Simpler, More Efficient Design

Borivoje Nikolić
Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA
E-mail: bora@eecs.berkeley.edu

***Abstract*—Design of custom integrated circuits has become prohibitively expensive for many application domains. As a result, these domains often choose to implement the desired functionality on programmable platforms, but those solutions are less energy efficient. This paper proposes several approaches for making the design process more efficient and enabling custom energy-efficient integrated circuits. Function generators, as opposed to function instances, should be designed, which combined with higher-level design abstraction improve design efficiency and foster reuse. The use of generators also enables modular designs, aiding design verification. Rapid design flow maps generated modules into silicon and enables design-space exploration for optimal efficiency. Open-source repository of function generators and their mappings into systems allow designers to selectively add value to the design. These principles are demonstrated on a design of a processor, based on an open-source instruction set architecture, with integrated switched-capacitor DC-DC converters implemented in 28nm FDSOI. The chip is designed with a relatively small team and features high conversion efficiency (80-86%) and high energy efficiency (26.2 DP GFLOPS/W).**

Keywords*—Energy efficiency, CMOS, microprocessor, design-space exploration, voltage regulators.

I. Introduction

The efficiency of designing and manufacturing integrated circuits has enjoyed enormous improvements in the past 50 years. This growth has been characterized through Moore's law, which states that IC design would be doubling its efficiency every two years. After initial rapid growth during the first decade, the semiconductor industry has been able to double the number of devices integrated on a single chip in each generation, roughly every two years, up to today [1]. This trend cannot continue indefinitely, and the end of CMOS technology scaling is expected to slow down in the immediate future. The ability to integrate an increasing number of devices on a silicon chip is illustrated in Fig. 1, which shows a plot of the number of transistors in Intel's microprocessors over time.

The design process has been able to utilize twice as many components available in each new technology in the past decade by replicating unit components and functions. The performance improvements in computing came through doubling of the number of processing cores.

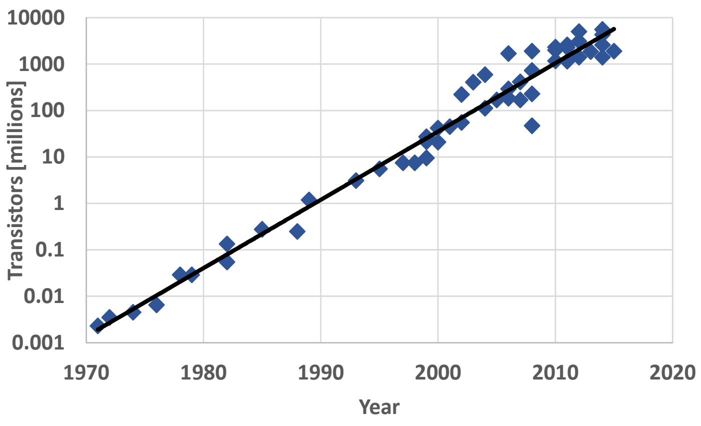

Figure 1: Transistor counts in Intel's microprocessors over the years. The trend line corresponds to doubling the number of transistors every two years.

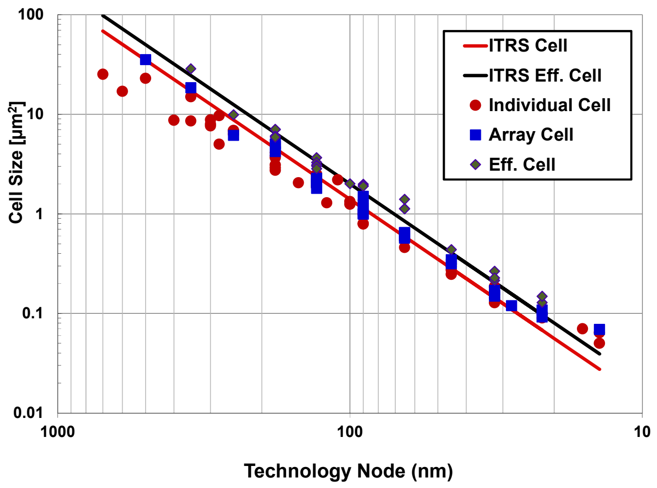

Figure 2: Scaling of the SRAM cell size across the technology nodes.

