# FlexNode: a reconfigurable Internet of Things node for design evaluation

Guillaume Patrigeon, Paul Leloup, Pascal Benoit, Lionel Torres

# FlexNode: a reconfigurable Internet of Things node for design evaluation

Guillaume Patrigeon[1], Paul Leloup[2], Pascal Benoit[1], Lionel Torres[1]
[1]LIRMM - University of Montpellier, CNRS - Montpellier, France - *firstname.lastname*@lirmm.fr
[2]University of Montpellier - Montpellier, France - paul.leloup@etu.umontpellier.fr

***Abstract*—Accurate evaluation of Ultra Low Power Systems on Chip (ULP SoC) is a huge challenge for designers and developers. In embedded applications, especially for Internet of Things end-node devices, ULP SoCs have to interact with their environment, but modelling a complete SoC and the peripheral components, their interaction and low-power policies, can be very complex in terms of developments and benchmarking. In order to cope with this challenge, an approach is to implement the desired system on FPGA with a monitoring infrastructure dedicated to fast and accurate evaluation. This paper presents a reconfigurable prototyping platform used for SoC architecture exploration and real-time application evaluation.**

***Index Terms*—FPGA, Internet of Things, sensor node, WSN**

## I. Introduction

Wireless Sensor Networks (WSN) are widely used in various monitoring application requiring space deployment, thanks to their flexibility and low implementation cost compared to wired solutions [1]. However, many of these applications puts high constraints to designers and developers in terms of performance, energy consumption and security. These limitations lead to explore new technologies and techniques to complete, replace or improve current solutions and so at every levels. Characterization of these new solutions and comparison with current ones are necessary to evaluate the potential gains. Moreover, they have to be done at application level to justify the use of these innovations and the investments required for their manufacturing and deployment. A typical sensor node is composed of a controller surrounded by sensors and actuators (both optional, depending on the application), at least one energy source and communication modules (Fig. 1). Because all these components work together, application level evaluation has to take account of interactions between each other and the contribution of each one. Moreover, it has to take account of the possible network impact on the node behaviour when bidirectional communication is used.

Many works present physical implementations with dedicated architectures ([2]) and using new technologies and power management techniques (example with non-volatile logic and non-volatile RAM: [3]). If it is possible to integrate them in an application for accurate evaluation of a solution, they are prototypes mostly used for validation, not flexible enough for design space exploration. Simulation is one of the solutions for sensor node evaluation that is flexible and affordable to perform this investigation at various level. However, there is a trade-off between accuracy and simulation speed: the more accurate we want to be, the slower the simulation will be and so at every level, from the SoC evaluation [4] to the radio model [5]. Moreover, some models are not exempt from bugs and some others are not accurate enough because of abstractions, simplifications and underestimated effects, especially for wireless communications [6], [7].

Power emulation using FPGA acceleration [8] is a solution between the flexibility of simulation tools and the accuracy and speed of a physical SoC. With FPGA prototyping, every signal of the implemented design is reachable like for RTL simulation. Performance and power estimation can be done by using monitoring tools with adapted models. As ULP SoCs usually run at low frequencies, it is possible to perform an evaluation with real-time execution, so the integration of an FPGA-based system inside an already deployed WSN is also possible for application level evaluation. Some other works use the speed advantage of FPGAs to accelerate a design evaluation. For example, Strober [9] is an evaluation tool that use an FPGA as an accelerator. The design is implemented on it, and random snapshots of the registers' state are taken during runtime. Each capture is transmitted to a computer that will replay it on a RTL simulator, and a power estimation flow is used to obtain the power consumption of the design at the moment of the capture. This method helps to accelerate the system evaluation, which is interesting for long runtime applications. However, the snapshot system used by Strober requires the transfer of the registers' state at multiple time during the execution. These interruptions do not allow real-time evaluation since some external events may be missed during this step. Unlike Strober, we propose another FPGA-based solution allowing real-time execution and interactivity. We present here the FlexNode, a reconfigurable prototyping platform used for design space exploration and evaluation of new technologies for ULP SoC. Section II introduces the design evaluation flow based on FPGA prototyping. Section III presents the prototype platform and the design evaluated in Section V. Section IV focuses on the activity monitor, a tool implemented in our design to capture the system's activity. Section V presents evaluations with two software benchmarks. Finally, Section VI concludes this paper.

