2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA) | 978-1-6654-3333-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISCA52012.2021.00012

Energy Efficiency Boost in the AI-Infused POWER10 Processor

Industrial Product

Brian W. Thompto, Dung Q. Nguyen, José E. Moreira, Ramon Bertran, Hans Jacobson, Richard J. Eickemeyer, Rahul M. Rao, Michael Goulet, Marcy Byers, Christopher J. Gonzalez, Karthik Swaminathan, Nagu R. Dhanwada, Silvia M. Müller, Andreas Wagner, Satish Kumar Sadasivam, Robert K. Montoye, William J. Starke, Christian G. Zoellin, Michael S. Floyd, Jeffrey Stuecheli, Nandhini Chandramoorthy, John-David Wellman, Alper Buyuktosunoglu, Matthias Pflanz, Balaram Sinharoy, Pradip Bose

International Business Machines Corporation, Armonk, NY, USA

***Abstract*—We present the novel micro-architectural features, supported by an innovative and novel pre-silicon methodology in the design of POWER10. The resulting projected energy efficiency boost over POWER9 is 2.6x at core level (for SPECint) and up to 3x at socket level. In addition, a new feature supporting inline AI acceleration was added to the POWER ISA and incorporated into the POWER10 processor core design. The resulting boost in SIMD/AI socket performance is projected to be up to 10x for FP32 and 21x for INT8 models of ResNet-50 and BERT-Large. In this paper, we describe the novel methodology deployed and used not only to obtain these efficiency boosts for traditional workloads, but also to infuse AI/ML/HPC capability directly into the POWER10 core.**

***Index Terms*—POWER10, energy efficiency, AI acceleration, microprocessor design methodology, pre-silicon modeling.**

I. Introduction

POWER10 is the latest processor in IBM's POWER systems roadmap [43]. The processor chip (Fig. 1) is a 7nm CMOS design, with a die size of 602 mm^2, using an 18-layer metal stack. It is available in single-chip or dual-chip sockets. The top-level system architectural attributes are captured in Table I. Two of the major distinguishing operational features of POWER10 over POWER9 include: (a) significant improvements in energy efficiency; and (b) AI enhancement of the core featuring ISA extensions and inline MMA (Matrix-Multiply Assist) acceleration. In this paper, we focus on these two aspects of the micro-architectural and design enhancement of POWER10. Our principal focus in this paper is on the fundamental building block: namely, the processor core.

The new micro-architectural design is built upon the foundation of a novel, multi-faceted pre-silicon power-performance modeling and analysis methodology. The baseline for comparison in this paper is the prior-generation POWER9 processor [1], [23], [32], [39], [45]. In POWER10, the team adopted a radically new (in-flight RTL optimization) pre-silicon design methodology, in which the evolving RTL model was used directly and continuously to track and tune power-performance efficiency. This was in lieu of relying on the cycle-accurate micro-architecture-level performance model alone during early-stage definition. As such, in this revamped methodology, there was a need to work with representative abstractions (proxies) of key benchmark suites, such that the relatively slow RTL simulation model could be used to project power-performance numbers with acceptable levels of accuracy. The benefit of having the prior-generation POWER9 hardware as an accurate reference platform, allowed us to develop validated POWER9 models, with subsequent scaling to project POWER10 specific numbers. At the same time, an innovative new power simulation methodology called APEX (exploiting IBM's AWAN accelerator platform [18]) that enabled a ∼5000× speedup over traditional RTL simulation, allowed us to validate full workload projections a bit later in the design cycle.

In terms of the scalar core performance, in this paper we focus only on SPECint codes [41] to show how the 2.6× boost in power-performance efficiency (Table I) was achieved. (The corresponding socket-level boost is up to 3×, relative to POWER9). For the MMA-enhanced core, the socket AI performance boost has been projected to be up to 21× for ResNet-50 and BERT-Large INT8 models, with an insignificant increase in per-core power. In the backdrop of these amazing numbers,

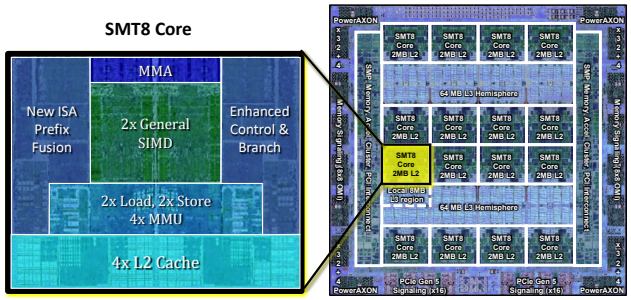


Fig. 1. POWER10 chip die photo with core inset. The primary focus in this paper is on the SMT8 core building block. Relative to the prior-generation POWER9 core, POWER10 has: 2x general SIMD capability; (new) Matrix-Multiply Assist (MMA) inline accelerator; 2x load and store processing capability, backed by 4x memory management unit (MMU) resource; 4x private L2 cache.

This paper is part of the Industry Track of ISCA 2021's program.

978-1-6654-3333-4/21/$31.00 ©2021 IEEE
DOI 10.1109/ISCA52012.2021.00012

TABLE I
POWER10 CHIP FEATURES & EFFICIENCY PROJECTIONS

Chip Attributes	Value (Enumeration)
Functional cores	15
SMT per core	8-way
L2 cache per core	2MB
L3 cache	Up to 120MB (chip)
Mem. Mgmt Unit (MMU)	4× relative to POWER9
Open Memory Interface	16 ×8 @ up to 1 TB/s
PowerAXON Interface	16 ×8 @ up to 1 TB/s
Energy efficiency (dual-chip socket)	Up to 3× relative to POWER9
Performance/watt (core)	2.6× relative to POWER9

the key contributions in this paper are:

- We provide a real-life industry perspective of how micro-architectural innovations, coupled with an agile pre-silicon power-performance methodology are used to achieve the quoted efficiency boost figures in one generation, in a technology-independent evaluation.
- We describe the rationale and micro-architectural innovations behind efficient AI-infusion at the core-level.

As an added benefit, the POWER10 experience has provided modern-day insights about how to scale for energy efficiency boost in the post-Moore/post-Dennard era such as:

- Power must be front and center from the start. Technology improvements and traditional clock gating efforts after mainline function entry are no longer sufficient. The microarchitecture must minimize wasted work, data movement, control overhead, and table lookups and provide power efficient acceleration for compute heavy workloads. Hence, POWER10 focused (among many other things) on improved branch predictors, removal of reservation stations, instruction fusion, Effective Address (EA) based L1 cache and power efficient inline AI acceleration support for AI/ML/HPC workloads.
- Designers need to shift to a mindset where all latch clocks are off by default and only enabled when needed.
- Efficient clock and data switching reduction efforts require tool support that provides continuous quick feedback for a wide range of different workload behaviors.
- Accurate absolute power projections in support of Workload-Optimized Frequency (WOF) and Power Frequency Limited Yield (PFLY)/Core Limited Yield (CLY) projections require chip models and large workload segments capturing the full chip behavior, which requires hardware-accelerated simulation capabilities.
- Enabling a detailed power information flow from actual implementation (RTL/Physical Design) to higher abstraction models is key to having a complete power/performance efficiency view over the different project stages.
- In addition to traditional synthetic test cases, early usage of RTL runnable proxies of real applications with high coverage is a must to permit early detection of core power/performance optimization opportunities.
- It is essential to consider Reliability, Availability and Serviceability (RAS) from an early stage of design in order to ensure a high level of soft-error rate (SER) resilience while minimizing power overheads.
- Full coverage of power behavior via a rich proxy workload set is needed to enable automated generation of models which will then speed up adoption at different levels in the design

II. CORE MICROARCHITECTURE OVERVIEW

In this section, we provide a high-level review of the POWER10 core design, in the evolutionary context of the prior-generation POWER9 design [32], highlighting some of the main contributors to the much higher efficiency.

A. Core Pipeline Depth Analysis

Selection of pipeline depth and associated clock frequency target places significant constraints on what microarchitecture structures can be used to provide a power/performance efficient design. It is hence one of the first decisions made in defining a new microarchitecture family. POWER9 [23], [32], [39] represented a brand new microarchitecture with POWER10 being a significant augmentation thereof but with a similar pipeline structure. To determine whether a change in pipeline depth, (or equivalently fanout-of-4 (FO4) per pipeline stage) was motivated going forward, a study was performed early in the POWER10 concept phase based on the mature POWER9 design. The POWER9 M0/M1 microarchitecture performance models (see Fig. 7) were used to analyze the optimal pipeline depth [42] in light of different target efficiency metrics as originally explained in [13], [52].

The pipeline-depth-dependent power model used detailed Einspower [20] reports separating out latch-clock, logic data-switching, array and register file components. These power contributors were individually scaled according to functions of the new target design pipeline depth. A range of different pipeline depths (FO4 per stage values) was explored for different core power targets to find the pivot points for potential product offerings as illustrated in Fig. 2. The modeling further accounted for the power limited frequency constraints to the core as dictated by the target power envelope of the chip. Thus, if the core power for a given FO4 depth exceeds the power envelope, the voltage and frequency is adjusted accordingly with a subsequent impact on the performance. The analysis in Fig. 2 showed that the optimal pipeline depth point held stable at 27 FO4 for the throughput metrics and range of power targets of interest (0.5x-1.0x of POWER9 baseline power) for the POWER10 core. While higher FO4 points were indicated as optimal for lower core power targets, those power/performance points were not of particular interest to current IBM POWER systems. Note that the earlier POWER4-based research [42] also illustrated the optimal pipeline depth to be quite stable, even under accuracy deviations in the early-stage power-performance modeling. As a result of this analysis, the base pipeline structure and nominal frequency target did not change significantly from POWER9 to POWER10.

