000 in Address Editor, then to read 64 elements from a FIFO and write them to DDR, you would set addr_offset = 0x8000_0000, and num_cols (i.e. num_elements) = 64.
    - Example 2: if tableau is located at 0x0100_0000 in DDR (and DDR is at 0x8000_000 in Address Editor), then you would set addr_offset = 0x8100_0000.
    - Example 3: if BRAM is located at 0x0ECE_0532 in Address Editor and you want to write an entire column, then set addr_offset = 0x0ECE_0532, num_elements = NUM ROWS OF TABLEAU

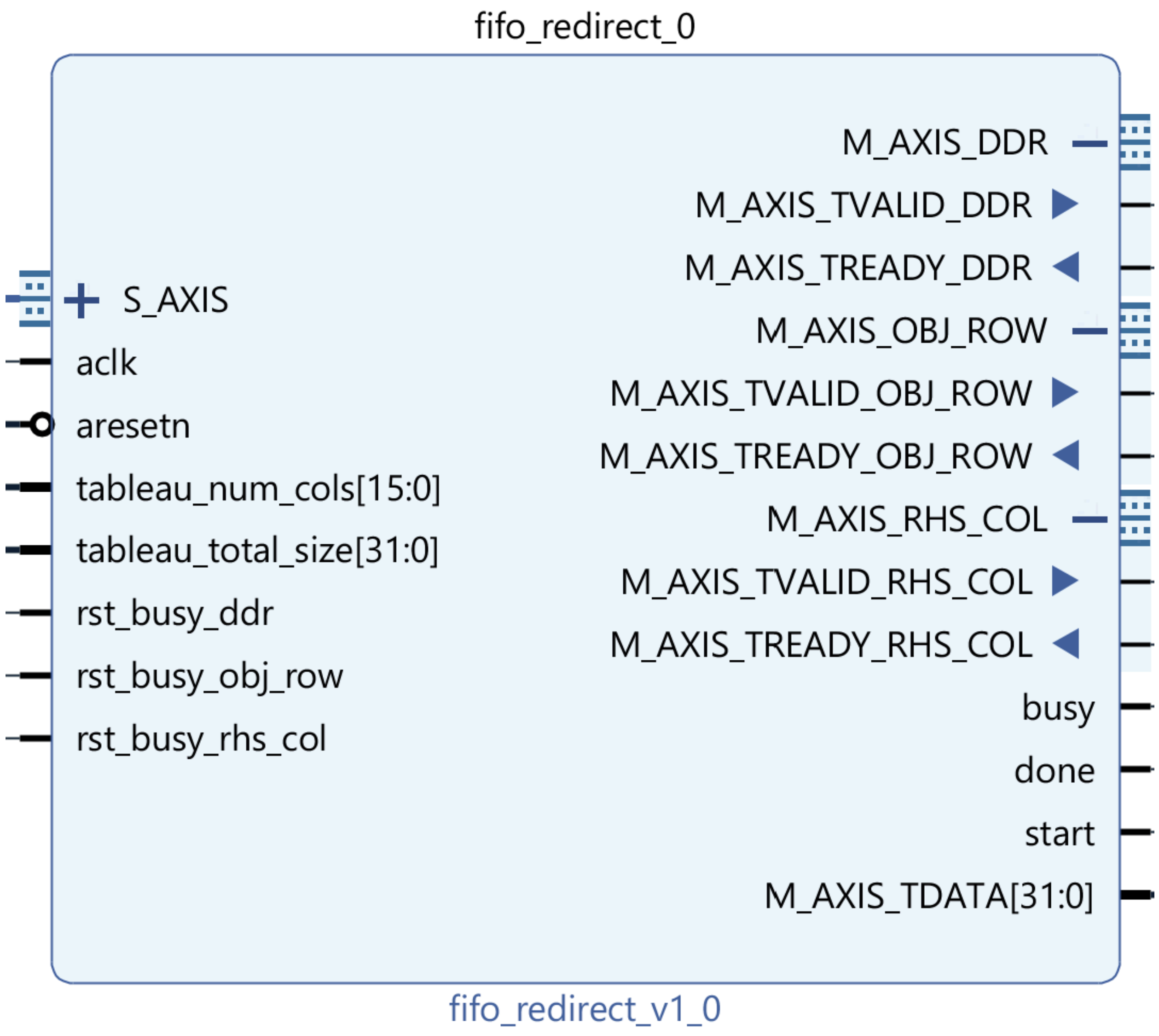
- **start:** start signal to begin reading from FIFO and writing results to memory, this signal should ONLY come from the **fifo_redirect** block

- **S_AXIS:** AXI Stream interface to read from FIFO

- **M_AXI:** Full AXI interface to write to memory (either DDR or BRAM)

- **busy:** set to 0 if FIFO_WRITE_INTERFACE is chilling, set to 1 if FIFO_WRITE_INTERFACE if currently in the process of reading from FIFO and writing to memory

FIFO_REDIRECT IO Description:



- **Summary:** this block takes data from the final update LP module, and redirects it to a FIFO based on its location in the tableau (i.e. if element is in objective row, it should be written to objective row BRAM through a FIFO. If element is in RHS column, it should be written to RHS column BRAM through a FIFO. As well, every single element should be written back to DDR through a FIFO).

