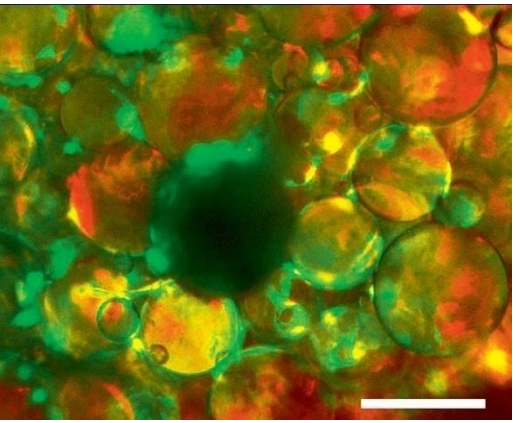


PacketFilter: A high-speed ethernet frame filter and switch

Embedded Systems
Spring 2025

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Motivation

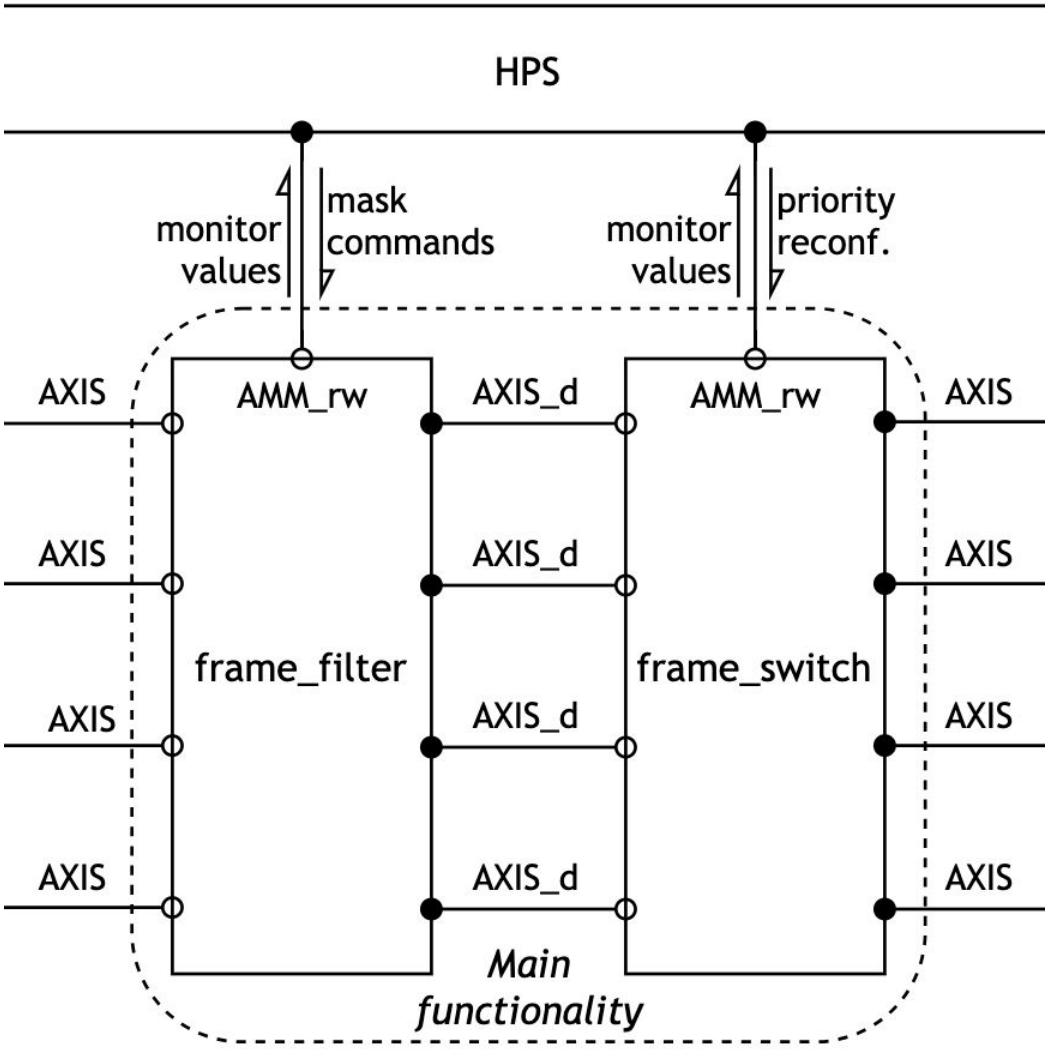
- Lightweight filtering close to the sensor
  - Quickly reduce downstream data volume
- Reconfiguration
  - Mask inputs/outputs
  - Priority
  - Base on monitor
- Global interface has minimal sideband