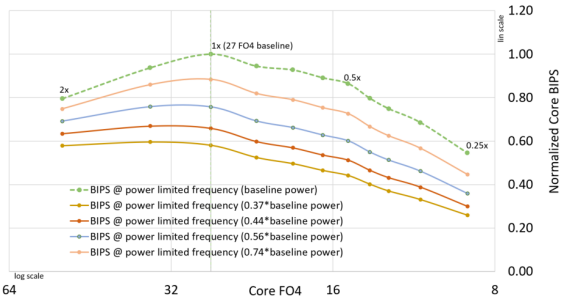


Fig. 2. Optimal Pipeline Depth Analysis: The x-axis shows pipeline depth expressed as logic FO4. The y-axis shows performance in BIPS (billions of instructions per second) at power limited frequencies for different power targets expressed as a fraction of baseline power. Performance is normalized to the baseline optimal FO4 and power target.

B. Improvements of POWER10 over POWER9

Fig. 3 shows the main processor core pipeline microarchitecture, as reported in [43]. POWER10 builds on the foundational modular architecture of POWER9 [32], and introduces a significant redesign of many micro-architectural aspects with a focus on performance efficiency. The target was to achieve at least a 25% boost in per-core throughput for general-purpose integer code, a significant improvement in single-thread performance and dramatically improve AI, ML, and HPC processing capability.

The design team was able to over-achieve on the above goal by delivering ∼30% additional throughput performance, while at the same time *reducing* power by ∼50% based on pre-silicon modeling. This computes to 2.6× increase in performance-per-watt per core. The figures quoted here are *iso-voltage and frequency*, in that it does not factor in any additional boost resulting from voltage-frequency advantages in going from the prior technology node (14nm) to the current (7nm). This improvement was sufficient to also enable up to 2.5× more cores per socket with operating efficiencies up to 3× measured at the socket level.

Obviously, for a design as complex as the POWER family, a large number of different micro-architectural design decisions cumulatively account for the large boost in performance and efficiency. In terms of performance, the enterprise-strength POWER10 core improves upon the POWER9 core in several directions. As indicated earlier in Fig. 1, the POWER10 core reflects selective doubling (or even quadrupling) of computational and cache resources. It doubles the traditional SIMD engine resources and corresponding load-store bandwidths, while integrating a brand-new inline AI/HPC accelerator - the Matrix Multiply Assist (MMA). POWER10 addresses larger working sets by increasing the L1 instruction cache, L2 and TLB sizes, while simultaneously reducing latency cycles across the entire cache hierarchy and TLB. POWER10 also provides deeper and wider out-of-order instruction windows and increases decode bandwidth by 33%. These and many other performance oriented improvements provided for a significant uplift in overall throughput and single thread performance along with dramatic improvements in AI / HPC throughput and efficiency.

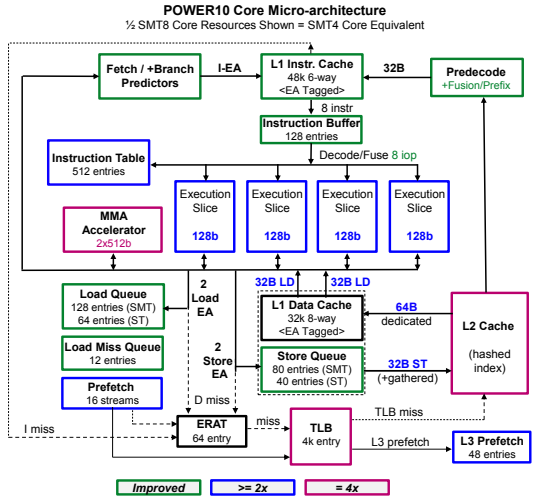


Fig. 3. POWER10 Core Microarchitecture.

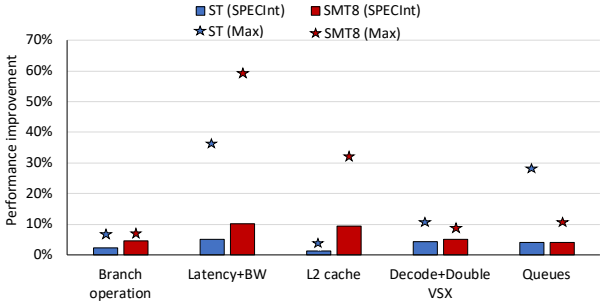


Fig. 4. Effect of design changes in different micro-architecture units from POWER9 to POWER10 and the consequent performance improvement. The results presented are averaged across various SPECint workloads for ST and SMT8 modes. The stars represent maximum gains seen across commercial, SPEC, Python and ISV workload groups of relevance to IBM Systems.

Fig. 4 shows the effect of select design changes in different units from POWER9 to POWER10, and the consequent performance improvement. Improvements are shown for several design changes, including branch prediction, cache latencies, memory bandwidth, L2 cache, SIMD engines, instruction decode and queue sizes. We observe an average performance benefit of 4% from optimized branch execution, 10% from improved latency and bandwidth, 9% from the increased L2 cache size, 5% from optimizing decode and SIMD engines and 4% from the increased queue sizes when running SPECint workloads on SMT8 cores. While Fig. 4 also identifies the impact to select workload groups, individual workloads of importance often see much more significant impacts from specific design changes. For example, we observe machine learning and analytic workloads that gain close to twofold the performance from doubling the number of VSX SIMD units per core.

Many of the fundamental structures and micro-architectural mechanisms were re-designed with power-efficiency as a focus. All major blocks were either re-designed or updated with structural efficiency changes that focused on reducing dynamic power consumption through switching reduction.

For instance, reservation stations were removed in favor of an unified sliced configuration for the main register files made from smaller building blocks. In the most prominent example, an unified register file was designed to hold the data for both general purpose (GPR) and vector-scalar (FPR/VSR) registers with a significant growth in out-of-order rename capacity, but trading off the addition of stages to the main execution pipeline compared with POWER9. The result was a significant reduction it total switching capacitance, and a net boost in performance. The improved structural elements are more efficient with only two write ports each, while the entire structure supports up to eight concurrent writes for each set of up to two threads.

Some of the highest impacts to overall performance efficiency came from changes that both boosted performance and reduced energy consumption per unit of work. Improvements to various aspects of branch prediction including the addition of new predictors for direction and indirect targets along with the doubling of selective prediction resources not only improved raw performance, but also reduced wasted / flushed instructions by 25% on average for SPECint in comparison to POWER9. For interpreted languages and business analytics workloads, the reduction is even higher, reaching 38%.

Significant changes for efficiency were made in the load-store (LSU), and memory management units (MMU) along with the instruction cache pipeline. The fundamental pipelines supporting load and store instructions as well as instruction fetch were re-designed with effective-address (EA) tagging in each L1 cache. The effective to real address translation is a relatively power-hungry operation that must be performed on each access to a real-address tagged cache. In POWER10, this address translation is only required for an L1 cache miss for both instructions and data. Additionally, the structure of the LSU was changed fundamentally from address-oriented units in POWER9 to slice-oriented units in POWER10 that were streamlined for the EA based cache. The result was not only a reduction in translation power, with a single translation pipeline for the core, but also reduced switching capacitance from basic load and store operations as they are able to utilize the cache index as an address proxy for typically accesses.

Across the core pipeline, a number of improvements in peak throughput and instruction window sizes were efficiently enabled with instruction pairing. The decode and completion pipelines for example, manage instructions in pairs to achieve higher throughput with reduced controls, enabling up to eight instructions per cycle versus six on POWER9. Additionally, the branch execution pipeline was completely re-designed and control flow and register renaming were merged into the same structures that manage basic arithmetic operations. This resulted in reduced energy overhead through structure removal, and also in improved performance by allowing for the reduced (as low as zero cycles) latency exchange of data between general purpose (GPR) and branch target registers.

A dramatic expansion in instruction fusion capabilities also paid double dividends to performance and efficiency. Over 200 different pairs of instruction types are detected in the instruction cache pre-decode stage and can be fused at decode resulting in reduced work (one operation instead of two), as well as reduced or zero latency for dependent operations. For example, dependent ALU operations can either be reduced to a single operation or can be issued from a single shared issue queue entry supporting optimized latency. In another example, store instructions to consecutive addresses are fused, resulting in a single address generation pipeline operation supporting two stores up to 16 bytes in length each –and for stores of eight bytes or less consuming only a single store queue entry–. In addition, stores to consecutive address blocks in the store queue can be merged dynamically, enabling the retirement of up to two store queue entries (up to four store instructions) per cycle to the L1 and L2 caches.

A pervasive focus on power efficiency continued with design entry. From the start, designers were expected to provide designs where latch clocks would be off by default, and enabled only when a related instruction required that portion of the logic. This was in contrast to previous designs, where robust clock gating was largely added only after the full logic function was in place. This new mentality of design resulted in a significant reduction in the latch clock activity and reduced the total cost of implementation by avoiding design re-work breakage (timing or functional) due to subsequent changes for clock gating improvements.

Clock gating, ghost and data switching were all tracked and made a focus for designers. *Ghost switching* is switching on data inputs that does not reflect a write to a latch, register file or array. Data switching for arrays and register files was explicitly tracked in RTL simulation and correlated with write actions; data input switching that did not correspond to a write was then flagged and addressed. Ghost switching of logic cones was minimized by careful attention to clock gating of upstream latches.