The amount of cache memory has been doubling in each generation of leading-edge microprocessors as well, owing to the continued scaling of SRAM cells. The 6-transistor SRAM cell has been shrinking in half in every technology node in the past two decades, but the reduction in size has been showing some signs of slowdown in the past two technology generations, as shown in Fig. 2. Furthermore, the density of large cache arrays has not been able to follow the 2x increase in each generation because of the challenges of maintaining robust read and write margins in SRAM. This is due to the reduction in column heights and the deployment of various assist techniques, all resulting in lower transistor densities in leading microprocessors.

The proliferation of multi-core processors utilizes the doubling of the transistors available in each new generation without doubling the design effort. To achieve the same processing throughput as a single-core processor, the N-core processor can operate at roughly N-times lower clock rate, which allows for frequency scaling and significantly increased energy efficiency. Continued increase in the number of cores relies on user's ability to extract parallelism in implemented applications.

Very large levels of parallelism can exist by customizing functions implemented on chip. This parallelism can be obtained by mapping concurrent functions into custom accelerators, or, ultimately, by mapping an algorithm into an application-specific circuit. However, ability to map custom functions onto silicon is limited by designer productivity and the overall cost of the design. Higher level of abstraction and IP core reuse have enabled more complex designs; however, the majority of digital designs are still described in traditional hardware description languages (HDLs), like Verilog and VHDL, while mixed-signal circuitry is still custom designed at transistor level. The penetration of high-level synthesis (HLS) methods in custom designs has been limited. Design mapping onto field-programmable gate arrays (FPGAs) addresses a part of the design cost, offering lower function cost in low volume products, but primarily it enables a rapid turnaround time for debug and verification/validation steps. The use of higher level abstraction in design synthesis is also more prevalent in FPGA designs. However, performance of FPGA designs is lower than that of ASICs, and energy per function is higher. This power-performance gap traditionally spans the 10-100x range [2]. Simultaneously, when accounting just for the hardware fabrication cost between the ASICs and FPGAs, the watershed point is reached with volumes of just several thousand for large designs.

The design process needs to change to provide a path to achieving energy-efficient, cost-effective, high-performance designs of the future [3]. The complexity of systems implemented in software nowadays by far outweighs the complexity of software that runs it. This is enabled by the fact that software design has a naturally rapid prototype turnaround time, has been able to move to higher levels of abstraction, has been modularized to employ large design teams, and largely relies on open-source components. Hardware design becomes more efficient by adopting the principles of software development, as outlined in this paper.

II. Designing Generators, Not Instances

Energy efficiency is achieved through hardware specialization, such that every function is implemented in an optimal way. However, designing specialized hardware for every possible application is very costly.

A. Hardware generators

An emerging solution is to design function generators instead of the specific instances for each function. For this to be practical, it is necessary to have language and tool support that will enable designers to naturally implement generators, rather than particular instances of a function [4-6]. Fortunately, languages for writing generators, such as Chisel, Genesis2, and Spiral are emerging to aid the designers in this objective. Chisel (Constructing Hardware in Scala Embedded Language) is a hardware construction language aimed at designing hardware by using parameterized generators [5]. Chisel is based on the Scala programming language, and is built by extending Scala. It supports a combination object-oriented and functional programming and supports good software engineering techniques.

The key to efficient design and reuse is in using generators, that is, parameterized descriptions of a set of design choices. Generators are naturally designed in Chisel, which offers a higher level of description and a natural parameterization of the design. The use of generators enables design space exploration through sweeping the parameters of the design. The Chisel code is compact, due to its higher level of description than traditional hardware description languages.

Hardware development in Chisel mimics the good practices from software engineering that are employed in large software projects.

B. Analog Generators

Synthesis of analog and mixed-signal functions has been a long-standing hard problem [7]. It is highly motivated by the fact that analog circuits are designed at a low level, through a process of manual iterative entry of schematics and layout, followed by extraction and simulation. Having a tight iteration loop through a layout phase is essential for analog designs, where the performance is largely dictated by layout-dependent parasitics. There have been numerous attempts at synthesis of analog circuits, both in research and in commercial enterprises, but there is no widespread adoption.

