Intel Tofino2 – A 12.9Tbps P4-Programmable Ethernet Switch

Anurag Agrawal & Changhoon Kim
Intel Corporation
Notices and Disclaimers

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available security updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.
Why do we need Programmable Switches?

“Future Computing Platforms Critically Require More Efficient, More Reliable, and More Flexible Networks”

Amin Vahdat – Engineering Fellow and Vice President – Google [ONF 2019]

Data plane programmability is critical to enable rapid innovation/experimentation to:

- Supports large scale disaggregation
- Non-blocking BW and “zero queuing” with new end-to-end congestion controls
- In network processing
- End-to-end visibility through data plane telemetry
- Application (load) aware networking
- …
“Programmable switches are 10-100x slower than fixed-function switches. They cost more and consume more power.” — Conventional Wisdom

Needs a Paradigm Shift
Our Observation

Single Chip Switch Silicon Area Distribution

- **128/256 High Speed SERDES** (IO: Hard)
- **PCS/MAC** (IO: Soft)
- **Forwarding Pipeline** (Ingress + Egress)
- **Traffic Manager** (Packet Buffer + QoS)

Only ~30% silicon needed **Significant Attention**

SRAM/TCAM make up large part of forwarding logic (Same for fixed or programmable design)

Even if logic increase by 2x → ~20-25% overall increase in chip Area/Power (not 10x)
PISA: Protocol Independent Switch Architecture

Multiple Match-Action Units for header transformation
(VLIW Instructions, ALUs, SRAM+TCAMs, counters, meters, ...)

Packet Header fields to ‘Registers’
(PHV: Packet Header Vector)

User Programs
“This is how you must process packets”

Intel® Tofino™: 6.5Tb/s, 16nm (Dec’2016)

Any header anywhere
## Intel Tofino Proof Point

### P4 Programmable “Tofino” vs. Fixed Function ASIC

<table>
<thead>
<tr>
<th>Feature</th>
<th>P4 Programmable “Tofino”</th>
<th>Fixed Function ASIC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technology</strong></td>
<td>16nm</td>
<td>16nm</td>
</tr>
<tr>
<td><strong>L2/L3 Throughput</strong></td>
<td>6.5Tb/s</td>
<td>6.4Tb/s</td>
</tr>
<tr>
<td><strong>Number of 100G Ports</strong></td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Max Forwarding Rate</strong></td>
<td>4.8B packets per sec</td>
<td>4.2B packets per sec</td>
</tr>
<tr>
<td><strong>Max 25G/10G Ports</strong></td>
<td>256/258</td>
<td>128/130</td>
</tr>
<tr>
<td><strong>Programmability</strong></td>
<td>Yes (P4)</td>
<td>No</td>
</tr>
<tr>
<td><strong>Typical System Power draw</strong></td>
<td>4.2W per port</td>
<td>4.9W per port</td>
</tr>
<tr>
<td><strong>Large Scale NAT</strong></td>
<td>Yes (100k)</td>
<td>No</td>
</tr>
<tr>
<td><strong>Large scale stateful ACL</strong></td>
<td>Yes (100k)</td>
<td>No</td>
</tr>
<tr>
<td><strong>Large Scale Tunnels</strong></td>
<td>Yes (192k)</td>
<td>No</td>
</tr>
<tr>
<td><strong>Packet Buffer</strong></td>
<td>Unified</td>
<td>Segmented</td>
</tr>
<tr>
<td><strong>Segment Rtg/Bare Metal</strong></td>
<td>Yes/Yes</td>
<td>No/No</td>
</tr>
<tr>
<td><strong>LAG/ECMP Hash Algorithm</strong></td>
<td>Full entropy, programmable</td>
<td>Hash seed, reduced entropy</td>
</tr>
<tr>
<td><strong>ECMP</strong></td>
<td>256 way</td>
<td>128 way</td>
</tr>
<tr>
<td><strong>Telemetry and Analytics</strong></td>
<td>Line-rate per flow stats</td>
<td>Sflow (Sampled)</td>
</tr>
</tbody>
</table>

### Benefits of P4-programmability

- **14% Greater Performance**
- **14% Lower Power**
- **Better Burst Absorption**
- **Real-time Visibility**
Can This Be Repeated?

Was Tofino just a fluke?

Is there an evidence that switches can be programmable and meet speeds & feeds?
Intel Tofino2: The 2\textsuperscript{nd} Generation of PISA

Performance
- 12.9Tb/s & 6B pkt/s of throughput

More Programmable PISA
- Fully programmable in P4 with more resources

Unified Traffic Manager
- Larger unified packet buffer

Power
- Significant better performance/watt

Industry-leading Process Node
- 7nm Technology

- 260x 56G SERDES
- PCIe Gen3 x4
Intel Tofino2: Multi-die Package

Package
- 2.5D CoWoS Packaging
- 260 lanes of 56G-PAM4 SerDes Tiles (x4)
- PCIe Gen3 x4 lanes

Switch Die
- 4 Pipe x 3200Gb/sec/piper Architecture
- 64MB Unified Packet Buffer
- 400G MACs (32x) + 100G MAC (1x)
  - Highest Radix as 256x10/25/50GE, 128x100GE, 64x200GE, 32x400GE
- Gen3 x4 PCIe Interface, GPIOs
- 7nm FF TSMC
Intel Tofino2: Match-Action Unit (MAU)

- New VLIW instructions, Multiple instructions per table
- Enhanced Stateful ALU capabilities
  - Max/min selection, 16b signed divide, learning/match, and more
- HW learning and Stateful FIFO/Stack primitives
Intel Tofino2: Traffic Manager

