A Scan-Chain Based State Retention Methodology for IoT Processors Operating on Intermittent Energy

Pascal Alexander Hager*, Hamed Fatemi†, Jose Pineda de Gyvez† and Luca Benini*‡
*Integrated Systems Laboratory, ETH Zürich, Switzerland, {hager,benini}@iis.ee.ethz.ch
†NXP Semiconductor, Eindhoven, The Netherlands, {fristname.lastname}@nxp.com
‡Electrical, Electronic, and Information Engineering, University of Bologna, Italy

***Abstract*—Future IoT systems are tightly constraint by cost and size and will often be operated from an energy harvester's output. Since these batteryless systems operate on intermittent energy they have to be able to retain their state during the power outages in order to guarantee computation progress. Due to the lack of large energy buffers the state needs to be saved quickly using residual energy only. In related work, the state is retained *in-place* by replacing all flip-flops with state retentive flip-flops (SRFF), which are powered by auxiliary supplies for retention or incorporate non-volatile memory cells. However, these SRFFs increase the power consumption during active operation impairing the overall systems efficiency. In this paper, we present a scan-chain based state retention approach, where the state is moved to memory using only 4.5pJ/b. Since our approach does not introduce any power overhead, this energy cost pays off after an on-time of just 100us compared to state-of-the-art in-place solutions. Moreover, compared to a software mechanism, our approach requires 6.6x less energy to move the state and is 5.8x faster.**

I. Introduction

In many Internet of Things (IoT) applications, such as sensor tags [1], an environmental variable is sampled once in a while, processed and sent to the cloud over a radio link. As many IoT systems cannot afford large energy buffers due to space and cost constraints, they are powered directly by harvesting intermittent power sources [2]. As a result of this powering scheme, the input power can be lost at any time, which complicates reliable system operation. One approach to operate reliably in such a setup, is to split the operation of a IoT node in *atomic tasks*, e.g., take one sample, process a few samples, transmit the processed samples. Each atomic task has then a known energy requirement and as soon as sufficient energy for a task was harvested on a tiny energy buffer, the task is executed [3]. While this scheme guarantees reliable operation, it can only be applied when the maximal required energy of a task can be reasonable bounded, which is often not the case for the processing tasks due to data-depended execution times. Design for worst-case execution energy is often unfeasible due to overly large energy buffers and splitting the task into many shorter sub-tasks heavily complicates the system design as the maximal energy consumption of each sub-task needs to be characterized.

This research was funded by the Swiss National Science Foundation under grant 157048: Transient Computing Systems. This work was partially supported by the EU FP7 ERC Advance project MULTITHERMAN (GA n. 291125).

Extending the scheme to support the execution of *non-atomic tasks*, alleviates these drawbacks. Non-atomic tasks are tasks that can be arbitrarily interrupted. They are launched as soon as sufficient energy is available to start them. If the input power dies and the buffered energy is depleting, they are halted. Once power is restored, they are continued.

Since these IoT systems operate entirely on harvested energy, it is not sufficient to just halt the execution and wait in a low-power mode and hope for power restoration, as power will be completely lost after a while, when all buffers are discharged. Thus, in order to guarantee computation progress and to be able to always continue an interrupted task after a complete power outage, the IoT microprocessor must retain its state, which includes both *memories* and *registers* alike.

The state of on-chip volatile *memories* is usually retained in-place and on-chip, as it is unpractical to move the entire content to a dedicated state retaining memory due to the required amounts of energy. This *on-chip memory retention* is either realized by using a non-volatile memory technology like FRAM [4] or by supporting an ultra low power retention mode in the case of SRAM, which consumes less than a pW per bit [5], [6]. This power consumption makes SRAM virtually non-volatile as a tiny 50uAh (3.8V) solid state bare die battery [7], [8] can be co-packaged with the processor, which supplies sufficient energy to retain 10kbit for over 2 years.

If all relevant *registers* are accessible by the processor, the state can be saved in *software* by copying it to a state-retaining memory. *Hardware* state retention implementations can be divided in two approaches: *In-place* (InP) state saving implementations replace all registers with State Retentive Flip-Flops (SRFFs), which retain the state, while the main power is cut by having an auxiliary supply or by incorporating a non-volatile memory cell. Similar to software approaches, *out-of-place* (OoP) hardware implementations move the state of the registers to a state-retaining memory. Using dedicated hardware reduces the required energy to move the state. Adding dedicated hardware can easily introduce a power overhead during active operation. Even when the active power is only increased by a few percents, any energy savings achieved with dedicated hardware are periled to be canceled after a brief on-time. Thus, it is absolutely crucial to keep the introduced overhead at bay, considering the scarce energy in an IoT node.