## II. Evaluation flow

The typical power estimation flow of a design consists of a power model driven by the system activity and the environment. This power model can be composed of individual ones, obtained from low-level simulation (for example, with

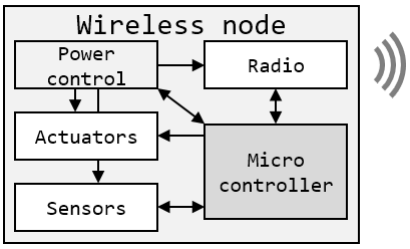


Fig. 1. Typical wireless node

Spice), from standard technology library cells given by manufacturers through design kits, from physical implementation measurements or from the literature. The system activity is dependent on the application behaviour. This activity can be obtained by multiple methods, depending on the nature of the design. Here we focus on a microcontroller design, that is, a synchronous digital circuits. For this kind of device the activity is typically obtained from the RTL design thanks to simulation tools (for example, ModelSim) and application test benches. In the case of programmable devices, which is the case with a microcontroller, part of a test bench come from a software code compiled from a programming language such as assembly or C. When external events are part of the test bench, they are injected during the simulation. The design activity recorded during this step is then provided to a power estimation tool with the power model previously generated.

This typical power estimation flow presents inconveniences. First, the simulation step is slow. Secondly, the external events are scripted, so critical cases could be forgotten. Our proposition is to modify this evaluation flow, replacing the RTL simulation by FPGA prototyping for application level analysis (Fig. 2). Power models are still generated like for a standard evaluation flow, with low-level hardware simulations, manufacturers' design kits and literature. It has been demonstrated that using Performance Monitoring Counters (PMC) can be used for power consumption estimation [10], [11]. Because, inside a bloc, some signals/events are more correlated with the bloc's power consumption than other ones, it is possible to use a simplified power model based on these events, which will be captured thanks to a dedicated activity monitor. The target design is synthesised and implemented on a FPGA board using FPGA development tools, with the fewest changes possible (some IPs need to be replaced in order to run on the FPGA, like memories or I/O buffers, but the behaviour is unchanged). Now that the design is implemented on FPGA, real peripherals can be attached to it to produce the external events required by the test bench. The data captured by the activity monitor are then used with the power model to produce power estimation.

## III. FlexNode

### A. Prototyping platform

The FlexNode prototyping platform consists of an electronic board with a controller slot, peripheral slots, power

