Intel Skylake

Ryan Estep
Vishakh Suresh Babu
A brief introduction

- Intel Skylake microarchitecture
  - 2015
  - designed for 14nm process
  - preceded by Broadwell
- Intel development process
  - Tick-Tock
  - Process-Architecture-Optimization
- Tick was new process, adapting old architectures
- Tock was designing new microarchitecture

<table>
<thead>
<tr>
<th>Intel development roadmap</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cycle</strong></td>
</tr>
<tr>
<td>Tock</td>
</tr>
<tr>
<td>Tick</td>
</tr>
<tr>
<td>Tock</td>
</tr>
<tr>
<td>Tick/Process</td>
</tr>
<tr>
<td>Architecture</td>
</tr>
<tr>
<td>Optimization</td>
</tr>
<tr>
<td>Optimization</td>
</tr>
<tr>
<td>Optimization</td>
</tr>
<tr>
<td>Optimization</td>
</tr>
<tr>
<td>Optimization</td>
</tr>
<tr>
<td>Optimization</td>
</tr>
</tbody>
</table>

*Source: [5]*
Die shot

~11.08 mm

~9.19 mm

~13.31 mm

Image source: [5]
Die shot (cont’d)

Image source: [5]
Pipeline (cont’d)

IO Fetch

- Branch Pred
- Instruction Fetch Unit
  - L1 ITLB
  - 32KB L1 I$ (8-way)
  - >20B
  - 16B Predecode, Fetch Buf
  - 6 x86 Instructions
  - 2x20 Instruction Queue
    - μcode
    - Complex Decode
    - Simple Decode
    - Simple Decode
    - Simple Decode
  - 5 x86 Instructions

IO Decode

- 1.5K μop Cache
- 2x64 μop Decode Queue
- 5 μops

OOO Issue and Late Commit

- Retire Unit
- 224 Entry ROB
  - 2x4 μops

Integer/FP Functional Units with OOO Writeback

- 160 Integer Registers
- 168 FP Registers
- 48 Entry BR Order Buffer
- 72 Entry Load Buffer
- 66 Entry Store Buffer

Load/Store Execution

- 97 Entry Unified Scheduler
  - Port 0, 1, 2, 3, 4, 5, 6, 7
    - ALU
    - ALU Branch
    - LEA
    - ALU Branch
    - SIMD
    - FMA
    - DIV
    - SORT
    - MUL

- L1 DTLB
  - 32KB L1 D$ (8-way)
- L2 DTLB
  - 256KB L2 Cache (8-way)

Image source: Page 21, [3]
Front end overview
MOP fusion & Decoding

- Pre-decoding buffer
  - mark instruction boundaries
  - prefix decoding (e.g. branches)
- IQ has the ability to fuse MOPs into a single instruction
  - improved bandwidth
- Decode complex and variable MOPs into fixed size μ-ops

μ-op cache

- μ-op cache (or Data Stream Buffer)-has cache lines of decoded μ-ops ready
- Bypasses the entire other path to IDQ (immensely preferred path)
- 1536 μ-ops → 32 sets, 8 lines/set, 6 μ-ops/line
- Competitively shared
- Hit rate > 80%
  - “Hot spots” ~100%

Execution engine overview
Renaming & optimizations

- Reorder Buffer for OoO Execution
  - in-order commit
  - increased size from predecessors
- Register Alias Table maps architectural registers to physical registers
- Speculative Execution
  - branch Order Buffer for mispeculation
- Renaming optimizations include Move Elimination, Zero or Ones Idiom

Source : Page 2-20, [1]; Page 24, 28 [3]  
Image source : Page 21, [3]
Scheduler & EUs

- Unified Reservation Station
- Scheduler for sorting μ-ops between ports and holding them until EU is ready
  - competitively shared and increased in size from predecessors
  - OoO oldest ready
- Ports are balanced between instructions for maximum performance

Memory subsystem overview

- Caches:
  - L0 μ-op cache
  - 3-level cache hierarchy
    - L1 cache
    - L2 cache
    - L3 cache/ LLC
  - eDRAM (on Skylake GPUs)
- TLB

*Source: Page 2-19, [1]*
Cache hierarchy

- **L1 cache**:
  - separate Instruction and Data caches
  - shared by 2 threads on the same core
  - L1D bandwidths:
    - load: 64 B/cycle
    - store: 32 B/cycle

- **L2 cache**:
  - unified
  - non-inclusive of L1
  - 64 B/cycle bandwidth to L1
Cache hierarchy (cont’d)

