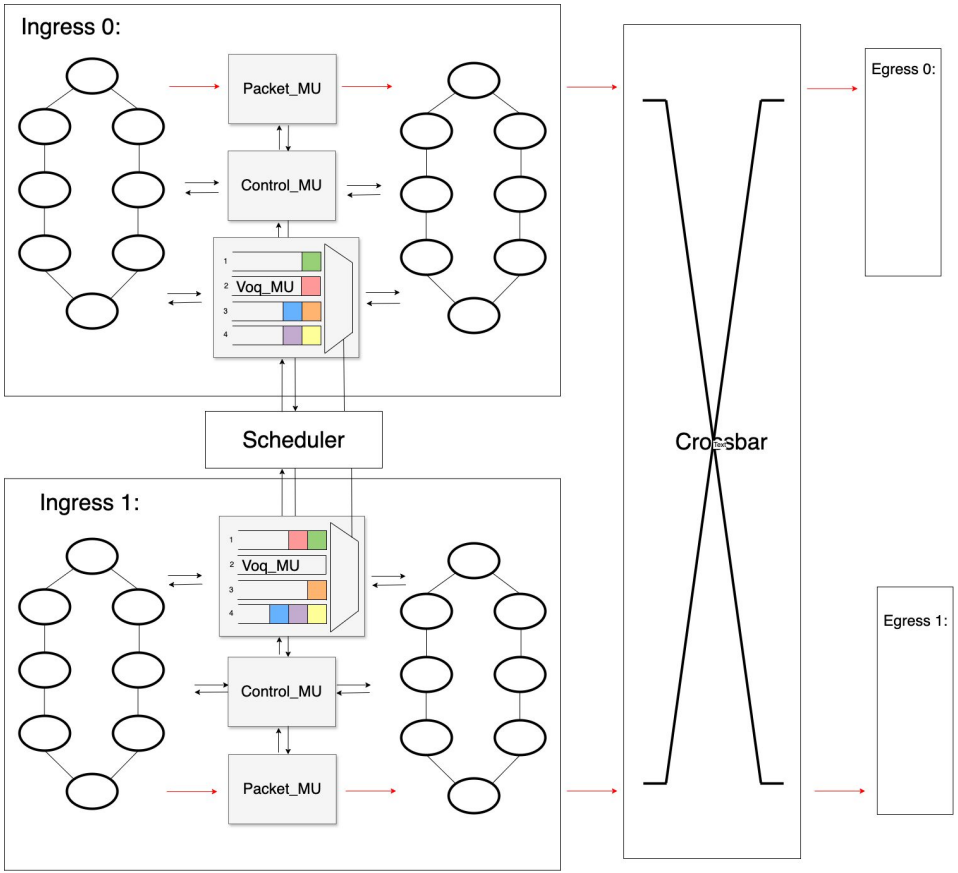
# daFPGASwitch

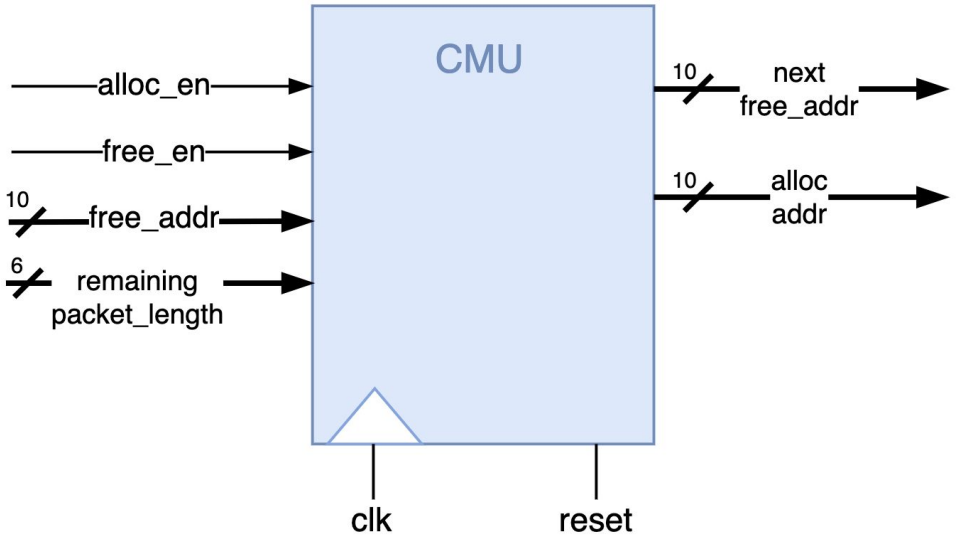
or Simple Switch?