A specific example of power-performance efficient design at the circuit level is in the design of floating point (FP) arithmetic units. Several different power levels of carry save adders (CSAs) were provided to enable power and delay optimization at the different levels of the logic cones. Efficient classical symmetric CSAs were used at the widest parts of the logic cones to save power at the expense of delay. Later stages in the logic cone used progressively optimized CSAs with duplication of the carry along with layout optimization to recapture the delay of the early stages. In parallel, the design team optimized across the boundary of two clock cycles to yield the lowest power level within the target cycle time. The net cumulative effect of these optimization steps have yielded significant power and area efficiency improvement in POWER10. Similar optimizations that also used a novel "sum" pass gate circuit have been shown to yield a remarkable 36% area reduction and $>$40% power reduction (for the FP unit) in an earlier 14nm processor product.

Additional research-mode ideas regarding register file optimization, coupled with layer-specific metal pitch reduction, improved utilization of multi-layer wiring and congestion control, preplacement of latches to reduce horizontal and

vertical wiring needs for the CSA tree, etc. promise significant additional improvements in power-performance efficiency for future processors.

C. The Inline AI/ML Accelerator - MMA

Dense numerical linear algebra computations are the basis of several important algorithms for scientific and AI/HPC workloads. To accelerate such workloads, the latest Power ISA Version 3.1 [36] introduces a new set of instructions –collectively known as the Matrix-Multiply Assist (MMA) facility– that implement numerical linear algebra operations directly on small matrices. These instructions are meant to accelerate computation-intensive kernels, such as matrix multiplication, convolution and discrete Fourier transform, with additional application to AI workloads including neural networks. These instructions are complemented by a doubling of general purpose SIMD units (8×128b units per core) and a doubling of load and store bandwidth (including new 32-byte load and store instructions) to address a broad range of machine learning and data preparation use cases.

The introduction of the MMA facility to the POWER architecture, combined with its area- and power-efficient design in POWER10, delivers high performance and high efficiency execution of AI and other workloads. The MMA unit leverages a 4×4 grid of processing elements and a set of local accumulators (a set of 8 architected 512-bit wide registers) to achieve area and power efficiency. This implementation leads to an efficient execution of the kernels resulting from two aspects. First, an increase in the overall computational throughput capacity. From the theoretical peak of 8 and 16 double-precision flops/cycle of vector code executed on POWER9 and POWER10 respectively to a maximum of 32 double-precision flops/cycle of MMA code on POWER10. Second, a significant reduction of data movement, thus reducing power consumption. Through the implementation of outer-product operations, MMA instructions can produce 512 bits of result data from just 2×128-bit vector inputs, while using the local accumulators as the additional inputs and outputs of the operations. Specific instructions are architected to move data from/to these local accumulators [36]. Overall, in comparison to existing approaches of wide vector facilities [3], [28], [40], MMA provides unique features that are listed below:

- The MMA unit is added with no impact on the rest of the architecture and minimal impact on the microarchitecture (as shown in Fig. 3), whereas widening the register file [26] or switching to a scalable vector [44] approach requires complex changes to existing structures and architecture. Also, the decoupled design allows for an effective power gating mechanism of MMA, which is leveraged by the Workload Optimized Frequency (Section IV-A) to boost performance when the unit is not utilized.
- MMA unit minimizes data movement for outer-products by keeping input/output operands on local accumulators, thus avoids the consecutive reads and writes to the register file that existing approaches require for each operation.
- MMA's 2D design grid aligns with the structure of the outer-product computation, whereas vector computations are one dimensional and need additional constructs to fold into a 2D arrangement.
- MMA provides direct support of outer-product operations (BLAS Level 2 operations [11], [31]), which are key for dense numerical linear algebra kernels, whereas vector instructions require additional steps –e.g. extra load or splat instructions– to convert the BLAS2 operations into one dimensional (BLAS Level 1 [11], [31]) operations.
- MMA design allows efficient back-to-back execution latency of MMA instructions in contrast to comparable vector instructions by having accumulators already in the functional unit.
- MMA instructions are more fine-grained than a complete matrix multiply unit and they can also be used as the building blocks of other computations such as convolution, triangular solve and discrete fourier transform.

The rest of the section provides an analysis of the significant efficiency improvements of these design decisions for the MMA implementation on POWER10 for key targeted workloads.

1) Evaluation of individual compute kernels: The power consumption of common computational loops has been analyzed using pre-silicon models at iso voltage-frequency between POWER9 and POWER10 to estimate the energy efficiency benefits from the MMA. Fig. 5 shows the FLOPs/cycle and core power consumption of an OpenBLAS-representative DGEMM Kernel of MMA and VSU code relative to the POWER9 baseline for single thread (ST) run. Note that the VSU version has not been specifically optimized to POWER10 microarchitecture. Performance and Core Power results are average measurements across multiple 5K cycle windows of the kernel to reduce data variability coming from cross-inner loop transitions and data switching effects, which we observed to be significant for the MMA unit.

In the VSU code, we observe the POWER10 performance is significantly increased, with a FLOPs/cycle 1.95× that of POWER9, and a reduction of 32.2% of in-core power consumption. When comparing POWER10 MMA code vs POWER9 VSU code, the performance is increased by 5.47×, while the power is reduced by 24.1%. Note that the MMA version of the code performs more work per instruction –i.e. 27.9 double precision FLOPs/cycle (87.1% of peak)– whereas the vector version achieves 9.94 FLOPs/cycle (62.1% of peak)– on POWER10. We expect higher utilization ratios once the kernels are fully optimized for POWER10 and higher SMT levels are used.

Overall, both versions of code –whether leveraging the new MMA unit or not– increase performance per cycle and also reduce core power, providing a significant increase of overall core energy efficiency that stems from the new energy efficient core design (POWER9 VSU vs POWER10 VSU) as well as from leveraging the new MMA execution unit (POWER9 VSU vs POWER10 MMA). As an example of that, during development of the MMA version of the code, we observed

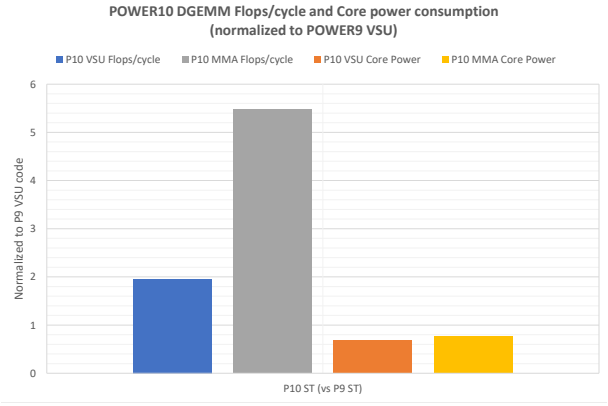


Fig. 5. Performance and core power normalized to POWER9 baseline VSU code showing the performance and power benefits of the new POWER10 core design. Same POWER9 VSU code on POWER10 achieves 1.95× FLOPs/cycle, and the new code leveraging the MMA unit achieves 5.47× FLOPS/cycle. In both cases, the core power consumption is also decreased by 32.2% and 24.1%, increasing the overall core energy efficiency significantly.

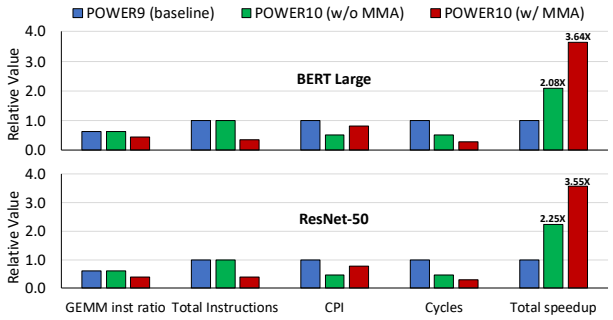


Fig. 6. Performance Comparison of POWER9 core, and POWER10 core (with the MMA disabled and enabled) running a PyTorch-based FP32 *ResNet-50* and *BERT-Large* models. The figures show the fraction of SGEMM instructions, and the total instructions, CPI, total execution cycles and overall speedup relative to the POWER9 core baseline. Due to the larger proportion of GEMM instructions the MMA-induced speedup in BERT-Large is slightly higher than in Resnet-50

that a not-fully optimized MMA version of the kernel was already achieving significant performance gains at lower core power when compared to the vector version of the code on POWER10.

2) Evaluation of end-to-end AI workloads: In addition to GEMM kernels, we modeled the benefit of the MMA when applied to end-to-end CPU-based AI inference workloads. Most AI inference applications spend a significant fraction of their execution in GEMM operations, and hence show substantial MMA utilization. We evaluate workload traces corresponding to two pre-trained PyTorch-based models, namely an image classification application using a *ResNet-50* [24] model on the ImageNet validation dataset [19] and a question-answering application using a *BERT-Large* [49] model on the SQuAD v1.1 dataset [35]. We use batch sizes of 100 and 8 for ResNet-50 and BERT-Large respectively. The MMA is enabled by linking to an optimized OpenBLAS library [50] during execution, which computes 8x16 SGEMM panels on the MMA.

Fig. 6 compares the estimated performance of a POWER10 core (with the MMA disabled or enabled) relative to a POWER9 core, for the ResNet-50 and BERT-Large models. When the MMA is disabled, the SGEMM kernels are mapped to the vector unit (VSU) executing Vector Multiply-Add instructions, and when enabled, they are mapped to MMA-dedicated Matrix Multiply instructions. These instructions are much more efficient in expressing the computations, since a single matrix-multiply instruction replaces several vector multiply-adds, resulting a much smaller instruction count when the computations are mapped to MMA instead of the VSU.

