Theme Article

Samsung M3 Processor

Jeff Rupley, Brad Burgess, Brian Grayson, and Gerald D. Zuraski Jr.
Samsung

***Abstract*—The M3 Processor is Samsung's third-generation custom microarchitecture. As performance demands continue to grow, major design improvements are required. In this generation, the core is enhanced with a 6-wide microarchitecture, deeper out-of-order resources, and faster instructions. The M3 delivers a new level of performance to the Android eco-system.**

■ **The competition for** a mobile smartphone is adding performance at a rapid pace, and the design team at the Samsung Austin R&D Center (SARC) is determined to respond. Building upon the first generation M1[1] design that was preparing for tape out in late 2014, the initial planning for the subsequent generation microarchitectures started in Q2 of 2014. The second-generation (M2) was originally planned as a shrink of the M1. RTL work for the third-generation (M3) begun starting in Q1 of 2015. By Q3 2015 it became clear that Samsung could opportunistically improve M2 by pulling a subset of features from the next generation. The third generation (M3) goals were reset to a much higher target.

The M3 processor is designed for flagship Samsung smartphones. Building upon the prior designs, the M3 has widened the bandwidths, deepened the out-of-order window, and improved latencies in both core operations and cache hierarchies. The result, launched early in 2018 and fabricated in Samsung 10LPP process technology, is a highly competitive design that achieves high performance with efficient power expenditure.

Digital Object Identifier 10.1109/MM.2019.2897556
Date of publication 5 February 2019; date of current version 15 March 2019.

GOALS

The performance goals for the reset M3 program were to push up IPC on a broad set of workloads within the available schedule, without losing frequency versus the prior generation. Power efficiency was prioritized over absolute power. Area growth was a concern, but acceptable if good performance gains could be shown.

PERFORMANCE

The performance analysis effort for the M3 processor was broken into three phases, each of which requires different abilities from the analysis team: early trailblazing/feature-mining, implementation tradeoffs, and final correlation/polish. In reality, the analysis phases for M3 were not crisply defined, and the correlation phase of M3 completely overlapped the trailblazing phase of the next core (M4), just as the correlation of prior designs overlapped with the beginning of M3.

Early Trailblazing and Feature-Mining

For the M3 processor performance effort, workloads that span both the current mobile ecosystem and industry-standard CPU benchmarks were used. For the mobile side, core benchmarks such as Geekbench V4 (GBv4) and AnTuTu were chosen, as well as web/system benchmarks like Octane, Sunspider, BBench, and Browsermark. These were augmented with internal real-world mobile workloads. From the CPU industry side, Dhrystone, CoreMark, SPEC CPU2000, SPEC CPU2006, and others were analyzed. Note that this workload collection was constantly evolving during the M3 program, as new compiler or library drops, workload revisions, and new benchmarks were released.

The M3 processor is designed for flagship Samsung smartphones. Building upon the prior designs, the M3 has widened the bandwidths, deepened the out-of-order window, and improved latencies in both core operations and cache hierarchies.

Due to the yearly cadence of the mobile processor market, the analysis team continued to look into deficiencies of the predecessor designs throughout their design timeline, using a cycle-accurate simulator. Any insights that could not be corrected in the M1/M2 family due to schedule or area had been put on the shelf for consideration on M3. The team had also been building a long-term roadmap of more ambitious microarchitecture improvements, which might take years to reach their final evolution. Performance modeling of several of these features began during the M1/M2 projects. As soon as the primary focus could be shifted to M3 performance analysis, there was a large backlog of ideas and partial model implementations to sift through.

Early in the M3 discussions, it was decided to widen the machine to 6-wide throughout (decode/rename/dispatch/retire). Additionally, the depth of the out-of-order window was grown, anchored by a target reorder buffer (ROB) size of 228 entries. These large increases in width and depth were chosen based on a mix of what was showing sufficient promise on IPC increase while still being on the edge of what the design team thought was buildable at frequency targets.

The M1/M2 had been balanced for a 4-wide core and ~100 entry ROB, so the M3 had to be rebalanced for the 50% growth in width and more than 2× depth. This required a series of studies for all of the major resources in the machine. Due to the interconnected relationships of these resources, it took an iterative approach to collectively tune all of these given the timing/area/schedule constraints.

