Minimizing the Energy Usage of Tiny RISC-V Cores

Asbjørn Djupdal, Magnus Själander, Magnus Jahre, and Snorre Aunet
asbjorn.djupdal@ntnu.no,magnus.sjalander@ntnu.no,magnus.jahre@ntnu.no,snorre.aunet@ntnu.no
Norwegian University of Science and Technology
Trondheim, Norway

ABSTRACT

There is an emerging class of applications for which minimizing energy usage is significantly more important than achieving high performance, and the energy consumption of a CPU core fundamentally depends on the supply voltage and the switched capacitance of the core. Two key approaches for minimizing energy usage are hence (i) to reduce the supply voltage into the near or subthreshold region, and (ii) to minimize processor core area to reduce leakage currents and total switched capacitance.

In this paper, we combine our subthreshold cell library with the PicoRV32 and SERV tiny RISC-V cores and evaluate the resulting energy per instruction across supply voltages from subthreshold 250mV to above-threshold 600mV. Reducing the supply voltage to 300mV decreases power consumption by three orders of magnitude and energy consumption by one order of magnitude compared to the nominal 1.2V supply voltage for both cores. While bit-serial SERV yields 54% lower power consumption than bit-parallel PicoRV32 because it is smaller, PicoRV32 yields 80% lower energy per instruction because its higher performance makes up for its higher power consumption compared to SERV.

1 INTRODUCTION

There is a niche for ultra low-power CPU cores where the energy per instruction (EPI) is more important than performance. Applications include, for example, biomedical devices, RFID, and self-powered IoT devices where data processing needs are very modest, and where the amount of available energy for computation is very limited. This can be because they run on battery and need extremely long battery life, or that they use some form of energy harvesting mechanism that has a very low power or intermittent output.

One approach for creating low-power CPU cores is to lower the supply voltage of the core close to, or below, the threshold voltage of the transistors. It is well known that reducing the supply voltage is the most direct and dramatic means of reducing power consumption. This is because dynamic power consumption is proportional to supply voltage squared. However, reduced voltage also reduces the speed at which gates can switch. Lower voltage implies lower currents and it, therefore, takes longer time to charge and discharge the same capacitance [10]. This is precisely the relationship leveraged by conventional dynamic voltage and frequency scaling (DVFS) [8, 14]. The voltages exploited by conventional DVFS are much higher than the absolute values of the inherent threshold voltages to maintain reasonable performance and reliability. Power consumption, however, continues to decline when reducing supply voltage, resulting in near- or subthreshold CPU cores yielding ultra-low power consumption while reducing maximum switching speeds / performance.

Designing cell libraries that enable the use of subthreshold operation throughout larger parts of a chip is challenging, mainly due to vulnerabilities to process-, voltage-, and temperature variations. The principles were exploited by Swiss watchmakers in the 70s [17], but it was not until about the last decade that a company like Ambiq Micro was able to provide microcontrollers exploiting subthreshold operation for larger parts of their chips [9].

A complimentary approach to reduce the power consumption of the core is to choose a smaller and simpler CPU architecture that reduces the total core area, i.e., the total number of transistors. By reducing the transistor count, both the leakage currents, switching currents, and switched capacitance will be reduced, resulting in circuits that consume less power. This, however, also leads to reduced performance because most or all of the performance enhancing hardware modules have to be left out to keep the transistor count down.

In this paper, we implement and evaluate both of these approaches. We consider two simple CPU cores, both designed with the goal of minimizing core size: 1) *PicoRV32* [21], which is a conventional but minimalistic 32-bit RISC-V core, and 2) *SERV* [11], which is a tiny bit-serial RISC-V core. They are compared to each other with respect to power and EPI. Both cores are evaluated at different supply voltages, ranging from 250mV (below the absolute values of the transistor threshold voltages of about 350mV) to 600mV (well within superthreshold operation). In addition, they are compared to cores running at the nominal supply voltage of 1.2V.

We find that the energy optimal supply voltage for the chosen process is close to 300mV. When running the PicoRV32 core at 300mV, the power decreases by three orders of magnitude and the energy usage decreases by one order of magnitude when compared to running the same core on the nominal supply voltage. At all supply voltages, the bit-serial SERV core is more power efficient than the larger PicoRV32 core, but turns out to use more energy per instruction due to its slow execution.