One promising approach to analog design automation is in building analog generators. Although analog and digital generators are similar in spirit, they employ different principles of operation. By starting from a hand-optimized design instance, the purpose of an analog generator is to recreate the design for a different specification or for a different target technology. The Berkeley analog generator (BAG) is a design framework that enables designers to completely capture and automatically re-execute the methodologies they use to design their analog/mixed-signal circuits [8]. Given a set of target specifications, generators written in BAG produce schematics and DRC/LVS clean layouts (along with associated verification infrastructure) of circuits that meet these requirements. Beyond quickly producing new designs as specifications and/or process technologies evolve/change, by exploiting hierarchy, these generators can inherently capture the designer's intent of how to best optimize the sub-components as the high level specifications of the overall system are changed.

Currently, BAG is a Python-based, hierarchical framework that allows designers to systematically capture the procedures they use to design their circuits given a set of specifications. It relies on Synposys' PyCell framework and a set of base classes that capture common layout styles [8].

C. *Embedded memories*

Efficient data processing relies on the designers' ability to integrate large amounts of embedded memory on chip. The memory blocks often have different requirements in size and in speed, depending on the function they are embedded in. Static random access memory (SRAM) has been the workhorse in embedded processing due to its favorable tradeoffs between size, speed and power.

SRAM arrays are typically generated by using memory compilers, which naturally fit the paradigm presented in this paper.

The most commonly used 6-transistor (6-T) SRAM cell has been a benchmark for CMOS technology scaling over the past several decades, illustrated in Fig. 2. With typical linear geometry shrinks of 0.7 between nodes, the 6T SRAM cell area is reduced by 50%. Although cell size continues to scale in the most recent technology nodes, its size has been lagging this ideal scaling trend.

Fig. 2 illustrates the slowdown in scaling of memory density. Individual cells in Fig. 2 have been presented at leading technology conferences over the years. While the individual cells were able to beat the requirements set by the International Technology Roadmap for Semiconductors (ITRS) in the past, now they barely meet them. As a result, array cells, representing cells used in practical products, typically fall behind the traditional scaling trend. Finally, when the array overhead is accounted for in the effective cell area, a further slowdown can be observed.

SRAM affects the energy efficiency of the entire system through its both active and leakage energy components. For overall system efficiency, in many applications it is essential that SRAM operates at the same supply voltage as the logic, as the decoupling of voltage domains incurs too large of a performance penalty. SRAM cell is accessed through a shared transistor for both read and write operations, which creates tradeoffs in achieving wide read and write margins, particularly at low voltages [9].

SRAM operation at low voltages often requires decoupling of read and write operations by the use of 8-transistor (8-T) SRAM and by using the assist circuit to aid the write. Both of these techniques reduce the overall memory density. An 8-T cell is typically 50% larger than a 6-T cell, and the assist circuitry additionally increases the effective cell size.

Unfortunately, there are no viable embedded memory alternatives on the horizon. Embedded DRAM (eDRAM) has been an attractive alternative for implementing large cache arrays in microprocessors, but its area efficiency suffers at smaller granularities needed for a broad range of systems. Numerous embedded non-volatile memories – such as spin-transfer torque (STT-RAM), resistive RAM (ReRAM), or phase-change memory (PCRAM) – are being actively investigated for integration with CMOS. They all need improvements in either energy efficiency or endurance to be viable alternatives to SRAM. When that happens, they would naturally need generators for ease of integration.

III. Open-Source Hardware

A major challenge for custom chips is the cost of their design. This is compounded by very rich functionality resulting in high complexity, while the economic value added by each feature is not evenly distributed across all functions. Modern chips must provide all interfaces necessary for seamless integration into complex products. Many of the interfaces are rather simple, and fully commoditized. Their design does not provide a challenge, but also does not add value to the final product, yet must be part of a chip – hence a proposal to include it on a standardized chip in a multi-chip module [10]. Examples include SPI and I2C, but also more complex interfaces such as USB, WiFi, or Bluetooth. The value proposition of new chip is often not in improving the traditional interfaces, but rather in the new functionality that the chip enables. From this standpoint, open-source hardware presents a path for aligning the added value in a chip with the design effort.