In this work, we present BRAINSHIFT, a scan-chain based state retentive power gating (SRPG) methodology for IoT

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

SoCs. BRAINSHIFT reuses the existing scan-chains to implement low-energy out-of-place state saving of power gated domains. We show that it introduces virtually no active power overhead and that it does not deteriorate the testability. While BRAINSHIFT can be used for SRPG in general, we demonstrate in this paper its application to a tiny IoT microcontroller executing non-atomic tasks in a battery-less system with intermittent energy availability. Our contributions are:

- We present a system methodology based on scan-chains to save/restore the state of a microcontroller to guarantee computational progress over intermittent power losses and describe the required hard- and software modules.
- We re-purpose existing SRPG techniques designed for leakage mitigation to build a microcontroller able to execute non-atomic tasks on intermittent energy availability.
- We compare our scan approach with a software solution on the same platform to show that active power is not increased while the energy to move the state is reduced by 6.6x.
- We compare with related work and show that we achieve similar save/restore energies (5pJ/b) without needing non-volatile memory, while using a retention power of 1pW/b.
- By not introducing any power overhead, we show that our approach gets more energy efficient after a brief on-time of around 100us compared to in-place solutions – even if they only introduce an overhead of 2% of 3mW active power.

This paper is organized as follows: After related work, we present the system-level architecture. Subsequently, we elaborate how BRAINSHIFT is integrated into a microcontroller. Next, we present and discuss the results and close with conclusions and future work.

II. RELATED WORK

We group related work depending on whether the state of the processor core is retained *in-place* (InP) or moved *out-of-place* (OoP) to be retained elsewhere, and whether a *volatile* or *non-volatile* element is used for retention.

In-place state retention avoids the movement of data altogether by using State Retentive Flip-Flops (SRFF). SRFF come in many flavors: The ballon or shadow-latch FF [9] uses a third always-on latch beside the gated master and slave latch to retain the state. Life-latch based SRFF retain the state in the slave latch by not power gating the slave latch and the clock signal. Both are *volatile* and require virtually no energy to switch to retention mode, but around 0.1-1nW/b [10] must be supplied to retain the state.

Using volatile SRFF for in-place state retention in processors has been widely researched: In [11], a sub-threshold ARM Cortex-M0+ in 65nm CMOS is presented with 80nW retention power for the core and 4K SRAM. SRFF with protected live slave latches are used for the SRPG of the core. [12] presents a 16b MSP430 CPU based microcontroller in a 90nm technology with an 8KB MTCMOS SRAM and 64KB non-volatile memory. MTCMOS balloon SRFFs are used for the sequential cells. The power dissipation in active mode is 17.5uW/MHz and 31nW at 0.72V for state retentive standby of the core and the SRAM. While the use SRFFs can be easily integrated into standard tool flows by replacing their conventional counterparts [13], they come with a few disadvantages: First, they require the routing of additional control signals to trigger the state saving/restoration to/from the shadow latch. In [14] a wire length increase of 29%-60% is reported for shadow-latch based designs. Second, SRFFs have a higher dynamic power consumption than their conventional counterparts: In [15] an active power overhead of 26% is reported for Ballon SRFF compared to standard FF. [10] reports an overhead of 46-49% for the Ballon SRFF, which matches with our own analog simulations of SRFF standard cells in 40nm of 27% and 42% for a live-latch and balloon SRFF respectively. By reusing the scan chains to implement an out-of-place state saving, as we propose, the conventional scan-flip-flops are not replaced and thus an increase in active power is avoided. Other representatives of SRFF incorporate a *non-volatile* memory cell like FRAM, which enables to implement in-place non-volatile state saving. While these NV-SRFF need no power for retention, 19.4pJ/b in the case of ferroelectric SRFF need to be spent to write and read to the cell [16] in addition to the introduced overhead.

Out-of-place state saving needs a mechanism to move the state and a retaining memory to move the state to. Dedicated off-chip memories are a bad choice due to the energy dissipated in the IO pads required to move the data out of the chip. Thus a retentive on-chip memory is required. Non-volatile memories provide zero power retention but have generally considerable high access energies, reliability issues and technology restrictions. Up to now, only FRAM provides comparable performance to SRAM and manageable reliability. Thus, we only consider FRAM-based non-volatile memories in our comparison. However, FRAM technology access is limited and it could not be yet be scaled further than 130nm to advanced and more energy efficient technology nodes.

