A Comprehensive and Accurate Energy Model for Arm's Cortex-M0 Processor

Kyriakos Georgiou, Zbigniew Chamski, Kris Nikov, Kerstin Eder

University of Bristol, UK

Abstract. Energy modeling can enable energy-aware software development and assist the developer in meeting an application's energy budget. Although many energy models for embedded processors exist, most do not account for processor-specific configurations, neither are they suitable for static energy consumption estimation. This paper introduces a comprehensive energy model for Arm's Cortex-M0 processor, ready to support energy-aware development of edge computing applications using either profiling- or static-analysis-based energy consumption estimation. The model accounts for the Frequency, PreFetch, and WaitState processor configurations which all have a significant impact on the execution time and energy consumption of edge computing applications. All models have a prediction error of less than 5%.

1 Introduction

One trillion new Internet of Things (IoT) devices is predicted to reach the market by 2035 [ARM]. These devices would be generating an unprecedented amount of data that would need to be pushed to the cloud for storing and processing [IDC]. Edge computing, however, has enabled a degree of processing at the data-source that avoids the need for transmitting all collected data to the cloud. Although this can significantly reduce response time and bandwidth requirements, it results in increased resource requirements from edge devices, such as processing power and energy.

Typically, IoT devices are not part of a power grid but rather are scattered in the environment and powered by limited energy sources, such as batteries or energy harvesting. Thus, IoT devices are mostly based on small embedded processors with a tiny energy footprint, such as the Arm Cortex-M0. This kind of processor is inherently limited in processing power, making edge computing challenging. Developers must apply extreme optimizations to trim down the processing time, memory, and energy consumption of algorithms to enable their execution on small embedded devices. A trending example of such optimization is the streaming down of traditional machine learning algorithms to enable their execution on tiny IoT devices [NS].

The burden now lies with the software engineers to develop edge computing applications that can fit on the limited memory of the IoT embedded devices, execute within reasonable timeframes, and run within the available energy budget.

Balancing the resource usage of an IoT application is a challenging act and typically a manual trial-and-error process that costs significant development time. Thus, practical methodologies that enable resource-aware software development will significantly help to tackle the challenges associated with the development of edge computing applications.

Execution time and code size are easy to measure and well understood by the typical software developer. On the contrary, energy consumption information is not readily accessible, and something most software developers never had to account for. For edge computing, however, energy consumption feedback during the applications' development cycle is at least equally important as execution time and code size. Even when the energy consumption is directly proportional to time, energy consumption figures are still needed to ensure that the application's available energy budget is met [Eea16, GCAG^{+}20, GBXdSE18]. Thus, development tools need to enable such feedback [GdSE17].

Hardware measurements are the most accurate way of acquiring a program's energy consumption information, but they are not broadly supported by the hardware vendors and not within the know-how of typical software developers. Energy modelling and the integration of energy models into the development toolchains can solve both of these issues [GdSE17]. Once an accurate energy model has been developed for a particular platform, it can be integrated into a toolchain to allow for energy estimations with each compilation.

The literature offers a plethora of energy consumption models for embedded processors [PBH09, BCF11, KE15b, GKCE17, YJK^{+}20]. For an energy model to be useful to the software developer, it must be able to convey energy consumption information at the source-code level. Thus, Instruction-Set-Architecture-based (ISA) energy models [TMWT96] became the most popular because modeling at the ISA level allows for attributing energy costs to software components, such as ISA Control Flow Graph (CFG) basic blocks. Therefore, ISA-based energy models can be utilized by compiler autotuning techniques to discover energy targeted compiler optimizations [GCAG^{+}20, GBXdSE18], and by static analysis tools [GKCE17, GKE15] that estimate the energy consumption of programs.

Although ISA-based energy modeling approaches have benefits, extracting such models is time-consuming and challenging. It requires devising often complex energy measuring procedures to capture the energy consumption of each instruction in the ISA. Typically, an instruction is executed in a tight loop while measuring the power dissipated together with the execution time. For instructions that can not be measured within such a loop, for example, branch instructions, regression analysis is needed to capture their energy consumption.

