Microarchitectural Comparison of Intel IceLake and AMD Ryzen processor

Jacob James
jacobk@iisc.ac.in

Kawin M
kawinm@iisc.ac.in

I. Introduction

IceLake processors are the Intel's 10th generation of processors based on Sunny Cove microarchitecture with 10nm process. On the other hand, AMD's latest Ryzen processors are based on Zen 2 microarchitecture with 7nm process. This paper goes on to compare the microarchitecture of IceLake and Ryzen processors together with the design rationale behind each decision and their impact on performance and power consumption. This paper is split into the following sections: Frontend, Backend, Memory Hierarchy, Memory Address Translation and Branch Prediction Unit.

II. Front-End

A. Instruction Fetch and PreDecode

In IceLake, due to the variable instruction size (ranging from 1 to 15B), the bytes fetched are analyzed by the pre-decoder to mark instruction (macro-ops in Intel's terminology) boundaries. The pre-decoder outputs a maximum of 6 macro-ops per cycle and no further fetching of 16B occurs until all the macro ops corresponding to the bytes fetched are inserted into the Instruction Queue (IQ). This can lead to a decrease in the throughput in terms of instructions fetched. [2] The instruction fetch unit in Ryzen fetches 32 bytes

	IceLake	Zen 2
Bytes fetched per Cycle	16 Bytes	32 Bytes

naturally aligned block from L1 I-cache every cycle. The fetch unit sends these bytes to the decode unit through a 20 entry Instruction Byte Queue (IBQ), each entry holding 16 instruction bytes. The IBQ acts as a decoupling queue between the fetch/branch-predict unit and the decode unit. The pick unit (predecoder) scans two of the 16B IBQ windows per cycle to determine the boundaries of up to four instructions. Only the first slot can pick instructions longer than 8 bytes.

The 16B instruction fetch in IceLake is a major bottleneck in frontend, as the L1 I-cache fetches 64B from the L2 cache every cycle. Whereas in AMD, even though, theoretically 32B are fetched per cycle, from observations made, in practice, atmost 16B are fetched per cycle. IBQ is suspected to be the bottleneck in this regard [1]. Comparatively, the size of the similar structure in IceLake, the IQ has 50 entries, whereas IBQ in Ryzen has 20 entries only.

B. Interpretation of Macro-op and Micro-op

Macro-op is a variable length x86 instruction which needs to be converted into simpler instructions called micro-ops that can be executed by functional units in the context of IceLake. Whereas in Zen 2, a macro-op is a fixed length, uniform representation of (usually) one x86 instruction and is the primary unit of work managed by the processor. Micro-ops are the primitive operations executed in the processor's execution units. Each Macro-op can normally contain up to two micro-ops.

AMD uses these fixed size Macro-ops throughout the processor except in the exception units because - 1) On average, Number of Macro-ops per instruction is smaller than the number of micro-ops per instruction. Thus, AMD can use small size buffers or queue (such as Op Queue, ROB) to achieve the same performance as Intel with larger buffers. This helps in reducing the power consumption. 2) As keeping Macro-ops fixed size, the processor design could be kept simple. 3) Also, the micro-ops can be scheduled to execution unit, so that it also adheres to the high performance principles as that of RISC.

C. Instruction Decode

In Icelake macro-ops are converted to simple micro-ops by 5 decoders (4 simple 1 complex). Instructions that generate more than 1 but less than 4 uses the complex decoder during which some simple decoders may be deactivated [2]. To exploit more ILP the usage of complex decoder must be kept to minimum by reordering instructions. Instructions that decode to more than 4 micro-ops are routed through a MicroSequencer ROM, the usage of which is very less than 1% of total clock cycles from our analysis which doesn't pose much overhead.

In Zen 2, decoder converts up to four instructions per cycle to macro-ops. The maximum throughput of 6 macro-ops per cycle can be obtained only if half of the instructions generate two macro-ops each. Instructions that generate more than 2 macro-ops are using microcode. These instructions take at least two clock cycles to decode so that the throughput can be no higher than one complex instruction every two clock cycles. There is no penalty for decoding instructions with many prefixes [1].

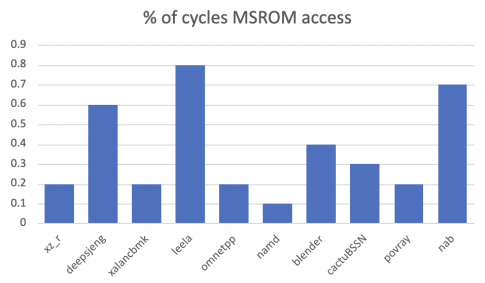


Fig. 1. Percentage of CPU cycles MSROM was accessed - Intel

Decoder unit is the major bottleneck in both the processors with a Decoding rate of 16 B per cycle and this hasn't improved even after several generations of improvement in microarchitecture. Bottleneck in the decoder appears to be difficult to overcome. This is a consequence of the variable size of the x86 instructions and it is complicated to determine the length of each instruction [12]. To compensate for this, an op cache is being used in both the processors.

