Towards an Integrated Strategy to Preserve Digital Computing Performance Scaling Using Emerging Technologies

Dilip Vasudevan(✉), Anastasiia Butko, George Michelogiannakis, David Donofrio, and John Shalf

Computer Science Department, Lawrence Berkeley National Lab, One Cyclotron Road, Berkeley, CA, USA
{dilipv,abutko,mihelog,ddonofrio,jshalf}@lbl.gov

Abstract. With the decline and eventual end of historical rates of lithographic scaling, we arrive at a crossroad where synergistic and holistic decisions are required to preserve Moore's law technology scaling. Numerous emerging technologies aim to extend digital electronics scaling of performance, energy efficiency, and computational power/density, ranging from devices (transistors), memories, 3D integration capabilities, specialized architectures, photonics, and others. The wide range of technology options creates the need for an integrated strategy to understand the impact of these emerging technologies on future large-scale digital systems for diverse application requirements and optimization metrics. In this paper, we argue for a comprehensive methodology that spans the different levels of abstraction – from materials, to devices, to complex digital systems and applications. Our approach integrates compact models of low-level characteristics of the emerging technologies to inform higher-level simulation models to evaluate their responsiveness to application requirements. The integrated framework can then automate the search for an optimal architecture using available emerging technologies to maximize a targeted optimization metric.

1 Introduction

Far from a physical law, Moore's law is a techno-economic observation on doubling the number of transistors per square inch of an integrated circuit. This led to Moore's subsequent observation that "shrinking the dimensions on an integrated structure makes it possible to operate the structure at higher speed for the same power per unit area" [8]. The expectation that early in the next decade 2D lithography will cease scaling, threatens the future of Moore's law. In response, a number of promising emerging technologies have been proposed. These alternative technologies appear throughout all levels of computing devices, ranging from transistors (devices), memory technologies and 3D integration to hybrid architectures, specialization, etc. [4]. While it is unlikely that a single technology will prevail to drive Moore's law, a combination of emerging technologies together with explicit understanding of application requirements is likely

© Springer International Publishing AG 2017

the solution [8]. This realization contradicts today's common practice of developing and evaluating each technology in isolation. Therefore, it is imperative to develop a comprehensive strategy to determine the optimal path forward to preserve digital computing performance scaling. To achieve this, we argue for a two-fold strategy that is composed of:

– an algorithmic methodology that takes into account each candidate technology's characteristics, and produces an optimal combination of emerging technologies for a given metric and application.
– a simulation and evaluation infrastructure across different levels (e.g., devices, circuits, architectures), that supports performance and cost models of emerging technologies.

2 Motivation and Background

Numerous emerging devices follow the traditional CMOS model but offer improved voltage–current characteristics. Devices such as tunnel FETs and carbon nanotube FETs have been demonstrated in practice, and initial studies discuss their impact to chip multiprocessors [1,7]. A recent influential study projected the future potential of several devices in terms of the energy–delay product to implement a 32-bit adder [5]. Similarly, new memory technologies offer attractive bandwidth, latency, and cost tradeoffs compared to traditional DRAM [6]. In fact, some memories such as resistive RAM are non-volatile and can be placed close to computational cores. Furthermore, 3D integration is growing capable of tens to hundreds of memory layers with fast and cheap inter-layer communication, and is projected to enable multiple logic layers as well in different interleaved patterns. Moreover, specialization is an attractive technique to remove performance and cost overhead of general-purpose architecture. Numerous other technologies are also promising candidates to preserve digital computing performance scaling, such as photonics and stochastic computing.

The heterogeneity of emerging technologies creates a challenge because different technologies are more suited for different applications and no clear winner is expected to emerge. In addition, a choice of one technology affects other choices when designing a complete chip or system. Finally, previous studies typically develop one solution in isolation and do not investigate any synergistic opportunities with other technologies. We propose a comprehensive strategy that paves the road forward for extending Moore's law using the entire range of emerging technologies.