2 ULTRA LOW VOLTAGE CIRCUITS

Dynamic power is proportional to the supply voltage V_{DD} squared. Lowering the supply voltage may reduce power consumption by several orders of magnitude. As mentioned in the introduction, this is commonly done in modern CPUs with DVFS, when the voltage is normally not lowered below the absolute value of the threshold voltages of the transistors. The threshold voltage V_{Th} is the minimum voltage needed between the gate and source terminals of a transistor for an NMOS transistor to start conducting significantly between source and drain, the transistor is commonly said to be turning on. A supply voltage below $|V_{Thp}, VThn|$, means that the transistors will never turn fully on, but operate in the subthreshold region. However, the transistors never turn completely off. Even in subthreshold there is a current flowing between source and drain. These currents can also be used for signalling, which is exactly what is exploited in subthreshold circuits.

There are some properties of the subthreshold currents that make designing subthreshold circuits more complex than conventional circuits. These currents are exponentially dependent on V_{Th}, unlike the currents flowing when $V_{GS} > V_{Th}$. Several exponential dependencies make the circuits prone to performance deviation or even malfunction due to variations in V_{Th} [3], temperature et cetera. Chip foundries are not able to control V_{Th} accurately, and V_{Th} can therefore vary significantly, both within a single chip (mismatch) and between chips (process variations). V_{Th} is also affected by temperature and a noisy supply voltage, and makes circuits perform very differently as the temperature or supply voltage changes. These V_{Th} variations can lead to, e.g., hold-time violations, and any subthreshold circuit must be designed to handle effects from process, voltage and temperature variations.

In addition, noise margins are worse in subthreshold [3]. It gets more difficult to achieve a gate output close to either V_{DD} or V_{SS}. The transistor can be regarded as a voltage controlled resistor, and in subthreshold the difference in resistance between a (somewhat) conducting state and a (mostly) non-conducting state is not very high. Combined with high loads on gate outputs, the output voltages may get degraded or take long to stabilize. This makes it important to keep a low number of transistors in series between a gate output and either V_{DD} or V_{SS} [7].

3 CUSTOM SUB-THRESHOLD LIBRARY

Vendor supplied standard cell libraries are commonly only characterized for nominal supply voltages. This paper uses a commercially available 130nm process where the nominal supply voltage is 1.2V and the absolute threshold voltage is around 350mV. Exact behavior of the cells at voltage levels near or below the threshold can not be known without characterizing them. Characterization requires access to the detailed layouts of the cells, and this is unfortunately not available to normal customers of the chosen 130nm process. It is therefore not possible for us to characterize these cells ourselves. This means that the synthesis tools cannot optimize the circuits for any other voltage than the nominal supply voltage. In addition, the standard cell library is not designed for ultra low voltage operation. As explained in Section 2, there are different design choices to be made when the supply voltage is very low, so it is expected that a cell library specifically targeting these voltages could be advantageous. Many cells in standard libraries will not work well, or at all, at lower voltages. When using standard cell libraries for subthreshold operations, often a selected subset is chosen, which can handle low voltages reasonably well, though not in an optimal fashion.

The dimensions of the transistors are highly dependent on the supply voltage the library is intended for, and are chosen to provide good tradeoffs with respect to speed, power etc. This also means that transistors in subthreshold cell libraries need different dimensions than for typical standard libraries. By sizing the transistors for balanced cells around the threshold voltage, it is expected that the noise margins and switching speeds will improve [7].

A custom cell library was, therefore, developed for the chosen 130nm process. This library consists of a small but sufficient number of standard cells. A summary of the cells is found in Table 1. The layout is conventional, but with transistors sized for low voltage operation. In addition, only cell types with low transistor stacks are included. It is important to keep the height of the transistor stacks in the cells low, which means few inputs to the cells. For logic gates, the number of inputs are often restricted to a maximum of 2 [13], which is also the case for this paper.

The library has been characterized and found working in simulations for all supply voltages between 250mV and 600mV, in 50mV increments.