Solution

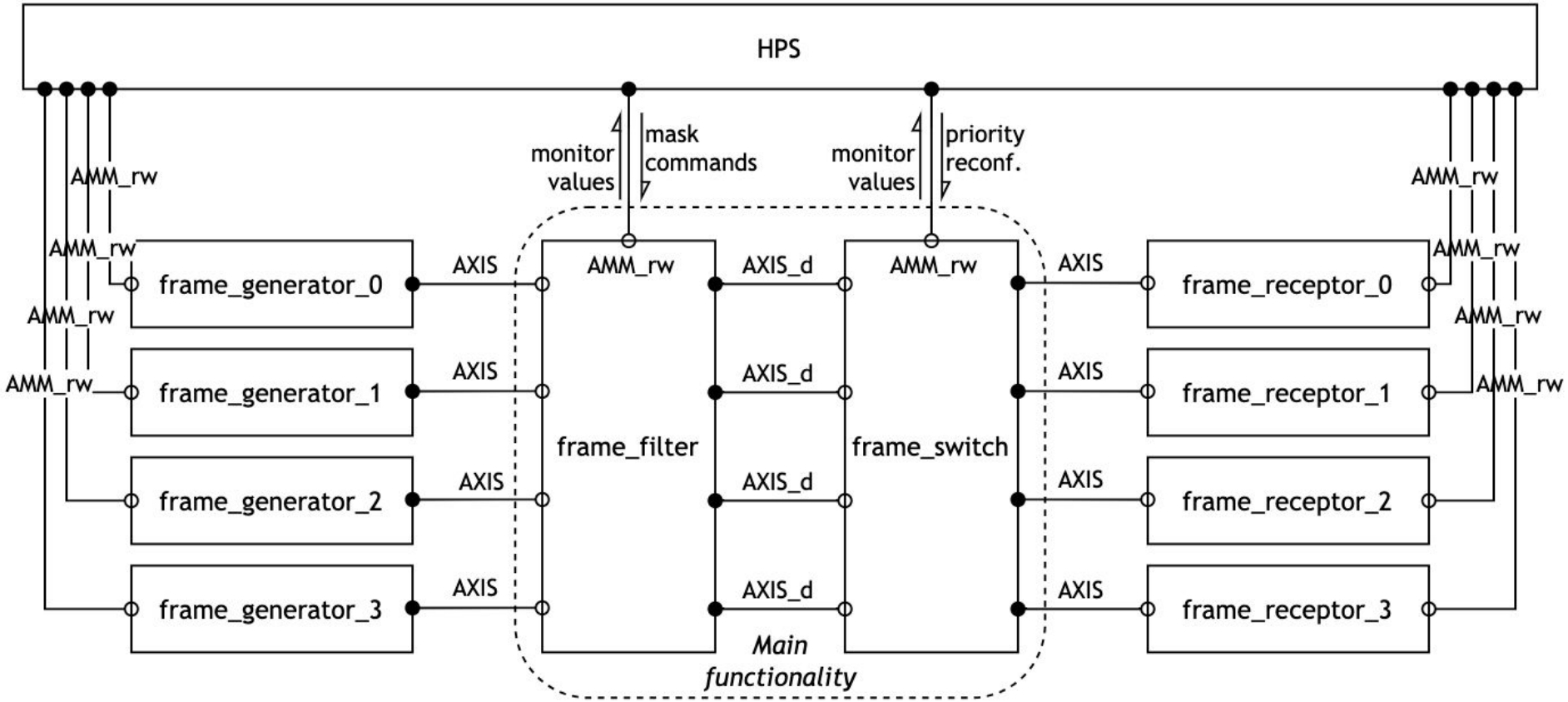
Primary function split in HW

Filter: drop and generate sideband

Switch: schedule frames to output



Total Contribution



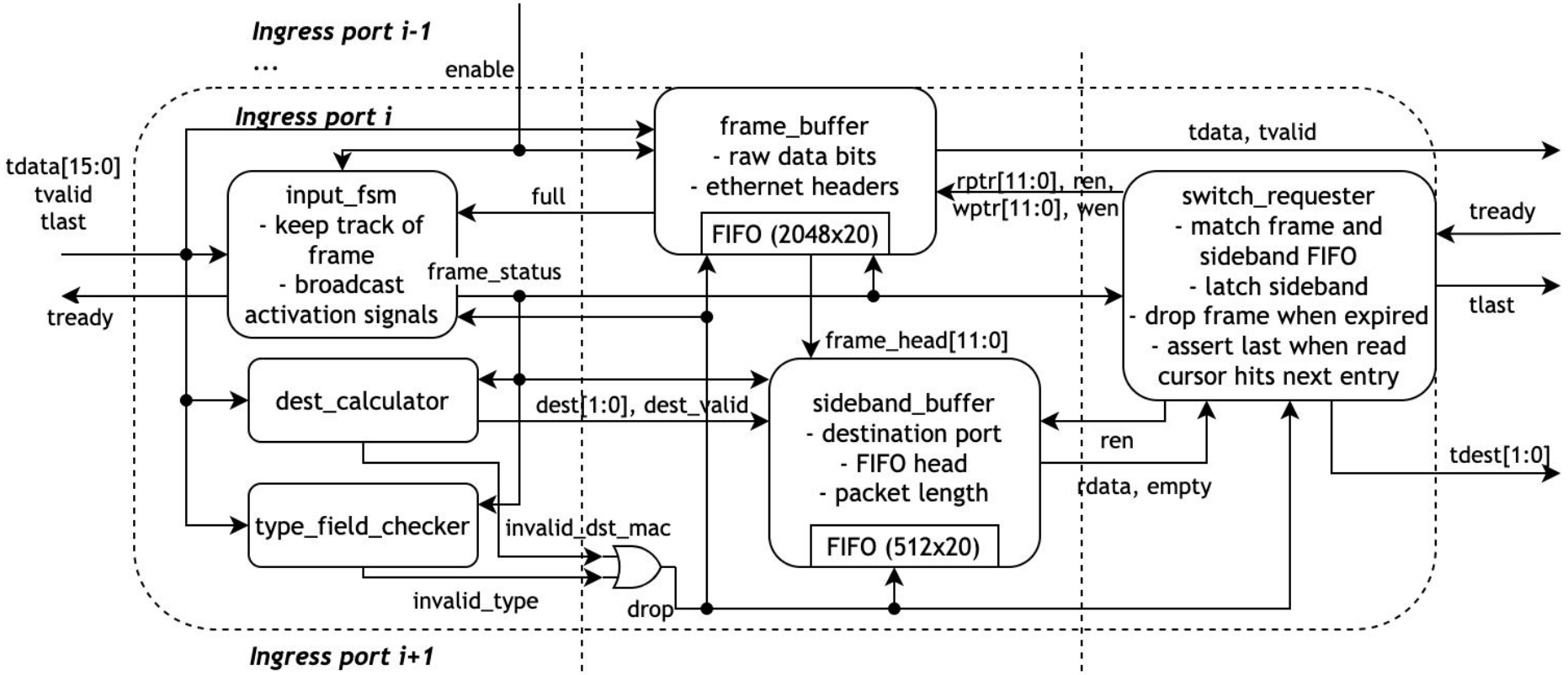
Base Addresses

Peripherals	Base Address
packet_filter_0	0x0000
packet_switch_0	0x1000
frame_generator_0	0x2000
frame_generator_1	0x2400
frame_generator_2	0x2800
frame_generator_3	0x2c00
frame_receptor_0	0x3000
frame_receptor_1	0x3400
frame_receptor_2	0x3800
frame_receptor_3	0x3c00

