Low-Power MCUs: Power Modeling, Estimation, and Optimization

Binqi Yang

School Electrical and Electronic Engineering, Newcastle University, Newcastle, UK

c4074774@newcastle.ac.uk

Abstract. Ultra-low-power microcontroller units (MCUs) are the foundation of IoT, wearable devices, and sensor networks, where energy efficiency directly determines system lifetime and reliability. This paper reviews power modeling, estimation, and optimization techniques tailored for low-power MCUs. Power consumption is analyzed across dynamic, short-circuit, and leakage components, with the fundamental relation of dynamic power expressed in (1). Estimation approaches at multiple abstraction levels—transistor-level, gate-level, RTL, and system-level—are compared in terms of accuracy, turnaround, and applicability. Gate-level power analysis, supported by SAIF-based switching activity, remains the reference standard, while RTL and system-level estimation enable rapid design-space exploration. Figure 1 illustrates the classical MCU pipeline structure, highlighting how switching activity from CPU, peripherals, and memory contributes to overall power. Optimization methods such as clock gating, power gating, dynamic voltage and frequency scaling (DVFS), near-threshold computing, approximate computing, and energy-aware logic synthesis are evaluated with their respective trade-offs in performance, energy savings, and design complexity. Recent trends emphasize machine learning–assisted estimation, automated power-intent synthesis, and hardware–software co-optimization. This comprehensive survey concludes that achieving decade-long MCU lifetimes requires integrating accurate modeling, fast estimation, and adaptive optimization strategies for real-world workloads.

Keywords: MCUs, Power estimation, Optimization, DVFS.

1. Introduction

Ultra-low power microcontroller units (MCUs) are at the core of the Internet of Things (IoT), wearable devices, and multi-hop sensor networks. Unlike high-performance CPUs, MCUs are designed to provide reliable sensing, control, and lightweight computing at power consumption ranging from micro-watts to a few milliwatts, typically powered by coin cells or energy harvesting. Battery replacement costs, product heat dissipation, and reliability in remote areas make power consumption the primary design goal. In practical deployments - environmental monitoring, industrial condition-based maintenance, medical patches - energy consumption per task and deep sleep leakage often determine the device's lifespan more than peak millions of instructions per second (MIPS).

Research areas indicate that power consumption is a cross-layer challenge. At the device level, leakage increases with technology scaling and temperature rise; at the circuit level, short circuits and glitches complicate dynamic power consumption; at the logic/register transfer level (RTL) and architecture level, clocks, memory/bus traffic, and peripherals determine switching activity; at the software level, workloads and scheduling determine duty cycles; at the system level, voltage/frequency selection and coordination of sleep states determine energy consumption over time. Accurate power estimation is challenging: transistor-level modeling is accurate but time-consuming; gate-level analysis relies on switching activity capture (VCD/SAIF) and glitch modeling; RTL power estimation relies on macro models and activity propagation. Each step involves a trade-off between accuracy and turnaround time, which directly affects the speed at which design teams can run the "estimate-optimize-verify" cycle.

In optimization, classic clock gating and power gating remain the mainstays of MCUs, while dynamic voltage and frequency scaling (DVFS), near-threshold voltage computing (NTV/NTC), approximate computing, and energy-aware logic synthesis have expanded the frontier, but they come

with different costs in terms of timing closure, verification, EDA process complexity, and software co-design. Since MCUs need to meet diverse real-time and reliability constraints (e.g., deterministic peripherals, SRAM retention, startup delay, security), a one-size-fits-all approach rarely works.

This paper focuses specifically on low-power MCUs. We (i) introduce basic terms and power consumption decomposition; (ii) review power estimation methods - RTL power estimation, gate-level power analysis, transistor-level power modeling, dynamic power estimation, leakage power modeling, switching activity estimation, and SAIF-based analysis - along with power modeling methods; (iii) investigate optimization strategies for microcontrollers - clock gating, power gating, dynamic voltage and frequency scaling, near-threshold voltage computing, approximate computing, and energy-aware logic synthesis - summarizing their principles, advantages and disadvantages, and application scenarios; (iv) conduct comparisons and trend analyses; (v) conclude with future directions.

2. BACKGROUND AND TERMINOLOGY

2.1.Logic computation pipeline in an MCU

A classical MCU pipeline (fetch–decode–execute–memory–writeback) sits alongside clock tree, power domains, SRAM/Flash, DMA, and low-power peripherals. Switching activity from CPU and peripherals drives dynamic power; static/leakage dominates in deep sleep, as shown in Fig.1.