We observe speedups of 3.55× and 3.64× for an MMA-enabled POWER10 core over POWER9, and 2.25× and 2.08× for the MMA-disabled, POWER10 core, when running ResNet-50 and BERT-Large respectively. Note that while the MMA-enabled POWER10 speedup is greater in BERT-Large than in ResNet-50, the speedup without the MMA is lower. This can be attributed to the core-level microarchitecture improvements being less impactful in BERT-Large due to its larger model size ($> 10\times$ more parameters than Resnet-50) and consequently, the greater contribution of data-loading and preprocessing to the overall execution time. We observe further improvements of 2.5× by increasing the per-socket core count from 24 to 60 and an estimated 1.1× due to improvements in bandwidth, software and other system-level configurations. This yields overall socket-level estimated speedups of up to 10× for both the Resnet-50 and BERT-Large models running on an MMA-enabled POWER10 processor. For INT8 models, our pre-silicon estimates project an additional increase in performance leading to as much as 21× that of POWER9. Our evaluations on pre-production POWER10 hardware are consistent with these projections and the hardware results show a difference of $< 5\%$ when compared to the above simulation results.

III. New Pre-Silicon Modeling Methodology

POWER10 was targeted to provide a major microarchitectural update, while also adopting a new high performance CMOS fabrication technology (Samsung 7nm HP, versus POWER9's 14nm HP GlobalFoundries technology). It was important to adopt a pre-silicon definition and modeling regimen that provided a much more agile environment (Fig. 7) than the traditional flow [29], [30], [37], [51]. As such, the decision was made to rely on in-flight RTL optimization for power-performance analysis. In other words, power-performance trade-offs and optimization decisions were made directly using analysis based on the evolving RTL model, starting from the reference POWER9 RTL baseline. There was less reliance on the POWER10/M1 model in the early stages of design; rather the prior-generation POWER9/M1 model was useful in establishing fundamental power-performance trade-offs that led to sound decisions on features to add to the POWER10/M3 model. Due to the significant magnitude of micro-architectural changes from POWER9 to POWER10, the POWER10/M1 model developed into full maturity later during the design cycle, as it factored in brand new elements of the design (e.g.

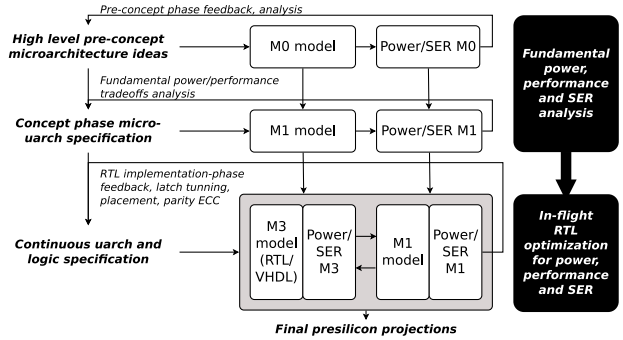


Fig. 7. High-level view of new pre-silicon modeling.

changes for power efficiency detailed in Section-II or the inline MMA accelerator detailed in Section II-B).

A. Generation of Workload Proxies and Traces

In order to pursue an in-flight RTL optimization-based microarchitecture definition process, there was a need to develop an accurate abstraction of the workload suite. This abstraction had to contain a variety of representative code snippets that are simple and small enough for the *partial* –i.e. not complete– and lower-speed RTLSim to process the code correctly and in reasonable time, but still accurately reflect the full-suite power-performance trade-offs. These snippets, in conjunction with well-known code kernels – e.g. *daxpy* – and synthetic microbenchmarks targeted to various aspects of the microarchitecture [4], [6]–[8], [17], [46] –which are designed to understand design corners– provided a rich coverage of workload behaviors from an early stage.

In order to generate early projections of power and performance for the SPECint benchmark suite, *SPECint proxy* workloads were generated using the Chopstix tool [15]. The top 10 most executed functions of each benchmark were extracted for each benchmark, achieving a coverage between 41% (e.g. *gcc* which has execution spread on many small functions) and 99% (e.g. *xz* which has the execution concentrated on few functions). Multiple invocations of these top most-executed functions of each benchmark, with the code and data state captured from memory, are used to generate the executable payload for RTLSim. The original captured core and data state of each traced invocation was automatically transformed into L1-contained (endless) loops running in real mode (without address translation) to have consistent and repeatable results during the duration of the project. In the end, 1935 *SPECint proxy* workloads were generated in total from the SPECint benchmarks and were selected for POWER10 processor characterization. These proxy workloads provided an average coverage of 70% for the *SPECint* suite.

Despite the limited size of these code snippets (ranging from few hundred to up to 22K instructions), the large variety of proxy workloads provided three main benefits. First, by avoiding interference of other (still being developed) elements that would require larger workloads to assess their performance, such as caches and branch predictors, these proxy workloads enabled designers to focus very early on core pipeline implementation issues using real code from applications not covered by traditional synthetic test suites inherited from previous generations. Results of these snippets were tracked and compared against POWER9 results to detect performance regressions (lower performance vs POWER9) and to pinpoint cases where core performance does not achieve the generational performance improvement goals for the project. Second, the L1-contained steady-state nature of the proxies allow them to be used on all the development stages: from RTL simulation right through to final hardware. At every stage, one can execute the workload proxies to perform cross-model/cross-environment validations on a large variety of behaviors, avoiding measurement granularity issues coming from the time scale disparities of each environment. Finally, the rich set of workloads enables generation of first-order power/performance projections for the entire suite (based on the weight assigned to each snippet with respect to the entire application). These projections can also be readily compared to the previous design, on which the proxies have been similarly executed.

As an alternative to detailed latch-accurate simulations using workload proxies, it is also possible to achieve highly accurate performance estimations (within 5% of hardware) by means of M1-model based simulations. However this requires representative instruction traces, both for validation against existing processors as well as projection on future processors. Existing *Simpoint*-based techniques [34] or their variations [16], [33], currently used for generation of representative traces, have several limitations, particularly in the case of AI workloads. The offline evaluation phase requires end-to-end simulation of the entire application which can take several days or weeks. Traces are generated in Simpoints by clustering Basic Block Vectors (BBVs) from the application. However, BBVs may not be the correct granularity for evaluation as several architecture parameters such as LLC misses, branch misses and application characteristics such as periodicity cannot be effectively captured. Finally, Simpoints have been empirically shown to be less accurate for interpreted languages such as Python which is the basis for most AI workloads.

To address these shortcomings, we devise a methodology called *Tracepoints* based on hardware performance counter data instead of simulation-generated BBVs. Performance counter information is collected at an epoch-level granularity of a few *ms* and these epochs are assigned to different histogram bins based on their CPI and/or other performance metrics such as number of cache misses, branch mispredictions, Integer, FPU and Vector operations. Individual epochs are picked from histogram bins, so as to match the aggregate performance of the actual application, and concatenated to form a trace. Tracepoint-based methods have also been empirically found to produce traces that closely match hardware performance for interpreted language-based workloads.

Since the MMA in POWER10 is capable of executing GEMM operations more efficiently than the Vector Unit in POWER9, this would result in a difference in the number of

instructions, cycles, compute operations and other parameters across the same instance of execution on the two machines. This makes it difficult to establish an equivalence between traces generated from hardware performance counter data collected on POWER9 and projected on POWER10. Hence, we generate *MMA-aware traces* using the Tracepoints methodology, that are representative of the end-to-end application, not only in terms of CPI, but also in terms of the number of BLAS [11], [31] API calls comprising of GEMM kernels dictating MMA utilization. Traces are validated against real hardware measurements on a POWER9 system.

B. Power Modeling with RTLSim

In previous generations, power modeling was performed continuously on well-known workloads of interest, including maximum power stressmarks, using IBM's Einspower methodology [20]. The increased focus on energy efficiency for POWER10 required that developers focus not only on feature implementation and performance, but also on power consumption in the early design stages. This in turn required new innovations in the existing power modeling of the RTLSim methodology to increase the power behavior visibility and to optimize power in the early stages.

New tools were developed to support the power optimization flow during development. Specifically, the IBM EDA team developed *Powerminer* to provide a full range of stats for logic activity directly related to power consumption, including logic/data/ghost switching stats and clock gating. The switching feedback from *Powerminer* helped designers to evaluate and optimize the design without requiring to execute the full and detailed *Einspower* characterization flow that involves going through the time consuming physical design flow. This complemented the existing range of tools available to the designers and was key to supporting the goal of energy efficient design and implementation as the designers were able to implement their own quick optimization loop –based on several metrics related to power consumption– without having to rely on other steps of the design methodology.

An additional focus of the POWER10 early stage analysis was to increase the behavioral coverage by expanding to use many newly-developed proxy workloads. This work significantly increased the number of workloads used to regularly evaluate the design, and provided more statistically meaningful data as to where to focus additional design effort. Instead of focusing on specific corner cases that showed maximum power consumption for particular units, designers were able to glean insight into the average power consumption and expected (i.e. typical) power ranges, as well as quickly detect behavioral outliers. Needless to say, the overall process required a much higher degree of automation to overcome the challenges from scaling-up the methodology. Not only were more workloads being regularly evaluated, but more tools were also being executed for each workload. The final implemented tool-flow is shown in Fig. 8.