Open-source model has revolutionized software development in the past ~15 years, and opening the source for hardware designs provides a similar opportunity.

A. *RISC-V: An Open ISA*

Processors are ubiquitous in complex chip designs, and multiple processor cores are often embedded in a typical modern SoC. A huge majority of these embedded processors are based on proprietary instruction-set architectures (ISAs), but much broader innovation is enabled with the existence of an open ISA. While there are several open ISAs in use, the emergence of RISC-V and its eco-system is an interesting one, as it closely follows the principles of open-source development [11]. It is a modern ISA, with small number of instructions, designed for extensibility. It supports the IEEE-754-2008 floating-point standard, variable-length instruction encoding, and 32-, 64- and 128-bit address spaces. A complete software tool stack is available including GCC, Linux, LLVM, and a verification suite.

RISC-V cores are built from Chisel generators (e.g. a "Rocket generator") and are freely available. They present a good example of the use of hardware generators. There has been several hardware implementations of RISC-V processors in 45nm and 28nm CMOS technologies.

B. *From Open-Standards to Open-Source*

Many of the chip and system interfaces are based on open standards (such as TCP/IP), yet implementations of a particular instance are generally proprietary. Like the way microprocessors can be open-sourced, many commodity interfaces can be developed in the open-source model as well. These designs can be implemented as generators, where a particular instance would correspond to a desired part of a standard needed for a SoC.

For instance, consider a generator for wireless basebands, illustrated in Figure 3.