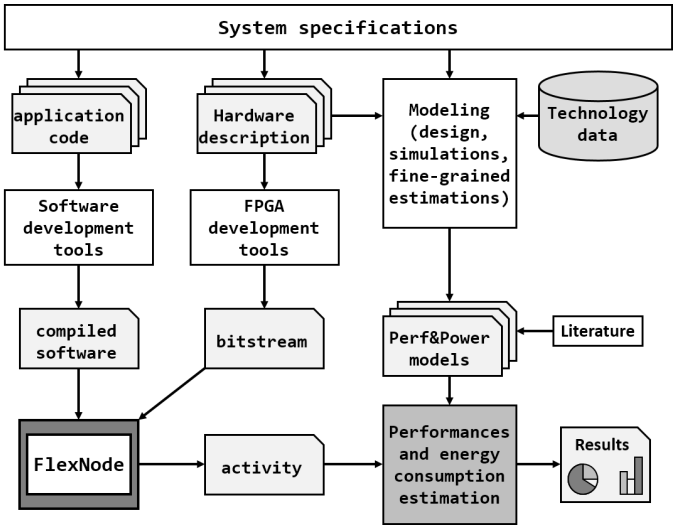


Fig. 2. Evaluation flow

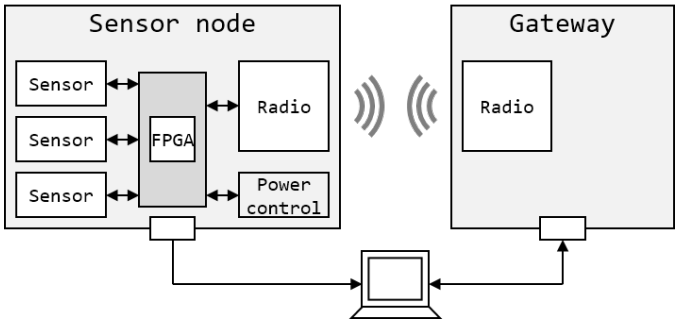


Fig. 3. Prototyping platform

distribution and the necessary components to perform power characterization of each element composing the node. The peripheral slots can be used to connect sensors, actuators or communication modules. It is possible to use this board for the characterization of a peripheral, and so to obtain a power model of it. This platform can be used for the evaluation and comparison of multiple controller solutions, with different node architectures and applications. This node can be directly connected to existing WSN to perform evaluation while taking account of communication and network hazards (Fig. 3).

If we use the Digilent CmodA7 as main controller of the node, it is possible to exchange the FPGA board with other kinds of controller, for evaluation and solution comparison. The CmodA7 is a small, 48-pin DIP form factor board built around a Xilinx Artix-7 FPGA. The CmodA7 35T use the XC7A35T-1CPG236C, which has 20 800 LUTs, 41 600 Flip-Flops, 225 kB of RAM and 1 MSPS ADC. We can explore different architecture solutions by reproducing the desired behaviour with the FPGA. With monitoring tools and power models, it is also possible to make estimations of a designed architecture's power consumption for a defined application.

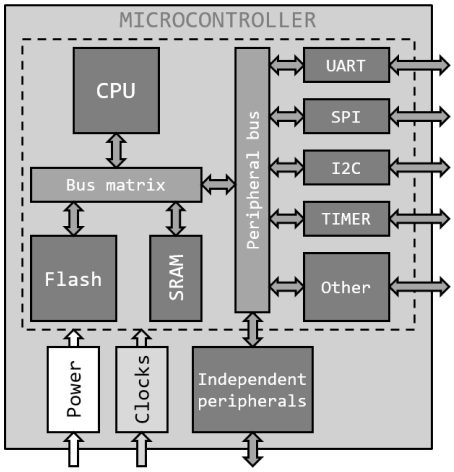


Fig. 4. Typical microcontroller architecture

### B. Controller architecture

The specifications of an application determine which microcontroller to use. Manufacturers generally offer a large variety of microcontrollers to answer to the large number of actual embedded applications and their specific constraints, as it is not possible to design one microcontroller architecture that will fit all applications. There are microcontrollers with different packages, number of input/output pins, processor, operating frequencies, peripherals, communication interfaces, analogic modules, low-power modes, memory technologies, memory capacities, and dedicated to different kind of applications (automotive for example). However, there are some similarities between all these different devices. Typical microcontrollers include at least one processor, a non-volatile memory (usually Flash for code instructions and read-only data), a volatile memory (usually SRAM for application data), a power management unit, a clock management unit, input/output peripherals, communication modules (UART, SPI, I²C, USB, CAN...) and timers. This typical architecture is depicted in Fig. 4. Some microcontrollers also include different types of non-volatile memories (ROM, EEPROM...) or have a multi-master system (multi-processors, Direct Memory Access (DMA)...).

### C. Architecture overview

Here is an example of a controller implementation we use in this work. The following system is used in the experiments described in Section V.

ARM Cortex-M are widely used in commercial low-power microcontrollers. We use the ARM Cortex-M0 r1p0 in our evaluations. This is a 3-stage 32-bit RISC processor that implements the ARMv6-M ISA, with a maximum frequency of 50 MHz. It includes a single AHB-Lite interface, 32 interrupt lines, 1 Non-Maskable Interrupt and a single-cycle multiplier. Existing products using this processor can be used as hardware references for performance evaluation comparison.