For each workload, the region of interest –i.e. the measurement window in cycles and instructions– is computed based on a baseline run (or previous runs). This ensures comparable results over time, since the measurement window (in cycles) may change as the RTLSim implementation evolves. The workload is then executed on RTLSim, generating the execution trace, performance stats and other logic activity files required to drive Powerminer and Einspower. Powerminer generates detailed logic activity stats, and Einspower generates accurate power reports by taking into account the physical design implementation. The execution trace is also converted into an M1-compatible format, and fed into the M1 model which generates its own higher-level performance stats. For each workload characterized, the summary selects ∼40K stats for modeling and analysis. The exact amount varies based on the target purpose of the modeling and analysis process. Note that this is a small subset of all the detailed stats that the different tools generate, which are stored in a database for deep-dive debugging and analysis if needed.

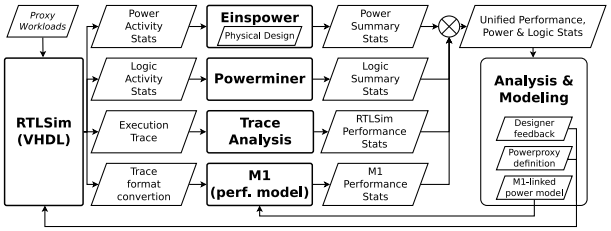


Fig. 8. Continuous power and performance modeling with RTLSim.

Finally, the enormous amount of data generated with the improved continuous characterization process made it necessary to also automate the process of summarizing and analysing the results. Tools were developed to generate summaries and find optimization opportunities. For instance, one can detect strong relationships between power and logic activity, and/or other high level performance metrics, outliers, mismatches between RTLSim and M1 model etc. At the end of the process, the designer obtains a unified view of power and performance at various levels for each workload snippet (i.e. regions of interest) being measured. This high level of automation also enabled the systematization of other steps of the modeling methodology, which are presented in Section III-D (the M1-linked power model) and Section IV-C (Power Proxy hardware definition).

During the duration of the POWER10 project, several metrics of interest were continuously tracked in order to assess the project execution. These metrics included, for instance, the Instructions per cycle (IPC), the Core power consumption, the Core efficiency, the Number of latches in the core, the % of Clock enabled (inverse of % Clock gating), Potential latch switching (potential latch activity when latch is clock enabled), and Observed latch switching ratio (the latch is clock enabled and signals actually switch) with respect to the number of latches in the design.

By retrospectively analysing the data, we see the expected variations during early stages of a design project, resulting from both design changes and improvements in the physical design synthesis methodology. As the project progresses, core

efficiency improved due to two primary factors. First, performance improvements (new features and performance fixes) are implemented in the new design as described in Section II. Additionally, the power consumption of design elements is significantly reduced. Initially, this reduction comes from the benefits of a new technology, but as the design matures, additional efficiencies are realized from improvements in both the physical design synthesis and more efficient implementation of features in the core design.

Overall, the tracking of these metrics, enabled by the pre-silicon modeling methodology, allowed a notable increase in core performance while maintaining the power consumption efficiency by providing continuous feedback to guide designer decisions. Specifically, these pre-silicon results of the SPECint proxy workloads were cross-validated with longer SPECint runs executed on APEX (See Section III-C) to project the 1.3× improvement in core performance at 0.5× the power consumption when compared to POWER9. This results in a 2.6× energy efficiency gain at the same voltage and frequency, without including efficiency boosts available from other aspects such as the compiler (i.e. using code specifically targeting the POWER10 architecture) and Workload Optimized Frequency (See Section IV-A). Overall, the new pre-silicon methodology provided a high degree of confidence in the projections and the initial measurements of test cases on early POWER10 hardware show significant improvements in efficiency in line with the estimations.

C. Power Modeling Acceleration with APEX

The snippet sized proxy workloads described in III-A provide a good means to drive early power/performance analysis. However, to accurately capture some effects such as branch prediction, flushes, cache, TLB misses, instruction and data prefetching, larger workload segments are needed. Power analysis for large workload segments is too slow using the software-based RTLSim simulator. A new methodology, the Awan Power Extractor (APEX) methodology (Fig. 9), leveraging the IBM internal Awan hardware accelerated simulation platform [18], was therefore developed. Awan can achieve simulation speeds of well over 100k cycles/second even for multi-core chip models. To track all clock and data switching activity in the design, the RTL is instrumented with edge- and level-triggered LFSR counters for the subset of signals used by Einspower for its power calculations. The number of clock, latch, array, and logic signals for which switching activity is tracked for a core+L2+L3 APEX model amount to about 8M, the majority of which are primary inputs and outputs and internal signals of logic macros. During simulation, a batch routine is called by the Awan runtime at configurable intervals, or at specific simulation events (such as start of stat collection after warmup) and the activity is extracted from the switching counters. The switching activity can be either stored to a file, to generate detailed power reports using Einspower, or a simplified power report can be generated on-the-fly by APEX using pre-extracted activity signal groupings and associated effective capacitance. This APEX solution provides

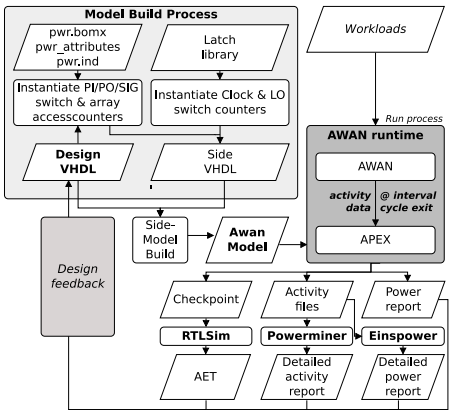


Fig. 9. APEX Tool Flow.

a 5000× speedup in power simulation over RTLSim, while providing identical accuracy. Since activity is extracted for every latch and signal of interest in power calculation, the APEX-produced activity files can also be used with other IBM tools, such as Powerminer, for further detailed switching and power analysis. Signal event trace files can also be extracted from the APEX runs providing a cycle-by-cycle trace of logic signal events to support further power debug deep dives by designers. Checkpoints corresponding to the event trace can also be used with a RTLSim side model to provide simulation based validation of eventual design fixes before being checked in to the main code branch.

Additionally, as the design matures and the RTL for components outside the core becomes available, it is desirable to model the power at the full chip level using the larger workload segments. A chip level model is therefore needed to account for the full cache hierarchy and memory latencies impact on power. The chip model provides the ability to detect and debug power/performance anomalies for real workload scenarios under real chip behavior. As an example, one power anomaly was detected for the *omnetpp* SPEC workload, where the power was unduly high relative to its IPC. The cause was quickly identified as stemming from the sequential operation of the memory unit in the core, causing undue performance driven speculative reissues during certain high miss-rate conditions. Enabling pipelining of the memory unit readily solved this issue.

Executing large benchmarks under real cache and memory bandwidth and latencies provides significant additional accuracy in modeling. Fig. 10 illustrates the difference in performance and core power for an APEX core model with infinite L2 (circle shapes) versus an APEX chip model with full cache and memory hierarchy (triangle shapes) for the SPECint benchmark suite in SMT2 mode. Memory bound workloads, such as illustrated by the gray colored entries in Fig. 10, have a significantly different power/performance characteristics in a chip model. This increased accuracy afforded by a chip model and larger workload segments also allows analysis of absolute, rather than relative power and performance measures. Absolute power analysis in turn allows pre-silicon projections useful

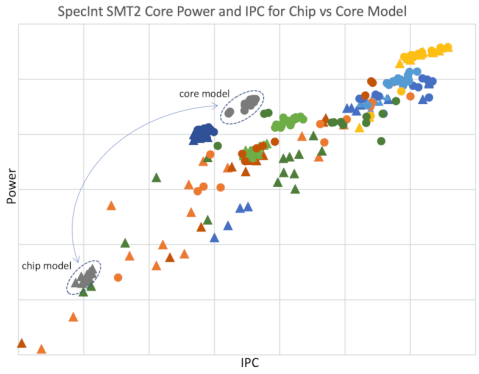


Fig. 10. POWER10 Core power for Core vs Chip RTL simulation models for the SPECint benchmark suite represented by 160 simpoints run in SMT2 mode. Power/IPC points for the APEX chip model is represented by triangle shapes and the APEX core model by circle shapes. The points are color coded by benchmark.

for WOF design and also feeds into PFLY and CLY analysis for product offering consideration. New APEX models are generated for major new RTL chip versions that are typically released at a monthly to semi-monthly cadence.

D. Power-Performance Modeling with M1 simulator

Early stage power modeling methodologies are important to perform accurate scaling of an abstracted model and to enable exploration of new microarchitectural features in the early design stages. The new pre-silicon modeling methodology for POWER10, with simultaneous development and experimentation in both the M1 and RTLSim models, was extended to allow the continuous transfer of the power information from the current RTLSim implementation to the higher-level M1 performance model. This capability provided an automatic means to keep the M1 model in sync with the developing RTL, and provided a means to investigate the power behavior of the latest hardware implementation on much larger and realistic workloads by running the fast M1 model using the latest power model data.