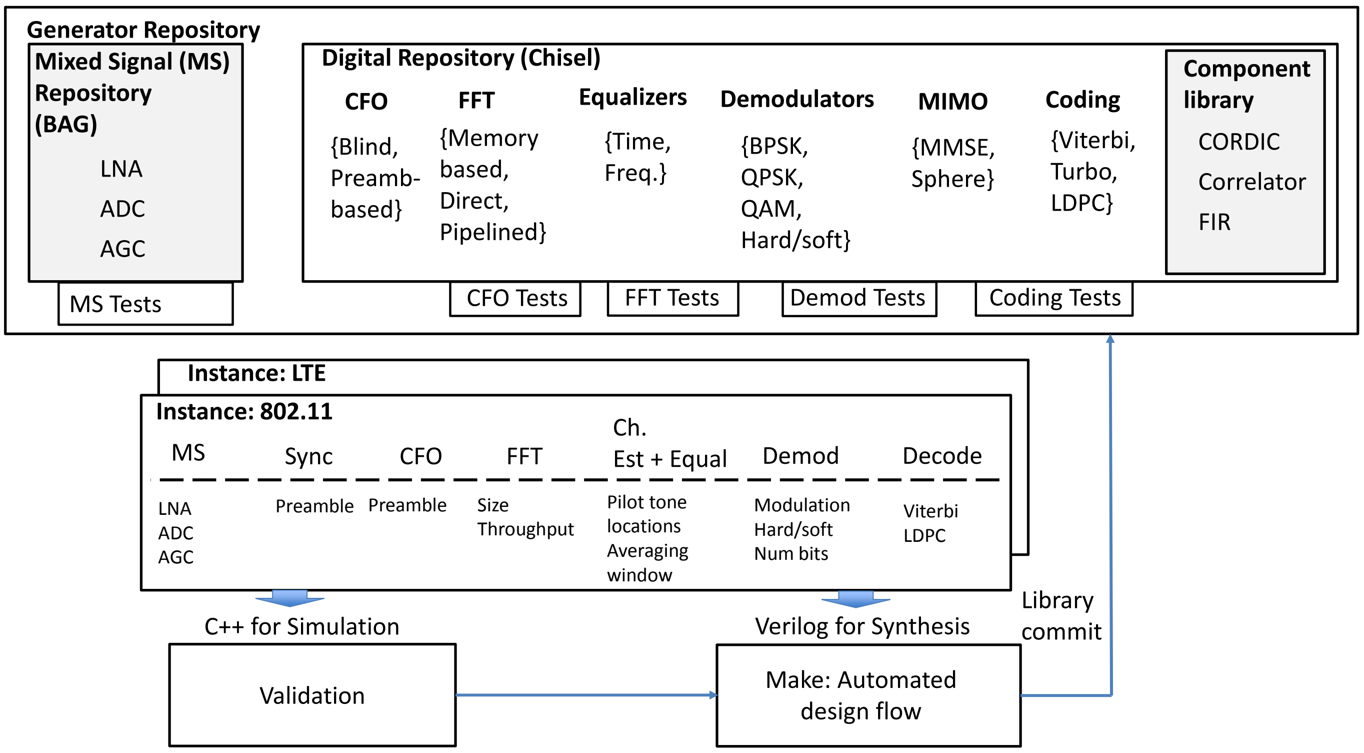

Figure 3: Example of a system of repositories for generating processing for wireless basebands.

Many of the modern wireless standards share common features. At the PHY layer, signal processing for e.g. LTE and WiFi modems is very similar, and this similarity can be further outlined by selecting the right module boundaries. Since both systems implement OFDM modulation, a fast-Fourier transform (FFT) generator could be used to generate 64-, 128-, 256-, 512-point FFTs for IEEE 802.11x standards and 128-, 256-, 512-, 1024-, 1536-, 2048-point FFTs for LTE.
A system of repositories, analogous to software repositories used in software development today, can be established to foster the collaboration in generator design and design reuse. The repositories support publishing, building, and installing of hardware components, where components can be built from sources or distributed in binary form on all major platforms. A generator repository would contain generators, both in Chisel and BAG, for all functions commonly found in a baseband PHY. Everything that constitutes the design is stored in a repository: its hardware description, simulation and verification modules, and documentation. A new repository would be set up for each particular instance of use, such as different implementations of WiFi standards, for different applications. Given that the majority of time in the development cycle is spent on verification, it is necessary that the validation is modularized as well.

IV. Rapid Design Flows

Agile hardware development relies on a rapid design flow that maps a description of hardware into layout [2]. Modern synthesis and place-and-route tools are robust enough that they can be scripted to produce a clean layout with a push-button flow. Such flow enables design exploration from a high-level description and by changing generator parameters. Even if each mapping of the design into silicon is suboptimal in terms of performance and energy efficiency, often the overall design is highly optimized through architectural tradeoffs.
Chisel allows users to code their designs in one source description and target multiple backends without rewriting their designs.

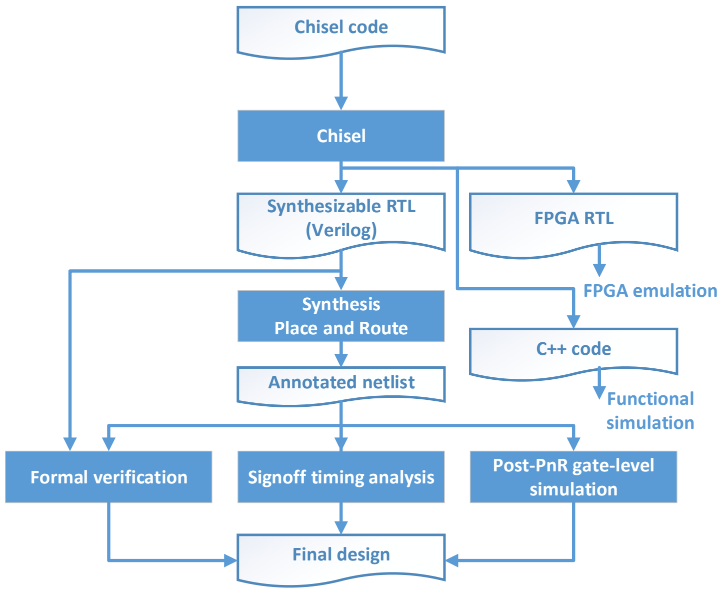

Figure 4: Illustration of a rapid design flow, for mapping Chisel-generated RTL code into hardware.

Chisel includes a C++ target as well as FPGA and ASIC Verilog targets. The C++ target is run using a fast C++ simulator for rapid design point evaluation; this simulator is approximately 8x faster than the Verilog VCS simulator. The fast C++ accelerates cycle accurate performance evaluations as well as testing and debugging. Finally, by providing memory abstractions and backend-specific implementations, Chisel also provides strong support for the production of Verilog tailored to either FPGAs or ASICs.