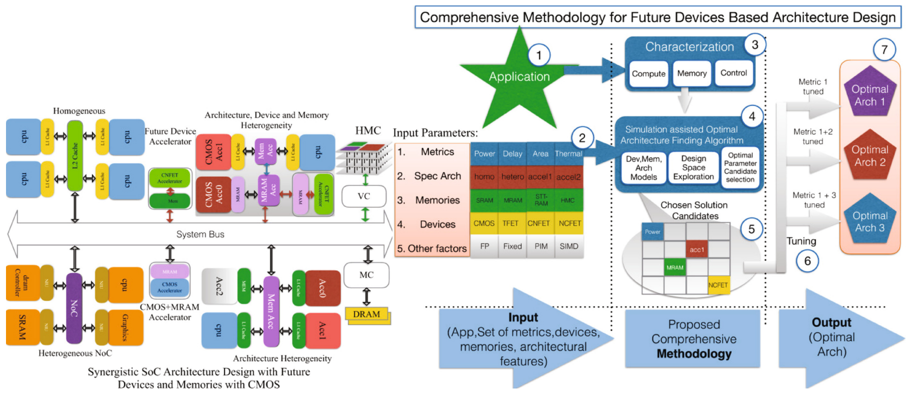
3 Towards an Integrated Methodology for Comprehensive Evaluation of Optimal Architectures

Developing an integrated methodology to preserve Moore's law performance scaling by designing future architectures that use emerging technologies is made particularly challenging by the sheer number of options (technologies) available at each level, the dependencies between these options, and the increasing demands from the applications. Namely, some of these options are:

– Heterogeneous or homogeneous architectures. This includes the entire range from fully to partially and then to non-programmable fixed-function specialized hardware.
– Devices like TFET (Tunnel Field-Effect Transistor), CNFET (Carbon-Nanotube FET), NCFET (Negative Capacitance FET) and superconducting circuits like RSFQ (Rapid Single Flux Quantum) devices, and many others. There are to the order of a dozen emerging devices in today's literature in different stages of maturity [5].
– Future memory technologies like MRAM, STT-RAM etc., some of which are non-volatile and can be integrated close to computation cores. There are to the order of ten emerging memory technologies.
– 3D integration with memory such as conventional DRAM or HMC (Hybrid Memory Cube), as well as future capabilities of deep 3D integration with hundreds of interleaved logic and memory layers, and different inter-layer technologies.
– Implementation constraints like thermal equilibrium, area and power budgets and other factors like floating point versus fixed point representation, SIMD (Single Instruction Multiple Data), PIM (Processing In Memory) based architectures etc.

To illustrate the dependency between application demands for performance per watt and area and the several choices impacting architecture design, let us consider a sample set of applications: namely a climate modeling application code (MPES), a memory intensive graphics application, and the Fourier transform (FFT). These applications are power constrained, perform complex computations, but also demand high throughput of data from memory. Given these application constraints, as well as an optimization metric such as performance per watt, the globally optimal architecture using the most optimal technologies in each layer from the ones mentioned previously could be the one shown in Fig. 1. That architecture uses accelerators implemented with CNFETs to satisfy computational demands, HMC memory to satisfy high memory access bandwidth, high heterogeneity to optimize for power, and other choices in both architectures and other technologies such as devices and memories always to satisfy application demands and optimization metrics. Note that to truly be the globally optimal solution, it must consider how a choice of technology affects others and avoid locally optimal solutions.

Our goal is to design a comprehensive methodology to perform multi-objective optimization and systematically design a globally optimal architecture for a given metric and for a given set of application characteristics, using all available technology options from all the different layers previously explained. To achieve this we have to operate and optimize at various levels of the system architecture from devices, circuits, accelerators and others, but also to include potential modifications to programming languages, compilers, ISAs, and others.


