Switch ON

Ayush Jain (aj2672)
Donovan Chan (dc3095)
Shivam Choudhary (sc3973)
An extremely efficient Hardware Switch!!
Architecture

- Userspace generates packets.
- Input module sorts and places in RAM(s).
- Scheduler avoids collisions between packets.
- Buffer stores data in output RAM(s).
Hardware Communication Protocol

Header Format

End of Packet is 32 Bits Zeros

Length(16 bits) → Seed (8 Bits) → Dst-Port(8 Bits)

Userspace (device driver) → Avalon Bus → Hardware (FPGA)

Userspace Sends WRITE_ENABLE_SCHEDULER and READ_ENABLE_SCHEDULER after packet's sent.
Hardware Scheduler - Single Input Queue

- Source & Destination modelled as 4 RAM(s) each.
- Individual scheduling to prevent collision.
- Greedy, no optimization for head of line blocking.
Hardware Scheduler - PPS

- Modeled as 16 RAMs at the input. (Parallel Packet Switch)
- Destination still modelled as 4 RAMs
- Prevents HOL, hence improves throughput.
- Requires additional hardware complexity and storage.
PPS vs Single Input Queue
PPS vs Single Input Queue (contd.)

**Worst-case Performance**

\[
\text{Speed} = \frac{32}{3} \text{ bits/cycle}
\]
\[
\text{Speed} = \frac{32}{3 \times 20 \times 10^{-9}}
\]
\[
\text{Speed} = 0.533 \times 10^9
\]
\[
\text{Speed} = 508.626 \text{ Mb/s}
\]

**Average Performance**

➔ Better average case performance for PPS.
➔ Higher variance.

**Best-case Performance**

➔ Same for PPS and Single Input Queue.
➔ Such a case is theoretically less probable to occur.
Timing Diagrams - RAM(altsync)

Input Signals - wren, wraddress, data, rden, rdaddress.

Output Signals - q (Data occurs after one clock cycle)
### Timing Diagrams - Scheduler

**Case:** Signals for different output ports.

<table>
<thead>
<tr>
<th>Signals</th>
<th>Waves</th>
<th>Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>10 ns</td>
<td>20 ns</td>
</tr>
<tr>
<td>clk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inp(0)(0)[31:0]</td>
<td>00000000</td>
<td>00000001</td>
</tr>
<tr>
<td>inp(1)(1)[31:0]</td>
<td>00000000</td>
<td>00000002</td>
</tr>
<tr>
<td>inp(2)(2)[31:0]</td>
<td>00000000</td>
<td>00000003</td>
</tr>
<tr>
<td>inp(3)(3)[31:0]</td>
<td>00000000</td>
<td>00000004</td>
</tr>
<tr>
<td>outp(0)[31:0]</td>
<td>00000000</td>
<td>00000001</td>
</tr>
<tr>
<td>outp(1)[31:0]</td>
<td>00000000</td>
<td>00000002</td>
</tr>
<tr>
<td>outp(2)[31:0]</td>
<td>00000000</td>
<td>00000003</td>
</tr>
<tr>
<td>outp(3)[31:0]</td>
<td>00000000</td>
<td>00000004</td>
</tr>
<tr>
<td>total_time[31:0]</td>
<td>00000000</td>
<td>00000001</td>
</tr>
</tbody>
</table>
Timing Diagrams - Scheduler

Case: Signals for same output ports.
Validator

Generate sequence from seed and compute total cycles.

Avalon Bus

Output RAM(s)

Userspace

FPGA
Results

Plot of Number of Clock Cycles against Transferred Data Size
Performance Constraints

- RAMs take up three clock cycles to change from one read location to another.
- Transfers are restricted to 32 bits at a time because of ioctl calls.
- Can also increase performance if we increase the number of parts at the cost of hardware complexity.
Lessons Learned

➔ It is named hard-ware for a reason.
➔ Timing diagrams save time.
➔ Simulations may be far from reality.
➔ You will often reduce to hard problems in polynomial time.
➔ Documentations need a lot of work.
Future Work

➔ Analyze and compare the performance for a maximum bipartite matching solution.
➔ Implement DMA.
➔ Produce test results for greater amount of data and different scenarios.
➔ Interface with Ethernet ports and test with real network.
Thank You!!

Code available on Github: https://github.com/shivamchoudhary/SwitchON
https://github.com/shivamchoudhary/SwitchONHW