README

Quickstart

To build the 4x4 switch, run

```
./build.sh
```

To test the 4x4 switch, run the following on separate tmux panes,

```
# On the first pane (do this first)
cd testbench
./testbench null Allto3
# You can write in the format <x>to<y> where x, y belong to {1, 2, 3, 4, All}
# The All case uses randomness.
```

```
# On the second pane (do this after the first pane's commands)
./ahir_system_test_bench
```

Documentation

Input Daemon

Pseudo Code

The Pseudo-code for the input daemon is fairly simple and close to the Input daemon in the 2x2 case. It is illustrated in the below codeblock.

C-style Pseudo Code

```
count_down = 0; // This is put in place to keep track of length of packet (in words)
remaining
last_dest_id = 0;


while(1) {
     input_word = read_uint32("input_word");
     new_packet = (count_down == 0);

     dest_id, pkt_length, seq_id = splitWord(input_word);
     count_down =   (new_packet ? pkt_length-1 : count_down-1);

     last_dest_id = (new_packet ? dest_id : last_dest_id);
     send_to_1    = (last_dest_id == 1);
     send_to_2    = (last_dest_id == 2);
     send_to_3    = (last_dest_id == 3);
     send_to_4    = (last_dest_id == 4);
     if(send_to_1)
          write_uint32("obuf_1_1", input_word);
     if(send_to_2)
          write_uint32("obuf_1_2", input_word);
     if(send_to_3)
```

```
            write_uint32("obuf_1_3", input_word);
        if(send_to_4)
            write_uint32("obuf_1_4", input_word);
}
```

This is written by tweaking the commented C code provided for the 2x2 switch case.

Output Daemon

The output daemon is also a simple extension of the logic found in 2x2 switch.

prioritySelect

Priority is no longer a simple boolean value. We need to maintain a pointer to keep track of the history.

- The pointer (`priority_index`) points to the first buffer at the start
- We move the pointer to the next available buffer (`priority_index + x` where `x` can be 0, 1, 2, 3) [1] after the current packet is processed (i.e. `down_counter==0`). Else, we sustain its value.
- In order to keep a track of the next available buffer, we have simple combinational logic which takes the buffer valid bits as well as `priority_index` as inputs.
- All of this is implemented in `utils.aa` as part of the `prioritySelect` . This gives us the next packet to use as well as next `priority_index` .

---

1. Note that `priority_index` is a 2-bit register thus even with this, the value always remains between 0 and 3.↩