Constraints

- Global AXIS interface has minimal signals
  - Valid, data, last, ready
  - Sideband is internal
- Fixed-weight round-robin
- Low memory footprint

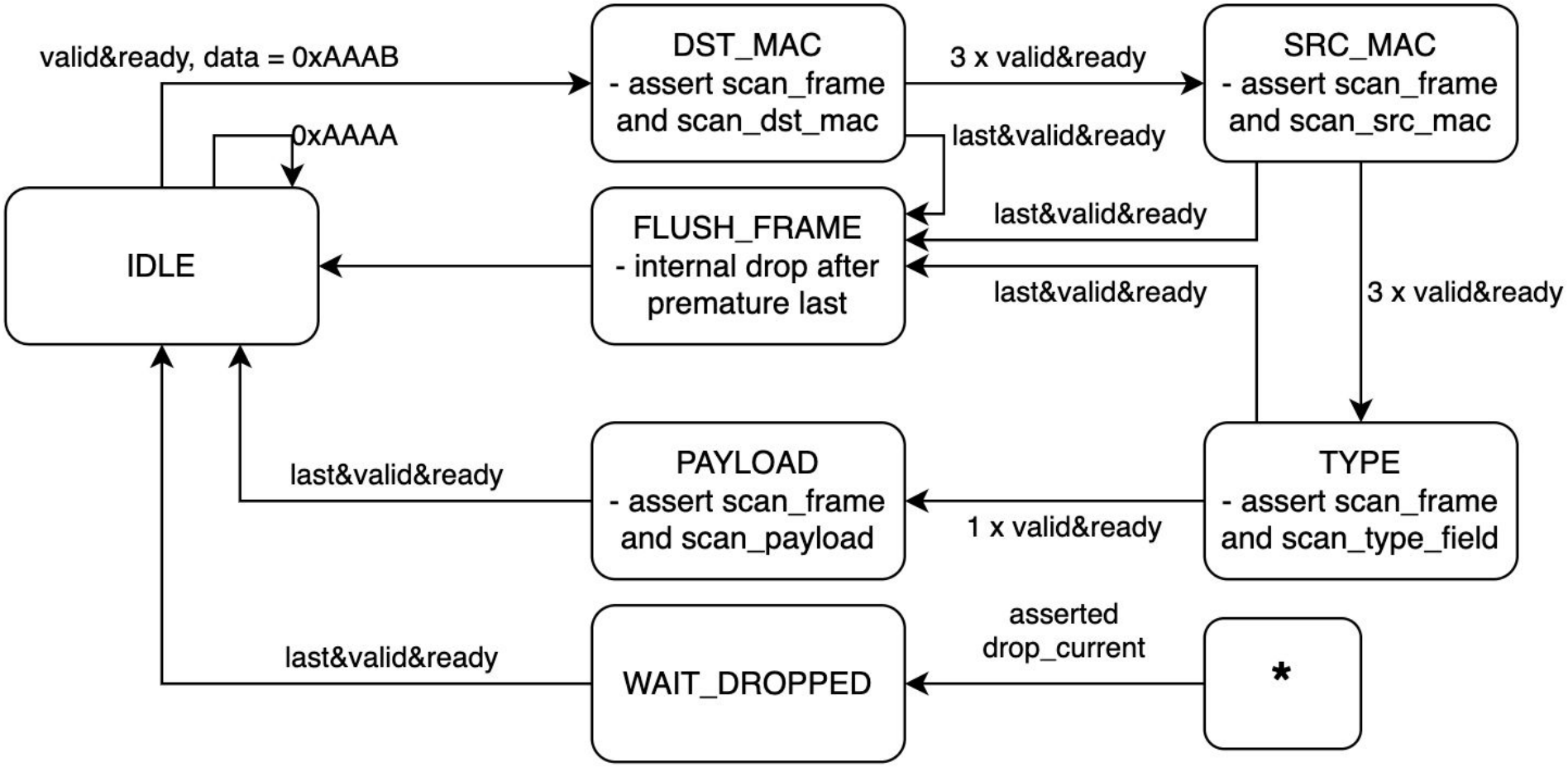
Filter


Ingress port i-1
...
enable
Ingress port i
tdata[15:0]
tvalid
tlast
tready
input_fsm
- keep track of frame
- broadcast activation signals
full
frame_status
frame_buffer
- raw data bits
- ethernet headers
FIFO (2048x20)
tdata, tvalid
rptr[11:0], ren,
wptr[11:0], wen
switch_requester
- match frame and sideband FIFO
- latch sideband
- drop frame when expired
- assert last when read cursor hits next entry
tready
tlast
tdest[1:0]
frame_head[11:0]
dest_calculator
dest[1:0], dest_valid
sideband_buffer
- destination port
- FIFO head
- packet length
FIFO (512x20)
ren
tdata, empty
type_field_checker
invalid_dst_mac
invalid_type
drop
Ingress port i+1