On the other hand, energy modeling using Performance Monitoring Counters (PMCs), also named event counters, is a simpler approach than ISA-level modeling. It requires measuring the energy consumption of representative programs, collecting execution statistics from PMCs and then deducting energy consumption coefficients for each counter via regression analysis [LEMC01].

This paper demonstrates how to build PMC-based models for multiple embedded processor configurations. The models can be used to attribute energy costs to software components and facilitate both profiling-based and static-analysis-based energy consumption estimation, similar to ISA-based models. Our main contributions are:

1. Because many deeply embedded processors used for IoT devices, such as the ARM Cortex-M0, do not support hardware PMCs, we customized an open-source Instruction Set Simulator (ISS) of the Arm `thumb` ISA, namely the `Thumbulator` [Thu], to produce accurate execution statistics needed for a PMC-style energy model.
2. We identified a set of PMCs that are both statically predictable at ISA basic block level, and at the same time offer a low energy consumption estimation error (a Mean Absolute Percentage Error (MAPE) of less than 5%).

Only a few of the existing ISA models for deeply embedded processors are parameterized by hardware configurations, but only cover the processor's frequency and voltage [YJK+20]. Other processor configurations, such as instruction buffer configurations, are equally important as they have a major impact on both the execution time and the energy consumption of an embedded processor.

3. Thus, we enhanced `Thumbulator` to include such configurations for the STM-32F0xx family of processors [STMb]. We tracked the use of the instruction PreFetch buffer (ON/OFF) which aims at increasing the efficiency of instruction fetching, and the number of WaitStates (0/1) required to correctly perform read operations from Flash memory [STMa]. Our energy models include all the permitted combinations of the selected configurations for a set of commonly used processor frequencies: 20, 24, and 48 MHz. These energy models allow software developers to select the configuration that will meet their application's energy consumption and execution time goals. They can also potentially be used to assess the application's risk of exposure to side-channel attacks (see Section 2.3).

2 Energy Modeling Methodology

Counter	**Description**
C_1	Executed instructions (no Muls)
C_2	Multiplication instructions - Muls
C_3	Taken branches
C_4	RAM data reads
C_5	RAM writes
C_6	Flash data reads

Table 1: Statically predictable PMCs for energy-modeling.

Two sets of benchmarks were used for model characterization and validation. First, the BEEBS benchmark suite [PHB13]; an open-source embedded-system benchmark suite designed for exploring the performance and energy consumption characteristics of embedded architectures. BEEBS supports the MAGEEC open-source energy measurement framework [Mag]. The framework provides triggers to start and end the measurements and a calibration factor for each benchmark. This ensures that the benchmark is repeatedly executed in a loop until an adequate sampling number is achieved, and thus, the measurements can be trusted. 76 out of the 88 BEEBS benchmarks have been used. The remaining twelve do not fit in the available memory of our STM32F051 target chip. The second set of benchmarks is based on an industrial edge computing application [Tea19a], developed by Irida Labs [IRI]. The application uses a Convolutional Neural Network (CNN) and implements a smart car-parking monitoring system that can monitor, in real-time, a car parking lot with multiple parking slots to determine whether a slot is occupied or not. The different layers of the CNN, namely convolutional, MaxPool, and Full-Connected, were isolated and configured with different hyper-parameters and optimizations, resulting in 154 distinct benchmarks [Tea19b]. The MAGEEC energy measurement framework was adapted to support the CNN benchmarks. Overall, a total of 230 benchmarks were used for the training and validation of our energy model. This number goes significantly beyond the average number of used benchmarks reported for existing energy models of embedded processors, ranging between 10-20 benchmarks (see [Ke15a], pages 22-23, Table 5.1).