An initial plan of record (POR) was drawn from a backlog of ideas, machine-growth studies, and incidental findings. This (M3 POR) was created as one configuration of the cycle accurate simulator and changed constantly as new changes were vetted and approved.

In order to not allow feature decisions to be adversely affected by the constantly evolving POR, an additional configuration was created of the performance model called "M3+" that was more idealistic, such as additional ROB entries and renames, lower latency caches, and better branch prediction. Vetting ideas against both Configurations (M3 POR and M3+) allowed higher confidence on whether an idea that might not meet be currently profitable on the evolving M3 could still be useful if other bottlenecks could be addressed.

Implementation Tradeoffs

The second phase of analysis was a gradual transition to tradeoff studies, as the design team began implementation. These studies were usually quick studies looking at an individual sub-block or feature. For example, the initial POR suggested a load/store hazard queue size larger than the final size of 32, but feedback suggested it would require significant effort to close timing unless it was reduced in size to 32 entries. A tradeoff study was performed to see how much performance was lost at each size reduction, in order to allow the team to make an appropriate judgment call between performance loss and extra effort to close timing. For some features, the design team was able to devise a different solution or a policy change to get most of the performance without an extra effort.

From simulated IPC comparisons at constant frequency taken at the end of the project across ~4800 unweighted traces, the M3 achieved an average of 2.01 IPC, compared to an M2 at 1.26. IPC will vary across specific applications and

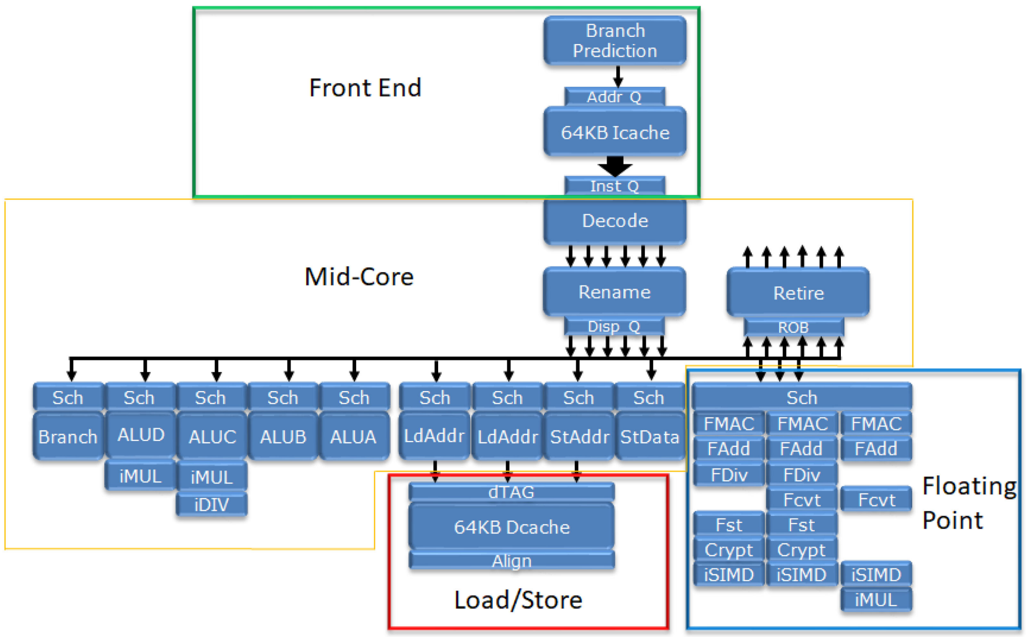

Figure 1. M3 core block diagram.

benchmarks, but this broad metric shows nearly a 60% speed-up in one generation.

Final Correlation and Polishing

The last phase of the analysis is the correlation effort. As the RTL design matured, RTL simulations of thousands to low millions of clocks were performed, to verify that the performance of simple bare-metal benchmarks met the desired projected performance. A set of around 700 workload snippets were used, from the simplest Dhrystone simulations to instructions extracted from real-world mobile workloads. Nightly performance regressions are run so that any backslide of performance can be caught as soon as possible.