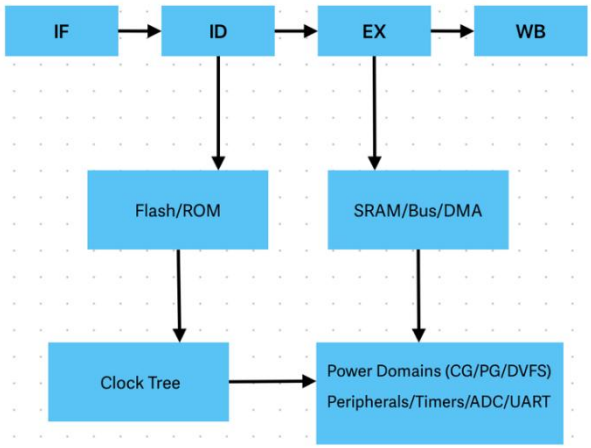

Figure 1. CPU Pipeline

2.2.Maintaining the Integrity of the Specifications

Dynamic power: charging/discharging load capacitances and glitching. A first-order model is

$$Pdyn = a \times C \times V^2 \times f \quad (1)$$

Where α is the activity factor (toggle probability per cycle), C the effective capacitance, V the supply, and f the clock. Accurate dynamic power estimation requires realistic switching activity estimation that captures correlation and hazards (glitches).

Short-circuit power: during input transitions, both pull-up and pull-down paths conduct simultaneously; increases with input slew and depends on gate sizes and VDD.

Static (leakage) power: subthreshold, gate-oxide tunneling, and junction leakage; strongly dependent on process, temperature, and Vth; dominates long standby in MCUs.

2.3. Common metrics

Power (average/peak, mW), energy (J), energy per operation (J/op) or per inference (J/inf) for edge AI, and composite metrics such as EDP or ED^2P to capture time-energy trade-offs.

Energy efficiency in MCUs is often reported as nJ per peripheral transaction, nJ/sample (ADC path), or µJ/packet (radio path), in addition to core compute nJ/op.

Activity files: VCD (Value Change Dump) and SAIF (Switching Activity Interchange Format) are used to record toggles; SAIF is compact and widely used to drive gate-level power analysis tools after back-annotation.

3. Existing Methods

This Chapter organizes the literature by estimation level—transistor, gate, RTL, and system/architectural; and optimization strategy—clock/power gating, DVFS, near-threshold, approximate computing, energy-aware logic synthesis—highlighting principles, pros/cons, and MCU-specific scenarios.

3.1.Power estiamtion methodologies

(1) Transistor level power modeling

SPICE/fast-SPICE solves device equations (e.g., BSIM) to capture waveform-accurate dynamic, short-circuit, and leakage components under PVT and realistic slews. Compact models also dissect leakage (subthreshold, gate tunneling, junction). It achieves highest fidelity, decomposes contributors, validates standard-cell libraries and SRAM macros at low-voltage corners (critical for NTV). However, it has drawbacks such as the extremely slow and input-pattern limited; impractical for full MCU SoCs or long workloads [1-4].

(2) Gate level power analysis

As shown in Fig.2, after logic synthesis and CTS, a gate-level netlist, cell library (.lib), parasitic (post-layout if available), and switching activity (VCD/SAIF) drive vector-based or vector less analysis. Glitch filtering and inertial delays are important for accuracy. It has advantages with high correlation to silicon. But requires representative workloads and long simulations; SAIF quality determines accuracy; turnaround can bottleneck iteration loops [5–7].

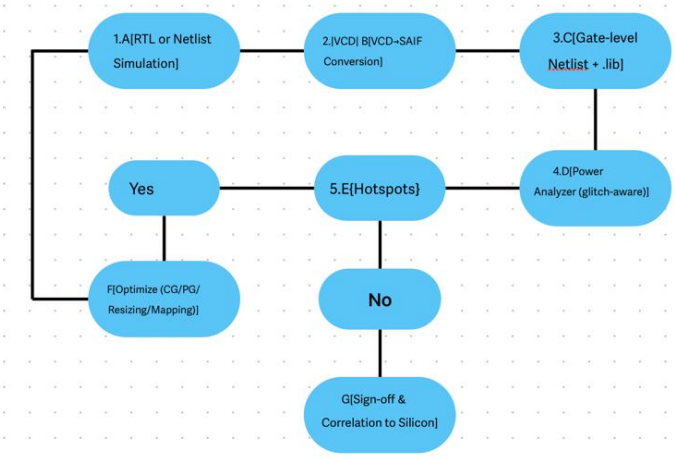

Figure 2. SAIF-driven gate-level power analysis

(3) RTL power estiamtion

Using RTL simulation (or emulation) to gather activity and multiply by macro-level power models (adders, multipliers, register files, buses, SRAM ports) calibrated from gate/transistor data. Machine-learning regressors increasingly map architectural counters to power. This methods are much faster than gate-level; supports rapid design-space exploration (e.g., bit-widths, pipeline depth, peripheral gating granularity). However, it is sensitivities to model calibration and activity realism; misses layout-induced capacitances and glitching; correlation depends on IP coverage [8, 9].