Hardware Configuration	Energy Consumption Model [nJ]	MAPE [%]	RESD [%]
[20, OFF, 0]	E $= 0.964258 \times C_1 + 1.652455 \times C_2 + 2.091986 \times C_3 + 1.109833 \times C_4 + 0.650563 \times C_5 + 0.633621 \times C_6$	2.80	3.60
[20, OFF, 1]	E $= 1.282474 \times C_1 + 2.110668 \times C_2 + 2.191545 \times C_3 + 1.185609 \times C_4 + 0.416602 \times C_5 + 1.178991 \times C_6$	2.97	3.60
[20, ON, 0]	E $= 1.003378 \times C_1 + 1.885309 \times C_2 + 1.802974 \times C_3 + 1.122833 \times C_4 + 0.849223 \times C_5 + 0.475831 \times C_6$	2.86	3.53
[20, ON, 1]	E $= 0.895879 \times C_1 + 2.185851 \times C_2 + 2.001178 \times C_3 + 1.493364 \times C_4 + 1.076354 \times C_5 + 1.573758 \times C_6$	3.68	4.61
[24, OFF, 0]	E $= 0.959172 \times C_1 + 1.888565 \times C_2 + 1.357556 \times C_3 + 1.089427 \times C_4 + 0.993145 \times C_5 + 0.562952 \times C_6$	3.22	3.63
[24, OFF, 1]	E $= 1.178558 \times C_1 + 2.540429 \times C_2 + 2.042475 \times C_3 + 1.190892 \times C_4 + 0.979651 \times C_5 + 0.891088 \times C_6$	3.16	3.90
[24, ON, 0]	E $= 0.985415 \times C_1 + 1.933276 \times C_2 + 1.448160 \times C_3 + 1.075671 \times C_4 + 1.011891 \times C_5 + 0.617510 \times C_6$	3.36	3.88
[24, ON, 1]	E $= 0.883755 \times C_1 + 2.156046 \times C_2 + 1.633465 \times C_3 + 1.436556 \times C_4 + 1.152560 \times C_5 + 1.455166 \times C_6$	4.15	5.02
[48, OFF, 1]	E $= 1.096677 \times C_1 + 2.364495 \times C_2 + 1.627854 \times C_3 + 1.173680 \times C_4 + 0.681475 \times C_5 + 0.652665 \times C_6$	3.65	4.08
[48, ON, 1]	E $= 0.816331 \times C_1 + 2.014612 \times C_2 + 1.372157 \times C_3 + 1.402116 \times C_4 + 0.835035 \times C_5 + 1.250446 \times C_6$	4.33	4.99

Table 2: Energy models for selected Cortex-M0 hardware configurations – Hardware Config. Format: [Frequency (MHz), PreFetch (ON/OFF), WaitState (0/1)], MAPE: Mean Absolute Percentage Error, and RESD: Relative Error Standard Deviation

2.1 PMC-based Code-level Energy Modelling

PMC-based energy consumption estimation models are typically obtained via multi-linear regression analysis, where coefficients, β_x, are determined for each counter, C_x, to predict the overall energy cost, i.e., $E = \sum_x(\beta_x \times C_x) + \alpha$, with α being the error term. The coefficients β_x are the constants in the energy model that are program independent while the counters C_x are the variables that depend on the program and its input. For a specific program with known counters, the energy model can be used to estimate the energy consumed during the program's execution.

For static-analysis-based energy consumption estimation, the overall energy consumption estimate of a piece of code is typically constructed from the estimates of the ISA basic blocks of the program [GKCE17]. Thus, a PMC-based energy model can enable energy consumption estimation via static analysis only if the counters used for the modeling and prediction can be statically predicted at the ISA basic block level.

2.2 Collection of Cortex-M0 Event Counters

Traditionally, there are two ways to collect execution statistics for an architecture. One is to collect them directly via PMCs while executing a program on the actual architecture, provided it offers PMCs. For architectures without PMCs, the second way is via an ISS, preferably cycle-accurate. The ISS simulates the execution of a program for a specific architecture, and thus, it can collect PMCs. Since the Cortex-M0 is a deeply embedded architecture with minimal resources available on-chip, it does not expose any PMCs. Thus, we modified an open-source ISS, namely `Thumbulator` [Thu], to extract the necessary event counters for our energy consumption modelling. The modifications wrt. the reference `Thumbulator` implementation [Thu] included four key aspects:

– Adaptation to reflect the memory organisation of the STM32F0xx processor family;
– Introduction of a model of the instruction fetch mechanism used in the STM32F0xx processors;
– Implementation of a range of event counters and the associated reporting mechanism;
– Calibration and improvement of the timing behaviour of the simulation to match the hardware's behaviour.