The generation of the M1-linked power model relied on the extensive power modeling performed at the RTLSim level. The implemented tool-flow executes the workloads at the RTLSim level and at the M1-level. By using the existing counter-based power modeling methodologies based on machine learning techniques [8]–[10], [29], the set of performance stats reported by the M1 model were systematically selected to create accurate power models based on the current RTLSim implementation. Overall, the large variety of workloads –more than 25K–, including the workload proxies as well as synthetic workloads, allowed the generation of robust M1-linked power models. Fig. 11 shows the average error on active power of the models generated for different number of inputs, modeling methods and constraints. Active power is defined as the workload-dependent part of the power consumption, that excludes the leakage and active-idle power. Note that the average error reported in this chart is further reduced when all such static components are taken into account. The results showed that when increasing the number of inputs, the error

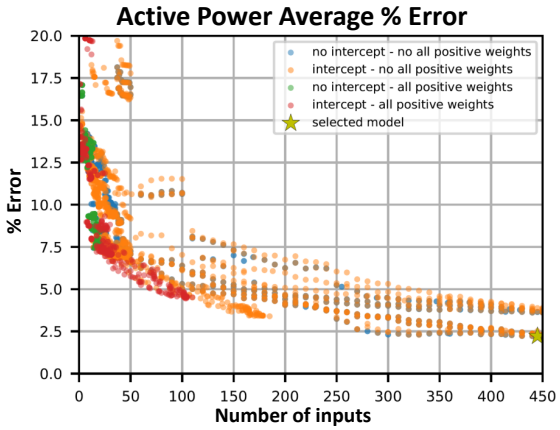


Fig. 11. M1 linked active power modeling accuracy for different number of inputs and modeling constraints.

is reduced, achieving less than 2.5% on active power when the number of inputs is maximized.

As the project progressed, the top-down core model was superseded by a bottom-up one. Per-functional-unit power models were implemented, thanks to the user-configurable granularity of the Einspower reports which can provide, for instance, hardware-macro level granularity. These fine-grain power models provide more detailed insights about which higher-level performance events are the drivers of the power consumption of each different component, and allowed the detection of corner cases and unusual power behavior. For this project, 39 components were defined and a counter-based power model was implemented for each of them by applying similar modeling design explorations and techniques applied for the coarse grained core power model but with the aim of minimizing the number of inputs –within reasonable error bounds–. The rationale for such an approach is to generate models with the few key performance events driving the power of each particular component. This results in simpler and more interpretable models for designers to understand. The breakdown was based on the contribution of each specific macro/component to the overall power. After the modeling process, besides regular validation of each and every model accuracy, per macro models were added to create a bottom-up core model and the estimations were validated against the coarse grain core model for consistency. Fig. 12 shows the correlation between the estimations of the two models when applied to 1480 large workload traces executed on the M1 performance model. On average both modeling approaches differ only by 3.42%, while still providing similar accuracy levels when compared to the reference Einspower output. The bottom-up macro model, which can be broken down into 39 separate power components, uses only a total of 72 events, which is far less than the number of events used by the top-down core model.

E. Power-Aware Latch Reliability Modeling and Optimization

Reliability is a key feature driving the POWER10 design right from an early stage. We introduce the *SERMiner*

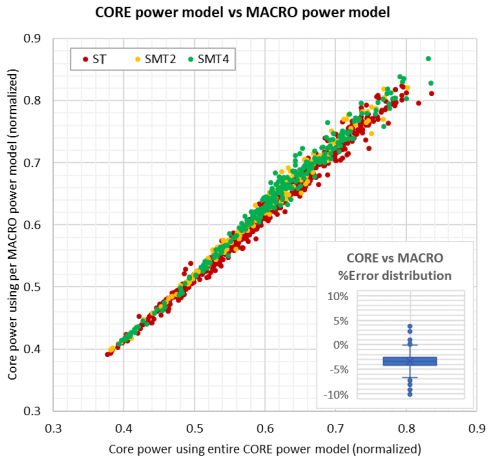


Fig. 12. M1 linked power models comparison.

methodology for mitigating processor vulnerability, particularly to radiation-induced soft errors, with minimal energy/performance overheads. SERMiner estimates vulnerability based on switching characteristics derived from latch-level simulations using RTLSim. It also determines key components of interest from among the latches, register files and flip-flops in the design that would most benefit from protection or hardening. Such an approach enables us to establish a more fine-grained and efficient method for RAS implementation.

Due to the fine granularity of clock-gating in POWER10, SERMiner adopts *clock utilization*, or the fraction of cycles for which a latch is clocked, as an effective proxy for vulnerability, as opposed to data-residency-based evaluations used in prior works [37]. This is because latch data is refreshed every cycle that the latch is clocked, regardless whether the value changes or not.

1) Latch Derating: A latch is said to be *derated* if an SER-induced bit flip does not manifest in a change in other latch values, and can be categorized as either *Static-* or *Runtime-*derated. Most latches that never switch through the entire execution period for any workload, with the exception of some configuration latches usually set during initialization, are assumed to be static-derated. Runtime-derated latches are those that have non-zero switching across one or more target workloads, but whose switching value falls below a specified threshold known as the *Vulnerability Threshold (VT)*. The VT determines the minimum switching value required for a latch to be termed as vulnerable. For instance, a VT=10% would mean that latches whose switching activity is within the top 10^{th} percentile across all workloads, are considered vulnerable. Higher the VT value, greater the number of latches that will be classified as vulnerable. Functional clock gated latches retain state even when not clocked, and are hence potentially vulnerable throughout the period of execution, including when they are clock-gated.

2) Evaluated Workloads: We carry out derating estimates on a combination of synthetic testcases generated by the Microprobe tool [8] for varying SMT (ST, SMT2 and SMT4),

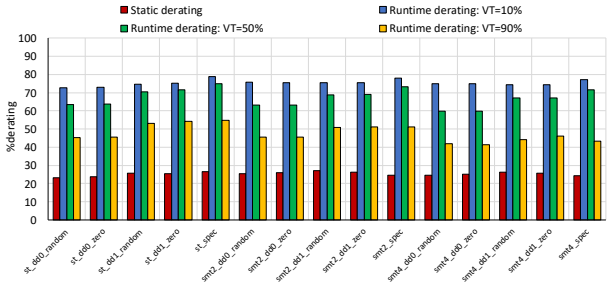


Fig. 13. Variation in static and runtime latch derating for different synthetic and real testcase suites. The figure shows runtime derating for different vulnerability threshold (VT) values of 10%, 50% and 90%.

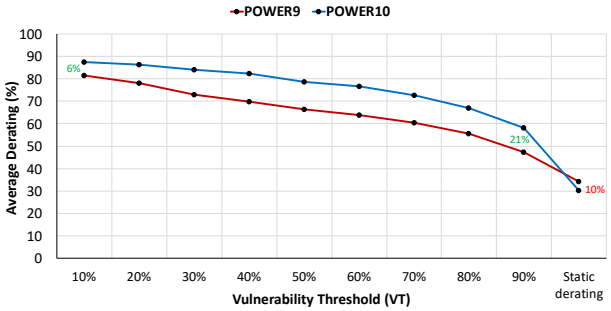


Fig. 14. Comparison of derating in POWER9 and POWER10 cores, averaged across all workloads. The values in green denote the difference in runtime derating between POWER9 and POWER10 at minimum (10%) and maximum (90%) VT, while the value in red denotes the corresponding difference in static derating. The improved runtime derating in POWER10, in spite of a higher latch count, can be attributed to a comprehensive RAS-aware design approach.

Dependency Distance (DD, 0 and 1 instructions) and latch data initialization (zero, random) parameters, and SPEC CPU 2017 proxies described in Section III-A. Fig. 13 shows the variation in derating observed across all workloads, for VT values of 10%, 50% and 90%. Designers may either choose to protect/harden all latches that are not statically derated as a conservative RAS protection policy, or may adopt more aggressive policies that protect only highly utilized latches. For instance, only around 25% of the total latches would be vulnerable for VT=10% , while VT=90% would result in 52% of the total latches being classified as vulnerable.

Fig. 14 compares derating between POWER9 and POWER10 cores (higher derating is better). We observe a higher runtime derating in POWER10, with the difference becoming larger with increasing VT. This implies that a smaller fraction of latches needs to be protected in POWER10 compared to POWER9 to achieve a comparable degree of resilience between the two processors. In contrast, static derating in POWER10 is lower by $\sim$10%, resulting in fewer inactive latches during execution. The higher runtime derating in POWER10 results in a more efficient RAS implementation, and hence a lower power overhead required to attain the same level of resiliency. As a result, POWER10 is able to enhance RAS while reducing the associated power overheads, compared with POWER9.

IV. Core Power Management

This section describes the core power management infrastructure developed as a key piece in optimizing power and performance of the system.

A. Workload Optimized Frequency - WOF

Power management techniques are used by most modern processors to provide maximum performance for a given power and thermal envelope [2], [14], [27], [38]. The POWER10 processor design supports *Workload Optimized Frequency*, where the frequency operating point and instruction throughput is adjusted to stay just under the power and thermal limits of the socket, providing a performance boost for typical workloads. This allows workloads that do not consume as much power as the system's thermal and voltage regulation design points (TDP/RDP) to operate at a higher clock frequency than specified by the nominal baseline, giving such workloads increased performance. IBM's WOF power management is unique in that it offers a deterministic performance boost which is a requirement for many customers. The same workload running on two POWER chips from the same sort (offering) with identical system configurations will deliver the same performance under typical ambient conditions. While frequency (and corresponding voltage) is the primary lever utilized by firmware to optimize the performance/power of the processor, the POWER10 core also employs a power proxy based core throttling scheme (see IV-C) which can control the throughput of individual cores.

The Workload Optimized Frequency (WOF) boost projections were primarily made using the SPECint benchmark suite as a representative customer workload. APEX (see Section III-C) was used to extract the activity from the SPECint workloads, and Einspower to calculate the power consumption. The resulting difference between the workload and the supported system design point is represented as an Effective Capacitance ratio which is fed into the PFLY and CLY analysis for workload specific frequency and performance projections.