Generated RTL code is synthesizable and enters an automated scripted design flow, one iteration of which is illustrated in Fig. 4.

V. Efficient Design Example

A project of designing a RISC-V microprocessor with integrated switched-capacitor DC-DC converters outlines many of the design principles highlighted in this paper. This high-performance, very energy efficient design has been executed with a small design team. The architecture of the processor is open, and integrates a vector accelerator. The use of generators throughout the design makes it scalable and the components are highly reusable.

Voltage regulators (VRs) are one of the remaining components to be integrated in a SoC. Efficient design requires many voltage domains with dynamic voltage scaling (DVS) to achieve optimal energy efficiency. In most practical systems today, voltage regulators are of a buck topology, on a separate chip, and rely on external magnetic components to achieve high efficiency of conversion, typically higher than 90%. In addition to the cost of components, the challenge of external VRs is typically long response to voltage changes, requiring complex system control. With the increase of the number of power domains in modern chip, it is impractical to have many external VRs, and there is a need to integrate them on chip.

Fully-integrated linear regulators have a fast response, but suffer from low efficiency at large step-down ratios [14]. Recent Intel's microprocessors integrate switches for buck converters on-chip, and use in-package inductors to achieve high efficiency (up to 90%) and fast switching times [15].

Switched-capacitor (SC) DC-DC converters can be completely integrated on a chip, but have remained in the research domain so far because of low efficiencies (about 80%) and high silicon area, due to low current densities [16]. To minimize the inherent supply ripple, SC converters typically switch capacitors in many phases, which limits efficiency because of charge sharing losses between the phases [17].

A processor with an integrated SC DC-DC converter has been designed based on many of the principles outlined in this paper. To achieve high conversion efficiency, the DC-DC converter is not interleaved, and its unit cells switch simultaneously to avoid charge sharing losses [18]. To utilize the charge under the ripple, an adaptive clock translates a higher instantaneous voltage into a higher frequency [19].

The 64-bit processor is implemented in a 28nm fully-depleted silicon-on-insulator (FD-SOI) process with ultra-thin body and buried oxide (UTBB) [20]. It is an implementation of the RISC-V ISA and is generated by the Rocket generator. The DC-DC converter is supplied with two external voltages, a 1.0V core and 1.8V I/O, and generates four dynamically reconfigurable average output voltages of 1.0V, 0.9V, 0.67V, and 0.5V. An adaptive clock generator adjusts the clock period of each cycle based on the instantaneous converter output voltage. The use of BAG in designing the DC-DC converter cells, shown in Fig. 5, allowed for the design framework to preserve many of the layout techniques previously employed in interleaved 65nm and 40nm designs. All custom blocks – DC-DC converters, clock generator, and custom low-voltage memories – have been characterized for use in an automated flow. The complete system has been synthesized and automatically placed and routed using the automated flow, where the run-time from an RTL change to the LVS-clean layout was less than 14 hours.

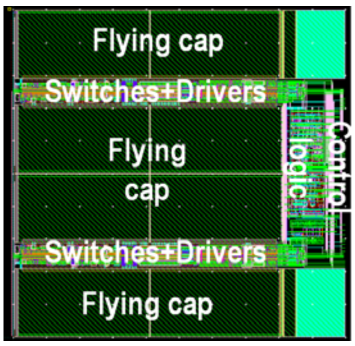

Figure 5: Sample layout of the DC-DC converter unit cell.