If an tiny auxiliary supply is available for retention, as we propose, *volatile* ultra-low leakage SRAM memories are usable, which are compatible to standard CMOS processes. In [6] a 7ns access-time 25uW/MHz 128kb SRAM in 65nm CMOS is presented with a density of 3.46um2/b and 27fA/b retention current at 1.2V. Similarly, [11] describes an ultra low voltage SRAM built from thin-oxide devices with 1.83pW/b leakage at 250mV with a density of 3.64um2/b.

A few implementations using out-of-place state retention are described in literature: In Hibernus [17] a OoP software state saving implementation is presented based on the Texas Instruments MSP430 with integrated FRAM. Prior to entering a state-loosing deep sleep power mode, the 524B of core registers are written to the internal FRAM by an interrupt routine triggered as a warning to an imminent power supply loss detected by monitoring the supply voltage. Moving data in software to FRAM is energy expensive: 2.7nJ/Byte are reported to store the registers in FRAM. With dedicated hardware support, as proposed in our work, this cost can be reduced. A pure hardware out-of-place state saving is demonstrated in [4], which features a SoC with a Cortex-M0 core, which can save/restore the system state (2537 FF) in

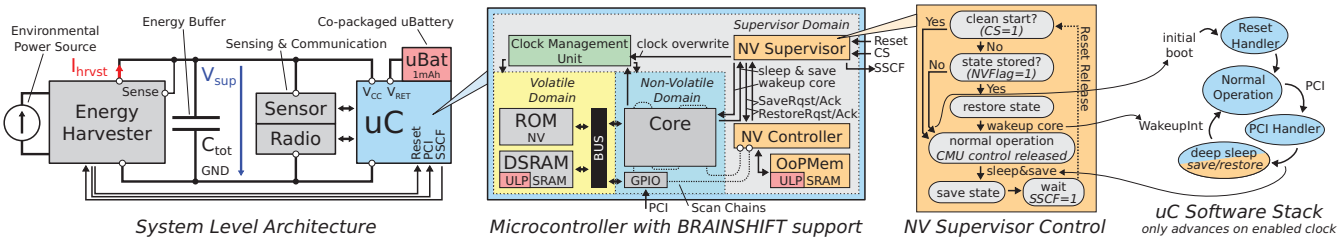

Fig. 1. The system level architecture of our exemplary IoT system with the microcontroller architecture supporting BRAINSHIFT and the main state diagrams.

320ns/384ns using 4.72nJ/1.34nJ to 10 distributed 256b non-volatile ferro-electric mini-memory-arrays. 256FF are connected in parallel over a 8:1 mux to the 32b mini-memory interface. Each FF was extended with an additional data in and enable port for state restoration, which adds an active power overhead of 6.9% to the FF. The overhead due to the inserted mux is not reported. Introducing overhead with the added HW for OoP was not avoided in this work.

The Synopsys ARM Leakage Technology Demonstrator (SALT) [18] implements both in-place state saving with balloon SRFF for *light sleep* as well as a scan-based out-of-place state saving for *deep sleep* for an ARM926 SoC. Their *scan-hibernate* uses 32 scan chains to shift the core state onto the system bus to store it on any on/off-chip memory. While *scan-hibernate* is similar to our proposed solution, active power overheads are not avoided as in-place retention is supported as well. Detailed comparison is impossible as relevant numbers are not reported in [18]. As the main purpose of SALT is to demonstrate aggressive leakage mitigation techniques, it does neither support it nor provide the required hardware functions to operate it on intermittent energy availability.

III. System Level Architecture

Fig.1 depicts the system level architecture we use for our case study of BRAINSHIFT. It consist of three parts: 1) An energy harvester, which extracts energy from the environment, 2) a microcontroller with BRAINSHIFT support responsible for processing and 3) some sensor and radio peripherals to interact with the outside world. The energy to operate the system is harvested from an unsteady power source. This may be a kinetic (vibration, wind), solar, or thermal source. Due to system cost and form factor constraints a large battery is not affordable and thus the system operates only when sufficient power is harvested. A tiny microbattery supplies a few nA to retain the state in the uC when no energy can be harvested. For active mode, the only available energy storage elements are buffer and decoupling capacitors on the supply net. The systems operation profile is shown in Fig. 2, which is similar to [17]. When sufficient energy is available, the harvester outputs a current, which charges the capacitance of the supply net (C_{tot}). If the supply voltage V_{sup} surpasses a threshold $V_{T,\text{on}}$, set high enough to operate the system, the harvester releases the reset. This assures that the system does not start preliminary when the V_{sup} is insufficiently high. The system can now start to acquire and process data. However,