The architecture used in this work, depicted in Fig. 5, is composed of the ARM Cortex-M0 r1p0 processor, a 2 kB ROM containing a bootloader code, a 128 kB RAM and a 16 kB RAM, peripherals for inputs/outputs control (44 I/O, PPS), serial communication (4x UART, 2x SPI, 2x I²C) and timing modules (4x 16-bit timers). All these elements are connected together thank to a single-master AMBA3 AHB-Lite system. A peripheral called *Activity Monitor* is used to report events and will serve as basis for the design evaluation flow described in Section II.

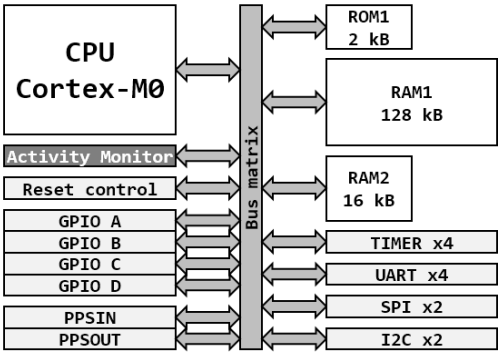


Fig. 5. Architecture example

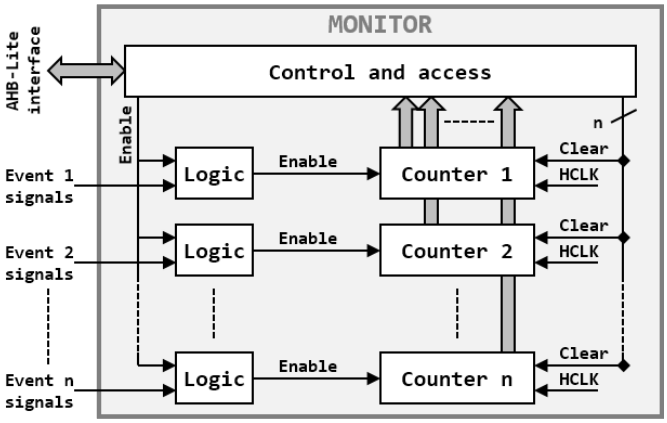


Fig. 6. Monitor block diagram

## IV. Activity Monitor

### A. Hardware

The activity monitor is a set of counters used to capture events. Its architecture is based on the principle of PMU, as described in Fig. 6.

In the work we present here, the activity monitor is designed to capture the following events related to the memory:

- Number of cycles
- Number of executed instructions
- Number of instruction fetches
- Number of RAM read accesses
- Number of RAM write accesses

The activity monitor is connected to the AHB-Lite bus system, and can be accessed by the processor as a peripheral. By connecting it to the main bus, it is possible to start, stop

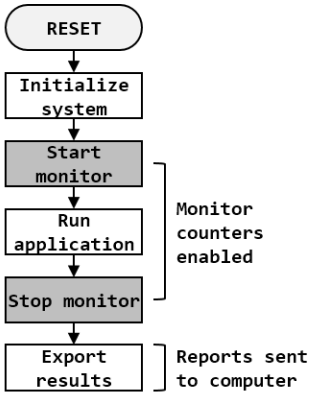


Fig. 7. Basic system initialization and run sequence

and reset the counters by software using a control register. Activity monitoring can be performed for a selected portion of code without adding external control hardware.

For cycle counting, a simple counter, always enabled, is used. The instruction counter is incremented each time the program counter (PC) changes. The RAM counters are incremented when a valid RAM access is detected.

The size of the counters depends on the desired use. In this work, we use 32-bit counters, which will overflow after 86 seconds at 50 MHz (ARM gives 50 MHz as the maximum frequency supported by the Cortex-M0 processor).

### B. Software interface

The software manages the activity monitor by writing into its control registers. At system reset, the application code initializes the platform, runs code and performs monitoring by using the activity monitor. Then, it retrieves the monitor's counter values and can share them or process the results. Fig. 7 shows the basic sequence to perform activity monitoring on a portion of code.