Once the RTL was sufficiently functional to boot, workloads are run on a full system emulator (including M3 cores and little cores, as well as a complete DDR model). This allows visibility for behaviors over billions of clock cycles, allowing full workloads like Geekbench or SPEC to be run, which frequently illuminates problems that cannot be seen on shorter instruction sequences. In particular, the M3 design had a branch predictor update bug that took millions of updates before its performance degradation became noticeable, which would not have been detectable through RTL simulations alone. Other noticeable areas of tuning from the emulation effort include prefetchers and L2/L3 cache policies.

M3 CORE MICROARCHITECTURE

The M3 implements ARMv8-A.[2] Figure 1 shows an overview of the M3 core microarchitecture, which is a 6-wide out-of-order design. The core features 64 KB L1 caches, advanced dynamic branch predictors, and an aggressive data prefetcher. While the sustained instructions per cycle is 6, a peak of 11 operations can be issued (4 ALU, 1 branch, 2 load, 1 store, and 3 Float/vector) in any given cycle.

Front End (FE)—Branch Prediction and Instruction Fetch

The microarchitecture of the M1 branch predictor and instruction cache was conceived considering three classes of applications: small kernels that fit in a micro branch-target-buffer (BTB), moderate sized applications that fit in the main BTB, and large applications that get coverage from an L2 BTB. The design of the neural net branch predictors of the micro and main BTBs are inspired by earlier work on perceptron-based branch predictors.[3,4] The new challenge for M3 was to keep a wider, more capable machine full of useful instructions.

Using a hierarchical approach allows tradeoffs between capacity, latency, bandwidth, and power. Kernels with a smaller footprint can be run in the micro BTB at low latency, high bandwidth, and low FE power. On M3, the micro BTB capacity was doubled to 128 entries to cover some slightly larger kernels. The accuracy of the micro BTB conditional branch predictor was improved by increasing the local history length, doubling the capacity of the weight tables, and tuning confidence mechanisms. As with prior designs, the M3 micro BTB includes a loop predictor and a return address stack.

As with M1/M2, the main BTB holds 4K branch prediction records for moderate sized applications. The main BTB latency to predict taken branches was reduced with a number of micro-architectural optimizations. Optimization of the storage records allowed an increased opportunity to predict 2 branches per cycle similar to the operation of the micro BTB. The accuracy of the main branch predictor was improved by doubling the capacity of the weight tables, trading of dynamic range for reduced aliasing, and by tuning the intervals used to select the weights for the neural network. Further improvements were also made to the indirect branch predictor. The M3 main BTB also includes a 64 entry return stack.

In order to better optimize for large-footprint applications, the sizes were doubled for the L2BTB and instruction TLB, and the instruction fetch bandwidth from L2 cache was doubled to 32 bytes/cycle.

The fetch pipeline for the 4-way associative 64-KB instruction cache is decoupled from the branch prediction pipeline by an address queue. The instruction cache bandwidth from FE to decoder was doubled to 48 bytes per cycle to support the more capable machine.

The net of the changes to the predictors yielded a 15% reduction in branch mispredicts per 1000 instructions (MPKI). Overall misprediction rates across 4800 traces were reduced from 3.92 MPKI to 3.29 MPKI. Improved accuracy of the branch predictor helped better utilize the larger out-of-order window, as well as compensate for the additional pipe stages that were added to build a wider machine.

Mid-Core (MC)

In Samsung designs, the “MC” refers to sections including instruction decode/rename/dispatch, the integer schedulers/register file/execution, and the retirement blocks. These blocks were redesigned for increased bandwidth and out-of-order capabilities, as well as instruction latency reduction.

Decode, Rename, and Dispatch. The decoder translates instructions to micro-operations (μops), and can support up to 6 instructions per cycle versus 4 in M1/M2. A small number of instructions require cracking into multiple μops. One example is the A32 “load multiple registers” instruction, which is primarily broken up into a series of “load register pairs” μops. Several fusion idioms are supported for improved perf/power.

The rename and dispatch blocks were improved to handle 6 μops/cycle, as well as supporting larger renames for the integer and vector physical register files (PRFs).