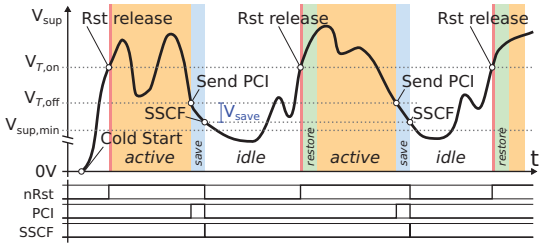

Fig. 2. Exemplary supply voltage over time for our IoT system and the system-level control signals below. The microcontoller is turned on when sufficient energy has been harvested to lift the V_{sup} above $V_{T,\text{on}}$. When the supply drops below $V_{T,\text{off}}$, the state is saved and the microcontroller is turned off. When the supply is up again, the state is restored and operation is resumed.

input power can be lost at any time. The capacitance on the supply net then provides a small energy buffer, which gives the microcontroller time to react onto the imminent power loss. Thereto, the harvester monitors the V_{sup} and as soon it drops below a threshold $V_{T,\text{off}}$, a power critical interrupt (PCI) is sent to the microcontroller. When the microcontroller has saved its state it asserts the State Save Complete Flag (SSCF) and the harvester reasserts the reset to keep the system in a defined state until the supply voltage reaches again $V_{T,\text{on}}$.

IV. BRAINSHIFT

The BRAINSHIFT protocol is summarized as follows: If power gets critical and the PCI is triggered, the microcontroller is first put to sleep before its state is being extracted with the scan-chains and saved in a state retentive memory. If power is available again and the reset has been released, the state is restored with the scan chains, before the microcontroller is woken up to resume its operation.

In Fig. 1, we depict the chip-level architecture of the microcontroller, which we extended with BRAINSHIFT. The selected microcontroller architecture consists of simple processor core, a bus, a instruction ROM, a data SRAM (DSRAM) and a few GPIO with interrupt support. By putting the microcontroller to sleep before the state is saved, we bring it into a defined state, in which all bus transactions between the core, the memories and the peripherals are completed. Then data consistency is assured and all registers in the bus no longer hold relevant data. We divide the microcontroller in two domains, the non-volatile domain (NVD) and the volatile domain (VD) depending on whether the registers in that domain are saved or not. In our case, we place the system bus in the VD. By doing so, we remove many non-architectural registers containing no relevant

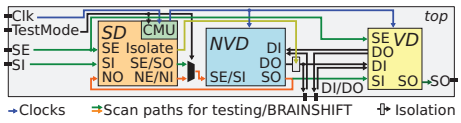

Fig. 3. Top view of the scan chain architecture with the required glue logic .

state from the state being saved. However, there are still a few non-relevant registers, like pipeline registers, found within the NVD. They could be identified and easily removed from the state being saved by specifying the NVD at a finer granularity.

For BRAINSHIFT three major components are added: The NV-Supervisor (NVS), which is the main state machine for BRAINSHIFT, an ULP SRAM (OoPMem) to save the state out-of-place and the NV-Controller (NVC), which controls the scan chains for state extraction and insertion and provides a centralized interface to the OoPMem. All are placed into a supervisor domain (SD), which is not saved. Furthermore the DSRAM is replaced by a ULP SRAM, which, as the OoPMem as well, retains its state during global power off by switching to the auxiliary retention battery supply V_{BAT}.

Fig. 1 shows the control flow of the NVS: After reset release, it is checked whether a previous state is stored. If none is stored, the microprocessor is normally booted, i.e. its clock is activated and the reset handler is executed. As soon as a power loss is imminent the harvester triggers the PCI. The PCI is handled by the microcontroller in software instead of directly activating the hardware state saving mechanism such that the sleep mode can be properly activated in order to bring the system into a defined state. Moreover, by properly configuring the interrupt controller, critical program sections can be protected from interruption and by extending the PCI handler, custom code can be executed before the system is saved and shut down. Once the microcontroller is sleeping, the NVS overwrites the controls of the clock management unit and disables all clocks to freeze the state of the microcontroller. The core can no longer wake up. Next, the NVC saves the state of the NVD: It enables the scan chains and the according clocks and shifts out the state into the OoPMem.