The modified simulator can be used to simulate any of the processors in the STM32F0xx family [STMb] and can collect a large number of event counters that represent various aspects of the architecture's runtime behaviour such as the effective RAM and Flash memory accesses, taken branches, per-opcode instruction execution statistics, and interactions between instruction- and data-related memory accesses. The timing behaviour validation of the modified `Thumbulator` against the actual hardware, using all the benchmarks, exposed a correctness bug in the implementation of the `ASR` instruction in the original `Thumbulator` code and identified a case of incorrect memory access counting. Both problems have been fixed in the version used to build the final energy model. The execution time model derived from event counts reported by `Thumbulator` is fully cycle-accurate wrt. hardware execution when the instruction PreFetch buffer is disabled or the WaitState count is 0. When the PreFetch buffer is enabled and the WaitState count is 1, the MAPE of the `Thumbulator`-based timing prediction is 1.55%.

Using the available architecture documentation and a series of modeling cycles, we constrained the number of event counters used for the modeling to the set of the counters that have the most significant impact on the energy consumption and are suitable for static analysis. These counters also yield the highest observed estimation accuracy compared to physical measurements when compared with the retrieved estimations of other event counter combinations. The selected counters are shown in Table 1.

2.3 Model Training and Validation

Both the BEEBS and CNN-based benchmarks have been compiled into two kinds of binaries. First, the benchmarks have been compiled for the STM32F0-DISCOVERY board in order to conduct energy consumption measurements. The hardware measured energy consumption of the programs provide the data for the dependent variables of our regression analysis. Second, the benchmarks have been compiled for the modified `Thumbulator` ISS in order to derive events counter values. The counter values provide the data for the independent variables of our regression analysis. The two sets of binaries are required because the simulator does not fully handle access to off-core peripherals, e.g., PLL clock generators; these should be skipped in `Thumbulator` binaries. However, the same location and alignment of benchmark code for both types of binaries was maintained.

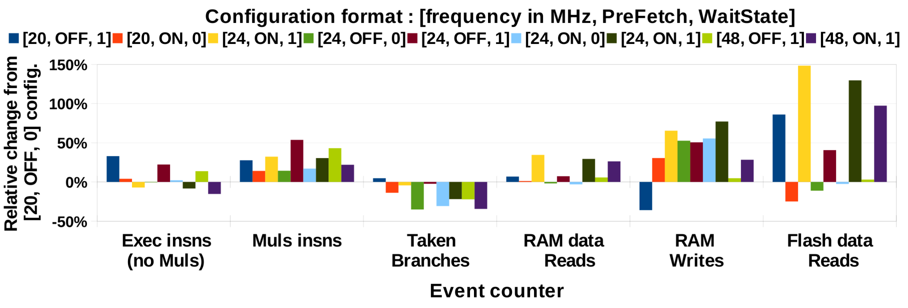

Fig. 1: Relative costs associated with events used in the energy model at distinct processor configurations.

When using regression modelling, it is critical to include as broad and representative a training sample as possible in the training phase. This ensures that the model is as generic as possible and can capture a large part of the space being modeled. Thus, instead of splitting our data into predefined training and testing sets, we included all data into the training, and we used k-fold cross-validation to ensure the retrieved models avoid overfitting and selection bias. If the cross-validation demonstrates a good estimation accuracy across all folds, then the final model using all the available data will exhibit a balanced variance and bias. Thus, the model will have a good chance of accurately capturing a big part of the space being modeled. In our case, we used 10-fold cross-validation and we used the R^2 to evaluate the performance of each of the ten models for each of the modelling configuration, shown in Table 2 under column *Hardware Configuration.* An R^2 value close to 1 demonstrates an excellent prediction. The 10-fold cross-validation yielded an R^2 mean value of close to 0.99 for all configurations, with a standard deviation of around 0.2%. This is a robust result as the R^2 score approaches the value of one across all the different folds, demonstrating that the counters selected for the model are accurately capturing the energy consumption of a variety of programs. Thus, for the final model, all the data points were used for training.

Energy models for the different hardware configurations and their accuracy are listed in Table 2. For all models the MAPE is less than 5%, with a standard deviation of less than 5%, compared to hardware energy measurements. An early version of the model, configured for 20 MHz frequency, PreFetch on, and WaitState 1 was evaluated in the context of static energy consumption estimation in [Tea19b] demonstrating the suitability of our models for static analysis.