The dispatcher is the last in-order part of the design before retire and primarily handles distributing μops to the various schedulers as well as to the ROB. The dispatcher checks to make sure all required resource credits are available. It also handles several special cases, such as “multidispatch”. This occurs when a single μop will be sent to more than one scheduler—for example, integer stores are sent to the store address scheduler as well as the integer store data scheduler.

Integer Scheduler, Integer Register File, and Integer Execute. The integer scheduler consists of 9 distributed schedulers, versus 7 in prior SARC designs. There are 4 ALU (1 additional), 1 simple branch, 3 AGU (1 additional), and 1 integer store data schedulers. The total integer scheduler capacity of 126 μops is more than 2× deeper than in M1/M2 designs.

Integer PRF is 192 entries of 64 bits, versus 96 in prior designs. While 32 entries are required to map architectural state, the remainder are available for use as speculative renames as part of building a large out-of-order window.

Integer execution has improved bandwidth by adding a 4th ALU as well as a 2nd integer multiplier. Faster computations are done by converting some 2-cycle latency μops on M1 into 1-cycle latency on M3. Also, the integer divider now uses radix 16 (iteration loop of 4 bits/cycle) versus prior design of radix 4 (2 bits/cycle).

Retire. The retire unit contains the ROB with a 228 μop capacity, up from 100 on M2. Up to 6 μops can be retired per cycle, up from 4 μops/cycle on M1/M2.

Floating Point Unit (FPU)

The FPU handles floating point and SIMD (including vector integer) instructions in a coprocessor style design. The design has been widened to sustain up to 3 ops per cycle (dispatch, schedule, and execute) versus 2 in M1/M2.

The M3 FPU scheduler capacity at 62 entries is nearly 2× the prior designs. The vector PRF has also grown, to 192 entries of 128 bits each. This combination allows for a much deeper out-of-order execution window.

Refer to Table 1 below for a summary of improvements to latency and bandwidth in the M3 FPU execution units. All execution units can work on 128-bit data and are generally fully pipelined. The floating-point multiply (FMUL) operation is handled inside the floating-point multiply-accumulate (FMAC) unit at 1 cycle less latency than an FMAC operation. With three 128-bit FMAC units, M3 can compute 24 SP or 12 DP floating-point operations per cycle, double the amount in M1/M2 (which saturates at one FMAC plus one FADD).

The primary exception to the pipelined nature of the FPU is the divider unit. The redesign of the floating-point divider to a radix64 algorithm allows computing of 6 bits/cycle of result, a 3× increase over the older radix 4 design.

Table 1. FPU execution summary.

	Latency (typical)		Execution Bandwidth (128-bit execution units)	
	M1/M2	M3	M1/M2	M3
FMAC (FMUL)	5 (4)	4 (3)	1	3
FADD	3	2	1	3
Fconvert	2	2	1	2
Fdivide	Variable (Radix4)	Variable (Radix64)	1	2
Vector Int ALU	1	1	2	3
Vector Int Mul	3	3	1	1
Crypto	1	1	1	2
FP Loads	5	5	1	2
FP Stores			1	2 (for store-pairs)

Load/Store Unit (LSU)

The major improvements of the M3 LSU can be categorized by bandwidth improvements, increased out-of-order capabilities, and capacity/latency improvements.

One of the larger improvements in M3 performance was to add a 2nd Load port. This allowed the M3 to support up to two independent loads plus one store per cycle. Additionally, the transfer bandwidth between L2 and L1 was doubled to 32B per cycle.

The out-of-order capabilities were improved by increasing scheduler depths and doubling the Store Buffer to 32 entries. Up to 12 outstanding misses are supported.

Effective data latency was improved by growing the data cache capacity from 32 KB/8-way on M1/M2, to 64 KB/8-way on M3, without growing load latency. The multistride pattern prefetcher was enhanced and hybridized with a new engine that learns additional types of patterns.

The TLB hierarchy was also upgraded. While the first level 32-entry dTLB was too timing critical to grow, a new mid-level 512-entry data TLB was added, and the unified (instruction and data) L2 TLB was quadrupled to 4K entries. These improvements allow for many larger footprint applications to fit within the core TLBs, without resorting to the higher latency of the hardware table walk engine.

Core Pipeline

The primary core pipe for integer execution and 4-cycle loads is shown in Figure 2.