If the saving procedure is completed, the NVS sets the state-save-complete-flag (SSCF) to signal the harvester to cut the global power supply. On the next reset release, the NVS requests the NVC to check for an existing state in the OoPMem. Since there is now a state available, the microcontroller is not booted, i.e. its clock is not activated and thus the Reset Handler is not executed. Instead the NVC controller shifts back the state into the NVD. Once the entire state is restored, the NVS enables the clock of the microcontroller by releasing the clock freeze overwrite signal. The core is now able to detect the internal wake-up signal, which is sent by the NVS. The core wakes up and executes the wakeup interrupt handler. If desired, re-configuration code of the peripherals can be executed or any other custom code before the core resumes the operation from the point where it was interrupted by the PCI. To guarantee proper state saving $V_{T,\text{off}}$ needs to be set such that there is sufficient energy stored on the supply net capacitance to complete the save procedure before the minimal operation voltage $V_{\text{sup,min}}$ of the microcontroller is reached and the supervisory reset mechanisms in the harvesters is activated, which also switches the power supply of the ULP SRAMs to the external battery. To force a clean start, the CleanStart (CS) port multiplexed with a GPIO pin can be asserted during reset release to a reboot irrespective of an existing stored state.

A. Scan Chain Architecture

To keep the active power overhead low, BRAINSHIFT reuses the scan-chain that have to be inserted for testing. The required hardware blocks of BRAINSHIFT (NVS, NVD, OoPMem) are inserted on RTL and connected to the microcontroller (Fig. 1). After synthesis during DFT insertion, scan chains partitions are inserted for SD, VD and NVD, and connected to hookup pins provided by the NVC using automatic scan insertion. Fig. 3 depicts the required glue logic to switch between functional and test usage. Note that during shift, critical NVD outputs are silenced with a logic isolation gate to avoid toggle propagation.

It is not a-priory clear how many scan chains should be used in parallel to extract the state. Since having no test structures is not an option for a real design, we define a design with one scan-chain as the baseline solution. We assume a full-scan design, meaning that all N registers in the NVD are in the scan chain. We define the active power overhead P_{oh} as the additional average power spend during active operation compared to the baseline design. The power dissipation during shift is primarily caused by the toggling in the scan registers, which is propagated into the logic. To reduce the toggling, we shift a constant logic value into the scan chains, while shifting the state out. Consequently, $K(K+1)/2$ state transitions occur, when the state of a scan chain of length K is shifted out. The same number of transitions is achieved for restore, if all register are reset before shift. No scan silencing technique, i.e. latch or nor insertion at the register output, is used to suppress the toggling propagation, as active power overheads of 31%-153% and 3%-75% respectively are reported in [19].

Using Q scan-chains in parallel enables to save/restore the state Q times faster, given that the same clock frequency is used. Moreover, the total number of state transitions $T_N(Q) = Q \cdot \lceil N/Q \rceil (\lceil N/Q \rceil + 1)/2$ over all scan chains is reduced by $T_N(1)/T_N(Q) \approx Q$ assuming $N \gg Q$. Consequently we expect a dynamic energy reduction of Q as well. In the most extreme case $Q = N$, each register has a direct connection to the OoPMem. Inserting so many scan chains will complicate the routing, increase the overall netload and ultimately increase active power consumption. Ideally, the optimal number of scan chains is obtained by sweeping Q and performing a power simulation. Which is what we do in the results section. Since this is not practical for larger design, we use, as a rule of thumb, $Q \approx \sqrt{N}$ scan chains for state extraction in order to have a reasonable trade-off between reduction of save and restore energy, and introduction of active power overhead.