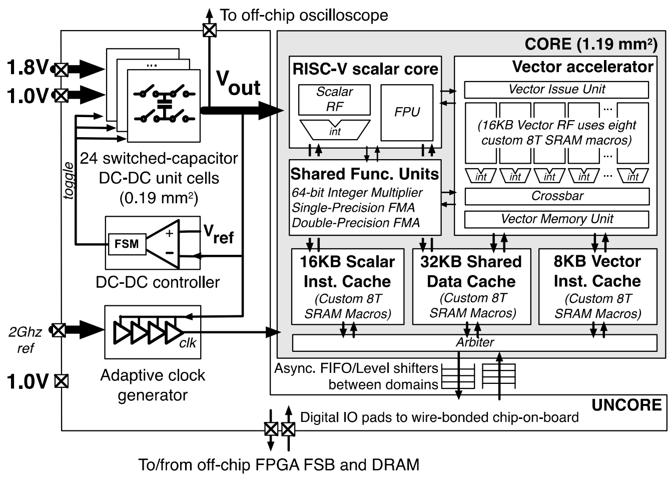
To off-chip oscilloscope
1.8V
1.0V
Vout
24 switched-capacitor DC-DC unit cells (0.19 mm²)
toggle
FSM
Vref
DC-DC controller
2Ghz ref
clk
Adaptive clock generator
1.0V
CORE (1.19 mm²)
RISC-V scalar core
Scalar RF
int
FPU
Vector accelerator
Vector Issue Unit
(16KB Vector RF uses eight custom 8T SRAM macros)
int
Crossbar
Vector Memory Unit
Shared Func. Units
64-bit Integer Multiplier
Single-Precision FMA
Double-Precision FMA
16KB Scalar Inst. Cache (Custom 8T SRAM Macros)
32KB Shared Data Cache (Custom 8T SRAM Macros)
8KB Vector Inst. Cache (Custom 8T SRAM Macros)
Arbiter
Async. FIFO/Level shifters between domains
UNCORE
Digital IO pads to wire-bonded chip-on-board
To/from off-chip FPGA FSB and DRAM

Figure 6: Block diagram of the processor with integrated DC-DC converters and adaptive clocking [13].

Block diagram of the chip is shown in Fig. 6 and its die photograph is in Fig. 7. Twenty-four DC-DC cells are integrated, and can be selectively turned off. Measured efficiency of the DC-DC converters themselves ranges from 80% to 95%, justifying the use of single-phase switching. The chip runs Linux and achieves the energy efficiency of 34GFLOPS/W when the DC-DC converter is bypassed, or 26GFLOPS/W with DC-DC and clocking losses taken into account [13].

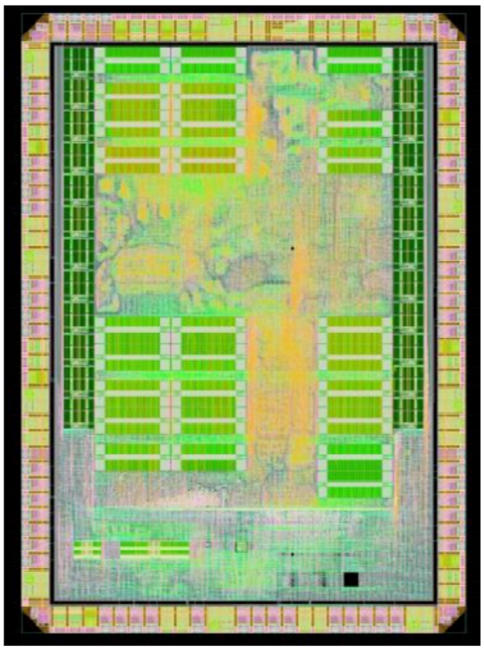
Figure 7: Processor die photo.

VI. Conclusion

In the final nodes of CMOS technology scaling it is necessary to achieve maximum performance with limited power out of billions of available transistors. The needed energy efficiency comes from tailoring functions integrated on chip to the desired application, but this needs to be accomplished with a reasonable design effort. The design these complex and efficient chips can be enabled by using generators and opening up the source code of commoditized components. Many of the principles used in developing complex software systems can be used to enable more efficient chip design.

A design example of a highly efficient microprocessor with an integrated supply regulators highlights that a very efficient design can be accomplished by a small team.

Acknowledgments

This work was in part supported by DARPA PERFECT Award Number HR0011-12-2-0016, SRC/TxACE 1836.136, and Intel. The chip fabrication was donated by STMicroelectronics. The author acknowledges BWRC, and ASPIRE students, faculty and staff, and their respective industrial member companies. Many of author's colleagues and current and former students have contributed to the ideas presented in this paper, including, Krste Asanovic, Jonathan Bachrach, Elad Alon, Robert Brodersen, Dejan Markovic, Vladimir Stojanovic, Brian Richards, Brian Zimmer, Yunsup Lee, Zheng Guo, Ruzica Jevtic.