D. Op Cache

Op cache is a cache of previously decoded instructions. When instructions are being served from the op cache, normal instruction fetch and decode are bypassed. This saves power consumption of the complex decoding unit and also gives better throughput for frontend. Thus, the op cache is adopted in both the processors. In Icelake, an entry in the Op cache

	IceLake	Zen 2
Op Cache Size	2304 micro-ops	4096 Macro-ops

stores the micro-ops of 64B of instructions (6 micro-ops in average) which is sent to the issue queue per cycle ensuring that IPC is not limited due to the restrictions imposed by decoders.

In Zen 2, each entry contains up to 8 sequential instructions ending in the same 64-byte aligned memory region, resulting in a maximum capacity of 4096 macro-ops. An interesting thing is that, Op cache in Zen 2 also stores the macro-fused branch instructions [3]. The op cache can send up to 8 macro-ops per cycle to the op queue, whereas, the legacy decoder unit sends atmost 4-6 macro-ops to the op queue (or IDQ in Intel). Thus, the throughput from the op cache is doubled in Zen 2.

From our performance evaluation of SPEC Benchmark programs, on an average of 65% of macro-ops are dispatched through the op cache. Thus, as AMD reports says the doubling of op cache size is a better trade off in terms of power consumption and performance is verified to be true.

E. Dispatch Width

The IDQ in Intel dispatches 6 micro-ops to the backend, whereas in Zen 2, the Op queue dispatches 6 macro-ops to the backend.

F. Frontend Optimizations

1) Stack Engine: Found in both IceLake and Zen2. It is dedicated hardware for handling PUSH, POP, and RET instructions. Without Stack engine, instructions involving the stack are decoded into 2 micro-ops - stack pointer update and store micro-ops which had to go through the complex decoder and then to separate execution units which can cause delay. Stack Engine takes the pointer modification part away from the pipeline so that only one micro-op is issued effectively. The Stack Engine consists of parallel adders for updating the stack pointer. For ensuring that the correct value of the stack pointer becomes available to instrutions certain synchronizing micro-ops are added in Icelake. [1]
Stack Engine in Zen 2, unlike that in IceLake, helps in eliminating dependencies on macro-ops performing stack operations. It consists of a sideband stack optimizer (SSO) and a stack tracker. SSO removes long chains of dependencies on the **rsp** register in a chain of PUSH, POP, CALL and RET instructions and the Stack tracker tracks the stack-pointer value [3]. Since referencing the system stack is frequent in programs involving procedure calls the extra hardware is beneficial which can be seen in both architectures adopting this feature.

2) Zero Latency Memory Mirroring: Memfile in Zen 2 predicts dependencies between stores and loads accessing the same data in memory. Both stack engine and memfile uses memory renaming to facilitate store-to-load forwarding bypassing the load-store unit. This could result in a zero latency memory operation for both loads and stack operations. This is done speculatively in frontend and it's impact on performance is yet to be know **??**.
Intel started adopting this strategy from IceLake where store-to-load Bypassing can take place with zero delay. It was observed that [1] there was no prediction like Zen2 implying bypassing is done after knowing the address. The exact details are not known whether there is register mirroring or some other strategy that IceLake is using.

3) MacroFusion and Microfusion: Two of the prominent features of the Intel processor are micro-op fusion and macro-op fusion. In the case of the latter operation, whenever a CMP instruction is followed by a JUMP instruction, the processor combines them, thereby reducing the number of instructions while increasing instruction throughput. For micro-op fusion, the processor considers instructions requiring two microop as a single microop through most of the pipeline(even though it will be executed as two instructions). Hence only one entry for such operations is needed in the reorder buffer and it also reduces the usage of the complex decoder. Microop fusion can be applied for ready-modify and read-modify-write

instuctions.
AMD uses macro-op fusion, in which a CMP or TEST instruction immediately followed by a conditional jump is fused into a single macro-op. The fused branch instructions can execute at a throughput of two such branches per clock cycle if they are not taken, or one branch per two clock cycles if taken. The macro fusion is done in the decode unit and the fused macro-ops are forwarded to Op cache [13]. Since Branch Fusion eliminates 1 macro op, it increases the dispatch, issue and retire bandwidth.