Depending on the implementation and software optimizations, the processor needs to execute some instructions to stop the monitor, which are captured by the counters. If stopping the monitor may add few extra cycles, instructions and RAM accesses to these counters, the overhead can be measured and is most of the time negligible (5 cycles, 3 instructions including 1 load and 1 store).

## V. Experiments

### A. FPGA resources

Synthesis is done using Vivado 2018.2 Synthesis tool, with Vivado default pre-set. Implementation is also done using Vivado 2018.2 Implementation tool, still with Vivado default pre-set. The Table I shows the resources utilization of the whole implementation, of the Cortex-M0 and of the Activity Monitor.

### B. Application benchmarks

Two application benchmarks are considered in this work.

TABLE I
Resources utilization

| Resources | Used | Available | Ratio |
|---|---|---|---|
| **Total utilization** | | | |
| Slice LUTs | 5915 | 20800 | 28.44% |
| Slice registers | 3432 | 41600 | 8.25% |
| Block RAM Tile | 36.5 | 50 | 73.00% |
| **Cortex-M0** | | | |
| Slice LUTs | 3118 | 20800 | 14.99% |
| Slice registers | 901 | 41600 | 2.17% |
| Block RAM Tile | 0 | 50 | 0.00% |
| **Activity Monitor** | | | |
| Slice LUTs | 305 | 20800 | 1.47% |
| Slice registers | 372 | 41600 | 0.89% |
| Block RAM Tile | 0 | 50 | 0.00% |

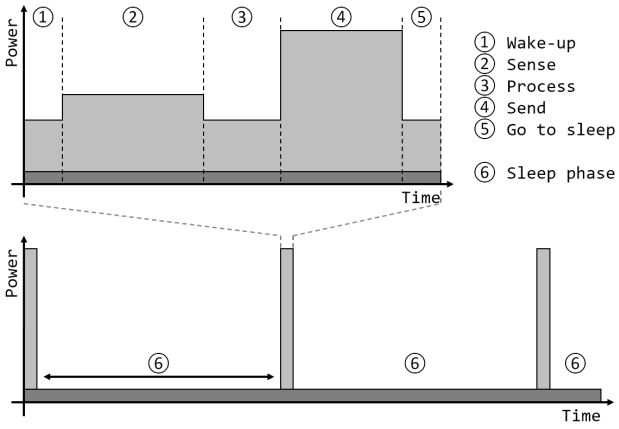


Fig. 8. Application behaviour

The first one is the ULPMark from EEMBC [12]. CoreProfile (ULPMark-CP) is an application designed to reproduce a periodic behaviour with active and sleep phases (see Fig. 8). The active phase of the CoreProfile is composed of math functions (linear approximation, filtering), conversion tables, string search, table copy, sorting, data permutations and output toggling. This application code is used to evaluate and compare the energy efficiency of Ultra-Low-Power microcontrollers for Internet of Things applications

The second benchmark used is based on a software code of a commercial agriculture application. The system's drivers are replaced to match the architecture implemented inside the FPGA, but the main algorithm is keep the same. Depicted by Fig. 8, this benchmark is composed of active phases separated by sleep periods. At each wake-up, the node performs sensing, processes the data and then sends them by radio before returning to sleep phase. We use the ADT7420, a digital temperature sensor with I²C interface, which is the one used by Digilent on the PmodTMP2. The radio module is the SX1272 from SEMTECH, a LoRa transceiver with SPI interface. The SX1272 is configured in LoRa mode with a spreading factor of 12. The schematic of this node is presented in Fig. 9.