V. RESULTS

The functional verification and post-layout power estimation of BRAINSHIFT was done on PULPino[1], an open microcon-

[1]Available on http://www.pulp-platform.org/

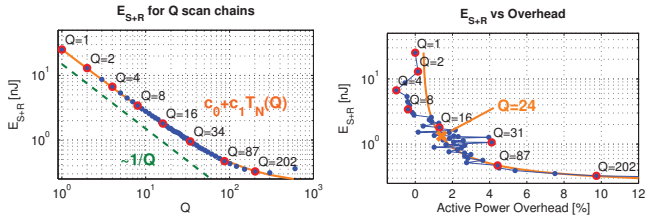

Fig. 4. The total save and restore energy $E_{\text{S+R}}$ (blue dots) for different number of scan chains Q in our $N = 604$ test circuit. The left plot shows $E_{\text{S+R}}$ depending on Q and the right one relates $E_{\text{S+R}}$ with the introduced active power overhead P_{oh} for different Q. The trend lines are in solid orange.

troller platform based on a RISC-V instruction set processor core. The RISC core along with several peripherals (GPIO, Interrupt Controller) are defined as the NVD ($N = 3222$), i.e. saved and restored using scan chains. The remainder of the microcontroller, i.e. the buses (AXI, APB) along with SPI, UART peripherals, were put into the VD, which means they are either not saved, since they do not contain relevant information, or in the case of the peripherals, they can be properly shut down and saved using software functions if the peripheral is in use. Synopsys Design Compiler 2014.09 was used for synthesis. Questa Sim 10.3a by Mentor Graphics Corp. was used for verification and post-layout simulation. Backend design and power simulations were performed with Cadence Innovus 15.20. All evaluations were done using a 65nm standard library (typical, 1.2V). The retention in the ULP SRAM was emulated by not applying a reset signal to the according memories. Two chip designs were made based on the PULPino platform: one to simulate software state saving as a baseline implementation to quantify the introduced active power overhead, and one implementing BRAINSHIFT.

The optimal number of scan chain to insert for state extraction was determined with a design space exploration (Fig. 4) on a simpler test circuit to facilitate full-automated post-layout simulations. A linear feedback shift register running at 10MHz was used with its outputs fed to a combination processing pipeline (modulo, look-up-table and xor operations). Using one scan chain (Q=1), the total energy required to save and restore the state ($E_{\text{S+R}} = E_{\text{S}} + E_{\text{R}}$) is 25nJ (41pJ/b). The average active power is 0.333mW, which is our active power baseline. Fig. 5 shows the power trace for Q=32. Note that unlike testing, power is lower during shifting than normal operation (see figure text). For a design with Q=202 with only 3 registers per chain, the $E_{\text{S+R}}$ goes below 0.33nJ (0.5pJ/b), but the overhead increases to 10%. The $E_{\text{S+R}}$ follows nicely our proposed $T_N(Q)$ with a constant offset. The larger Q, the lower the $E_{\text{S+R}}$, but at a growing overhead. Note the overhead figures are quite noisy, thus a trend line $P_{\text{oh}}(Q) = c_0 + c_1/Q$ has been fitted. With our rule of thumb, we get $Q = \sqrt{N} \approx 24$, which corresponds to an $E_{\text{S+R}}$ of 1.33pJ/b with an introduced P_{oh} of 1.42%. The reported test coverage is still at 99.93% by using only ATPG test vectors and no functional tests.

The reference software implementation operates similar to the BRAINSHIFT protocol: When the power critical interrupt (PCI) is triggered the implicit context switch for the interrupt

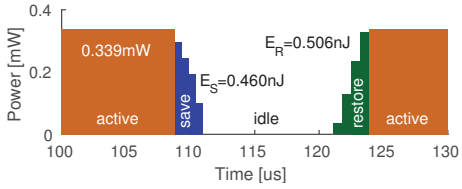

Fig. 5. Simulated power trace of one save/restore cycle with Q=32 on our test circuit. Note the drop in power consumption during state extraction (save) as zeros are shifted into the circuit causing the toggle to die out. The opposite profile can be observed during restore as the state is shifted in after reset.

handling pushes part of the context onto the stack. The handler then copies the remaining state information into a retaining memory and puts the processor to sleep waiting for the power to go out. When power is restored, the microcontroller reboots and the reset handler is executed, which checks if a previous state was stored. If so, the state is restored and the task is resumed by switching back to the context before the PCI was triggered. For the hardware implementation, $Q = 64$ scan chains were inserted, following our rule of thumb $Q = \sqrt{N} = 57$ and rounding the value up to match two 32b memories. The E_{S} and E_{R} excluding the OoPMem power consumption obtained from a post-layout power simulation running at 50MHz were 4.4nJ (1.4pJ/b) and 3.1nJ (1.0pJ/b) respectively, which was a 6.6x improvement in energy (5.8x in time) compared to the software state saving. The state was saved in 1.61us (81 cycles) and restored in 1.2us (60 cycles). Only 51 cycles are required for shifting. The remaining cycles are used for interrupt (PCI) handling during saving and for flag checking and wake-up during restore. The active power consumption while executing a sha256 calculation without pads was 3.28mW, both for the microcontroller extended with BRAINSHIFT and the reference implementation performing software state retention. Thus we conclude that the introduced power overhead is negligible, or in any case below the accuracy limit of post-layout power analysis.