4) Loop Stream Detector: This is an optimization specific to Intel dealing with loops. Microops detected as part of a loop are repeatedly streamed from the allocation queue to the backend until the loop condition becomes false. The frontend will be deactivated during this time. The size of the loop is bounded by the allocation queue which is the queue storing microops ready to be dispatched to the backend [2]. According to my observation this is only a power saving feature as the microop cache can perform the same task.

III. Back-End

A. ReOrder Buffer

	SkyLake	IceLake	Zen	Zen 2
Reorder Buffer Size	224	352	192	224

For a superscalar processor to be efficient the IPC must match the instruction retire rate. The number of instructions retired is influenced by the size of the ROB. If the instruction retire rate doesn't match the IPC improvement then instruction retirement can become a bottleneck. Icelake had an IPC improvement of 18% over Skylake. Therefore the reorder buffer size must be atleast 1.18 more than Skylake which is true. Similarly, the IPC improvement from Zen to Zen 2 is 15% which correlates with the increase in ROB size by 15% in Zen 2 [3]. Upto 8 macro-ops can retire per cycle in Zen 2 and the ROB is shared between both execution units. The penalty in the power consumption is compensated by the smaller process node as we we get the advantage of a more power budget.

As said earlier in II B, the reasoning behind Zen 2 using a smaller ROB than IceLake is that the ROB in Zen 2 stores macro-ops, whereas, the ROB in IceLake stores micro-ops. Thus, Zen 2 could store the information for same number of instruction with less number of entries.

B. Scheduler

In both the processors, the scheduler is responsible for issuing micro-ops to the various execution units out-of-order subject to dependencies. In Zen 2, the scheduler also breaks the received macro-ops into micro-ops.
IceLake feeds the execution units from a Unified Reservation Station, which is logically partitioned comprising of 160 entries. The advantage of a Unified Reservation Station will be the maximum utilization of available entries compared to distributed ones in which majority of the entries corresponding to a functional unit may be idle, causing poor utilisation of hardware. [7]
Zen 2 consists of 6 dedicated schedulers - four single issue 16 entry ALU schedulers, one three issue 28 entry Address Generation Unit (AGU) scheduler and one four issue 36 entry FP scheduler. Schedulers receives upto 6 Macro-ops, where they are broken into micro-ops. The entries in the schedulers are micro-ops, and not macro-ops like that in other buffers. Scheduler tracks operand availability and dependency information as part of its task of issuing micro ops to be executed. Each scheduler tracks the availability of operands and issues one micro-op per cycle which is ready for execution, oldest first, to the Execution units.

C. Register Renaming

In IceLake Register Renaming is done by the Register Allocation Table (RAT). There are a total of 280 integer and 224 vector registers. The size of each vector register is unknown, but it might be a mixture of 512-bit and 256-bit registers from the vectorization capability of IceLake. The count for the registers has been made taking into account the issue width; therefore renaming has not been found to be a bottleneck in IceLake.

Zen 2 has a dedicated renaming unit for both the execution units. The ROB and the integer and floating point rename units form the retire control unit (RCU) tracking instructions, registers, and dependencies in the out-of-order execution units. The remaining registers are available for out-of-order renames. Renaming of Floating point operations is done by the dedicated FP renaming unit and it can rename 4 micro-ops per cycle. The integer physical register file is made up of 180 registers (up from 168 in Zen) and the floating point register file is made up of 160 registers (same as in Zen, but the register width is doubled 128 bits to 256 bits), with up to 38 per thread mapped to architectural state.

D. Optimizations in Renaming Unit

1) Zero Cycle Move: Both IceLake and Ryzen processors supports move elimination, performing register to register moves with zero latency in the rename unit while consuming no scheduling or execution resources. This is implemented by mapping the destination register to the same physical register as the source register and freeing the physical register previously backing the destination register. Given a chain of move instructions registers can be renamed several times in one cycle.

2) Idioms for Dependency removal: A number of instructions can be used to clear a register and break the dependency without the need to load an immediate value of 0 or 1. These are referred to as Zeroing or Ones Idioms respectively. Both

IceLake and Zen 2 recognizes zeroing and ones idioms such as XOR-ing, subtracting or Comparing a register with itself to eliminate the dependency on the source register.

E. Execution Unit

The execution unit in IceLake is unified, whereas, there are 2 dedicated execution units for integer and floating point or vector operations in Zen 2.

1) Issue Width: In IceLake, the maximum issue width is 10 micro-ops per cycle, i.e, 1 micro-op per each port per cycle; however, on an average, only 5 micro-ops are issued due to constraints imposed by dependencies. In Zen 2, together all schedulers in the core can issue up to 11 micro-ops per cycle, this is not sustainable however due to the available dispatch and retire bandwidth [3].