- **L3 cache/ LLC**:
  - inclusive of L2
  - shared among all cores
  - split into slices connected by 4 rings:
    - data, request, acknowledgement & snoop
    - to increase the bandwidth
    - uses an undocumented hash function, mapping cache lines almost evenly across slices
  - per core bandwidths (@ ring clock):
    - read & write: 32 B/ cycle (two times that of Haswell)

## Cache parameters

<table>
<thead>
<tr>
<th>Level</th>
<th>Capacity</th>
<th>Associativity</th>
<th>Line size (bytes)</th>
<th>Fastest latency (cycles)</th>
<th>Update policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1I</td>
<td>32 KB</td>
<td>8</td>
<td>64</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>L1D</td>
<td>32 KB</td>
<td>8</td>
<td>64</td>
<td>4</td>
<td>writeback</td>
</tr>
<tr>
<td>L2</td>
<td>256 KB</td>
<td>4</td>
<td>64</td>
<td>12</td>
<td>writeback</td>
</tr>
<tr>
<td>L3</td>
<td>Up to 2 MB per core</td>
<td>Up to 16 ways</td>
<td>64</td>
<td>44</td>
<td>writeback</td>
</tr>
</tbody>
</table>

_Data source_: Page 2-20, [1]
Cache parameters (cont’d)

- L2 cache has been reduced from an 8-way (in Haswell) to 4-way set associative.
  - Theoretically, half the associativity \(\Rightarrow\) \(\uparrow\) in miss rate.
  - Practically,
    - \(\downarrow\) in power on a successful data access
    - saves area on the silicon die
  - \(\uparrow\) in miss rate countered by
    - doubling bandwidth to L2 misses
    - improvement in cache and page miss handling
  - Net effect: A performance comparable to Haswell @ a reduced power consumption.

Source: Page 2-20, [1]
**eDRAM based cache**

*Haswell & Broadwell:*

- eDRAM access through L4 tags in LLC.
- eDRAM acts like a victim cache for LLC.
- eDRAM fetches from processor:
  - earlier tag checking
  - faster
- Other devices require eDRAM data:
  - go through LLC & do the L4 tag conversion
  - slower

*Source*: [4]  
*Image reference*: Page 17, [7]
eDRAM based cache (cont’d)

**Skylake:**

- eDRAM behaves as a buffer!
- Other devices requiring eDRAM data do not need to navigate through the on-chip LLC.
- Graphics workloads need to circle around the system agent.
- All memory accesses through MC get looked up in eDRAM.
  - hit: use value from eDRAM.
  - miss: value stored on the eDRAM.
- Available in 2 sizes: 64 GB & 128 GB
  - (48 EU)  (72 EU)

*Source: [4]  Image reference: Page 18, [7]*
## TLB parameters

<table>
<thead>
<tr>
<th>Level</th>
<th>Page size</th>
<th>Entries</th>
<th>Associativity</th>
<th>Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITLB</td>
<td>4 KB</td>
<td>128</td>
<td>8</td>
<td>dynamic</td>
</tr>
<tr>
<td></td>
<td>2 MB/ 4 MB</td>
<td>8 per thread</td>
<td>8</td>
<td>fixed</td>
</tr>
<tr>
<td>DTLB</td>
<td>4 KB</td>
<td>64</td>
<td>4</td>
<td>fixed</td>
</tr>
<tr>
<td></td>
<td>2 MB/ 4 MB</td>
<td>32</td>
<td>4</td>
<td>fixed</td>
</tr>
<tr>
<td></td>
<td>1 GB</td>
<td>4</td>
<td>4</td>
<td>fixed</td>
</tr>
<tr>
<td>STLB</td>
<td>4 KB and 2 MB/ 4 MB</td>
<td>1536</td>
<td>12</td>
<td>fixed</td>
</tr>
<tr>
<td></td>
<td>1 GB</td>
<td>16</td>
<td>4</td>
<td>fixed</td>
</tr>
</tbody>
</table>

*Data source: Page 2-20, [1]*
Parallelism summary

- Client Dual-core or quad-core
- Dual-thread
  - competitively shared
- Skylake (Server)
  - doubled bandwidth after front-end
  - mesh Interconnect
  - up to 28-cores (56 threads)
  - AVX-512

Source: Page 17, 29 [3]; [6]  
Image source: [6]
More special features

- Configurable core
  - Client (14nm)
  - Server (14nm+) → higher drive current, lower power

- Focus on graphics
  - wanted to improve performance and power consumption during video...
  - new IPU/ISP in mobile units