So far the energy spent in the OoPMem was not considered. Given the active power consumption of a state of art ULP SRAM of 25uW/Mhz [6] another 2pJ/b (6.6nJ) is spent for the memory during the 2x51 active cycles required to write and read the state. Finally, we expect a save and restore energy $E_{\text{S+R}}$ of 4.5pJ/b (12.15nJ) for our scan-based out-of-place state saving solution. Moreover, given literature values, a retention power of 0.03-1.83pW/b @ 65nm and an area of 3.46-3.64um2/b @ 65nm [6], [11] is expected, which results in a retention time of over 2 years assuming 10kbit are retained using a tiny 50uAh (3.8V) solid state bare die battery [8].

VI. DISCUSSION & COMPARISON

Table I compares our results with related work. The FRAM-based state saving implementations, (OoP-FNV) [4] and (InP-FNV) [16] report similar transition costs compared to ours. While there is a clear indication that a power overhead is introduced in the OoP-FNV implementation, the overall power overhead is not reported for both. Even if no power overhead were introduced, our method is competitive as long as an ULP SRAM can be used for retention, as our approach does not

TABLE I
COMPARISON WITH RELATED WORK (CORE ONLY)

System	E_{S+R}	P_{ret}	P_{oh}	Tech.
ours software OoP-V	15.2 pJ/b	1pW/b	0%	65nm
ours hardware OoP-V	4.5 pJ/b	1pW/b	≈0%	65nm
[11] InP-V[a]	0	n/a	n/a	65nm
[12] InP-V[a]	0	n/a	n/a	90nm
[16] InP-FNV	19.4pJ/b	0	n/a	130nm
[4] OoP-FNV	2.9pJ/b	0	6.9%[b]	130nm

[a] For InP-V approaches we assume a de-facto zero E_{S+R}.
[b] Reported only for FF

rely on non-standard technology options (such as embedded FRAM), which often are not portable to scaled and more power efficient technology nodes. Both InP-V implementations provide a defacto $E_{S+R} = 0$ [11], [12], but they introduce significant active power overheads to the FF of 27%-49% as previously noted. In addition, P_{ret} of volatile SRFF is orders of magnitudes worse compared to ULP SRAMs (below 2pW/b @ 65nm [6], [11]): In our simulations of SRFF standard cells we got 88 and 108pW @ 1.1V 25C in 40nm for the live and balloon SRFF. [10] report 1.3-1.5nW @ 0.9V 25C in 80nm for the balloon SRFF. Thus we conclude that compared with our approach, using SRFFs results in lower transition cost, while P_{ret} and overheads are higher.

To finally determine which state saving method is better suited for an IoT system operating on intermittent energy, we compare the total energy E_{cyc} consumed in one power cycle. A power cycle of time t_{cyc} contains an active time of $t_{on} = Dt_{cyc}$, which is interrupted by a state-retaining power down.

$$E_{cyc} = E_{S+R} + t_{cyc}\left((P_{on} + P_{overhead})D + P_{ret}(1-D)\right) \quad (1)$$

Figure 6 shows a comparison between InP-V in general and our OoP-V state saving of a system with $N = 1000$ and $P_{on} = 3$mW. We observe that for an operation profile with an on-time below 100us (5000 cycles at 50MHz) and power cycles occurring more frequently than 5Hz, InP-V requires the lowest energy. This is due to low transition costs and the little impact of active power overhead when on-times are brief. In all other scenarios, i.e. longer on times and less frequent power cycles, our out-of-place state saving requires less energy overall. Moreover, our hardware OoP-V has always a lower E_{cyc} irrespective of the operation profile compared to our software OoP-V as it provides a 6.6x lower E_{S+R} while introducing no overhead and requiring the same retention power. We thus conclude that an OoP-V state saving approach is best for an IoT system operating on intermittent energy.