4 THE CORES

Two cores are compared in this paper: PicoRV32 and SERV. Both are RISC-V cores, and both are designed to be tiny in area.

PicoRV32 [21] is a multi-cycle CPU with a 32-bit datapath. A multi-cycle architecture differs from a pipelined architecture in that instructions do not overlap in time. For example, to execute an `add` instruction, the core fetches and decodes the instruction in the first cycle, fetches operands in cycles 2 and 3 and executes the instruction and writes back the result in the fourth cycle; the core continues with fetching and decoding the next instruction in the fifth cycle. In PicoRV32, instructions execute in 3 to 15 cycles, where most instructions require 4 cycles as in the example above. This means that without memory stalls, the CPI (cycles per instruction) is around 4, depending on the instruction mix in the program.

The SERV [11] core is also a multi-cycle CPU, but with a bit-serial datapath. This architecture operates on one bit at a time, resulting in a lot more cycles needed for executing an instruction. Instructions execute in 35 to 76 cycles, where most instructions require 35. As for PicoRV32, there is no pipelining, so the CPI is somewhere between 35 and 76 depending on the instruction mix. The upside to this is a modest area footprint. In our physical implementation, SERV is only 57% as large as the size of PicoRV32.

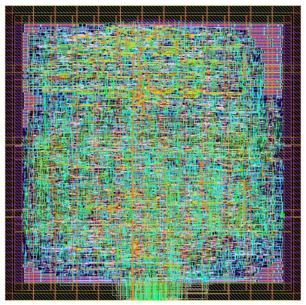
To enable a fair comparison between the PicoRV32 core and the SERV core, we have configured them with similar feature sets. More specifically, we configured both cores to support the RV32E ISA [20], which is identical to the more common RV32I ISA, except for having half the number of registers. They also include only the minimum set of hardware modules, i.e., all optional hardware has been excluded.

In these tiny cores, the register file dominates both area and power consumption, and it is therefore important to implement the register file efficiently. The easiest approach is to use D flip-flops, but this is very inefficient as each D flip-flop consists of two latches. For this reason, most ASIC implementations use either SRAM or latches for the register file. As indicated in [1], latch-based register files can be more area efficient than SRAM-based register files when the register file is small (1Kib or less). Since the RV32E register file is 512 bits, we implemented latch-based register files in both the PicoRV32 and SERV cores.

The interfaces of the cores have been modified to be the same on both cores, making each core fully compatible with each other. They can, therefore, be simulated with the same testbench.

Table 1: Custom cell library

Name	Size variants	Description
IV	X1, X2, X5, X10, X20, X40, X80, X160, X320	Inverter
NAND2	X1, X2, X5, X10	2-input NAND
NOR2	X1, X5, X10	2-input NOR
XOR		2-input XOR
XNOR		2-input XNOR
AOI		2-input And-Or-Invert
OAI		2-input Or-And-Invert
BF	X2, X5, X10, X20, X40, X80, X160, X320	Buffer
DLY	X4, X8	Delay
LDHQ		Latch
DFPQ		D-flipflop
DFPRQ		D-flipflop with async reset
DFPSQ		D-flipflop with async set
FA		Full adder
MUXI21		2-input multiplexer


(a) PicoRV32

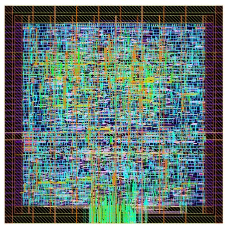

(b) SERV

Figure 1: Layout of the cores. Same scale for both cores.

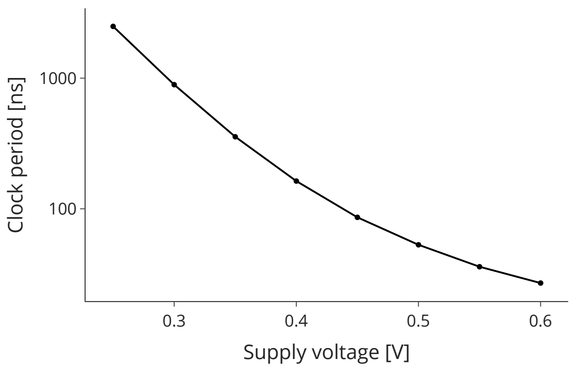


Figure 2: Clock Period. Log-scale

5 EXPERIMENTAL SETUP

The cores from Section 4 have gone through synthesis (Cadence Genus [4]) and the full physical design flow (Cadence Innovus [5]) using the custom cell library from Section 3 at 250mV. The cores are given a square floorplan with an area resulting in around 80% utilization after routing. All I/O pins are placed at the bottom edge. The finished layouts of the cores are shown in Figure 1 and the synthesis and implementation statistics are shown in Table 2.