IceLake

- Fed from a unified Reservation Station which is logically partitioned.
- 10 Execution ports (4 ALU, 2 Store Data , 4 Address Calculation Units for Loads and Stores). Only 5 execution ports on average were found to be active on a given cycle due to constraints imposed by dependencies. Therefore the issue width on average is 5 while the theoretical maximum would be 10.
- Execution ports are shared among the integer floating point and vector instructions.(Port 0,1,5,6 Integer Arithmetic, Ports 0,1 Floating Point)
- Most of the operations take unit clock cycle except for floating-point divide and square root and memory operations.
- Seperate address calculation units for load (2 units) and store (2 units) enables load and store to take place parallely. The L1 Data Cache Bandwidth was increased so that more loads and stores can take place parallely. The size of the Load Store Queue was also increased appropriately to handle more loads and stores.

Zen 2

Integer Execution Unit: The processor contains 4 integer execution pipes. There are four asymmetric ALUs capable of all integer operations with the exception of multiplies, divides, and CRC which are dedicated to one ALU each.

There are 3 AGUs (Address Generation Unit) pipelines compute a linear memory address from the operands of a load or store micro-op. Two of them can generate load addresses, while all three can generate store addresses.
The third dedicated AGU for store operation is added in Zen 2, the possible reason could be the fully utilize the capability of L1 D-cache of being able to read two 256 b and write one 256 b per cycle. Thus, the 3 ports can be utilized properly by the 3 AGU, otherwise, the 3 read ports could be under-utitlised with 2 AGUs.

Floating-Point Execution Unit: The floating point unit in Zen 2 utilizes a coprocessor architectural model with dedicated scheduler, renaming unit, register file and 4 execution pipes (2 for FMUL and 2 for FADD operations) which can execute an operation every cycle. There is a 64-entry non-scheduling queue (NSQ) that decouples dispatch and the FP scheduler. This allows dispatch to send operations to the integer side, in particular to expedite floating point loads and store address calculations, while the FP scheduler, whose capacity cannot be arbitrarily increased for complexity reasons [3], is busy with higher latency FP operations. There are dedicated busses to enable fast moves between the floating point registers and the general integer registers in the EX unit. Load data is written to the register file after going through the load convert (LDCVT) logic to convert data to the internal register file format.

F. Vectorization

IceLake supports AVX 512 instructions, operations of which are performed through the ports 0,1 and 5, having sizes 256,256,512 bits, respectively. A 256-bit vector operation has a throughput of three instructions per clock cycle. 512-bit vector operations is performed by combining port 0 and port 1 and along with port 5, giving only a throughput of two instructions per clock cycle but able to process more data at a time than 256 bit instructions. Latency is the same for both types of instructions. The AVX 512 instruction set has the support for masking unneeded bits in a vector register. There is no extra cost in masking, latency remains the same, but there might be restrictions in register transfers.
Zen 2 doubles the width of the physical registers, execution units, and data paths to 256 bits. So, as to efficiently execute AVX 256 operations in single cylce, in Zen it takes 2 cycles to execute AVX 256. Zen 2 handles x87, MMX, SSE, and AVX and AVX 2 instructions. But still Zen 2 does not have the support for AVX 512 instructions.

G. Load Store Unit

	Load Queue	Store Queue
IceLake	128	72
Zen 2	44	48

IceLake

- The ratio of Load and Store Queue size favours loads more than stores. This according to Intel is a user focussed design where day to day workloads favour loads more than stores [6]
- Store to Load Forwarding takes place with a delay of 4 clock cycles. IceLake in addition uses Memory renaming (in section E 2) for forwarding with zero latency.

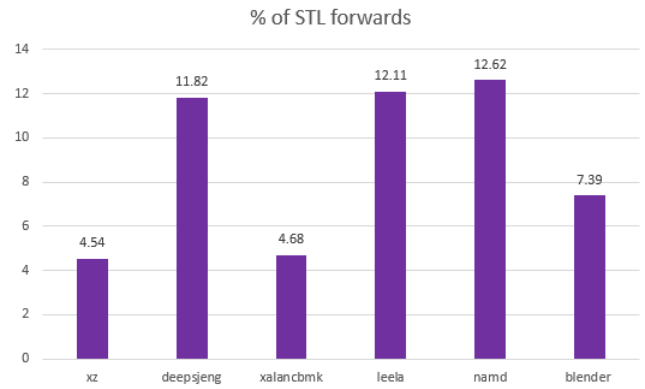
Zen 2

LS unit dynamically reorders operations, supporting both load

operations bypassing older loads and loads bypassing older non-conflicting stores. The processor uses address bits 11:0 to determine STLF eligibility. Avoid having multiple stores with the same 11:0 address bits, but to different addresses (different 47:12 bits) in-flight simultaneously where a load may need STLF from one of them. Loads that follow stores to similar address space should use the same registers and accesses should be grouped closely together, avoiding intervening modifications or writes to the base or index register used by the store and load when possible. Also, minimize displacement values such that the range will fit within 8 bits when possible. The LS unit can track up to 22 outstanding in-flight cache misses in the Miss Address Buffer (MAB).