Homogeneous
Future Device Accelerator
Architecture, Device and Memory Heterogeneity
HMC
VC
System Bus
MC
DRAM
CMOS+MRAM Accelerator
Heterogeneous NoC
Architecture Heterogeneity
Synergistic SoC Architecture Design with Future Devices and Memories with CMOS
Comprehensive Methodology for Future Devices Based Architecture Design
Application
Characterization
Compute
Memory
Control
Input Parameters:
1. Metrics
2. Spec Arch
3. Memories
4. Devices
5. Other factors
Power Delay Area Thermal
homo hetero accel1 accel2
SRAM MRAM STT-RAM HMC
CMOS TFET CNFET NCFET
FP Fixed PIM SIMD
Simulation assisted Optimal Architecture Finding Algorithm
Dev,Mem, Arch Models
Design Space Exploration
Optimal Parameter Candidate selection
Chosen Solution Candidates
Tuning
Metric 1 tuned
Metric 1+2 tuned
Metric 1 + 3 tuned
Optimal Arch 1
Optimal Arch 2
Optimal Arch 3
Input (App,Set of metrics,devices, memories, architectural features)
Proposed Comprehensive Methodology
Output (Optimal Arch)

(a) *Future Architecture* (b) *Proposed Methodology*

Fig. 1. a. An example of a future architecture implemented with heterogeneous devices, b. Comprehensive and synergistic methodology for finding a global optimum by exploring the different levels of system design, including emerging devices and specialized architectures (only a subset of options is shown)

Methodology: Figures 1 and 2 shows our proposed methodology that involves seven constructive steps to be taken to arrive at a globally optimal architecture. The steps are:

1. Gathering any software engineering requirements of the application (functional and non-functional), such as any compiler and programming language requirements.
2. Secondly, this methodology requires comprehensive details and models of available emerging technologies in several levels of the system ranging from devices to architecture (logic design). In this step, a set of optimization metrics is also provided.
3. Next, characterization of the application code for compute, memory and control intensive instructions, to derive performance and other application requirements such as data movement, floating point computation, etc.
4. In the next step, the inputs generated from the previous three steps drive the optimal architecture finding algorithm. This step includes constraint optimization to make sure the solution is acceptable. This algorithm is further described below.
5. The fifth step produces a baseline architecture using the choices in technologies made by the algorithm in the previous step.
6. In this step, the architecture constructed is further tuned toward a subset of target optimization metrics.
7. Finally, a set of architectures and choice of emerging technologies targeting the given application will be generated. Each architecture will be a globally optimal solution for a choice of given metrics.

To formulate the inputs that step four (the algorithm) requires, we need a way to first identify and record the relevant characteristics and needs of applications, and then input those to our framework. This includes an array of factors such as data movement, memory access, floating point compute, and others shown as a row named "other factors" in Fig. 1. Each application's needs translate to different weights for each of these factors. Another input is the available technologies (step two). For those, we need to quantify the impact of each technology to application needs. Doing so requires reliable performance and cost models for each technology, which we describe in Sect. 4.

Algorithm: The algorithm itself (step four) will be formulated as a graph optimization problem with the design space represented as a graph with weighted vertices and edges. The algorithm is illustrated in Fig. 2. As shown, each vertex represents one computational unit, which is comprised of a choice of device technology (dev), memory technology (mem), logic design (logic), and as a result of these and other microarchitectural and emerging technology choices has a certain energy–delay (ED) product. Only four parameters are shown in this example. In other words, each vertex is just one of the possibly many computational units of the architecture. Edges, on the other hand, represent cost of communication in delay and energy between computational units. Using this notation, finding an optimal architecture (step six in the Figure) is a multi-objective graph optimization problem of finding a path in the graph where a given metric, such as performance over watt, is optimized. This path represents an architecture where the different components were chosen because their corresponding vertices were in the chosen path of the graph. This algorithm builds from obtaining and embedding into the feature vector of each vertex (step two in the Figure) simulation and energy–delay (ED) results for the various general-purpose cores, adders, multipliers, FFT accelerator blocks, and other computational blocks, for a given set of devices, memories, logic design, etc. From the chosen path, a new feature vector will be constructed for each computation block in the resulting architecture (step five in the Figure). In the example in Fig. 2, the feature vectors found in the enumerated red path will suggest a computational block with 2 adders, a multiplier, a FFT implemented using different devices and a 32KB MRAM for a constraint of low power for multiplier and a high memory density.