In addition to workload driven variations in power, WOF also takes advantage of idle regions of the chip that are powered off by power management firmware. For the core power management, one such example is the MMA unit which can by dynamically powered off to save leakage power that is instead applied to achieve higher performance. The MMA architecture was carefully planned to minimize the impact of power on latency by not requiring array initialization or restoration of scan rings, other than scan latches required for circuit performance or timing in response to process variation. To minimize the impact of MMA power-on latency on performance, special hint instructions are provided in the architecture to proactively wake up the MMA unit. In addition, the firmware can select how long the MMA must be idle before powering off.

B. Core Throttling

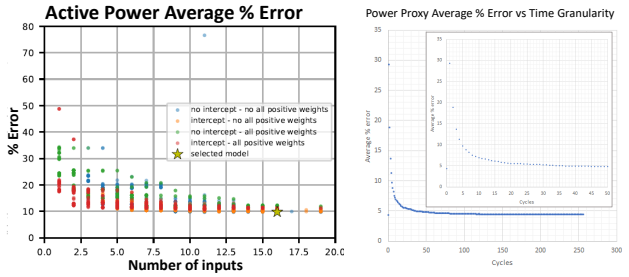
To support per core power and voltage droop management beyond DVFS, WOF supports two flavors of core throttling mechanisms. For customers that require a fixed frequency operation or in situations where the core already operates at Fmin, a fine-grained instruction throttling mechanism is provided to ensure the system operates within allowed current and thermal limits. In this operation mode core power proxy feedback allows for faster learning, yielding more efficient adaptive control loops.

Additionally, a set of coarse grained throttling mechanisms are employed at numerous control points in the core pipeline, execution engines, and caches/queues to respond to voltage droops caused by a sudden change in workload. This can manifest as an instantaneous swing in active current causing on-chip droops on the local power grid or a system level droop on the voltage supply rail caused by over-currenting the voltage regulator [7]. A Digital Droop Sensor (DDS) –i.e. a new generation CPM-like sensor [46]– is embedded in each core to measure timing margin as seen by the transistors in the sub 1ns timescale, where it is used to quickly detect and engage the coarse throttle controls to preserve adequate circuit timing margin.

C. Core Power Proxy

The use of power proxies is common in most high-performance processor designs [12], [22], [47] to generate fast predictions of power consumption, which are utilized by the power management mechanisms such as WOF. Previous implementations relied on designer expertise to manually define the set of inputs to track [5], [21], [22], [25], [47], [48]. With the new POWER10 methodology, the process of defining the inputs and their validation now leverages a rich data-set to automatically select the tracked inputs from the set of signals available in the design. This minimizes the risk of escapes by guaranteeing that the right set of inputs is selected.

During the continuous RTLSim power modeling process flow many regular signals, as well as instrumentation counters added by designers to debug and validate design functionality and efficiency, were tracked. While the debug counters are not implemented in the final core, they were used in the POWER10 power proxy design. These counters were analyzed using machine learning techniques similar to those used when generating the M1-linked power model [8]–[10]. In contrast to the M1 software based power model, actual implementation constraints, e.g. number of counters to implement, had to be taken into account. From the $\sim$500 counters analyzed, thousands of models were generated with different modeling constraints, such as number of inputs, coefficient ranges (all positive or not), intercepts (with and without). The detailed design space analysis allowed the selection of the most effective set of counters for the final design power proxy implementation. Once the power proxy implementation was selected, the design was included in subsequent RTLSim runs, and the analysis was repeated to validate counter selection. Overall, this methodology minimized the risk of escapes and



(a) (b)

Fig. 15. (a) Power Proxy active power accuracy for different number of inputs and modeling constraints. (b) Average error in core power predictions versus time granularity in cycles.

built confidence in the design. Fig. 15(a) shows the accuracy trade-off for different numbers of input counters. The figure shows modeling accuracy of the active power consumption. For POWER10, a simple design with 16 counters was selected with a 9.8% error on active power, which reduces to <5% when including static power contributors (i.e. leakage and active-idle). Another key aspect of the design was to determine the time-granularity at which the Power Proxy is capable of generating accurate information. The time-granularity has implications on the efficiency of the power management mechanisms that rely on the Power Proxy –for example, noise mitigation mechanisms require very fast reactions to power variations–. Given a specific Power Proxy definition, Fig. 15(b) shows the average error of the total core power prediction at different time-scales. Predicting every 50 cycles or more, yields a near-best case accuracy, although the error starts to increase dramatically as we reduce the granularity further.

V. Conclusions

We highlight a number of the micro-architecture and logic/circuit innovations of a new server-class processor core (POWER10) focused on improving performance and energy efficiency. We describe the use of novel, agile and accurate pre-silicon modeling methodologies to help achieve a SPECint performance increase by 30% at the core-level, while at the same time, reducing power consumption by 50% in comparison with the prior generation POWER9 based on pre-silicon projections. In terms of domain-specific acceleration, the inline accelerator (matrix multiplication assist - MMA) was estimated to boost socket-level performance of a class of AI/ML/HPC workloads by up to 10× over POWER9 for FP32 computations and up to 21× for INT8 precision inferencing.

Acknowledgment

POWER10 definition and design involved a large team spread across multiple sites across USA, Europe, Israel and India. For such a large enterprise, it is not possible to acknowledge the full team because of space constraints. Nonetheless, for this particular paper, with a special focus on general efficiency and AI-infusion, the following additional names (beyond the co-authors listed) need to be mentioned: Hung Le, Stephen Bergman, John Griswell, Tharunachalam Pindicura, Daniel Stasiak, Bryant Cockcroft, Doowon Lee, Vivek Britto, Steven Aden, Adarsh Subramanya, Mohit Karve, Vinod Ramadurai, Rex Berridge, Eric Fluhr, Lisa Ferrera, Cindy Reynolds, Niels Fricke, Jeff Brownscheidle, Matthew Cooke, Jim Bishop, Sanjeev Ghai, Sam Kirchhoff, Steve Battle, Daniel Howe, Wolfgang Roesner, Michael Genden, Viresh Paruthi, John Schumann, Phillip Restle, Rajesh Veerabhadraiah, Abraham Mathews, Sanaka Sriram, Puneeth Bhat and Khajistha Fattu.

References

[1] B. Abali *et al.*, "Data compression accelerator on IBM POWER9 and z15 processors : Industrial product," in *47th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2020, Valencia, Spain, May 30 - June 3, 2020*. IEEE, 2020, pp. 1–14. [Online]. Available: https://doi.org/10.1109/ISCA45697.2020.00012

[2] AMD Corporation. Turbo Core Technology, [online]. Available: https://www.amd.com/en/technologies/turbo-core.

[3] M. Arafa *et al.*, "Cascade lake: Next generation intel xeon scalable processor," *IEEE Micro*, vol. 39, no. 2, pp. 29–36, 2019. [Online]. Available: https://doi.org/10.1109/MM.2019.2899330

[4] C. J. Berry *et al.*, "2.7 IBM z15: A 12-core 5.2ghz microprocessor," in *2020 IEEE International Solid- State Circuits Conference, ISSCC 2020, San Francisco, CA, USA, February 16-20, 2020*. IEEE, 2020, pp. 54–56. [Online]. Available: https://doi.org/10.1109/ISSCC19947.2020.9063030

[5] C. J. Berry *et al.*, "IBM z14 design methodology enhancements in the 14-nm technology node," *IBM J. Res. Dev.*, vol. 62, no. 2/3, p. 9, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8276267/

[6] C. J. Berry *et al.*, "IBM z14: Processor characterization and power management for high-reliability mainframe systems," *IEEE J. Solid State Circuits*, vol. 54, no. 1, pp. 121–132, 2019. [Online]. Available: https://doi.org/10.1109/JSSC.2018.2873582

[7] R. Bertran *et al.*, "Voltage noise in multi-core processors: Empirical characterization and optimization opportunities," in *47th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2014, Cambridge, United Kingdom, December 13-17, 2014*. IEEE Computer Society, 2014, pp. 368–380. [Online]. Available: https://doi.org/10.1109/MICRO.2014.12

[8] R. Bertran *et al.*, "Systematic energy characterization of CMP/SMT processor systems via automated micro-benchmarks," in *45th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2012, Vancouver, BC, Canada, December 1-5, 2012*. IEEE Computer Society, 2012, pp. 199–211. [Online]. Available: https://doi.org/10.1109/MICRO.2012.27

[9] R. Bertran *et al.*, "Decomposable and responsive power models for multicore processors using performance counters," in *Proceedings of the 24th International Conference on Supercomputing, 2010, Tsukuba, Ibaraki, Japan, June 2-4, 2010*, T. Boku *et al.*, Eds. ACM, 2010, pp. 147–158. [Online]. Available: https://doi.org/10.1145/1810085.1810108

[10] R. Bertran *et al.*, "Counter-based power modeling methods: Top-down vs. bottom-up," *Comput. J.*, vol. 56, no. 2, pp. 198–213, 2013. [Online]. Available: https://doi.org/10.1093/comjnl/bxs116

[11] BLAS Technical Forum, [online]. Available: http://netlib.org/blas/blast-forum/.

[12] D. Bouvier *et al.*, "AMD "kabini" APU SOC," in *2013 IEEE Hot Chips 25 Symposium (HCS), Stanford University, CA, USA, August 25-27, 2013*. IEEE, 2013, pp. 1–25. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/HOTCHIPS.2013.7478299

[13] D. M. Brooks *et al.*, "Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors," *IEEE Micro*, vol. 20, no. 6, pp. 26–44, 2000. [Online]. Available: https://doi.org/10.1109/40.888701

[14] M. Broyles *et al.*, IBM Energy-Scale for POWER9 Processor-Based Systems. [online]. Available: https://www.ibm.com/downloads/cas/6GZMODN3.