H. Optimizations in LS Unit

1) Store-to-Load Forwarding: In both IceLake and Ryzen, the LS unit supports store-to-load forwarding (STLF) when there is an older store that contains all of the load's bytes, and the store's data has been produced and is available in the store queue. The load does not require any particular alignment relative to the store or to the 64B load alignment boundary as long as it is fully contained within the store. Performance evaluation of Benchmark programs on Zen 2



based processor shows that around 9% of the Load operations got its value through STL forwarding.

2) Superforwarding: Superforwarding in Zen 2 is the capability of a processor to send (forward) the results of a load instruction to a dependent floating-point instruction bypassing the need to write and then read a register in the FPU register file. As, floating point unit has high latency operations, this can help reduce lower the latency and also results in better saves the FP scheduler space, as most of the time, the FP execution unit is idle rather than the scheduler being empty.

IV. Memory Hierarchy

A. Cache Hierarchy

B. L1 Instruction Cache

L1 I Cache in both the processors are similar with 32 KB 8 way associative caches and a cache line size of 64 B. The major difference is in bandwidth, L1 I$ in IceLake fetches 64

	SkyLake	IceLake	Zen	Zen 2
L1 D-Cache	**32 KiB/core** **8 way** WriteBack 4 cycle latency	**48 KiB/core** **12 way** WriteBack **5** cycle latency	32 KiB/core 8 way Write Back	**32 KiB/core** **8 way** Write Back **4 – 5** cycle latency for Int **7 – 8** cycle latency for FP
L1 I-Cache	32 KiB/core 8 way 4 cycle latency	32 KiB/core 8 way 4 cycle latency	**64 KiB/core** **4 way**	**32 KiB/core** **8 way** 64 B line size
L2 Cache	256 KiB/core 4 way >= 14 cycles latency	512 KiB/core 8 way >= **14** cycles latency	512 KiB/core 8 way	512 KiB/core 8 way >= **12** cycle latency **Write Back** **Inclusive of L1**
L3 Cache	2 MiB/core 16 way Inclusive	**2 MiB/core** 16 way >= **50 cycles** latency **Inclusive**	4 MiB/CCX 16-way **40 cycles** average latency	**4 MiB/CCX, shared** 16-way **Write back, Victim Cache** 39 cycles average latency **Non inclusive**

B per cycle, whereas in Zen, only 32 B are fetched. This is one of the major bottleneck in Ryzen processors in Instruction fetching, which was tried to solve in Zen by increasing the cache size to 64 KB and decreasing the associativity to 4. In Zen 2 it was optimized further by bringing the size back to 32 KB and 8 way associativity. But, doubling the size of the Op Cache.

C. L1 Data Cache

L1 D Cache size and bandwidth was increased from SkyLake to support the increase in parallel load and store operations. The increase in associativity has increased the cache latency in L1 D-Cache. Zen 2 still keeps a smaller L1 D-cache so as to keep the cache hit latency minimum, so as to keep their clock cycle time lower. This is a write-back cache that supports two 256-bit loads and one 256-bit store per cycle which gets well utilized because of the three AGU units.

On Observation,

- Both IceLake and Zen 2 are adopting the same cache configuration.
- Due to cycle time advantage Zen 2 has smaller cache latencies compared to IceLake [10].

D. Optimizations

1) Micro Tag way Predictor: L1 data cache tags contain a linear-address-based microtag (utag) that tags each cacheline with the linear address that was used to access the cacheline initially. Loads use this utag to determine which way of the cache to read using their linear address, which is available before the load's physical address has been determined via the TLB. This linear address based lookup enables a very accurate prediction of in which way the cacheline is located

prior to a read of the cache data. This allows a load to read just a single cache way, instead of all 8. This saves power by not using the remaining 7 ways and reduces bank conflicts [13].

2) Hardware Prefetching: On misses, the L1 instruction cache generates fill requests for the naturally-aligned 64-byte block that includes the miss address and up to thirteen additional blocks. These blocks are prefetched from addresses generated by the Branch Predict unit. Because code typically exhibits spatial locality, prefetching is an effective technique for avoiding decode stalls.