The outcome of this algorithm is an architecture with combination of technologies that reach a global maximum *for a specific metric and application.* From this, we can "average out" for a general-purpose solution and shape a general strategy to preserve digital computing scaling. Certainly, this output should consider the potential each technology has on top of its current state. Thus, for the example set of applications (MPES and FFT and graphics) introduced earlier in this section, the algorithm will find an optimal set of input parameters (technologies) while considering optimization metrics such as for power and memory bandwidth, and arrive at the selected parameters shown in step five in Fig. 1. After further tuning of the derived architecture using these metrics in step six will generate the required optimal architectures tuned for a specific set of metrics

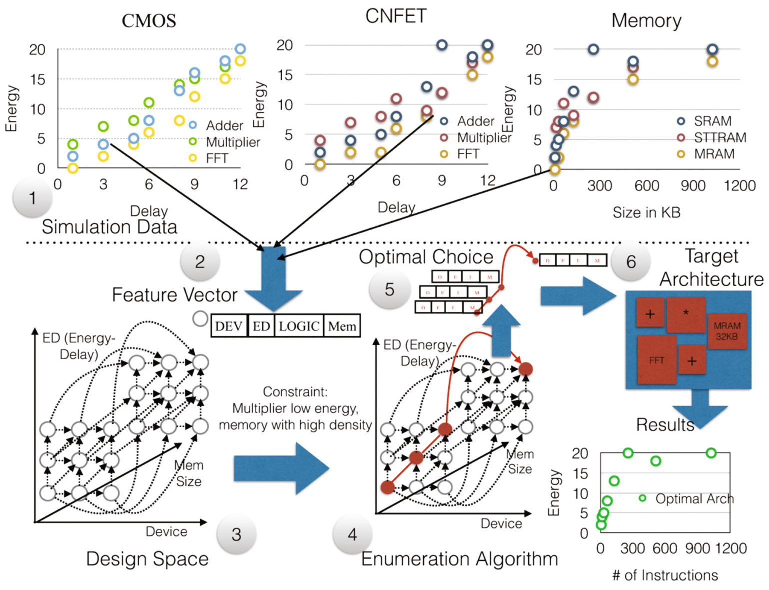


Fig. 2. Illustration of the algorithm for the optimal architecture selection. The energy–delay (ED) graphs are for illustration purposes only and not in scale.

shown as step seven in Fig. 1. Therefore, for our applications we will arrive at possibly at a different architecture per application.

4 Modeling Environment

4.1 Simulation Infrastructure

In order to generate the technology models necessary to conduct realistic experiments and therefore guide our methodology, we propose our simulation infrastructure as shown in Fig. 3. The proposed infrastructure consists of four main modeling levels: (i) device models, (ii) logic gates, (iii) logic and memory blocks and (iv) architecture. Each level contains three key components illustrated with colored blocks. Yellow blocks represent the *Target Modeling Unit*, which can be a device model, logic gate, accelerator, etc. depending on the modeling level. Blue blocks represent a set of *Evaluation tools* capable of providing target *Output metrics*. These output metrics are then used as *Input metrics* in the next modeling level. Here, we describe each modeling level in detail.