To get accurate power estimates, the circuits are simulated in SPICE. These simulations need transistor-level netlists, which are created from the cell-level netlists produced after physical design. These transistor-level netlists can be created both with and without parasitic extraction from the layout. Simulations with extracted parasitics are more accurate but increases simulation time. In addition, access to the full layout is required to extract the parasitics. The layout is available when using the custom library and transistor-level netlists are, therefore, created both with and without parasitic extraction.

To be able to compare the performance of the custom cell library with the standard cell library, the cell-level netlist for PicoRV32 is transformed into a new cell-level netlist by substituting all custom cells with the equivalent cells from the standard library. This is then used to create the transistor-level netlist. Since we do not have access to the full layout of the standard library, we cannot perform parasitic extraction.

The transistor-level netlists are SPICE simulated using Cadence Xcelium [6] with a testbench in Verilog. The testbench contains the memory, which means that memory is excluded from all power and energy numbers. The memory has a single cycle response time. The testbench fills the memory with a small program. The chosen test program is a CRC32 calculator, and the input given is a single byte buffer containing the letter 'a'. CRC32 was chosen because it is a well known and widely used algorithm, while at the same time is simple enough to be SPICE simulated in reasonable time. The testbench resets the core, and then waits until the core runs through the program and writes the correct CRC32 result to a specific memory address. From reset to the result is written, the core executes in total 26 instructions.

	PicoRV32 (custom lib)	PicoRV32 (standard lib)	SERV (custom lib)
Area	0.23mm^2	N/A	0.13mm^2
Utilization	79.45%	N/A	80.40%
Cells (synth)	7215	Same netlist as with custom	2973
Cells (final)	10643	Same netlist as with custom	4349
Transistors	54632	62212	27008

Table 2: Core statistics after full layout

Clock periods for various supply voltages have been determined with a static timing analysis tool (Synopsys PrimeTime [16]) using the custom cell library. This was done for both the PicoRV32 core and the SERV core, and found to differ with less than 5%. The shortest clock period that works for both cores was chosen for each supply voltage. These periods are shown in Figure 2. The standard cell library has not been characterized for near-threshold voltages, so the clock frequencies found with the custom library are also used when simulating with the standard cell library.

6 RESULTS

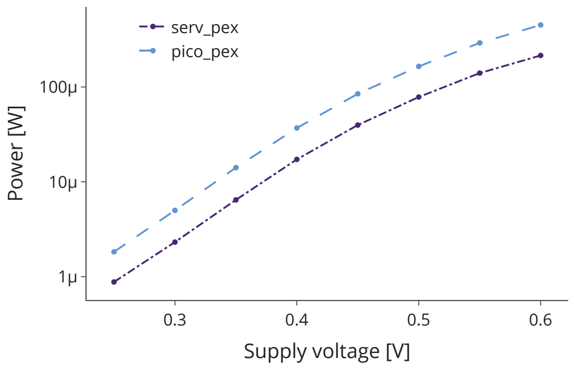
We now present our evaluation of the PicoRV32 and SERV cores in which we consider the following configurations:

- **pico**: PicoRV32, custom library, no parasitic
- **pico_pex**: PicoRV32, custom library, with parasitic
- **pico_std**: PicoRV32, standard library, no parasitic
- **serv**: SERV, custom library, no parasitic
- **serv_pex**: SERV, custom library, with parasitic