[15] C. Bulla *et al.*, "Chopstix: Systematic extraction of code-representative microbenchmarks," in *2018 IEEE International Symposium on Workload Characterization, IISWC 2018, Raleigh, NC, USA, September 30 -*

October 2, 2018. IEEE Computer Society, 2018, pp. 80–81. [Online]. Available: https://doi.org/10.1109/IISWC.2018.8573473

[16] T. E. Carlson *et al.*, "Barrierpoint: Sampled simulation of multi-threaded applications," in *2014 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2014, Monterey, CA, USA, March 23-25, 2014*. IEEE Computer Society, 2014, pp. 2–12. [Online]. Available: https://doi.org/10.1109/ISPASS.2014.6844456

[17] P. I. Chuang *et al.*, "26.2 power supply noise in a 22nm z13™ microprocessor," in *2017 IEEE International Solid-State Circuits Conference, ISSCC 2017, San Francisco, CA, USA, February 5-9, 2017*. IEEE, 2017, pp. 438–439. [Online]. Available: https://doi.org/10.1109/ISSCC.2017.7870449

[18] J. A. Darringer *et al.*, "EDA in IBM: past, present, and future," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 19, no. 12, pp. 1476–1497, 2000. [Online]. Available: https://doi.org/10.1109/43.898827

[19] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, pp. 248–255. [Online]. Available: https://doi.org/10.1109/CVPR.2009.5206848

[20] N. R. Dhanwada *et al.*, "Efficient PVT independent abstraction of large IP blocks for hierarchical power analysis," in *The IEEE/ACM International Conference on Computer-Aided Design, ICCAD'13, San Jose, CA, USA, November 18-21, 2013*, J. Henkel, Ed. IEEE, 2013, pp. 458–465. [Online]. Available: https://doi.org/10.1109/ICCAD.2013.6691157

[21] M. S. Floyd *et al.*, "Introducing the adaptive energy management features of the POWER7 chip," *IEEE Micro*, vol. 31, no. 2, pp. 60–75, 2011. [Online]. Available: https://doi.org/10.1109/MM.2011.29

[22] M. S. Floyd *et al.*, "Adaptive energy-management features of the IBM POWER7 chip," *IBM J. Res. Dev.*, vol. 55, no. 3, p. 8, 2011. [Online]. Available: https://doi.org/10.1147/JRD.2011.2114250

[23] E. J. Fluhr *et al.*, "IBM POWER9 circuit design and energy optimization for 14-nm technology," *IBM J. Res. Dev.*, vol. 62, no. 4/5, p. 4, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8383685/

[24] K. He *et al.*, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[25] W. Huang *et al.*, "Accurate fine-grained processor power proxies," in *45th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2012, Vancouver, BC, Canada, December 1-5, 2012*. IEEE Computer Society, 2012, pp. 224–234. [Online]. Available: https://doi.org/10.1109/MICRO.2012.29

[26] Intel Corporation. Intel AVX512 Instructions., [online]. Available: https://software.intel.com/content/www/us/en/develop/articles/intel-avx-512-instructions.html.

[27] Intel Corporation. Intel Turbo Boost Technology 2.0, [online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html.

[28] A. Jackson *et al.*, "Investigating applications on the A64FX," in *IEEE International Conference on Cluster Computing, CLUSTER 2020, Kobe, Japan, September 14-17, 2020*. IEEE, 2020, pp. 549–558. [Online]. Available: https://doi.org/10.1109/CLUSTER49012.2020.00078

[29] H. M. Jacobson *et al.*, "Abstraction and microarchitecture scaling in early-stage power modeling," in *17th International Conference on High-Performance Computer Architecture (HPCA-17 2011), February 12-16 2011, San Antonio, Texas, USA*. IEEE Computer Society, 2011, pp. 394–405. [Online]. Available: https://doi.org/10.1109/HPCA.2011.5749746

[30] H. M. Jacobson *et al.*, "Empirically derived abstractions in uncore power modeling for a server-class processor chip," in *International Symposium on Low Power Electronics and Design, ISLPED'14, La Jolla, CA, USA - August 11 - 13, 2014*, Y. Xie *et al.*, Eds. ACM, 2014, pp. 147–152. [Online]. Available: https://doi.org/10.1145/2627369.2627619

[31] C. L. Lawson *et al.*, "Basic linear algebra subprograms for fortran usage," *ACM Trans. Math. Softw.*, vol. 5, no. 3, p. 308323, Sep. 1979. [Online]. Available: https://doi.org/10.1145/355841.355847

[32] H. Q. Le *et al.*, "IBM POWER9 processor core," *IBM J. Res. Dev.*, vol. 62, no. 4/5, p. 2, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8409955/

[33] H. Patil *et al.*, "Pinpointing representative portions of large intel® itanium® programs with dynamic instrumentation," in *37th Annual International Symposium on Microarchitecture (MICRO-37 2004), 4-8 December 2004, Portland, OR, USA*. IEEE Computer Society, 2004, pp. 81–92. [Online]. Available: https://doi.org/10.1109/MICRO.2004.28

[34] E. Perelman *et al.*, "Picking statistically valid and early simulation points," in *12th International Conference on Parallel Architectures and Compilation Techniques (PACT 2003), 27 September - 1 October 2003, New Orleans, LA, USA*. IEEE Computer Society, 2003, pp. 244–255. [Online]. Available: https://doi.org/10.1109/PACT.2003.1238020

[35] P. Rajpurkar *et al.*, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, 2016, pp. 2383–2392.

[36] IBM Corporation, "IBM Power ISA^{TM} Version 3.1," [online] Available: https://ibm.ent.box.com/s/hhjfw0x0lrbtyzmiaffnbxh2fuo0fog0.

[37] J. A. Rivers *et al.*, "Phaser: Phased methodology for modeling the system-level effects of soft errors," *IBM J. Res. Dev.*, vol. 52, no. 3, pp. 293–306, 2008. [Online]. Available: https://doi.org/10.1147/rd.523.0293

[38] T. Rosedahl, IBM OpenPOWER On-chip Controller (OCC). [online]. Available: https://openpowerfoundation.org/on-chip-controller-occ.

[39] S. K. Sadasivam *et al.*, "IBM POWER9 processor architecture," *IEEE Micro*, vol. 37, no. 2, pp. 40–51, 2017. [Online]. Available: https://doi.org/10.1109/MM.2017.40

[40] M. Sato *et al.*, "Co-design for A64FX manycore processor and "fugaku"," in *SC '20: The International Conference for High Performance Computing, Networking, Storage and Analysis, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, C. Cuicchi *et al.*, Eds. IEEE/ACM, 2020, pp. 47:1–47:15. [Online]. Available: https://dl.acm.org/doi/10.5555/3433701.3433763

[41] SPEC CPU 2017, [online]. Available: https://www.spec.org/cpu2017.

[42] V. Srinivasan *et al.*, "Optimizing pipelines for power and performance," in *Proceedings of the 35th Annual International Symposium on Microarchitecture, Istanbul, Turkey, November 18-22, 2002*, E. R. Altman *et al.*, Eds. ACM/IEEE Computer Society, 2002, pp. 333–344. [Online]. Available: https://doi.org/10.1109/MICRO.2002.1176261

[43] W. J. Starke *et al.*, "IBM's POWER10 processor," in *IEEE Hot Chips 32 Symposium, HCS 2020, Palo Alto, CA, USA, August 16-18, 2020*. IEEE, 2020, pp. 1–43. [Online]. Available: https://doi.org/10.1109/HCS49909.2020.9220618

[44] N. Stephens *et al.*, "The ARM scalable vector extension," *CoRR*, vol. abs/1803.06185, 2018. [Online]. Available: http://arxiv.org/abs/1803.06185

[45] B. W. Thompto, "POWER9: processor for the cognitive era," in *2016 IEEE Hot Chips 28 Symposium (HCS), Cupertino, CA, USA, August 21-23, 2016*. IEEE, 2016, pp. 1–19. [Online]. Available: https://doi.org/10.1109/HOTCHIPS.2016.7936223

[46] C. Vezyrtzis *et al.*, "Droop mitigation using critical-path sensors and an on-chip distributed power supply estimation engine in the z14™ enterprise processor," in *2018 IEEE International Solid-State Circuits Conference, ISSCC 2018, San Francisco, CA, USA, February 11-15, 2018*. IEEE, 2018, pp. 300–302. [Online]. Available: https://doi.org/10.1109/ISSCC.2018.8310303

[47] T. Webel *et al.*, "Robust power management in the IBM z13," *IBM J. Res. Dev.*, vol. 59, no. 4/5, 2015. [Online]. Available: https://doi.org/10.1147/JRD.2015.2446872

[48] T. Webel *et al.*, "Proactive power management in IBM z15," *IBM J. Res. Dev.*, vol. 64, no. 5/6, pp. 15:1–15:12, 2020. [Online]. Available: https://doi.org/10.1147/JRD.2020.3008143

[49] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Oct. 2020, pp. 38–45.

[50] Z. Xianyi *et al.*, "OpenBLAS : An optimized BLAS library," openblas.net.

[51] M. Ziegler *et al.*, "Machine learning techniques for taming the complexity of modern hardware design," *IBM J. Res. Dev.*, vol. 61, no. 4-5, p. 13, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8032559/

[52] V. Zyuban *et al.*, "Balancing hardware intensity in microprocessor pipelines," *IBM J. Res. Dev.*, vol. 47, no. 5-6, pp. 585–598, 2003. [Online]. Available: https://doi.org/10.1147/rd.475.0585