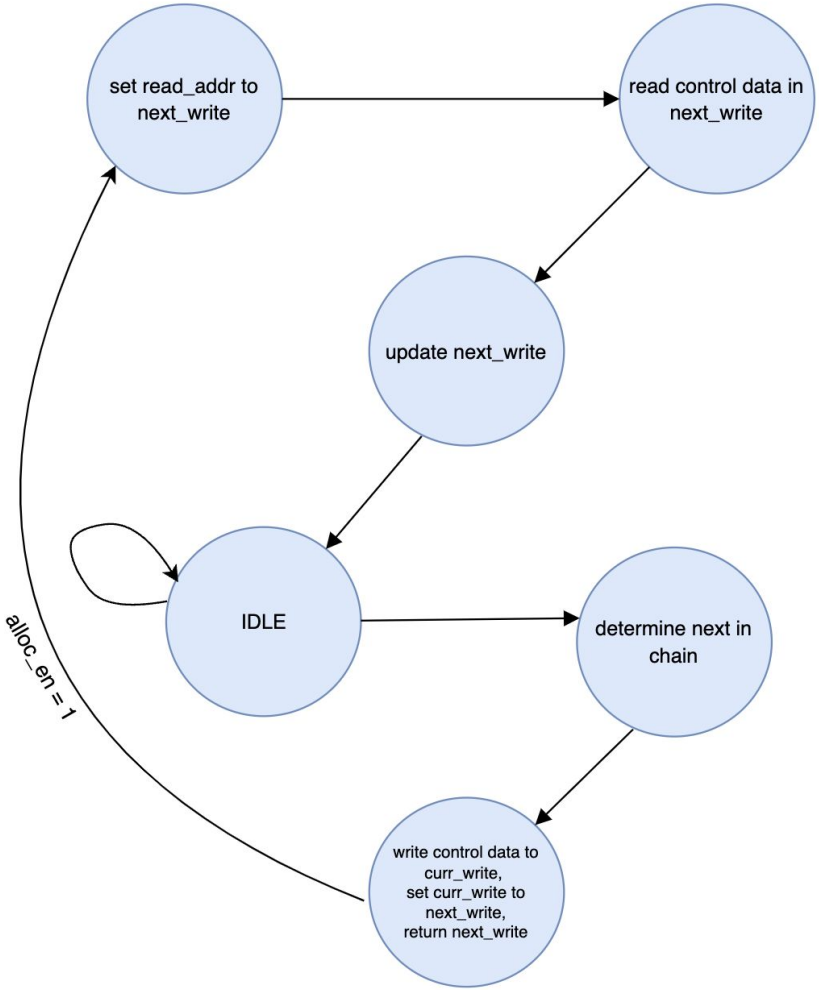
# daFPGASwitch "Original Design"



# Central Memory Unit (CMU)

(Demo with code)




set read_addr to next_write
read control data in next_write
update next_write
IDLE
determine next in chain
write control data to curr_write, set curr_write to next_write, return next_write
alloc_en = 1

# Scheduling algorithm: Doubly Round Robin

- Targeting for fairness.
- By average, each ingress gets a equal change to pick first. (First RR)
- By average, each virtual queue of ingress gets a equal change to be picked first. (Second RR)
- Takes 5 cycles (Falls within the 8 cycle heartbeat)
  - One cycle for continuing unfinished packet
  - One cycle for each ingress, deciding with one to pick (decided by all_comb module pick_voq)

# MetaData

| Dest Port (2 bits) | Src Port (2 bits) | Length (6 bits) | Start Time (11 bits) | End Time (11 bits) |
|---|---|---|---|---|