2.4 Potential Use-Case in Cyber Attacks

A comparative analysis of energy model coefficients extracted at distinct processor configurations (see Figure 1) shows significant variation in relative weights of the different events across the hardware configurations. It follows that by comparing the energy consumption of the processor at distinct frequency, PreFetch, and WaitState settings and by applying statistical model fitting techniques, an observer can potentially predict the proportion of each event in the program. Subsequently, the observer will also be able to predict the type of processing being performed (e.g., data- vs. control-centric). By increasing the time resolution of analysis (narrowing the observed time window), the observer could also identify distinct program execution phases. From a security standpoint, such information leakage forms a potential side-channel for attacks. These attacks can be directed against the secret information contained in the program code (e.g., encryption keys or ciphering algorithms) or against the data being processed, e.g., in autonomous medical diagnostics devices.

3 Conclusion and future work

This paper offers an open-source, ready-to-use energy model for the Arm Cortex-M0 processor. The model can be used for profiling-based analysis to estimate the actual energy consumption, and in static analysis to estimate the energy consumed by the worst-case execution path of an edge-computing application. Furthermore, the model accounts for the frequency and the flash instruction-buffer configurations of the processor that can significantly affect the execution time and energy consumption of an application, namely, PreFetch, and WaitState instruction-buffer configurations. Our customized open-source ISS is also readily available to profile the execution time and energy consumption of edge computing applications for any of the STM32F0xx family of processors. This allows developers to choose the hardware configuration that can meet the resource requirements. Preliminary analysis indicates that our models can be exploited for side-channel attacks to reveal information about the type of application and the processing an application performs. Future work will test these observations.

Acknowledgments

This research is supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 779882, TeamPlay (Time, Energy and security Analysis for Multi/Many-core heterogeneous PLAtforms).

References

ARM. ARM. A trillion devices — A trillion dollars. Accessed: 2020-11-02. URL: `https://learn.arm.com/route-to-trillion-devices.html`.

BCF11. C. Brandolese, S. Corbetta, and W. Fornaciari. Software energy estimation based on statistical characterization of intermediate compilation code. In *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, pages 333–338, Aug 2011. `doi:10.1109/ISLPED.2011.5993659`.

Eea16. K. Eder and et al. ENTRA: Whole-systems energy transparency. *Microprocess. Microsyst.*, 47, Part B:278–286, November 2016. `doi:10.1016/j.micpro.2016.07.003`.

GBXdSE18. Kyriakos Georgiou, Craig Blackmore, Samuel Xavier-de Souza, and Kerstin Eder. Less is more: Exploiting the standard compiler optimization levels for better performance and energy consumption. In *Proceedings of the 21st International Workshop on Software and Compilers for Embedded Systems*, SCOPES '18, pages 35–42, New York, NY, USA, 2018. ACM. `doi:10.1145/3207719.3207727`.

GCAG$^+$20. Kyriakos Georgiou, Zbigniew Chamski, Andres Amaya Garcia, David May, and Kerstin Eder. Lost In Translation: Exposing Hidden Compiler Optimization Opportunities. *The Computer Journal*, Aug 2020. `doi:10.1093/comjnl/bxaa103`.

GdSE17. K. Georgiou, S. Xavier de Souza, and K. Eder. The IoT energy challenge: A software perspective. *IEEE Embedded Systems Letters*, PP(99):1–1, 2017. `doi:10.1109/LES.2017.2741419`.

GKCE17. Kyriakos Georgiou, Steve Kerrison, Zbigniew Chamski, and Kerstin Eder. Energy transparency for deeply embedded programs. *ACM Trans. Archit. Code Optim.*, 14(1), Mar 2017. `doi:10.1145/3046679`.

GKE15. Kyriakos Georgiou, Steve Kerrison, and Kerstin Eder. On the value and limits of multi-level energy consumption static analysis for deeply embedded single and multi-threaded programs. abs/1510.07095, 2015. URL: `http://arxiv.org/abs/1510.07095`, `arXiv:1510.07095`.