LEVEL 1: Device Models. In this level, we start from low-level device physics and generate current–voltage curves using Xyce, a open-source SPICE tool. In this step, detailed knowledge of device physics and operating conditions is required. Open-source Verilog-A models are already available for many of the emerging devices, but care must be exercised in verification of those models,

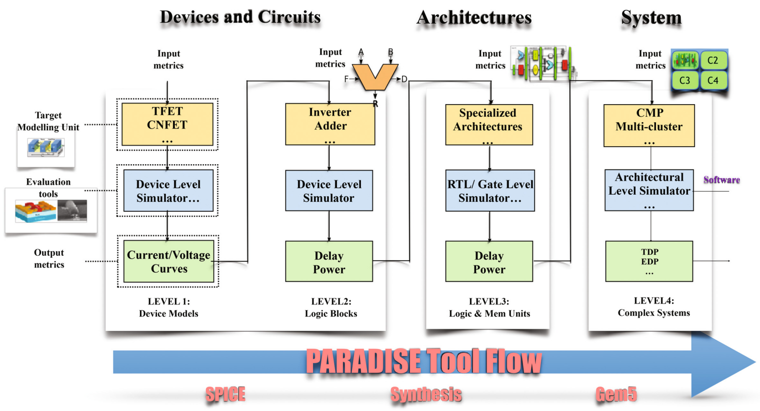


Fig. 3. Design space exploration flow. The example refers to results in Fig. 4. (Color figure online)

capturing the future potential of each device, and generating models for devices for which no models are available. Further metrics of interest such as error rates can also be captured in this step.

LEVEL 2: Logic Gates and Blocks. In this level, we use device models from the previous level to construct small functional blocks such as logic gates, adders, ring oscillators, and others. We do this by adding Verilog-A models from the previous emerging devices to Xyce, and describing the small functional blocks of this level in an Xyce netlist that uses emerging device models. From these experiments we extract delay and power for each block, and use that as inputs to the next higher level. Adders and multipliers result shown in Fig. 4-a and b are built using these logic gates.

LEVEL 3: Logic and Memory Blocks. In the logic and memory level, several new technologies can be modeled such as memories, 3D integration, and specialized architectures. Emerging memories have sometimes vastly different access times, energies, volatility, error rates, etc. In fact, memory access times for particular data may depend on the location of the data and the previous sequence of requests, which means that a careful study is required to investigate the level of accuracy that is sufficient versus the complexity of models. Furthermore, 3D integration affects distances and relevant performance-cost tradeoffs between any two points, especially when considering inter-layer communication technologies such as TSVs. 3D integration models need to include future capabilities such as deep integration with hundreds of layers, with multiple combinations of logic and memory. Finally, specialized architectures such as accelerators and fixed-function hardware have a range of different delays and energy costs to perform an operation. Non-programmable application specific hardware.

In order to rapidly model the technologies of this level, we can extend a modern HDL modeling tool such as Chisel [2] and use it to describe different

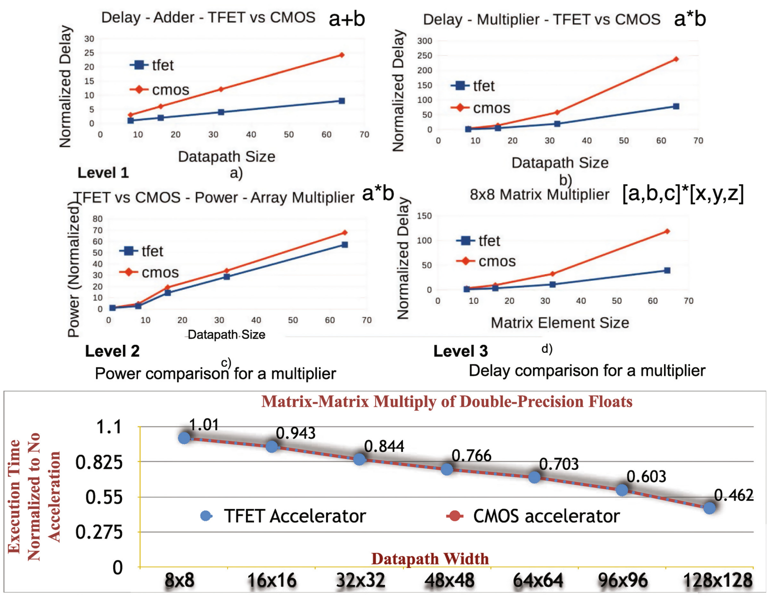


Fig. 4. a–d. Delay and Power comparison for adders and multipliers implemented with TFET and CMOS, e. Execution time impact for a matrix-matrix multiply code using an accelerator compared to a non-accelerated version. Results are for a single core co-located with the accelerator. While the CMOS accelerator is 3× slower resulting in two cycles instead of one for the TFET accelerator, the architectural-level impact is negligible.