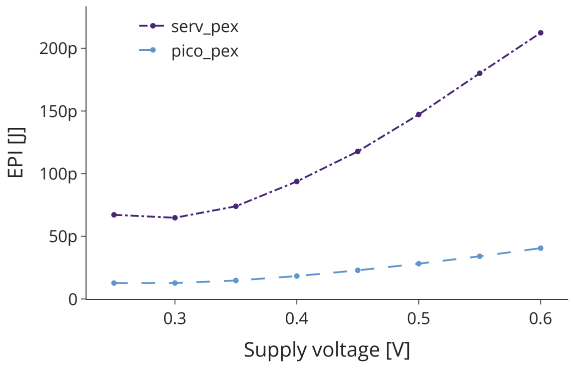
6.1 Energy Optimal Supply Voltage

Figure 3a shows how the average power varies with supply voltage on the two cores. As is expected, the power decreases as the supply voltage decreases. The lowest power is found with the lowest possible supply voltage. However, as Figure 2 shows, the runtime of the RISC-V program running on the core increases as the supply voltage decreases. Energy is average power multiplied with runtime, and for many applications low energy usage is of higher importance than power.

Figure 3b shows how the energy per instruction (EPI) varies with supply voltage on the cores. It is evident from the figure that the energy minimum for our custom library lies somewhere close to 300mV. In other words, the most energy efficient supply voltage for our custom library is 300mV. This is the same for both cores. This is in line with earlier findings by other researchers [12, 19]; the energy minimum typically lies below the absolute values of the inherent threshold voltages.



(a) Average power. Log-scale



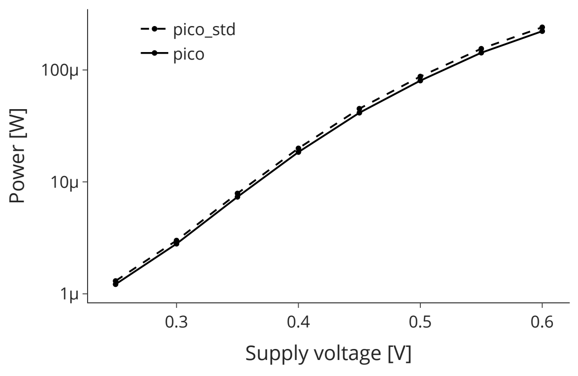
(b) Energy Per Instruction (EPI)

Figure 3: Comparing cores

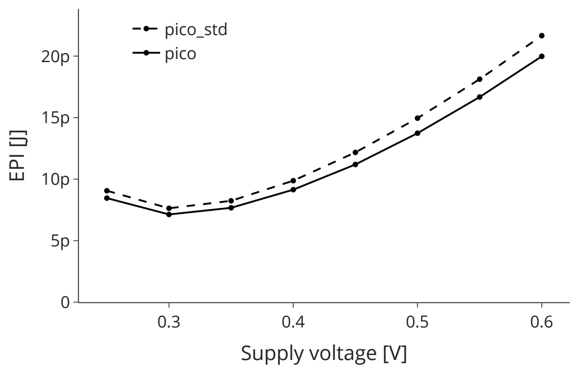
6.2 Bit-Serial vs. Bit-Parallel Core Architecture

When comparing the PicoRV32 core to the SERV core, it is clear from Figure 3a that the SERV core is more power efficient than the PicoRV32 core. With the custom library, PicoRV32 has 2.2 times the power consumption of SERV at 300mV. This is mostly due to the lower area of the SERV core. The size differences between the cores are visualized in Figure 1, but is also evident from Table 2 that gives an overview of the transistor counts of the cores. The size difference more or less coincides with the difference in power.

The SERV core is, however, significantly less energy efficient than the PicoRV32 core, which can be seen in Figure 3b. The EPI of SERV is 5.1 times the EPI of PicoRV32 at 300mV. This shows that the high CPI of the bit-serial SERV core makes the total energy usage of the SERV core much higher than for the more conventional designed PicoRV32 core, even though the average power is lower. The transistors saved by going to a bit-serial architecture do not weigh up for the increase in runtime.



(a) Average power. Log-scale



(b) Energy Per Instruction (EPI)

Figure 4: Comparing libraries

6.3 Custom Library vs Standard Library

Our custom library has been compared to the standard library in Figure 4. Both power and energy numbers are smaller for the PicoRV32 core using the custom library, mostly due to a larger transistor count in the standard library. The difference is, however, not very large and indicates that the custom library is a reasonable substitute for the standard library, with the added benefit of enabling parasitic extraction and subthreshold characterization.