- Security technology
  - protection from attacks
  - SGX, MPX --now deprecated

- Speed Shift power management
- Turbo Boost Technology
  - turbo mode: cores run faster than the rated frequency
  - algorithmic overclocking

Source: Page 18, 19 [3]; [5]; [6]; [12]
Power management

● Previously, OS responsible for DVFS based on the current workload.
  ○ eg: CPU utilisation peaked ⇒ ↑ f to cope up with it
  ○ limitation: granularity of OS response time - 10s of milliseconds

● “Speed Shift” - new power management.
  ○ quickly alternate core frequencies in response to power loads
  ○ a new unit called Package Control Unit (PCU)
    ■ full-fledged microcontroller
    ■ collects and tracks many SoC statistics
  ○ speed shift kicks in ~1 ms

OS bases P-state control can be as slow as 30 ms

Source: [5]
Skylake vs Kaby Lake

- **Turbo boost**:  
  - Skylake: 3.1 GHz  
  - Kaby Lake: 3.5 GHz

- **Encoding & decoding video codecs (10-bit 4K HEVC video codecs as well as 4K VP9)**:  
  - Skylake: software support  
  - Kaby Lake: hardware support

<table>
<thead>
<tr>
<th>Playing</th>
<th>Battery-life improvement in Kaby Lake</th>
<th>Power consumption (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Skylake</td>
</tr>
<tr>
<td>10-bit 4K HEVC video</td>
<td>2.6 x</td>
<td>10.2</td>
</tr>
<tr>
<td>4K video on YouTube</td>
<td>1.7 x</td>
<td>5.8</td>
</tr>
</tbody>
</table>
References


Supplementary slides
Image source: [5]
Fetch & Pre-decoding

- Fetching is dual-thread
  - Shared evenly
- 16B chunks of code
- Pre-decoding buffer
  - Mark instruction boundaries
  - Prefix decoding (e.g. branches)
- BPU-branch prediction
  - Further “vision” than predecessors

*Source*: Page 2-16, 2-17 [1]; Page 22 [3]  
*Image source*: [3]
Instruction queue & MOP fusion

- 25 entries/thread
- Instruction queue holds macro-ops until the decoder is ready
- Has the ability to fuse MOPs into a single instruction
  - Improved bandwidth

Source: Page 2-17, 2-18, [1]; Page 22 [3]
Decoding

- 5-way decoder
  - 1 complex and 4 simple
- Decodes complex and variable MOPs into fixed size µ-ops
- Supports 5 µ-ops sent down the pipeline
- Complex decoder=1-4 µ-ops
- More than 4 µ-ops->microcode sequencer

Source: Page 2-18, [1]; Page 22 [3]  
Image source: [3]
μ-op cache & Allocation queue

- Allocation queue (or Instruction Decode Queue)-interface between the in-order fetch/decode and OoO execution engine
  - Partitioned (non-competitive) 64 entries/thread
  - Loop stream detector detects loops and repeats μ-ops (server only)
- μ-op cache (or Data Stream Buffer)-has cache lines of decoded μ-ops ready
  - Bypasses the entire other path to IDQ (immensely preferred path)
  - 1536 μ-ops--32 sets, 8 lines/set, 6 μ-ops/line
- Competitively shared

Source: Page 2-18, Page 2-20 [1]; Page 150, [2]  
Image source: [3]
Intel Turbo Boost

- Some programs are memory-bound & some CPU-bound
  ⇒ need not always run the CPU at max frequency.
- Turbo Boost as an energy-η soln to this problem:
  - run at base clock speed for lighter workloads.
    - less power consumption
    - less heat dissipation
  - dynamically switch to a greater clock rate for heftier loads.
  - upto a max turbo boost frequency.
    - still within the safe power and temp limits.
  - “algorithmic overclocking”

Source: [12]
## Intel Turbo Boost (cont’d)

<table>
<thead>
<tr>
<th>Processor</th>
<th>Processor base frequency (GHz)</th>
<th>Max turbo boost frequency (GHz)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core i7-6700HQ</td>
<td>2.6</td>
<td>3.5</td>
<td>Mobile processor</td>
</tr>
<tr>
<td>Intel Core i7-6700T</td>
<td>2.8</td>
<td>3.6</td>
<td>Mainstream desktop processor</td>
</tr>
<tr>
<td>Intel Core i9-9960X X-series</td>
<td>3.1</td>
<td>4.4</td>
<td>High end processor</td>
</tr>
</tbody>
</table>

*Data source: [9], [10], [11]*