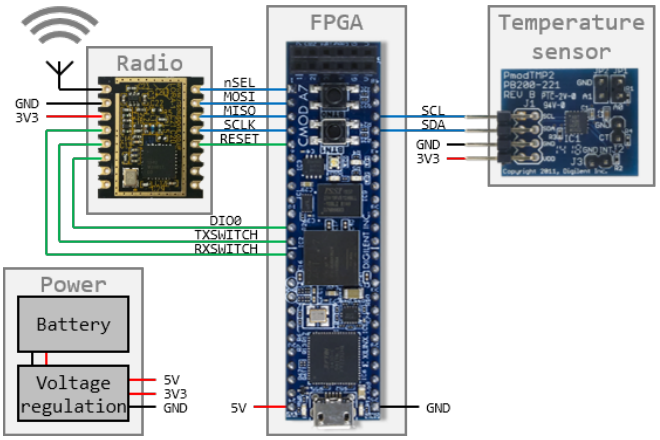


Fig. 9. Schematic of the sensor node

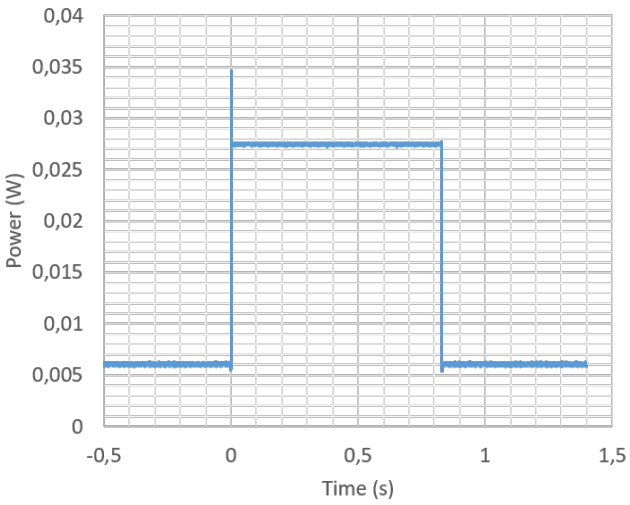


Fig. 10. Power consumption of the LoRa module when transmitting a message

### C. CoreProfile

The activity monitor starts at wake-up, and stops before calling *SLEEP* instruction. One active phase of the CoreProfile benchmark runs 49767 clock cycles with 32773 executed instructions. Memory operations for each benchmark (one active phase) are detailed in Table II. *Total reads*, which is the number of data read operations, does not include *Instruction fetches*. One instruction fetch corresponds to a 32-bit read operation. *Idle cycles* is the number of cycles without read nor write request to the memory.

TABLE II
MEMORY OPERATIONS FOR ONE COREPROFILE ACTIVE PHASE (49767 CLOCK CYCLES, 32773 EXECUTED INSTRUCTIONS)

| Memory location | Code | Data |
|---|---|---|
| **Idle cycles** | 29532 | 38280 |
| **Instruction fetches** | 18693 | 0 |
| **Total reads** | 1542 | 8151 |
| **Total writes** | 0 | 3336 |
| 8-bit writes | 0 | 1000 |
| 16-bit writes | 0 | 516 |
| 32-bit writes | 0 | 1820 |

### D. Agriculture application

Like for CoreProfile benchmark, the activity monitor starts at wake-up, and stops before calling *SLEEP* instruction. One active phase of the agriculture application runs 9936204 cycles with 2173 executed instructions. The memory operations are detailed in Table III. The high amount of cycles is due to wait states during communication through SPI and I2C, and during the radio emission. Some time is spent into communication: SPI currently works at 3 MHz and I²C bus works at 400 kHz whereas CPU frequency is 12 MHz. However, the main loose of time is due to the LoRa communication, which takes more than 800 ms to send one message. Regarding the application specifications in term of energy consumption, radio range and amount of data to transmit, it is possible to optimize the radio communication to save energy and so expand the battery life. Some optimisation can be done by forcing sleep mode of unused components.

TABLE III
MEMORY OPERATIONS FOR ONE ACTIVE PHASE OF THE AGRICULTURE APPLICATION (9936204 CYCLES, 2173 EXECUTED INSTRUCTIONS)

| Memory location | Code | Data |
|---|---|---|
| **Idle cycles** | 9935295 | 9935786 |
| **Instruction fetches** | 783 | 0 |
| **Total reads** | 126 | 192 |
| **Total writes** | 0 | 226 |
| 8-bit writes | 0 | 55 |
| 16-bit writes | 0 | 3 |
| 32-bit writes | 0 | 168 |

Over the 9936204 clock cycles, the processor is active (that is, executing instructions) only for 2173 cycles. It is put in sleep mode the rest of the time. The temperature value is fetched in 270 µs and the sensor as an average power consumption of 0.77 mW. In comparison, the LoRa module consumes 27.5 mW during 827 ms (Fig. 10). These results clearly show that for this sensor node, potential gains in term of energy saving will be on the radio communication.