The project timing goals to hold frequency despite the design stress from increased width and depth led to tradeoffs. While in many areas the team was able to redesign to accommodate the goals, by the end stages were added. The net branch misprediction pipeline on M3 of 16 cycles is 2 cycles more than M1/M2.

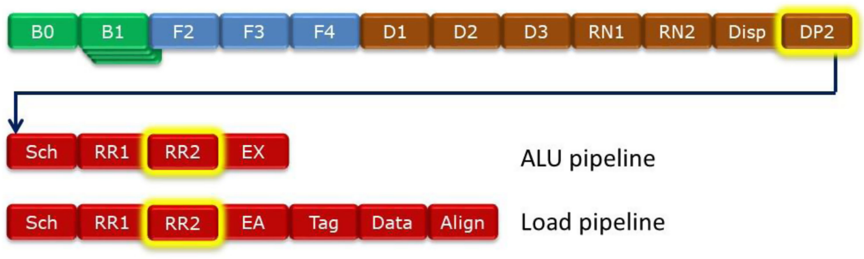

Figure 2. M3 core pipeline, with stages added to M3 highlighted.

The DP2 stage was added as part of routing dispatch to the μop schedulers and resultant stress on the allocation into those schedulers. The RR2 stage was added to deal with the larger PRF, as well as to better decouple the scheduler loop from the ALU execution loop.

M3 CACHE HIERARCHY

The M3 Cache hierarchy was significantly redesigned for improved effective latencies, and the ability to support increased capacities. Bandwidth links were generally doubled to 32B per cycle at all levels inside CPU cluster.

Figure 3 shows the 4 CPU cluster used in M1/M2 with a 2 MB shared L2 (sL2) with 4 banks b0-b3. The configuration in M3 was redesigned with 512 KB private L2's per core, backed by a 4 MB shared L3 (sL3) (broken into 41 MB banks) which are exclusive of the private L2 (pL2).

The L2 is inclusive of the data cache. It supports a 12 cycle latency versus the 22 cycle 2 MB shared L2 on previous designs.

The M3 L3 consists of a 4 MB shared cache, which is exclusive of L2. The L3 cache is a nonuniform cache access design with a typical L3 access of 25 cycle additional latency beyond the L2 (net typical 37 cycles from L1 address generation to L3 hit, and back to consumer operation). The L3 is designed in a 4-bank slice topology that allows for improved configurability for potential alternative configurations.

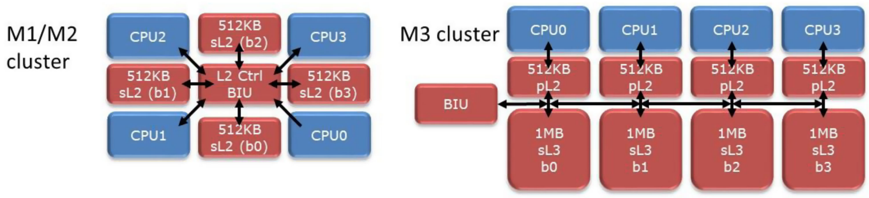

Figure 3. Comparison of M1/M2 with M3 CPU/cluster organization.

The M3 Bus Interface Unit supports up to 80 outstanding transactions versus 56 on M1/M2.

SILICON PERFORMANCE

While there are many ways to compare performance, Geekbench is an important CPU stress test for smartphones. Combining both IPC gains and a higher productized frequency, M3 achieves a net speedup on GBv4 single thread of >85% versus M2.

Figure 4 shows a Geekbench v4 single thread performance comparison relative to 2.3 GHz M2 (as productized in Samsung Galaxy S8) versus the 2.7 GHz M3 and 2.8 GHz A75 (both as productized in the Samsung Galaxy S9).

POWER

While the M3 process technology offered some power relief versus the M2 (3%–5%), with the large performance increases demanded in M3 it was known that absolute power would go up. The goal of the power team was to drive performance/power efficiency in the right direction as much as possible, using techniques like clock gating.