(4) System estiamtion

Analytical/empirical models at task or OS level estimate energy under DVFS, duty-cycling, and sleep-state transitions, often using performance counters or calibration experiments on boards. For the methods, it captures real workloads and mode orchestration (active, sleep, retention, peripheral-only). But with Low physical detail; requires careful calibration per SKU/process [10].

3.2.Optimization strategies for low-power MCUs

(1) Clock gating

As shown in Fig.3. Disable clock toggling for idle flops/blocks (automatic or explicit enable conditions), including gated-clock FSM synthesis for controllers. It has Large dynamic power reduction with minor area; synergistic with peripheral-centric MCU designs where duty cycles are low. For drawbacks, it need enable logic/timing constraints; verify for glitch-free gating and CDC effects; limited impact on leakage [11].

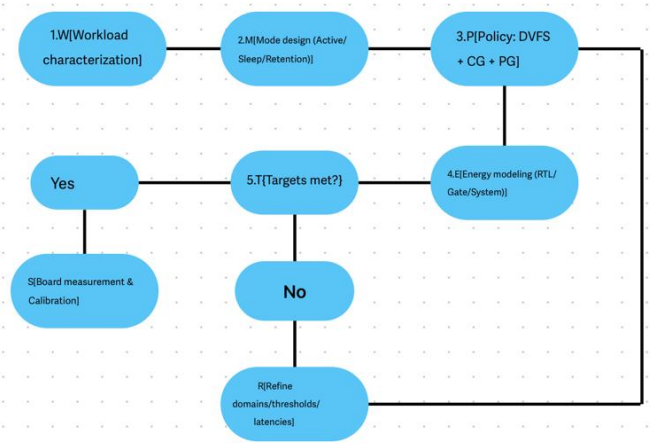

Figure 3. MCU power optimization loop across modes

(2) Power gating

Use header/footer sleep transistors, isolation, and state-retention to fully shut off blocks, eliminating leakage by 10–50× in sleep; requires staged wake-up to manage rush currents. It has pros. as Dominant technique to extend standby battery life; aligns with MCU multi-domain power managers. Also, it has Cons such as area/time overhead for isolation or retention; wake up latency and verification complexity; SRAM and I/O retention must be engineered [12].

(3) Dynamic voltage and frequency scaling (DVFS)

Adjust V/f to track workload or deadlines; energy $\propto$ V2 while delay increases as V drops. Real-time dynamic voltage and frequency scaling (DVFS) takes advantage of idle time; the implementation of microcontrollers combines DVFS with duty cycle adjustment and peripheral clock control. It has significant energy-saving advantages in computationally intensive burst tasks; it can be optimized in coordination with the scheduling of radios/analogue-to-digital converters (ADCs). However, it has drawbacks in terms of voltage conversion delay, regulator efficiency, and software complexity; at very low voltages, due to leakage dominating, the energy-saving effect gradually weakens [13].

(4) Near-threshold Voltage Computation (NTV))

Operating near the threshold voltage (Vth) to approach the "minimum energy point" typically enables an order-of-magnitude reduction in energy. This requires fault-tolerant circuits, timing elasticity, and SRAM data retention management. It performs well in terms of energy per operation; it is suitable for always-on sensing and intermittent computing. However, significant performance loss, process-temperature-voltage (PVT) sensitivity, SRAM stability, and error rates necessitate careful trade-offs between always-on and burst domains [14].

(5) Approximate computing

Trade bounded precision for lower energy consumption through reduced precision, inexact computation, cyclic perforation, or voltage overshoot with error detection. On MCUs, approximate kernels (DSP/ML) cut nJ/inf substantially. It has potentially large energy savings with minor QoS loss; orthogonal to DVFS/CG/PG. But requires application aware quality metrics and safety guards; limited to error tolerant workloads [15].