## VI. CONCLUSION

We have presented in this paper the FlexNode, an FPGA-based prototyping platform for fast, real-time, accurate and low cost evaluation and prototyping of ULP SoCs for sensor nodes. This platform can be easily interfaced with sensors, actuators and other peripherals for application level analysis. The activity monitor we presented can be exploited for architecture optimization and exploration purposes, and is easily customizable and adaptable as wished. The flexibility provided by the FPGA also makes possible to capture events that could not be available in a software model, with an accuracy similar

to RTL simulations but with real time execution. Moreover, the FlexNode can be integrated into existing WSN to evaluate network hazards effects for large-scale deployment.

## ACKNOWLEDGMENT

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 687973 (GREAT project) and the French National Research Agency under grant ANR-15-CE24-0033-01 (MASTA project).

## REFERENCES

[1] P. Tiwari, V. P. Saxena, R. G. Mishra, and D. Bhavsar, "Wireless Sensor Networks: Introduction, Advantages, Applications and Research Challenges," *Wireless Sensor Networks*, vol. 14, p. 11, 2015.

[2] G. Lallement, F. Abouzeid, M. Cochet, J.-M. Daveau, P. Roche, and J.-L. Autran, "A 2.7 pJ/cycle 16 MHz, 0.7 µW Deep Sleep Power ARM Cortex-M0+ Core SoC in 28 nm FD-SOI," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 7, pp. 2088–2100, Jul. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8351905/

[3] S. Izumi, K. Yamashita, M. Nakano, S. Yoshimoto, T. Nakagawa, Y. Nakai, H. Kawaguchi, H. Kimura, K. Marumoto, T. Fuchikami, Y. Fujimori, H. Nakajima, T. Shiga, and M. Yoshimoto, "Normally Off ECG SoC With Non-Volatile MCU and Noise Tolerant Heartbeat Detector," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 5, pp. 641–651, Oct. 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7265104/

[4] S. Fontaine, L. Filion, and G. Bois, "Exploring ISS Abstractions for Embedded Software Design." IEEE, 2008, pp. 651–655. [Online]. Available: http://ieeexplore.ieee.org/document/4669297/

[5] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K.-c. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan, "Effects of Detail in Wireless Network Simulation," p. 10.

[6] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental Evaluation of Wireless Simulation Assumptions," p. 5.

[7] S. Kurt and B. Tavli, "Path-Loss Modeling for Wireless Sensor Networks: A review of models and comparative evaluations." *IEEE Antennas and Propagation Magazine*, vol. 59, no. 1, pp. 18–37, Feb. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7809115/

[8] J. Coburn, "Power Emulation: A New Paradigm for Power Estimation," Anaheim, CA, USA, Jun. 2005, p. 6. [Online]. Available: https://ieeexplore.ieee.org/document/1510422

[9] D. Kim, A. Izraelevitz, C. Celio, H. Kim, B. Zimmer, Y. Lee, and J. Bachrach, "Strober: Fast and Accurate Sample-Based Energy Simulation for Arbitrary RTL," Jun. 2016, p. 12.

[10] M. E. Ahmad, M. Najem, P. Benoit, G. Sassatelli, and L. Torres, "Adaptive Power monitoring for self-aware embedded systems." IEEE, Oct. 2015, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/7364364/

[11] R. Bertran, M. Gonzelez, X. Martorell, N. Navarro, and E. Ayguade, "A Systematic Methodology to Generate Decomposable and Responsive Power Models for CMPs," *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1289–1302, Jul. 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6189333/

[12] "CPU Energy Benchmark – MCU Energy Benchmark – ULPMark – EEMBC Embedded Microprocessor Benchmark Consortium." [Online]. Available: https://www.eembc.org/ulpmark/