Filter processing

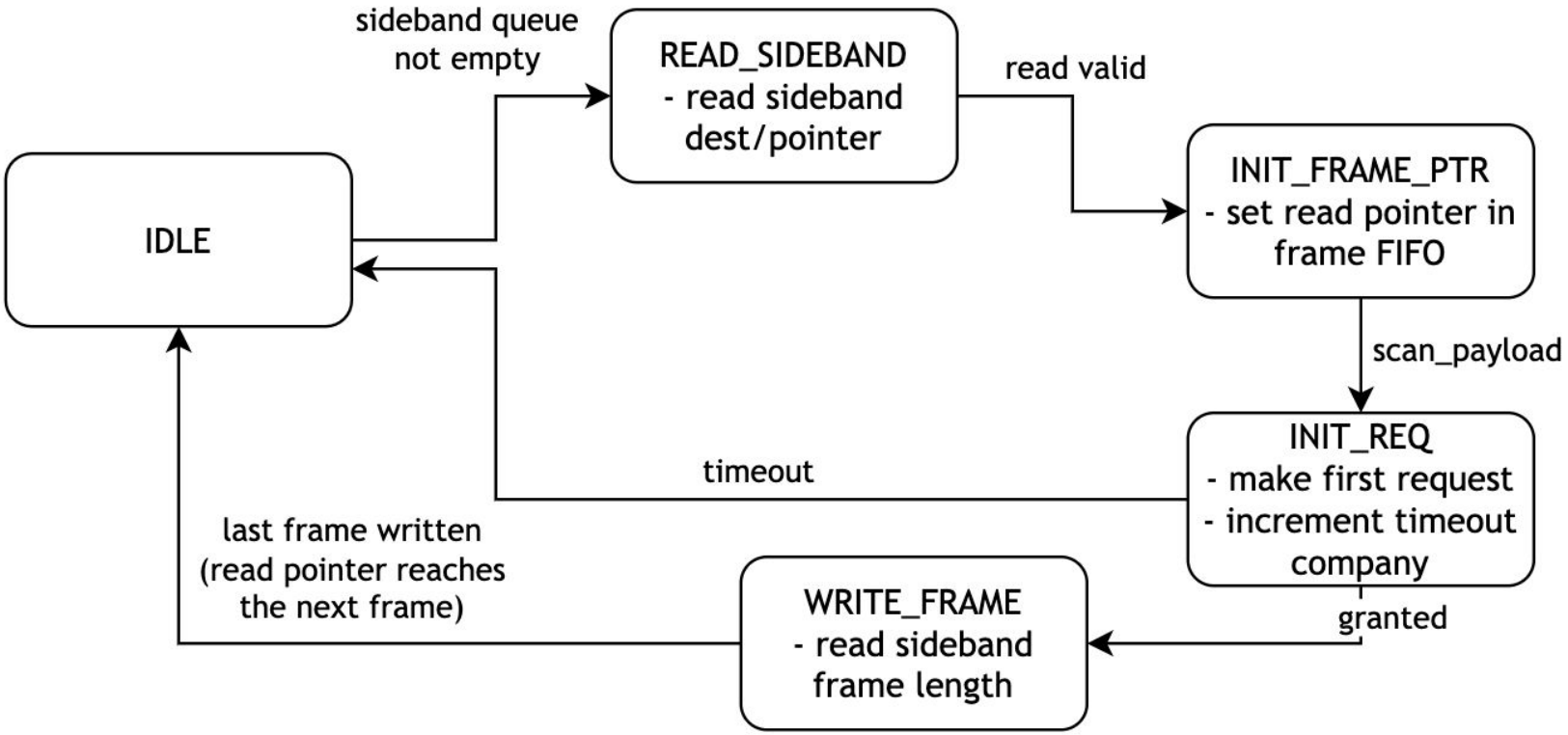
- **Modularity**: can add complexity to destination computation (standalone)
- **Configurability**: ingress masks, exposed CSRs for statistics
- Generates internal sideband
  - Destination
  - Start pointer in buffer