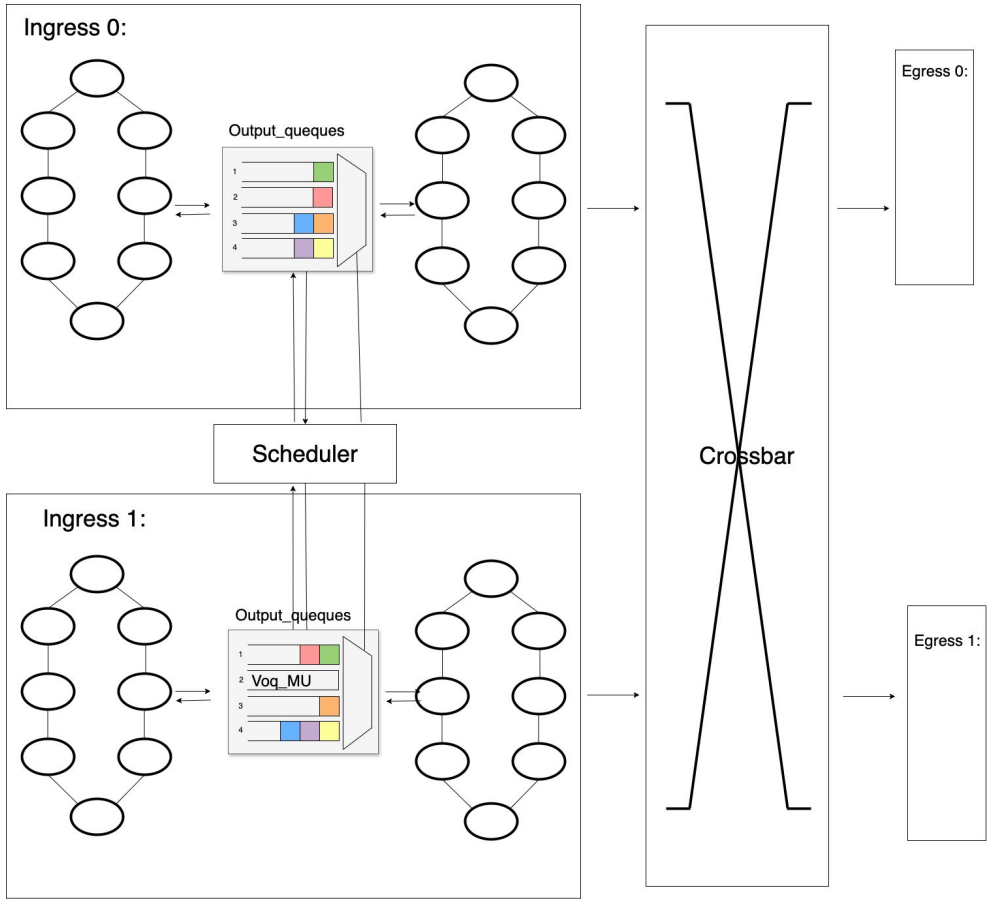
# Packet Data

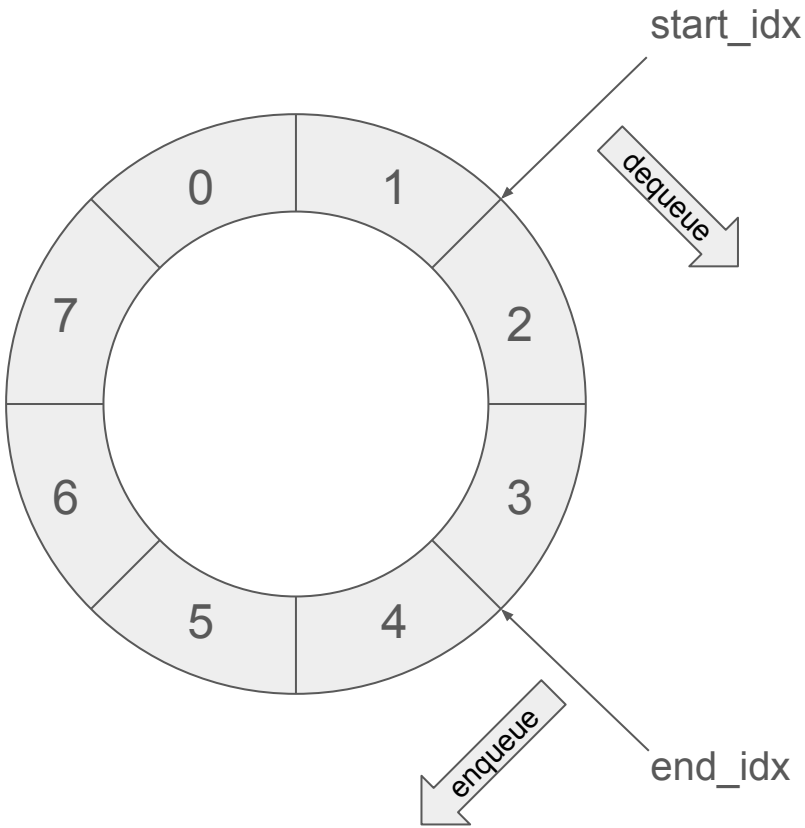
| Length (2 bytes) | Dest MAC (6 bytes) | Start Time (4 bytes) | End Time (4 bytes) | Src MAC (6 + 2 empty bytes) | Data Payload Variable length |
|---|---|---|---|---|---|

# Simple Switch "Updated Design"

Simple Switch is the version loaded on the FPGA.

But daFPGASwitch has more interesting features we would like to present.