(6) Energy aware logic synthesis

Synthesis/technology mapping driven by power cost functions (switching activity × capacitance), including pin-swapping, gate sizing, multi-Vt assignment, state re-encoding, and gated-clock FSM

synthesis. The structural reductions available before floorplanning; integrates with RTL power estimation loops. But it needs accurate activity estimates and library power models; may complicate timing closure and testability [16].

4. Comparisons among the methods

Table 1 shows the performance comparison among the mentioned methods. Transistor-level analysis remains the gold standard for cell/macro validation and NTV studies, but its runtime precludes whole-SoC coverage. Gate-level is the main workhorse for sign-off, accuracy hinges on SAIF quality and glitch modeling. RTL power estimation provides the best productivity for MCU-class iterations and is increasingly accurate when macro models are calibrated from gate-level data and when switching activity estimation accounts for correlation. System-level modeling closes the loop by validating policies (DVFS, duty-cycle, sleep states) with firmware in the loop and board calibration.

Table 1. Comparison Among the Different Methods

Method (keyword)	Accuracy	Turnaround	Typical MCU Use Case
Transistor-level power modeling	★★★★★	★	Library/SRAM corners, NTV feasibility
Gate-level power analysis	★★★★☆	★★	Sign-off, clock tree & bus hotspots, CG efficacy
RTL power estimation	★★★☆☆	★★★★	Early budgeting, IP/SoC exploration, firmware choices
System/architectural energy modeling	★★☆☆☆	★★★★★	DVFS/duty-cycling policies, sleep orchestration
Clock gating (low-power MCU)	—	—	Dynamic power cut on idle logic/peripherals
Power gating in microcontrollers	—	—	Leakage cut in deep sleep, state-retentive domains
DVFS (MCU)	—	—	Energy vs. deadline, bursty workloads
Near-threshold voltage computing	—	—	Always-on sensing, energy-harvesting nodes
Approximate computing (MCU)	—	—	Error-tolerant DSP/ML on Cortex-M-class cores
Energy-aware logic synthesis	—	—	Control/datapath power at RTL/gate pre-layout

Optimization trade-offs. Clock gating is nearly free energy, widely automated, and synergizes with peripheral-centric workloads; power gating provides order-of-magnitude leakage cuts but adds state-retention/isolation overhead and wake-up latency—critical for GPIO/RTC wake. DVFS is powerful for compute bursts and edge AI inferences; it faces diminishing returns near the minimum-energy point and must consider regulator efficiency/latency. NTV/NTC yields the lowest energy/op but demands resiliency to variability and SRAM stability; it often pairs with domain partitioning (always-on NTV vs. nominal-V bursts). Approximate computing provides orthogonal wins where quality-of-result is elastic (e.g., HAR, keyword spotting) and is compatible with CG/PG/DVFS. Energy-aware logic synthesis, which can obtain structural advantages as early as possible, should be retained in the inner loop of the design along with RTL power estimation and SAIF-based re-calibration.

5. Trends and Challenges

The latest trends in low-power design and analysis are clearly moving towards higher levels of abstraction, automation, and hardware-software co-optimization. Firstly, estimation methods are gradually moving away from detailed waveform activity files. System-level and register transfer level

prediction frameworks leveraging machine learning, using architectural counters and macro models, have enabled fast and scalable power estimation while maintaining sufficient accuracy. Additionally, AI-accelerated alternative models are emerging for predicting hotspots, glitches, and workload categories, thereby reducing the need for long simulation runs and enabling rapid design space exploration.

In terms of tools, EDA automation is moving towards tighter integration of power intent specifications such as UPF/CPF. Automatic clock gating insertion with formal verification, systematic power gating domain synthesis (including isolation and retention strategies), and one-click SAIF-driven power closure and simulation are becoming increasingly practical. Meanwhile, power management for microcontrollers is moving towards domain specialization, with vendor-provided power management units (PMUs) featuring fine-grained multi-rail dynamic voltage and frequency scaling (DVFS), autonomous peripheral gating, SRAM retention islands, and voltage scaling in deep sleep modes to maintain state at the lowest energy consumption.

At the edge-AI frontier, co-design between algorithmic compression and circuit-level management is gaining traction. Techniques such as quantization-aware DVFS, near-threshold operation, and approximate MAC selection per neural network layer enable significant energy savings while balancing accuracy requirements.