3) Write-Combining Operations: Write-combining is the merging of multiple memory write cycles that target locations within the address range of a write buffer Zen 2 includes a Write Combining Buffer (WCB) that consists of multiple write buffers that are 64-byte aligned to cache-line boundaries. The write buffers aggressively combine multiple memory-write cycles of any data size that address locations within the 64-byte aligned regions. Maximum throughput is achieved by write combining when all quadwords or doublewords are valid and the processor can use one efficient 64-byte memory write instead of multiple 16-byte memory writes. The processor can gather writes from 8 different 64B cache lines (up to 7 from one thread).
WCB in Zen 2 is used to reduce bus usage and increase the memory bandwidth.

On Observation, all these memory optimizations in Zen 2 are done because of their smaller Cache fetch rate and larger (atleast 4 times) memory latency compared to IceLake [14]. So, Zen 2 architecture is designed to access main memory as much low as possible to minimize the average latency of memory access.

V. Memory Address Translation

A. TLB

Icelake has support for a large virtual address space (57 bits) and has 5 levels of paging. IceLake is the first processor from Intel to begin this support. Whereas, in Ryzen, both the virtual and physical addresses are of 48 bits. There is hardware support for large pages - 2 MB and 1 GB type pages in both the processors.

B. L1 TLB

The L1 TLB in both the processors are split into dTLB and iTLB, each for handling data and instruction address translation separately. As the clock cycle is faster in Zen 2, both the L1 iTLB and dTLB are kept fully associative with smaller size. Whereas, the L1 TLBs are partitioned into 3 sections - one for each type of page size and the it is 4 way associative for 4 KB and 2 MB pages and fully associative for 1 GB pages. Either way, L1 TLB consumes same amount of power in both the processors as one is smaller with full associativity and the other is larger with less associativity.

IceLake	Type	Entries	Associativity
L1- DTLB	4 KB	64	4 Way
L1- DTLB	2 MB	32	4 Way
L1- DTLB	1 GB	8	Full
L1 - ITLB	4 KB + 2 MB	8	Full
L1-ITLB	4 KB + 2 MB + 1 GB	16	Full
L2	4 KB + 2 MB	1024	8 Way
L2	4 KB + 1 GB	1024	8 Way

Zen 2	Type	Entries	Associativity
L1-DTLB	4 KB + 2 MB + 1 GB	64	Full
L2-DTLB	4 KB + 2 MB + PDE	512	8 Way
L1-ITLB	4 KB + 2 MB + 1 GB	64	Full
L2-ITLB	4 KB + 2 MB	2048	16 Way

One interesting thing is that, Zen 2 uses page coalescing concepts in its L1 and L2 TLBs together with large pages.

C. L2 TLB

The L2 TLB in IceLake is an unified TLB for both Instructions and Data address translations. It is again split into two types - one 1024 entry TLB supporting 4 KB and 2 MB pages and another 1024 entry TLB supporting 4 KB and 1 GB pages. Both of these 8 way set associative. The L2 TLB in IceLake was found to have a 100% hit ratio almost all the time for Data Address Translations and the TLB overhead was less for instructions too from our performance evaluation of KabyLake processor through Vtune.

Whereas, even L2 TLBs are split TLBs in Zen 2 - with one 512 entry 8 way associative dTLB and another 2048 entry 16 way associative iTLB. L2 TLBs in Zen 2 does not have a support for 1 GB pages and these are smashed into 1 MB pages in L2 TLB. Overall, Zen 2 has a larger L2 TLB than IceLake. Also, the L2 dTLB holds PDEs, which are used to speed up tablewalks by skipping three levels of page table reads.

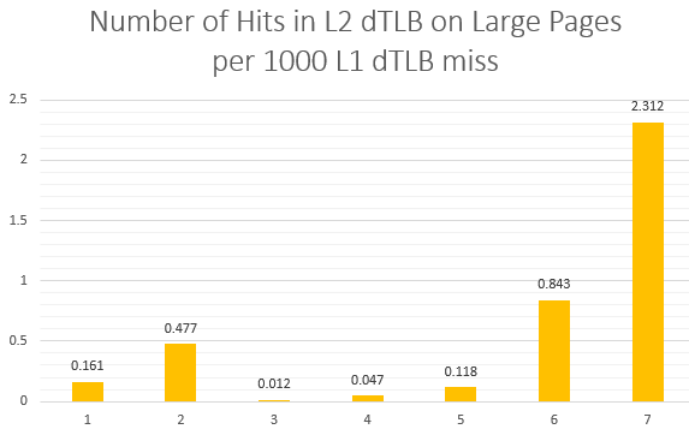
D. Hardware PageTable Walker

Zen 2 has two hardware page table walkers to handle L2 TLB misses. Misses can start speculatively from either the instruction or the data side. In addition to the PDE storage in the L2 dTLB, table walker includes a 64 entry Page Directory