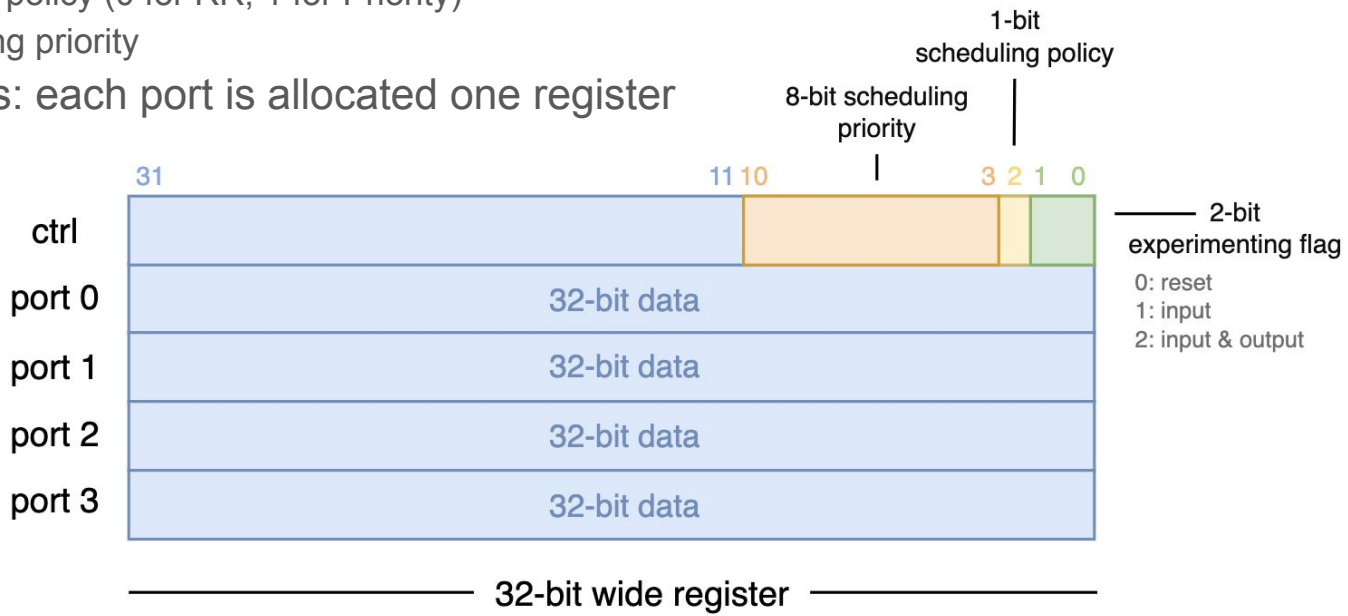

start_idx
dequeue
0
1
2
3
4
5
6
7
enqueue
end_idx

# Scheduling algorithm: Doubly Round Robin

- How to be biased -> Give some ingress/egress more cycles
- We have a list that sets the priority of each egress. When each ingress considers its scheduling decision, it will proritize to send the packets with higher priorities.
- (Demo the code)