Filter processing

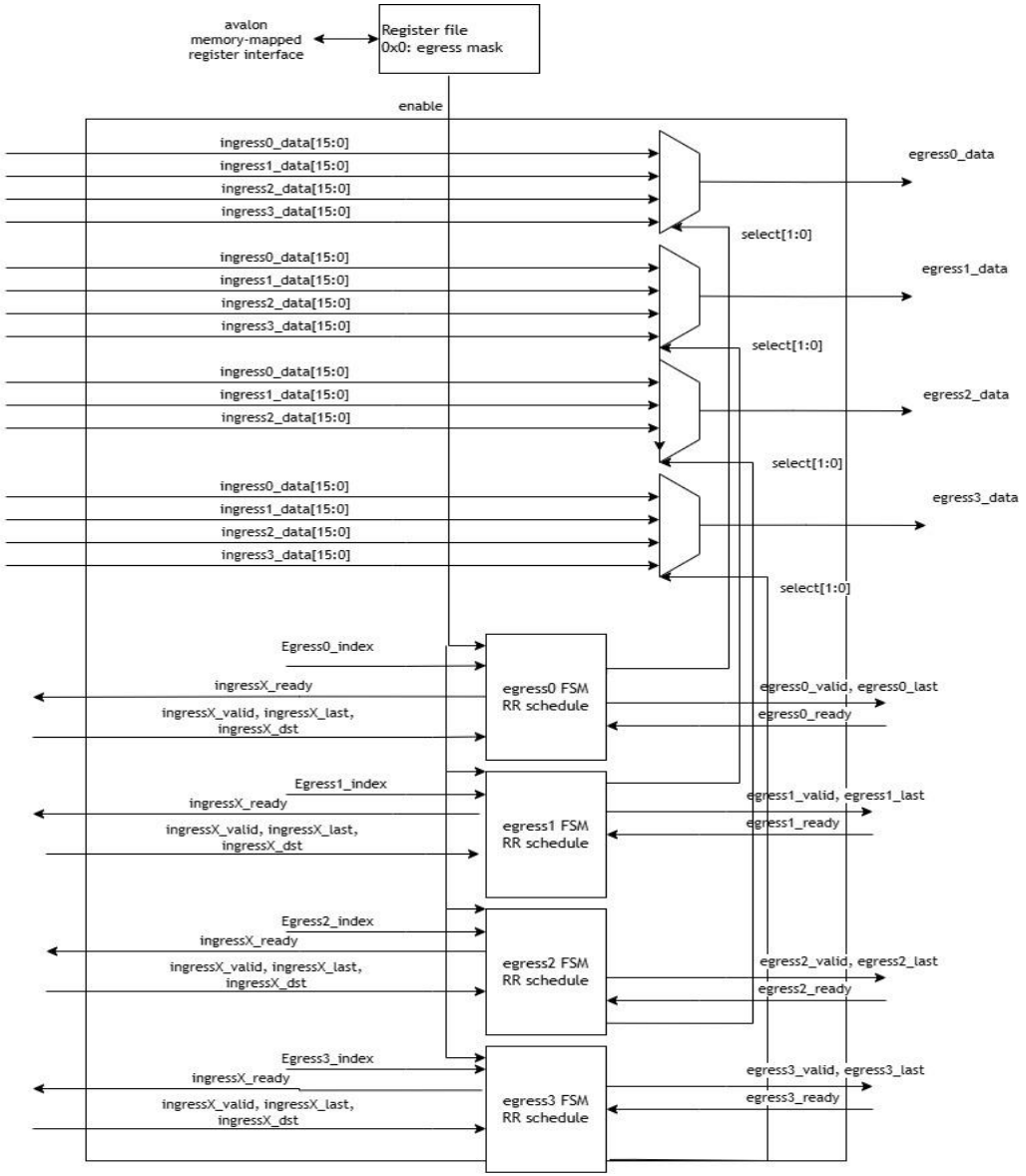


Filter buffering

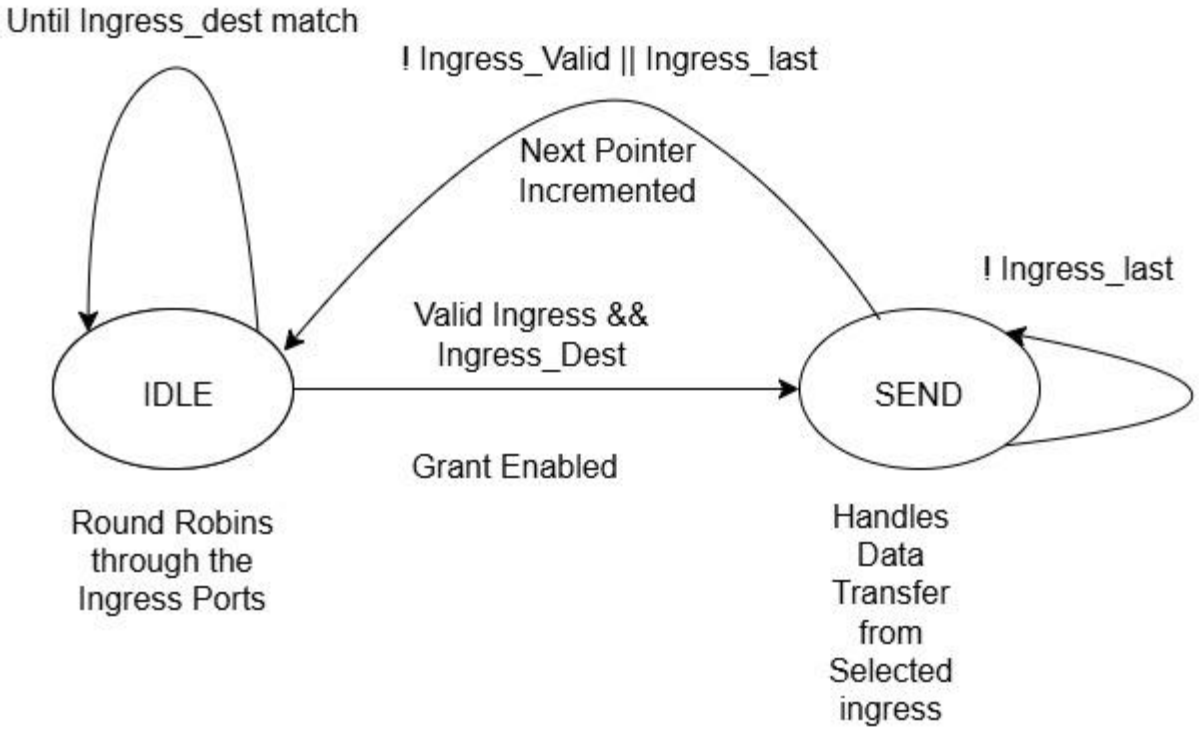
- Controllable FIFO cursors



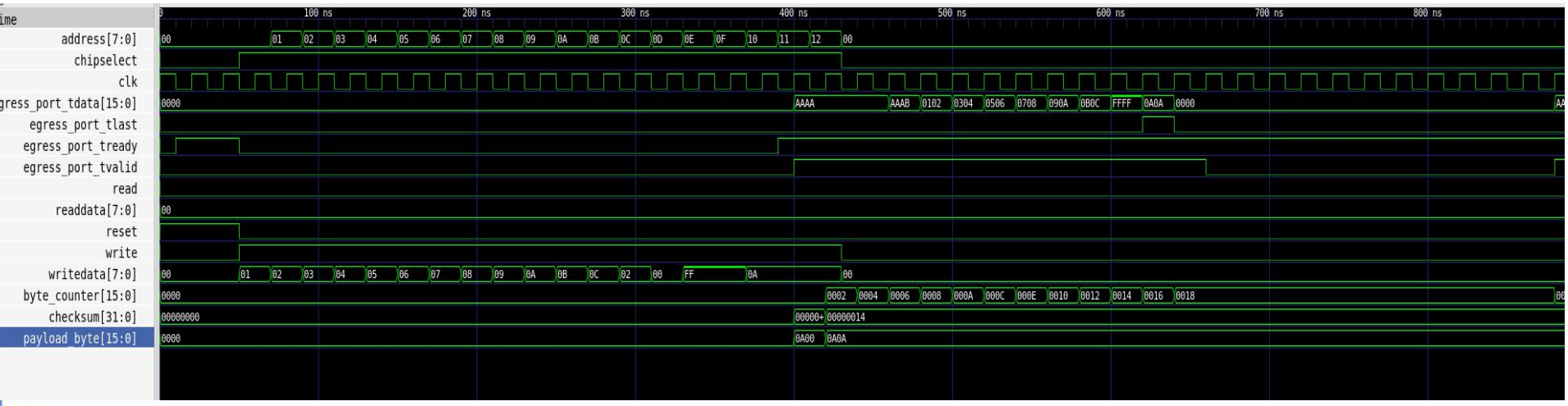
Switch


avalon
memory-mapped
register interface
Register file
0x0: egress mask
enable
ingress0_data[15:0]
ingress1_data[15:0]
ingress2_data[15:0]
ingress3_data[15:0]
egress0_data
select[1:0]
egress1_data
egress2_data
egress3_data
Egress0_index
ingressX_ready
ingressX_valid, ingressX_last,
ingressX_dst
egress0 FSM
RR schedule
egress0_valid, egress0_last
egress0_ready
Egress1_index
egress1 FSM
RR schedule
egress1_valid, egress1_last
egress1_ready
Egress2_index
egress2 FSM
RR schedule
egress2_valid, egress2_last
egress2_ready
Egress3_index
egress3 FSM
RR schedule
egress3_valid, egress3_last
egress3_ready

Switch FSM and Arbitration

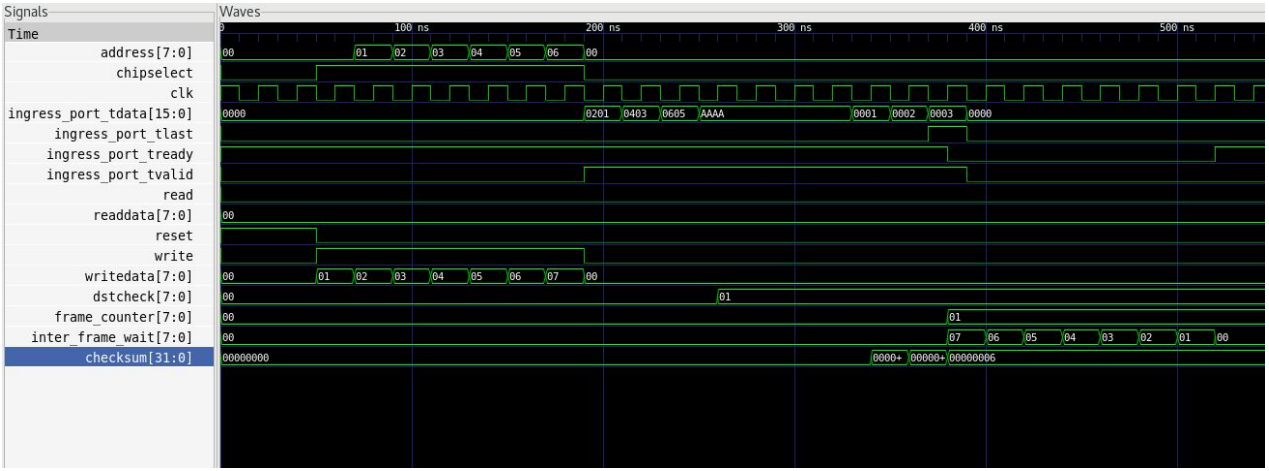


Frame Generator




```
/**
 * Register mapping
 *
 * Byte / mode | Name             | Meaning
 *         0W  |  Destination MAC |  Destination MAC byte 0.
 *         1W  |  Destination MAC |  Destination MAC byte 1.
 *         2W  |  Destination MAC |  Destination MAC byte 2.
 *         3W  |  Destination MAC |  Destination MAC byte 3.
 *         4W  |  Destination MAC |  Destination MAC byte 4.
 *         5W  |  Destination MAC |  Destination MAC byte 5.
 *         6W  |       Source MAC |       Source MAC byte 0.
 *         7W  |       Source MAC |       Source MAC byte 1.
 *         8W  |       Source MAC |       Source MAC byte 2.
 *         9W  |       Source MAC |       Source MAC byte 3.
 *        10W  |       Source MAC |       Source MAC byte 4.
 *        11W  |       Source MAC |       Source MAC byte 5.
 *        12W  |    Payload length |    Payload length byte 0.
 *        13W  |    Payload length |    Payload length byte 1.
 *        14W  |        Type field |        Type field byte 0.
 *        15W  |        Type field |        Type field byte 1.
 *        16W  | Inter-frame wait |  Cycles to wait between frames.
 *        17R  |         Checksum | Payload checksum byte 0.
 *        18R  |         Checksum | Payload checksum byte 1.
 *        19R  |         Checksum | Payload checksum byte 2.
 *        20R  |         Checksum | Payload checksum byte 3.
 */
```