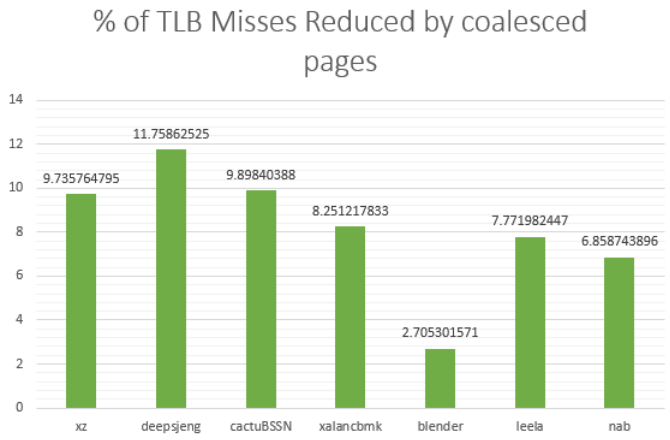
Cache (PDC) which holds page-map-level-4 entries (PML4Es) and page-directory-pointer entries (PDPEs) to speed up table walks. The PDC entries and the PDE entries in the L2 dTLB are usable by all tablewalk requests, including instruction-side table walks.

E. Optimizations

1) Large Pages: There is a support for Large Pages - 2Mb and 1 GB in both the processors. The evaluation performed on process based on Zen 2, shows that there is an average of 0.5 L2 dTLB hits on Large pages per 1000 L1 dTLB miss.



2) Coalescing Pages: Zen 2 supports page table entry (PTE) coalescing. When the table walker loads a PTE, which occupies 8 bytes in the x86-64 architecture, from memory it also examines the other PTEs in the same 64-byte cache line. If a 16-Kbyte aligned block of four consecutive 4-Kbyte pages are also consecutive and 16-Kbyte aligned in physical address space and have identical page attributes, they are stored into a single TLB entry greatly improving the efficiency of this cache. Our performance evaluation on Zen 2 shows that,



coalescing of pages helps in reducing an average of 8% of L2 dTLB misses. Also, the page coalescing technique helped in reducing the number of dTLB misses per 1000 instruction in AMD in xalancbmk benchmark program.

VI. Branch Prediction Unit

Branching can reduce throughput when instruction execution must wait on the completion of the instructions prior to the branch that determine whether the branch is taken. The processor integrates logic that is designed to reduce the average cost of branching by attempting to predict the outcome of a branch decision prior to the resolution of the condition upon which the decision is based. Intel keeps almost all of its details of Branch Prediction Unit as a secret.

A. Next Address Logic

The next-address logic in Zen 2 determines addresses for instruction fetch. When no branches are identified in the current fetch block, the next-address logic calculates the starting address of the next sequential 64B fetch block. This calculation is performed every cycle to support the 64 byte per cycle fetch bandwidth of the op cache. When branches are identified, the next-address logic is redirected by the branch target and branch direction prediction hardware to generate a non-sequential fetch block address. One thing to note here is that, fetch unit is able to fetch a maximum of 32 B per cycle. So, Next Address Logic stays ahead of fetching and also starts it's branch prediction early on.

B. Branch Target Buffer (BTB)

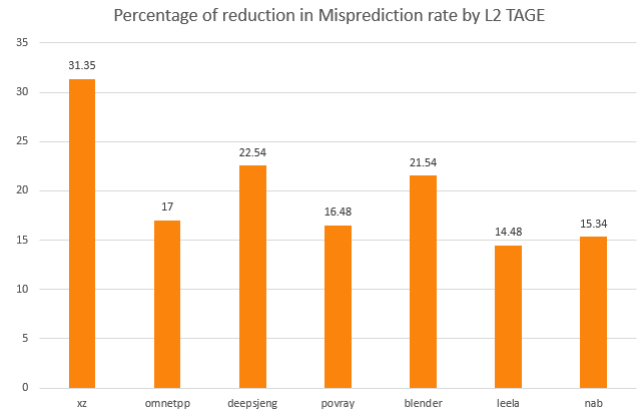
BTB in Zen 2 is a three-level structure accessed using the fetch address of the current fetch block. Each BTB entry includes information for branches and their targets. Each BTB entry can hold up to two branches if the branches reside in the same 64-byte aligned cache line and the first branch is a conditional branch. Each level of BTB holds an increasing number of entries, and prediction from the larger BTBs have higher latencies.