- Non-blocking any-to-any communication
  - Radix: 32x400GE ... 256x50GE
- ~25Tb/s of write and read from 4 pipes
  - Like a 25000 bits wide memory accessed at 1 GHz
- ~6B pkt/s of enqueue & dequeue operations
- 128Q per 400G port with Hierarchical-QoS
  - 1.68ns for 64B packet @ 400G
- Exporting rich metadata to forwarding pipeline
**Intel Tofino2: Traffic Manager Metadata to Ingress Pipe**

- **Programmable Parser**

- **Ingress stages (pre-switching)**
  - Q-occupancy (TM Metadata)

- **Egress stages (post-switching)**

- **De-Parse**

- **Buffer**

- **Use Cases:**
  - Congestion-aware routing
  - Congestion Notification from ingress
  - ...

- **Egress queues' info at all ingress pipes**

- **Control plane configures queues to monitor**
Co-packaged Optics (CPO) Demonstration at 12.9Tb/s

March 2020 Demonstration

- Intel Tofino2 P4-programmable Ethernet switch co-packaged with integrated photonic engines
- Fully functional switch demonstrating 400G Ethernet traffic interoperability
- Compliant with applicable standards for I/O interfaces

How Is Programmability Used?

1. Data-plane Telemetry and Real-time Control
1. “Which path did my packet take?

   “I visited Switch 1 @780ns, Switch 9 @1.3µs, Switch 12 @2.4µs”

2. “Which rules did my packet follow?”

   “In Switch 1, I followed rules 75 and 250. In Switch 9, I followed rules 3 and 80.”

---

<table>
<thead>
<tr>
<th>#</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>192.168.0/24</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
3. "How long did my packet queue at each switch?"

4. "Who did my packet share the queue with?"

"Delay: 100ns, 200ns, 19740ns"
3. "How long did my packet queue at each switch?"
   - "Delay: 100ns, 200ns, 19740ns"

4. "Who did my packet share the queue with?"
The Network Should Answer These Questions

1. “Which path did my packet take?”
2. “Which rules did my packet follow?”
3. “How long did it queue at each switch?”
4. “Who did it share the queues with?”

Intel Tofino™ + Deep Insight™ can answer all four questions. At full line rate. Without generating any additional packets!
Flow Reporting: INT-MD Mode

**Add:** SwitchID, Arrival Time, Queue Delay, Matched Rules, ...

**INT Source**
Instruments packets for Telemetry

**INT Transit**
Adds metadata based on INT instructions

**INT Sink**
Removes metadata

Original Packet

Log, Analyze and Visualize

**[Change Detector]**
“Monitor every packet, but report only what matters!”
- Generate reports upon
  - Flow initiation and termination
  - Path or latency changes
  - Special field values
- Change detectors are reset periodically (e.g., once every sec)
Flow Reporting: INT-XD Mode

Increase of report-traffic volume is marginal!

[Change Detector] ▪ Reset frequently (e.g., once every sec)

[Change Detector] ▪ Reset much less frequently (e.g., once every 10 sec)

SwitchID, Arrival Time, Queue Delay, Matched Rules, ...

Log, Analyze and Visualize
Viewing Microbursts (to the nanosecond)
HPCC: INT-based High Precision Congestion Control

Published at SIGCOMM 2019 by Alibaba, Harvard, Univ. of Cambridge, and MIT

Using INT as explicit and precise feedback

- Very fast convergence via MIMD (multiplicative increase & multiplicative decrease)
- Near-zero queue
- Few parameters

Adjust rate per ACK

Sender → pkt → pkt → pkt → pkt → Receiver

Link-1

Link-2

ACK

INT

INT
Key Benefits of HPCC:
Extremely Low Latency And Very High Throughput At The Same Time
How Is Programmability Used?

2. Accelerating Apps Beyond Networking
Cambrian Explosion Of Beautiful New Apps

Graphics
- OpenCL™
- Compiler
- GPU & GPGPU

ML/DL - Inference
ML/DL - Training
Self-driving Vehicles
HPC & Big Data
Virtual Reality
Augmented Reality
Gaming/Professional Graphics
2D/3D Graphics
Basic Graphics

Networking
P4
Compiler

Programmable Data Planes

More...
- Storage/Memory/ML Interconnect
- In-line Crypto
- DNS Cache
- KV Caching & Replication
- Extremely-Accurate Time Sync
- Pattern Matching & Stream Processing
- Consensus Acceleration
- Network Packet Broker
- Firewall & DDoS
- Accelerated VNFs
- E2E Telemetry & Analytics
- Adv Congestion Control
- Enterprise/Telco/SP Switching
- Cloud Switching
Accelerating DNN Training

- Training over huge data requires distributed processing
- With faster workers, sharing learned parameters becomes a bottleneck

![ResNet 269 (Sec/Iteration) Graph](image-url)
SwitchML: Streaming Aggregation via Switches

~100's of MB

~10's – 100's of KB
Combined Switch-NIC Architecture

NIC Responsibilities

- Chunking up vectors
- Quantization and scaling
- Detecting and recovering from packet drops

Switch Responsibilities

- Integer vector addition (32 elements per packet)
- Counting and comparison to detect complete slots
How Much Faster is SwitchML?

SwitchML provides a speedup from 20% to 300% compared to Tensorflow with NCCL (with direct GPU memory access)

Combined Switch-NIC Architecture Leads To Innovations