# Register Allocation

- One control register
  - Lower 2 bits: 0 means reset, 1 means taking input but do not output, 2 means input while output
  - Third bit: sched policy (0 for RR, 1 for Priority)
  - 4~11: Scheduling priority
- Four port registers: each port is allocated one register for packet data



# How SW-HW talks

Software "polls" with ioread32, which generates a high "read" signal for the Avalon slave. For HW, the read signal is like an ack signal ("read" means that sw has already consumed the packet segment).

Software "interrupts" with iowrite32, which generates a high "write" signal for the Avalon slave. For HW, the write signal is like an enable signal ("write" means that sw has already put the packet_data on the writedata wire.)

a picture of the Avalon slave here.

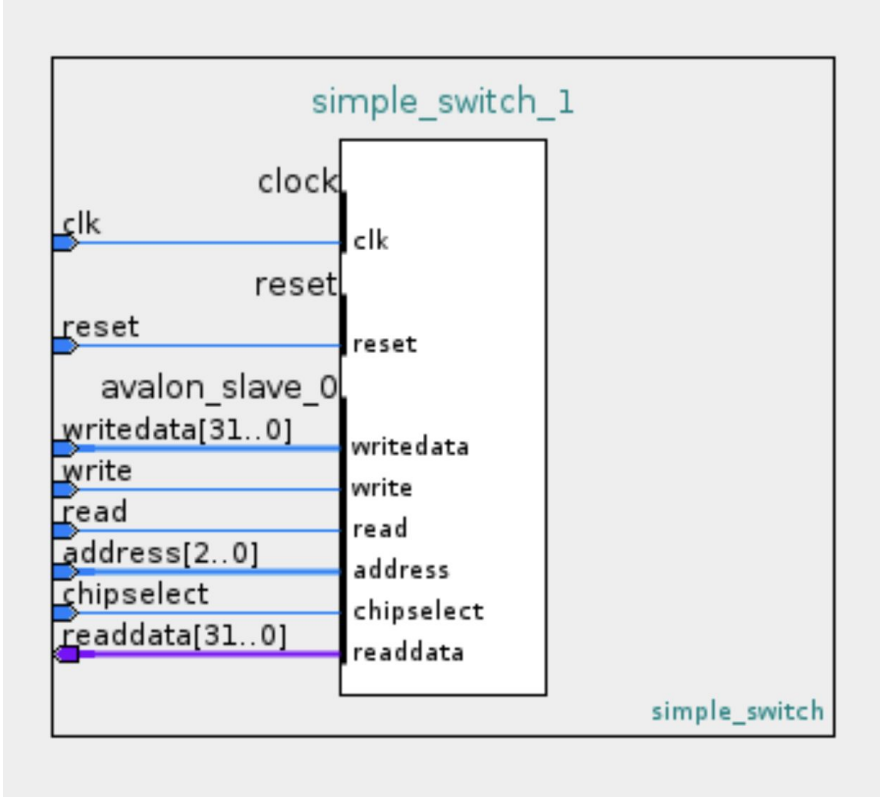
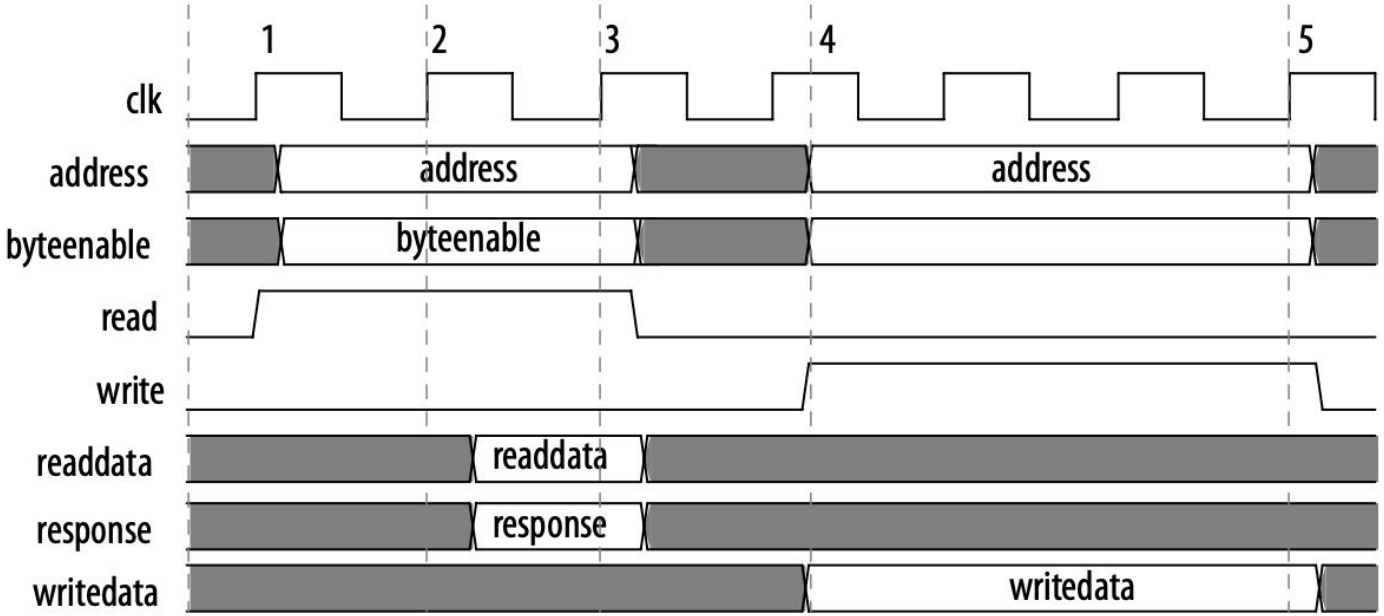
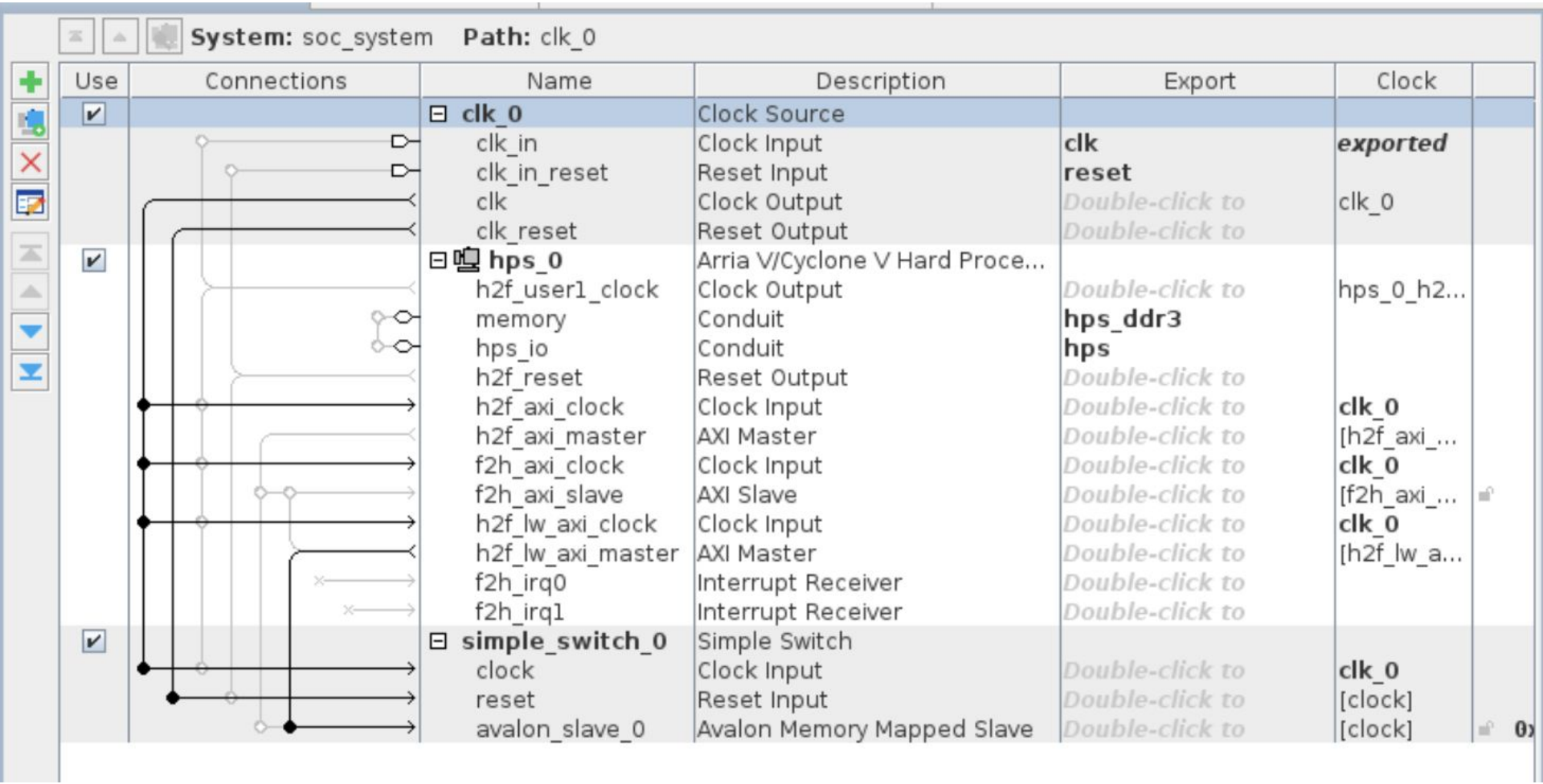