Despite these advances, challenges remain. A central issue is bridging the accuracy–time tradeoff, ensuring that RTL or system-level estimates correlate to silicon results without exhaustive vectors. Verification costs also escalate with the interactions of UPF/SAIF/STA/DFT flows, power-gating wake-up sequences, and CDC behavior under clock gating. Applicability also requires that the technology meet the real-time constraints of the microcontroller, startup and wake-up delays, as well as safety/security requirements. Finally, repeatable board-level energy characteristic descriptions and on-chip monitors are crucial for continuously calibrating models and validating assumptions under various operating conditions.

6. Conclusion

Low-power microcontroller design requires a closed loop covering modeling, estimation, optimization, and verification. Accurate gate-level power analysis and high-quality SAIF data remain crucial for final confirmation, while RTL power estimation enables rapid exploration and should be calibrated from transistor/gate-level data. In terms of optimization, clock gating and power gating are fundamental; dynamic voltage and frequency scaling (DVFS) and near-threshold computing can expand the energy range when combined with well-designed strategies and retention planning; approximate computing and energy-aware logic synthesis can bring application-aware and structural benefits.

Looking ahead, the field is moving towards higher-level, machine learning-assisted estimation, automatic low-power intent (UPF/CPF) synthesis, and strategies for co-design with firmware. For microcontrollers powering the Internet of Things (IoT) and wearable devices, integrating domain-specific power managers, learned workload models, and hardware-software co-optimization will be key to achieving a lifespan of over a decade under real-world workloads.

References

[1] S. Mittal, “A survey of architectural techniques for near-threshold computing,” ACM J. Emerg. Technol. Comput. Syst., vol. 12, no. 4, Art. 46, Dec. 2015.

[2] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, “Near-threshold computing: Reclaiming Moore’s Law through energy-efficient integrated circuits,” Proc. IEEE, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[3] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, “Near-threshold computing: Overcoming performance degradation from aggressive voltage scaling,” in Proc. Workshop Energy-Efficient Design, 2009, pp. 44–49.

[4] S. Salamin, H. Amrouch, and J. Henkel, “Selecting the optimal energy point in near-threshold computing,” in Proc. Design, Automation & Test in Europe (DATE), 2019, pp. 1691–1696.

[5] F. N. Najm, “A survey of power estimation techniques in VLSI circuits,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 2, no. 4, pp. 446–455, Dec. 1994.

[6] F. N. Najm, “Transition density: A new measure of activity in digital circuits,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 12, no. 2, pp. 310–323, Feb. 1993.

[7] C.-Y. Tsui, M. Pedram, and A. M. Despain, “Power-efficient technology decomposition and mapping under an extended power consumption model,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 13, no. 9, pp. 1110–1122, Sept. 1994.

[8] M. Pedram, “Low-power RT-level synthesis techniques: A tutorial,” Integration, the VLSI Journal, vol. 34, no. 1–2, pp. 1–26, 2003.

[9] L. Benini and G. De Micheli, “System-level power optimization: Techniques and tools,” ACM Trans. Des. Autom. Electron. Syst., vol. 5, no. 2, pp. 115–192, Apr. 2000.

[10] P. Pillai and K. G. Shin, “Real-time dynamic voltage scaling for low-power embedded operating systems,” in Proc. ACM Symp. Operating Syst. Principles (SOSP), 2001, pp. 89–102.

[11] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Álvarez, “Power-aware scheduling for periodic real-time tasks,” IEEE Trans. Computers, vol. 53, no. 5, pp. 584–600, May 2004.

[12] E. Bassetti, L. Benini, and G. De Micheli, “Enhanced dynamic voltage scaling for energy-efficient systems,” in Proc. Int. Conf. Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), Lecture Notes in Computer Science, vol. 4017, 2006, pp. 380–389.

[13] D. Altan, M. Yilmaz, and K. Köse, “DVFS and duty-cycling for ultra-low-power IoT,” Electronics, vol. 11, no. 21, Art. 3445, Nov. 2022.

[14] A. Saifullah, D. B. Shin, K. G. Shin, X. Hu, and J. Liu, “On the interplay of DVFS and DPM in real-time embedded systems,” ACM Trans. Embed. Comput. Syst., vol. 7, no. 2, Art. 21, Jan. 2008.

[15] J. Chen, C. Chiang, and S. K. Gupta, “Control-theoretic dynamic voltage scaling for embedded processors,” in Proc. IEEE Real-Time Systems Symp. (RTSS), 2006, pp. 390–401.

[16] J. Pouwelse, K. Langendoen, and H. Sips, “Dynamic voltage scaling on a low-power microprocessor,” in Proc. 7th ACM Int. Conf. Mobile Computing and Networking (MobiCom), 2001, pp. 251–259.