In addition, transistor sizing for subthreshold might have made the custom library more robust and better performing at subthreshold, but this can not be concluded without further investigations.

6.4 Subthreshold vs. Nominal Supply Voltage

As discussed in Section 6.1, the most energy optimal supply voltage for our custom library is 300mV. Table 3 shows how a 300mV core compares to a core running at 1.2V. As can be seen, the power decreases by three orders of magnitude when going from 1.2V to 300mV , which is similar to results in [2]. Frequency decreases by two orders of magnitude, and the result is that EPI decreases by one order of magnitude, similar to what has been observed by other researchers [2, 13].

	pico_std @ 300mV	pico_std @ 1.2V
Power	2.99μW	5147.05μW
Runtime	66.26μs	0.47μs
Energy	198.38pJ	2409.17pJ
EPI	7.63pJ	92.65pJ
Frequency	1.77MHz	250.00MHz

Table 3: Subthreshold vs nominal V_{dd}

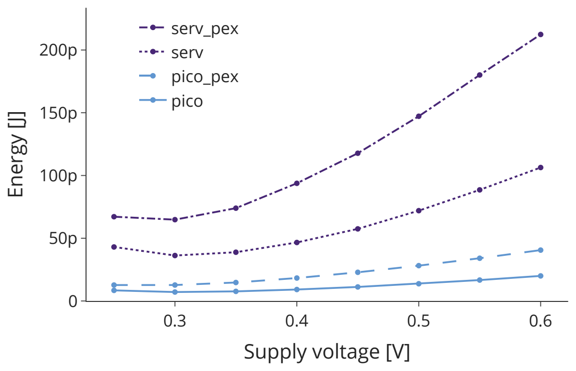


Figure 5: EPI, comparing simulations with and without extracted parasitics

6.5 Simulations with Extracted Parasitics

The simulations in Section 6.3 and Section 6.4 are done without parasitic extraction. As mentioned in section 5, this leads to less accurate results than when simulating a netlist where parasitics from the layout are included. Since this is not possible with the standard library, it was decided to exclude parasitic extraction for all simulations in Section 6.3 and Section 6.4, such that the comparisons are fair.

Figure 5 shows the effect of excluding parasitics. Both cores (using the custom library) have been simulated, both with and without extracted parasitics. As can be seen, the EPI increases significantly (78% higher for both cores at 300mV) when simulated with parasitics, but the trends are the same. This indicates that the conclusions based on simulations without extracted parasitics are correct, even though the absolute numbers are not.

7 RELATED WORK

Several CPU cores have been implemented for subthreshold operation. One example is found in [13] where they designed and produced a 65nm chip containing two 32-bit RISC cores with SRAM caches, running on a supply voltage as low as 200mV. Another example is [15] where a subthreshold ARM Cortex M0 was designed and built in a 65nm process, complete with chip prototype. They also found that some samples were operational down to 200mV. [18] implemented a near-threshold RISC-V core in 28nm and successfully ran a prototype chip on 250mV.

What separates the work in this paper from other published papers on subthreshold CPU cores is that two subthreshold RISC-V cores that are both tiny in size, but with widely different architectures, are compared with respect to EPI, giving insight into architectural choices on low-energy cores.

8 CONCLUSION

This paper has investigated the energy efficiency of two cores running at ultra-low voltages: PicoRV32 and SERV. PicoRV32 was found to be the most energy efficient of the two cores, with an EPI of 12.73pJ compared to an EPI of 64.81pJ for SERV when running at the energy optimal supply voltage of 300mV.

ACKNOWLEDGEMENTS

The work has been funded in part by the European Union's Horizon 2020 research and innovation program through the BOWI project (grant agreement number 873155).