However, at the system level, additional tradeoffs can come into play. Mobile smartphone systems consist of a mix of big and little cores. The system tends to start tasks on the little core and only migrate to the big core when performance demand justifies it. Also, the system software includes DVFS controls that boost and throttle cores as needed, and tasks will tend to start at a lower frequency and only get boosted as further performance is demanded. Many user interactivity scenarios only require brief bursts of higher performance to satisfy demand, which can be dealt with by temporarily boosting core frequency.

Figure 5 shows silicon measured GBv4 single-thread subtest comparison of perf/W between M2 (productized at 2.3 GHz) and an M3 at comparable voltage/frequency (2.3 GHz) as well as the productized frequency (2.7 GHz). Across these tests, at similar V/F points the M3 has improved

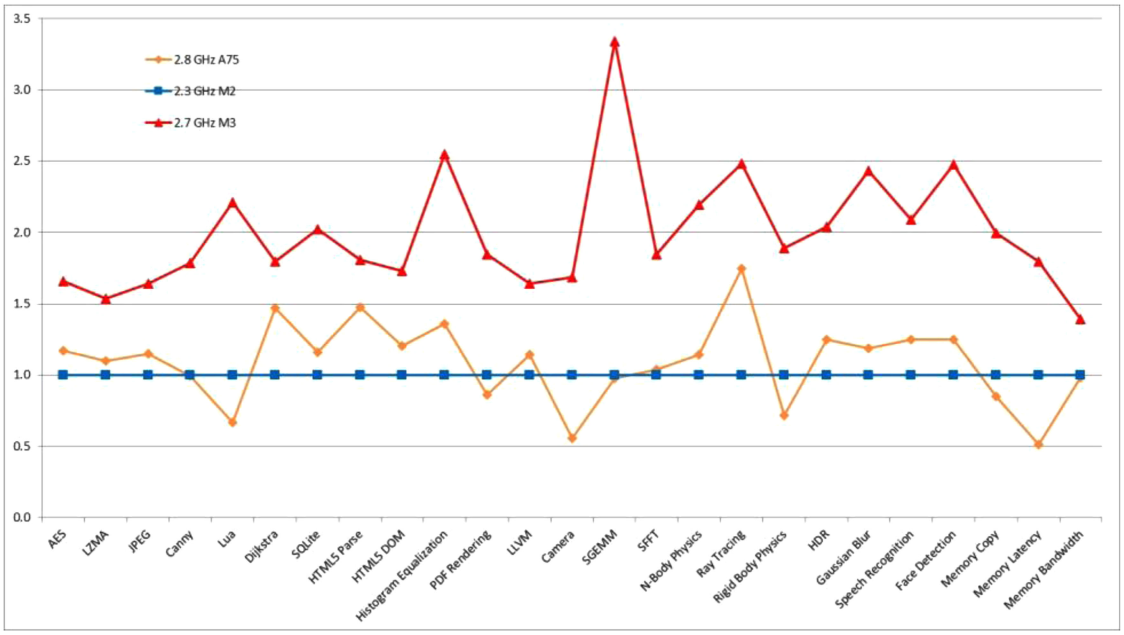

Figure 4. Silicon performance comparison of GBv4 single-thread subtests, normalized to M2.