logic and memory blocks. Chisel can be extended to capture and back annotate activity factors of logic gates and wires to models of emerging technologies from the previous two levels, and therefore generate the output metrics (delay, power, etc.) that are necessary for the next level. Figure 4-c and d show the delay comparison of adders and multipliers implemented using CMOS and TFET. Normalized delay and power shows that TFET based arithmetic units have higher performance and power benefit.

LEVEL 4: Architectural Level. To conduct high-level experiments, we use a software architectural-level simulator such as Gem5 [3]. A preliminary study of the impact of accelerators using TFET and CMOS devices is shown in Fig. 4-e. This shows that even though a TFET-based accelerator is 3× faster, the impact to the application is negligible. Still, this different can become substantial for large matrices, to which we are extending our study. This infrastructure will be capable of evaluating potentially vastly heterogeneous systems with all emerging technologies as options. This may require a parallel architecture simulator. In addition, we plan to extend high-level area and power models to include new devices.

5 Conclusion

Building systems using future devices and other technologies involves several levels of modeling with many factors to consider. The choice of components at each level impacts the choice in other levels and consequently the overall power and performance of future systems. Optimizing at only one level, such as solely focusing on new transistors/devices will lead only to a local optimal optimization point. However, a holistic approach to optimize the system at all levels for given optimization metrics and application needs will lead to a globally-optimal solution. In this paper, we present both a comprehensive methodology and a detailed modeling approach for emerging technologies capable of paving the path forward for preserving the performance scaling of digital computing.

References

1. Aly, M.M.S., Gao, M., Hills, G., Lee, C.S., Pitner, G., Shulaker, M.M., Wu, T.F., Asheghi, M., Bokor, J., Franchetti, F., Goodson, K.E., Kozyrakis, C., Markov, I., Olukotun, K., Pileggi, L., Pop, E., Rabaey, J., Rè, C., Wong, H.S.P., Mitra, S.: Energy-efficient abundant-data computing: the N3XT 1,000x. Computer **48**(12), 24–33 (2015)
2. Bachrach, J., Vo, H., Richards, B., Lee, Y., Waterman, A., Aviienis, R., Wawrzynek, J., Asanovi, K.: Chisel: constructing hardware in a scala embedded language. In: DAC Design Automation Conference 2012, pp. 1212–1221 (2012)
3. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M.D., Wood, D.A.: The Gem5 simulator. SIGARCH Comput. Archit. News **39**(2), 1–7 (2011)
4. Cavin, R.K., Lugli, P., Zhirnov, V.V.: Science and engineering beyond Moore's law. In: Proceedings of the IEEE 100(Special Centennial Issue), pp. 1720–1749, May 2012
5. Esch, J.: Overview of beyond-CMOS devices and a uniform methodology for their benchmarking. Proc. IEEE **101**(12), 2495–2497 (2013)
6. Poremba, M., Mittal, S., Li, D., Vetter, J.S., Xie, Y.: DESTINY: a tool for modeling emerging 3d NVM and eDRAM caches. In: 2015 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1543–1546, March 2015
7. Saripalli, V., Mishra, A., Datta, S., Narayanan, V.: An energy-efficient heterogeneous CMP based on hybrid TFET-CMOS cores. In: 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 729–734, June 2011
8. Shalf, J.M., Leland, R.: Computing beyond Moore's law. Computer **48**(12), 14–23 (2015)