REFERENCES

[1] Oskar Andersson, Babak Mohammadi, Pascal Meinerzhagen, Andreas Burg, and Joachim Neves Rodrigues. 2016. Ultra Low Voltage Synthesizable Memories: A Trade-Off Discussion in 65 nm CMOS. *IEEE Transactions on Circuits and Systems I: Regular Papers* 63, 6 (2016), 806–817. https://doi.org/10.1109/TCSI.2016.2537931

[2] V. Beiu, S. Aunet, J. Nyathi, R.R. Rydberg, and A. Djupdal. 2005. On the advantages of serial architectures for low-power reliable computations. In *2005 IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP'05)*. 276–281. https://doi.org/10.1109/ASAP.2005.48

[3] David Bol et al. 2011. Robust and energy-efficient ultra-low-voltage circuit design under timing constraints in 65/45 nm CMOS. *J. Low Power Electron. Appl* 1, 1 (2011), 1–19.

[4] Cadence. 2023. Genus web page. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html.

[5] Cadence. 2023. Innovus web page. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html.

[6] Cadence. 2023. Xcelium web page. https://www.cadence.com/en_US/home/tools/system-design-and-verification/simulation-and-testbench-verification/xcelium-simulator.html.

[7] Benton H Calhoun, Alice Wang, and Anantha Chandrakasan. 2005. Modeling and sizing for minimum energy operation in subthreshold circuits. *IEEE Journal of Solid-State Circuits* 40, 9 (2005), 1778–1786.

[8] S. Eyerman and L. Eeckhout. 2010. A counter architecture for online DVFS profitability estimation. *IEEE Trans. Comput.* 59, 11 (2010), 1576–1583.

[9] Scott Hanson. 2011. Ultra-low power microcontrollers for compact wireless devices. https://www.eetimes.com/ultra-low-power-microcontrollers-for-compact-wireless-devices/

[10] Mark Horowitz, Thomas Indermaur, and Ricardo Gonzalez. 1994. Low-power digital design. In *Symposium on Low Power Electronics*. 8–11.

[11] Olof Kindgren. 2023. SERV Git Repository. https://github.com/olofk/serv.

[12] Dake Liu and Christer Svensson. 1993. Trading speed for low power by choice of supply and threshold voltages. *IEEE journal of solid-state circuits* 28, 1 (1993), 10–17.

[13] Sven Lutkemeier, Thorsten Jungeblut, Hans Kristian Otnes Berge, Snorre Aunet, Mario Porrmann, and Ulrich Ruckert. 2013. A 65 nm 32 b Subthreshold Processor With 9T Multi-Vt SRAM and Adaptive Supply Voltage Control. *IEEE Journal of Solid-State Circuits* 48, 1 (2013), 8–19. https://doi.org/10.1109/JSSC.2012.2220671

[14] Rustam Miftakhutdinov, Eiman Ebrahimi, and Yale N. Patt. 2012. Predicting performance impact of DVFS for realistic memory systems. In *Proceedings of the International Symposium on Microarchitecture*. 155–165.

[15] James Myers, Anand Savanth, Rohan Gaddh, David Howard, Pranay Prabhat, and David Flynn. 2016. A Subthreshold ARM Cortex-M0+ Subsystem in 65 nm CMOS for WSN Applications with 14 Power Domains, 10T SRAM, and Integrated Voltage Regulator. *IEEE Journal of Solid-State Circuits* 51, 1 (2016), 31–44. https://doi.org/10.1109/JSSC.2015.2477046

[16] Synopsys. 2023. PrimeTime web page. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html.

[17] Yannis Tsividis. 2008. Eric Vittoz and the Strong Impact of Weak Inversion Circuits. *IEEE Solid-State Circuits Society Newsletter* 13, 3 (2008), 56–58. https://doi.org/10.1109/N-SSC.2008.4785782

[18] Roel Uytterhoeven and Wim Dehaene. 2018. A sub 10 pJ/Cycle Over a 2 to 200 MHz Performance Range RISC-V Microprocessor in 28 nm FDSOI. In *ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC)*. 236–239. https://doi.org/10.1109/ESSCIRC.2018.8494259

[19] Alice Wang, Benton H Calhoun, and Anantha P Chandrakasan. 2006. *Subthreshold design for ultra low-power systems*. Vol. 95. Springer.

[20] Andrew Waterman and Krste Asanović. 2019. *The RISC-V Instruction Set Manual, Volume I: User-Level ISA*. RISC-V Foundation.

[21] Yosys. 2023. PicoRV32 Git Repository. https://github.com/YosysHQ/picorv32.