**Figure 11. Read and Write Transfer with Fixed Wait-States at the Agent Interface**




System: soc_system Path: clk_0
Use
Connections
Name
Description
Export
Clock
clk_0
Clock Source
clk_in
Clock Input
clk
exported
clk_in_reset
Reset Input
reset
clk
Clock Output
Double-click to
clk_0
clk_reset
Reset Output
Double-click to
hps_0
Arria V/Cyclone V Hard Proce...
h2f_user1_clock
Clock Output
Double-click to
hps_0_h2...
memory
Conduit
hps_ddr3
hps_io
Conduit
hps
h2f_reset
Reset Output
Double-click to
h2f_axi_clock
Clock Input
Double-click to
clk_0
h2f_axi_master
AXI Master
Double-click to
[h2f_axi_...
f2h_axi_clock
Clock Input
Double-click to
clk_0
f2h_axi_slave
AXI Slave
Double-click to
[f2h_axi_...
h2f_lw_axi_clock
Clock Input
Double-click to
clk_0
h2f_lw_axi_master
AXI Master
Double-click to
[h2f_lw_a...
f2h_irq0
Interrupt Receiver
Double-click to
f2h_irq1
Interrupt Receiver
Double-click to
simple_switch_0
Simple Switch
clock
Clock Input
Double-click to
clk_0
reset
Reset Input
Double-click to
[clock]
avalon_slave_0
Avalon Memory Mapped Slave
Double-click to
[clock]

# Results (32 packet in total)

| Total Latency | Priority based (egress 0 has highest priority) | Doubly RR |
|---|---|---|
| Even load | 6.17 | 6.21 |
| Overload egress 0 | 4.5 | 4.9 |

They obtain similar average performance.
Keep in mind that priority based implementation guarantees low latency for egress 0 by compromising the performance of other ports.

# Takeaway:

- Implementing algo in hw is different
- State machine is hard
- Manual is so helpful (Avalon bus, read signal length)
- Start with drawing timing diagram: One cycle at a time.

# Demo