C. Dynamic Conditional Branch Predictor

Zen 2 introduces a TAGE(TAgged GEometric length predictor) predictor [8] in second layer along with a hashed perceptron predictor in BPU, whereas, in IceLake, from analysis it is assumed that a TAGE predictor is used, TAGE uses many prediction tables each indexed using a hash function of the Program Counter and some bits from the Global Branch History taken in a geometric series for each table. The prediction from the table with longest global history match is used. The predictor is found to be effective in modern day architectures but requires considerable time to build the history therefore Zen 2 uses TAGE as a second level predictor.

The introduction of a Level 2 TAGE predictor is effective as it lowers misprediction rate by L1 Perceptron predictor by 30% reported by AMD, the average is measured to be around 20% on the benchmark programs.

Misprediction penalty in Zen 2 is 12 to 18 cycles, on average 16 cycles. It is found to be 15 to 20 cycles in IceLake. The performance evaluation of Zen 2 on SPEC benchmark shows that, the average misprediction rate is 2.5%. Our analysis of Branch predictor with Microbenchmark programs in Zen 2 shows that, it predicts any conditional branches, nested if-else



statements, loops and nested loops very accurately with around 0.5 - 1% misprediction rate. There is 1 misprediction per outer loop at the end of it.

D. Return Address Stack (RAS)

In Zen 2 32-entry return address stack (RAS) to predict return addresses from a near call. Far control transfers (far call or jump, SYSCALL, IRET, etc.) are not subject to branch prediction. 31 entries are usable in single-threaded mode, 15 per thread in dual-threaded mode. At the address of a CALL instruction the address of the following instruction is pushed onto the stack, at a RET instruction the address is popped. The RAS can recover from most mispredictions, is flushed otherwise. It includes an optimization for calls to the next address, an IA-32 idiom to obtain a copy of the instruction pointer enabling position independent code.

E. Indirect Target Predictor

Zen 2 1024-entry indirect target array used to predict the target of some nonRET indirect branches - such as function calls. If a branch has had multiple different targets, the indirect target predictor chooses among them using global history at L2 BTB correction latency.

VII. Bottlenecks

The design principle of both the processors is to maintain simplicity in design and keep power consumption minimal, which as we can see, they both have achieved in most of the feature. Still in both the processors, frontend seems to be the bottleneck. Theoretically, the retire width and issue width is greater than the dispatch width on both the processors. Also, these two microarchitectures are struggling with a complex decoding unit, which consumes more power and also increases latency.

VIII. Conclusion

We have presented a detailed microarchitectural analysis of IceLake and Zen 2. Both processors differ in various aspects ranging from branch predictors to execution units and memory hierarchy. The processors gain the advantage of a smaller process node from their previous generation processors. The microarchitectures of both processors differ in aspects like the partitioning of the execution units, differences in the interpretation of a primitive operation etc. Apart from the differences in the microarchitecture, specific optimizations are found to be common among both the processors, like the usage of Macro Fusion, Large Pages and the usage of Op caches for already decoded operations. Both Intel and AMD are trying to imitate each other in terms of these optimizations, whose latest example could be seen in memory disambiguation technique used in IceLake. Both companies try to improve upon their processors by adding more and more features as well as optimizing already existing features to their base microarchitectural design so as to provide more performance within a power budget in each iteration. It is impossible to determine which processor is the best as both have its own advantages as well as disadvantages due to design which can be seen in real life workloads

References

[1] https://www.agner.org/optimize/microarchitecture.pdf
[2] https://en.wikichip.org/wiki/intel/microarchitectures/sunny_cove
[3] https://en.wikichip.org/wiki/amd/microarchitectures/zen_2
[4] https://www.amd.com/en/technologies/zen-core
[5] Anandtech "AMD Zen 2 Microarchitecture Analysis", https://www.anandtech.com/show/14525/amd-zen-2-microarchitecture-analysis-ryzen-3000-and-epyc-rome
[6] AnandTech, "Examining Intel's IceLake Processor", https://www.anandtech.com/show/14514/examining-intels-ice-lake-microarchitecture-and-sunny-cove/
[7] https://medium.com/adamedelwiess/high-performance-computer-architecture-16-unified-reservation-stations-weakest-link-7dd6e833cece
[8] https://www.irisa.fr/caps/people/seznec/L-TAGE.pdf
[9] https://shorturl.at/rCHI2
[10] https://www.agner.org/forum/viewtopic.php?t=48
[11] https://www.anandtech.com/show/14664/testing-intel-ice-lake-10nm/2
[12] https://www.agner.org/forum/viewtopic.php?t=56
[13] Software Optimization Guide for family 17h https://developer.amd.com/resources/developer-guides-manuals/
[14] https://news.ycombinator.com/item?id=20370636