Frame Receptor



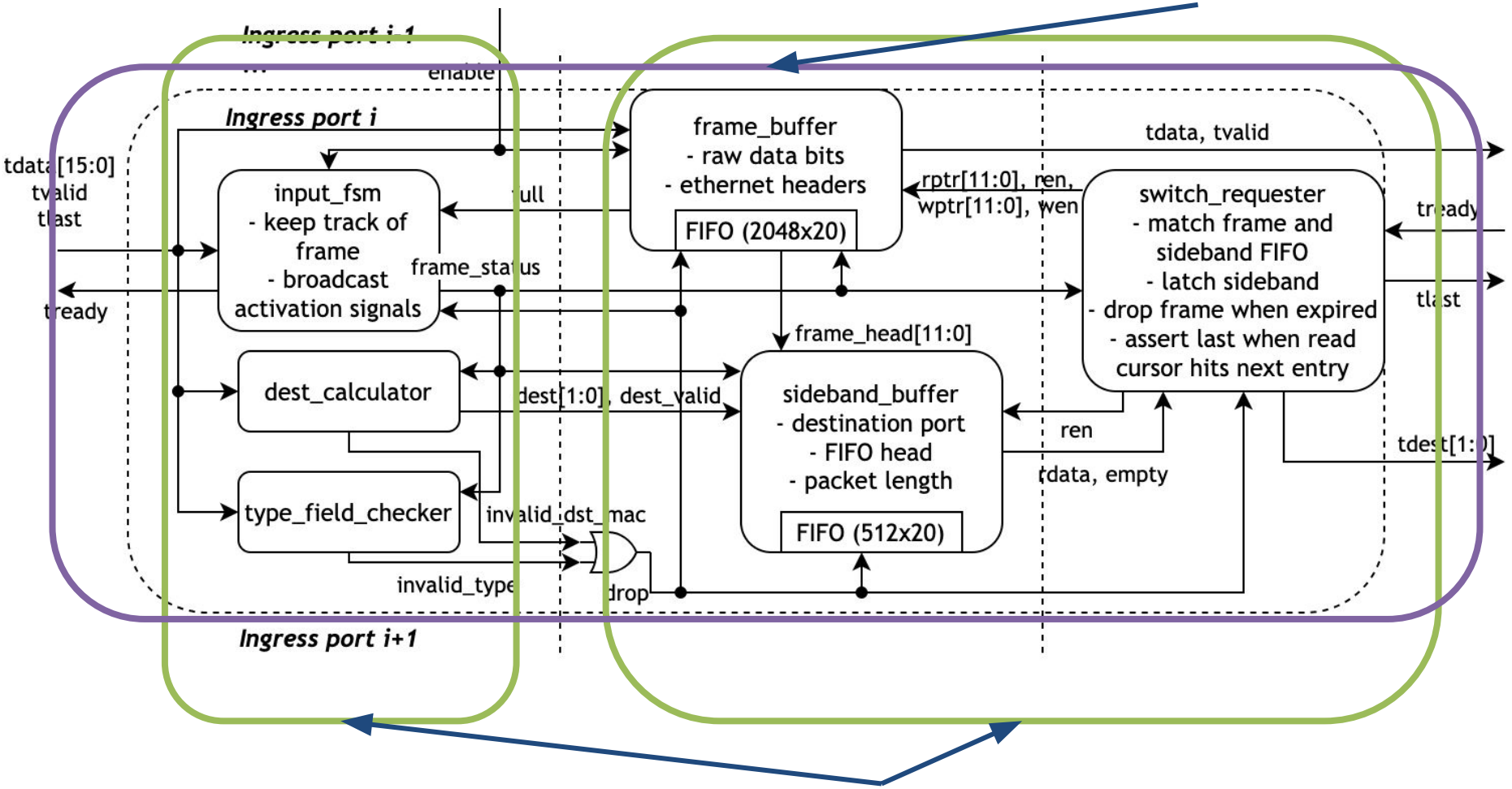

```
/**
 * Register mapping
 *
 * Byte / mode | Name             | Meaning
 *         0W  |  Destination MAC |  Destination MAC byte 0.
 *         1W  |  Destination MAC |  Destination MAC byte 1.
 *         2W  |  Destination MAC |  Destination MAC byte 2.
 *         3W  |  Destination MAC |  Destination MAC byte 3.
 *         4W  |  Destination MAC |  Destination MAC byte 4.
 *         5W  |  Destination MAC |  Destination MAC byte 5.
 *         6W  | Inter-frame wait |  Cycles to wait between frames.
  *        7R  |          dstcheck |  Destination check.
 *         8R  |          Checksum |  Payload checksum byte 0.
 *         9R  |          Checksum |  Payload checksum byte 1.
 *         10R |          Checksum |  Payload checksum byte 2.
 *         11R |          Checksum |  Payload checksum byte 3.
 */
```

Results - Sim. Verification

- Assertion-based and sequences through Verilator
  - Assertions enforce interface
  - Sequences abstract away AXIS