- **S_AXIS:** the AXI stream signal coming from the update_lp block

- **tableau_num_cols / total_size num_rows:** [Constant] the number of cols and the total size of the tableau, should be provided by the mblaze_lp_bridge

- **rst_busy_$$$:** see FIFO_read_interface and FIFO_write_interface. These signals should come from the **wr_rst_busy** ports from each FIFO respectively. (i.e. the rst_busy_rhs_col signal should be connected to the wr_rst_busy port of the FIFO that is connected to the FIFO_write_interface that writes to the RHS Col BRAM) [See block diagram]

- **Important:** all writeback FIFOs share the same TDATA signal, and the "steering" is performed solely by the fact that each FIFO has its own TVALID and TREADY signal. For example, let's say there is an element I want to write to both DDR and RHS COL BRAM, then I would assert M_AXIS_TVALID_DDR and M_AXIS_TVALID_RHS_COL. This effectively saves us having to instantiate 3 full-on AXIS interfaces, which should help with routing.

- **M_AXIS_TDATA:** the data going to the FIFO, and is shared by all three writeback FIFOs.

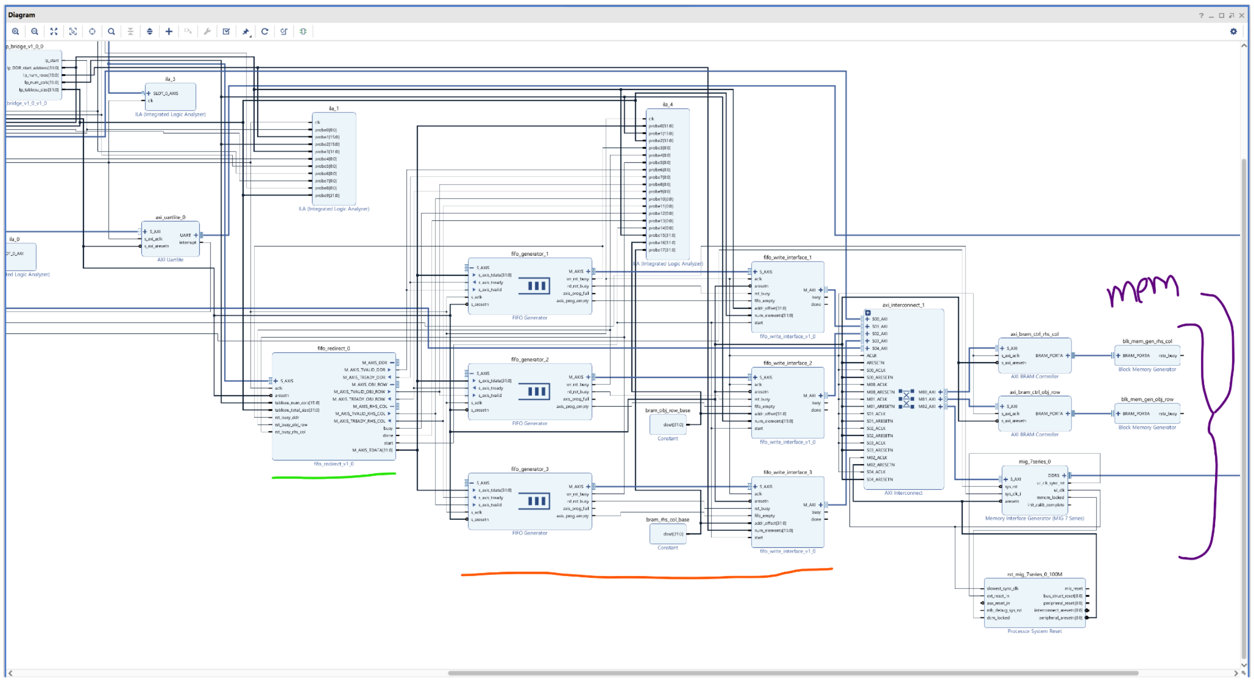
- **M_AXIS_TVALID/TREADY_$$$$:** the "steering" control signals for each individual FIFO. If I want to write to FIFO XYZ, I assert M_AXIS_TVALID_XYZ. I can write to multiple FIFOs simultaneously this way.

- **busy:** this signal is set to 0 when the redirect module is not currently steering the write operation of the entire tableau back to DDR (and steering the required elements to BRAM simultaneously), otherwise it is set to 1.

- **done:** this signal is asserted for a single clock cycle once the steering is complete and all elements go to where they need to go.

- **start [IMPORTANT!!!]:** this signal should only be going to all **fifo_write_interface** blocks. It is a handshaking signal that synchronizes when to start reading from FIFOs and writing to memory (see block diagram below).

Connectivity Diagram of FIFO_WRITE_INTERFACE and FIFO_REDIRECT



Green = FIFO_redirect, orange = FIFO_write + the actual FIFOs, purple = memory (RHS COL BRAM, OBJ ROW BRAM, DDR)

Much higher quality image link: https://ibb.co/znNcN8c