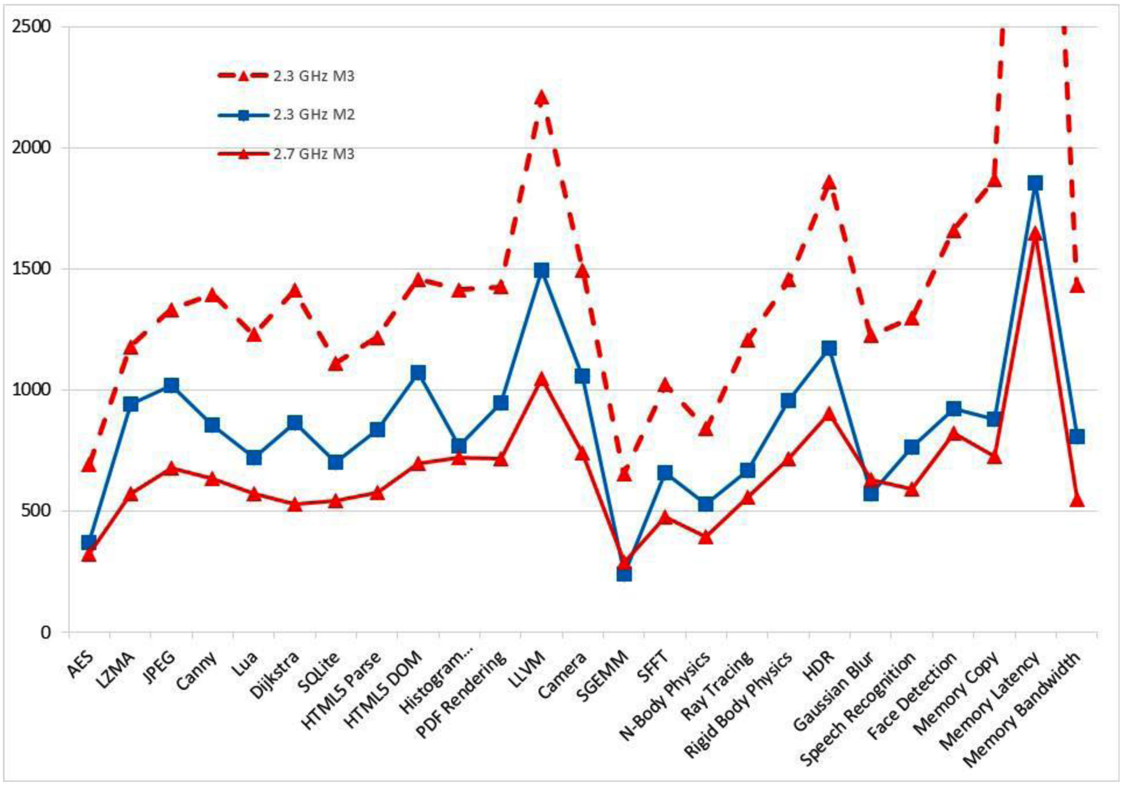

Figure 5. Silicon Perf/W comparison across GBv4 single thread subtests.

efficiency over M2. At higher/voltage frequency (comparing the M3 dashed line at 2.3 versus M3 solid line at 2.7 GHz), the power costs grow faster than the performance. While this can be useful for brief periods, the system software must be careful to control total power over time.

CONCLUSION

Smart phone technology is advancing rapidly and the competition is not standing still. To service the annual cadence of the premium smartphone market and to provide customers with better experiences, expect SARC to deliver competitive CPU performance uplifts and efficiency improvements every year.

REFERENCES

1. B. Burgess, "Samsung's Exynos-M1 CPU," in *Proc. Hot Chips 28 Symp.*, 2016, pp. 1–18.
2. ARM Architecture Reference Manual ARMv8, ARM, 2013.
3. D. Jimenez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proc. 7th Int. Symp. High Perform. Comput. Archit.*, 2001, pp. 197–206.
4. D. Jimenez, "Strided sampling hashed perceptron predictor," in *Proc. JWAC-4: Championship Branch Prediction*, 2014

Jeff Rupley is a Senior Principal Engineer with Samsung, Austin, TX, USA, working as a Lead Architect for future cores. His interests include CPU microarchitecture and embedded computing. He received the M.S. degree in electrical engineering from Purdue University, West Lafayette, IN, USA in 1996. He is a member of IEEE and ACM. Contact him at jeff.rupley@samsung.com.

Brad Burgess is a Senior Vice President and Chief CPU Architect with Samsung, Austin, TX, USA. He received the M.S. degree in electrical engineering from Texas A&M University, College Station, TX, USA in 1985, and is a member of IEEE. Contact him at b.burgess@samsung.com.

Brian Grayson is a Senior Principal Engineer with Samsung, Austin, TX, USA, working as a Lead for the performance analysis team. He received the Ph.D. degree in electrical and computer engineering from University of Texas at Austin, Austin, TX, USA in 1999, and is a member of IEEE and ACM Contact him at b.grayson@samsung.com.

Gerald D. Zuraski Jr. is a Senior Principal Engineer with Samsung, Austin, TX, USA, working as a Lead Architect for future cores. His research interests include CPU microarchitecture, GPU microarchitecture, and machine learning. He received the B.S. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA in 1992. Contact him at g.zuraski@samsung.com.