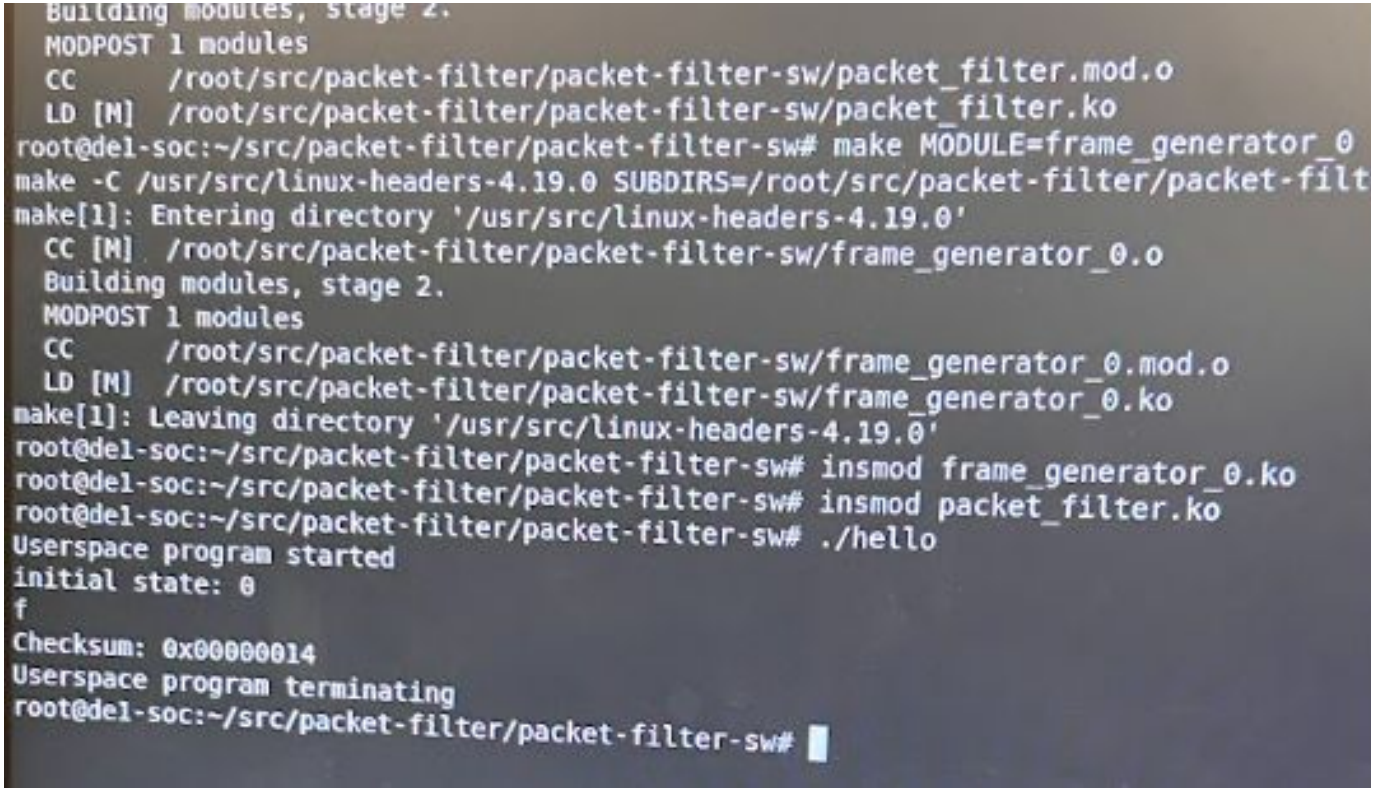
Results - Sim. Verification

- Verification structure



Results - FPGA Testbench

1. Sending packets to frame_generator, read checksum from frame_generator and frame_receptor



Results - Throughput

- Fmax

Fmax	Restricted Fmax	Clock Name
76.62MHz	76.62 MHz	clock_50_1
1237.62MHz	717.36MHz	soc_system:soc_system0

- Maximum output BW (w/o backpressure)

16 bits/cycle * 76.62 MHz

= 1225 Gbps = 153 MBps

Results - Throughput

- Monitor values (from simulation)

Ingress port	Ingress packets	Transferred packets	Ingress frames	Transferred frames
0	12450	11550	100	100
1	9050	7350	100	100
2	17150	13550	100	100
3	14050	12750	100	100

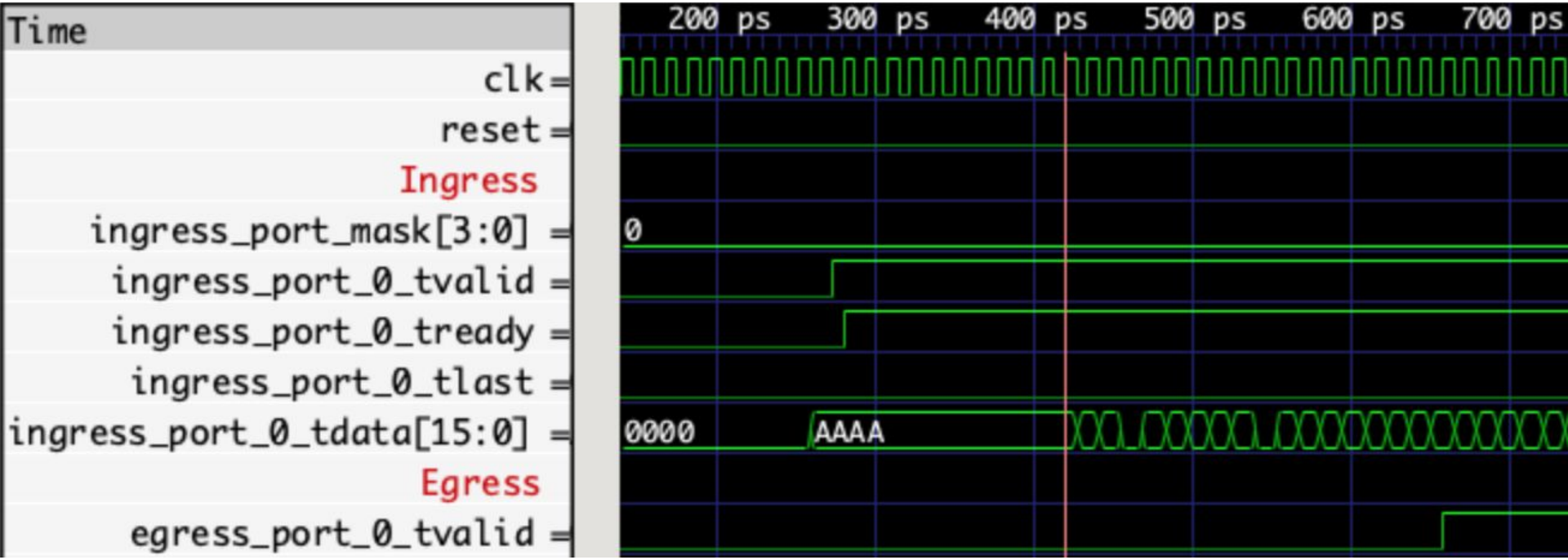
- Also verified packet drop

Request is high from 658ns to 7842ns (timeout of 512 cycles)

Results - Filter Response Time

- Waveform from simulation
  - SFD to first egress request: 420ns to 658ns

=> 238ns, or 17 cycles



* testbench in picoseconds

Results - Resource Usage

- Memory estimate from design document: 20/397 blocks (5%)

Total block Memory bits	159,744 / 4,065,280 (4%)
Total Pins	362 / 457 (79%)
Logic Utilization (in ALMs)	4,118 / 32,070 (13%)
Total DLLs	1 / 4 (25%)
Total Registers	5556
Total RAM Blocks	20/397 (5%)

Future work

- More flexibility with filtering
- Complete integration: bring results for the filter to the top-level
- FPGA-based testbench