References

[1] G.E. Moore, "No exponential is forever: but "Forever" can be delayed!" *IEEE International Solid-State Circuits Conference, Digest of Technical Papers. ISSCC'2003,* San Francisco, CA, Feb. 2003, pp.20-23.

[2] W. R. Davis, et al, "A design environment for high throughput, low power dedicated signal processing systems," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 3, pp. 420-431, March 2002.

[3] M. Horowitz, "Computing's energy problem (and what we can do about it)," *IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, Feb. 2014, pp. 10-12.

[4] M. Puschel, *et al*, "SPIRAL: Code Generation for DSP Transforms," *Proc. IEEE* , vol.93, no.2, pp.232,275, Feb. 2005.

[5] J. Bachrach, *et al*, "Chisel: Constructing hardware in a Scala embedded language," *49th ACM/EDAC/IEEE Design Automation Conference (DAC), June 2012*, pp.1212-1221.

[6] O. Shacham, *et al*, "Rethinking Digital Design: Why Design Must Change," *IEEE Micro,* vol.30, no.6, pp.9-24, Nov.-Dec. 2010.

[7] R. Rutenbar, "Analog circuit and layout synthesis revisited," *IEEE International Symposium on Physical Design*, Mar. 29-Apr. 1, 2015.

[8] J. Crossley, *et al*, "BAG: A designer-oriented integrated framework for the development of AMS circuit generators," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD),* Nov. 2013, pp.74-81.

[9] K. Zhang, ed. *Embedded Memories for Nano-Scale VLSI*, Springer, 2009.

[10] S. Sutardja, "The future of IC design innovation," *2015 IEEE International Solid- State Circuits Conference, Digest of Technical Papers,* Feb. 2015.

[11] www.riscv.org

[12] Y. Lee, "A 45nm 1.3GHz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators," *40th European Solid State Circuits Conference, ESSCIRC 2014,* Sept. 2014, pp.199-202.

[13] B. Zimmer, *et al*, "A RISC-V vector processor with tightly-integrated switched-capacitor DC-DC converters in 28nm FDSOI," 2015 Symposium on VLSI Circuits, June 15-19, 2015.

[14] Z. Toprak-Deniz, *et al,* "Distributed system of digitally controlled microregulators enabling per-core DVFS for the POWER8TM microprocessor," *IEEE Int'l Solid-State Circuits Conference Digest of Technical Papers, Feb. 2014,* pp.98-99.

[15] N. Kurd, *et al*, "Haswell: A family of IA 22nm processors," *IEEE Int'l Solid-State Circuits Conference Digest of Technical Papers, Feb. 2014,* pp.112-113.

[16] S. Clerc, *et al*, "8.4 A 0.33V/-40°C process/temperature closed-loop compensation SoC embedding all-digital clock multiplier and DC-DC converter exploiting FDSOI 28nm back-gate biasing," *IEEE Int'l Solid-State Circuits Conference Digest of Technical Papers, Feb. 2015,*

[17] H.-P. Le, *et al.*, "Design techniques for fully integrated switched-capacitor DC-DC converters," *IEEE Journal of Solid-State Circuits*, vol.46, no.9, pp.2120-2131, 2011.

[18] R. Jevtić, *et al*, "Per-core DVFS with switched-capacitor converters for energy efficiency in manycore processors," *IEEE Transactions on Very-Large Scale Integration (VLSI) Design*, vol. 23, no, 4, pp. 723-730,April 2015.

[19] J. Tschanz, *et al.*, "Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging," *IEEE Int'l Solid-State Circuits Conference*, *Digest of Technical Papers,* 2007, pp. 292-293.

[20] N. Planes *et al*, "28nm FDSOI technology platform for high-speed low-voltage digital applications," *2012 Symposium on VLSI Technology,* June 2012, pp.133-134.

[21] P. Flatresse, *et al.*, "Ultra-wide body-bias range LDPC decoder in 28nm UTBB FDSOI technology," *IEEE Int'l Solid-State Circuits Conference*, *Digest of Technical Papers,* 2013, pp.424-425.