IDC. IDC. The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025. Accessed: 2020-11-02. URL: `https://www.idc.com/getdoc.jsp?containerId=prUS45213219`.

IRI. Irida Labs. Accessed: 2020-12-28. URL: `https://iridalabs.com/`.

Ke15a. Liu Ke. *A Simulation Based Approach to Estimate Energy Consumption for Embedded Processors.* PhD thesis, 2015. Accessed: 2020-12-28. URL: `http://www.diva-portal.org/smash/get/diva2:877703/FULLTEXT01.pdf`.

KE15b. Steve Kerrison and Kerstin Eder. Energy modeling of software for a hardware multithreaded embedded microprocessor. *ACM Trans. Embed. Comput. Syst.*, 14(3), April 2015. `doi:10.1145/2700104`.

LEMC01. Sheayun Lee, Andreas Ermedahl, Sang Lyul Min, and Naehyuck Chang. An Accurate Instruction-Level Energy Consumption Model for Embedded RISC Processors. *SIGPLAN Not.*, 36(8):1–10, aug 2001. `doi:10.1145/384196.384201`.

Mag. MAGEEC measuring setup. Accessed: 2020-08-14. URL: `http://mageec.org/wiki/Workshop`.

NS. Danny Loh Naveen Suda. Machine Learning on Arm Cortex-M Microcontrollers — White paper. Accessed: 2020-11-02. URL: `https://www.arm.com/resources/guide/machine-learning-on-cortex-m`.

PBH09. S. Penolazzi, L. Bolognino, and A. Hemani. Energy and Performance Model of a SPARC Leon3 Processor. In *Digital System Design, Architectures, Methods and Tools, 2009. DSD '09. 12th Euromicro Conference on*, pages 651–656, Aug 2009. `doi:10.1109/DSD.2009.147`.

PHB13. James Pallister, Simon J. Hollis, and Jeremy Bennett. BEEBS: open benchmarks for energy measurements on embedded platforms. *CoRR*, abs/1308.5174, 2013. URL: `http://arxiv.org/abs/1308.5174`, `arXiv:1308.5174`.

STMa. STM32F030x4/x6/x8/xC and STM32F070x6/xB advanced ARM-based 32-bit MCUs - Reference Manual. Accessed: 2020-12-28. URL: `https://www.st.com/resource/en/reference_manual/dm00091010-stm32f030x4x6x8xc-and-stm32f070x6xb-advanced-armbased-32bit-mcus-stmicroelectronics.pdf`.

STMb. STM. The STMF32F0xx family of processors. Accessed: 2020-11-09. URL: `https://www.st.com/en/microcontrollers-microprocessors/stm32f0-series.html`.

Tea19a. TeamPlay Consortium. Deliverable D4.3 - Report on Energy, Timing and Security Modeling of Complex Architectures - Section 7.2 CEs Architectural Description, 2019. URL: `https://gitlab.inria.fr/TeamPlay_Public/TeamPlay_Public_Deliverables/-/blob/master/D4.3.pdf`.

Tea19b. TeamPlay Consortium. Deliverable D4.4 - Final Report on Architecture-Level Energy Usage, Timing and Security Modeling and on Prototype - Section 5 , 2019. URL: `https://gitlab.inria.fr/TeamPlay_Public/TeamPlay_Public_Deliverables/-/blob/master/D4.4.pdf`.

Thu. The Thumbulator Git repository modified for collecting performance monitoring counters for the STM32F0-Discovery board. branch: prefetch-model, Git-tag: teamplay-D4.5. Accessed: 2020-11-09. URL: `https://github.com/PicoPET/thumbulator-stm32f0x.git`.

TMWT96. V. Tiwari, S. Malik, A. Wolfe, and M. Tien-Chien Lee. Instruction level power analysis and optimization of software. *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 13(2-3):223–238, 1996. `doi:10.1007/BF01130407`.

YJK+20. Yahya H. Yassin, Magnus Jahre, Per Gunnar Kjeldsberg, Snorre Aunet, and Francky Catthoor. Fast and Accurate Edge Computing Energy Modeling and DVFS Implementation in GEM5 Using System Call Emulation Mode. 36(8), may 2020. doi:10.1007/s11265-020-01544-z.