VII. CONCLUSIONS AND FUTURE WORK

In this work, we have presented a light-weight method to save the state of a microcontroller using scan chains and compared it to other state saving methods considering their suitability for an IoT system operating on intermittent energy. Compared to volatile in-place state saving, our method takes more energy for save- and restoring, but introduces no active power overhead and achieves a lower retention power. Consequently, our method requires less energy in scenarios

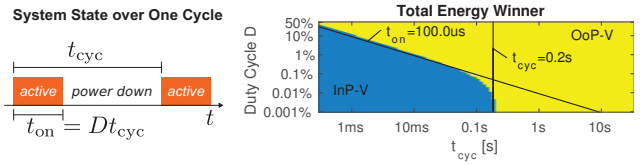

Fig. 6. Comparison InP-V vs OoP-V: The left plot shows the system state during one power cycle of t_{cyc} with t_{on} active time. The right plot compares qualitatively volatile in-place (InP-V) with out-of-place (OoP-V) state saving. The color is chosen according to the method using less energy E_{cyc} over complete power cycle. We set for the two methods (InP-V, OoP-V) E_{S+R} to (0pJ/b, 6pJ/b), P_{ret} to (1pW/b, 30pW/b) and P_{oh} to (2%, 0%) respectively.

with on-times longer than 100us and power losses occurring less frequently than 5Hz, which is the scenario most IoT systems are operating in. Moreover, our method surpasses software approaches, as less energy is required to move the state, while providing equivalent performance otherwise. In the next step, we will validate our results in silicon.

REFERENCES

[1] Texas Instruments, "CC2650 - SimpleLink Wireless MCU," 2015.
[2] Y. Zhang *et al.*, "A batteryless 19uW MICS/ISM-band energy harvesting body sensor node SoC for ExG applications," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 199–213, 2013.
[3] A. Gomez, L. Sigrist, M. Magno, L. Benini, and L. Thiele, "Dynamic Energy Burst Scaling for Transiently Powered Systems," *Des. Autom. Test Eur. Conf. Exhib. (DATE), 2016*, pp. 349–354, 2016.
[4] S. C. Bartling *et al.*, "An 8MHz 75μA/MHz zero-leakage non-volatile logic-based cortex-M0 MCU SoC exhibiting 100% digital state retention at VDD=0V with <400ns wakeup and sleep transitions," *IEEE Int. Solid-State Circuits Conf.*, vol. 56, pp. 432–433, 2013.
[5] Ambiqmicro, "Apollo Datasheet ULP MCU Family," 2015.
[6] T. Fukuda *et al.*, "13.4 A 7ns-access-time 25uW/MHz 128kb SRAM for low-power fast wake-up MCU in 65nm CMOS with 27fA/b retention current," in *IEEE Int. Solid-State Circuits Conf.*, 2014, pp. 236–237.
[7] G. Chen, H. Ghaed, and R.-u. Haque, "A cubic-millimeter energy-autonomous wireless intraocular pressure monitor," *IEEE Int. Solid-State Circuits Conf.*, pp. 138–139, 2011.
[8] CYMBET, "EnerChip Bare Die Datasheet," 2014.
[9] S. Shigematsu, S. Mutoh, Y. Matsuya, Y. Tanabe, and J. Yamada, "A 1-V high-speed MTCMOS circuit scheme for power-down application circuits," *IEEE J. Solid-State Circuits*, vol. 32, no. 6, pp. 861–869, 1997.
[10] Z. Liu and V. Kursun, "New MTCMOS Flip-Flops with Simple Control Circuitry and Low Leakage Data Retention Capability," in *IEEE Int. Conf. Electron. Circuits Syst.*, 2007, pp. 1276–1279.
[11] J. Myers *et al.*, "8.1 An 80nW retention 11.7pJ/cycle active subthreshold ARM Cortex-M0+ subsystem in 65nm CMOS for WSN applications," in *IEEE Int. Solid-State Circuits Conf.*, feb 2015, pp. 144–146.
[12] V. K. Singhal, V. Menezes, S. Chakravarthy, and M. Mehendale, "A 10.5uA/MHz at 16MHz single-cycle non-volatile memory access microcontroller with full state retention at 108nA in a 90nm process," in *IEEE Int. Solid-State Circuits Conf.*, feb 2015, pp. 148–149.
[13] D. Flynn, "Power gating applied to MP-SoCs for standby-mode power management," *Proc. 50th Annu. Des. Autom. Conf. - DAC '13*, 2013.
[14] J. Seomun and Y. Shin, "Design and Optimization of Power-Gated Circuits With Autonomous Data Retention," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 2, pp. 227–236, 2011.
[15] H. Jiao and V. Kursun, "Low-Leakage and Compact Registers with Easy-Sleep Mode," *J. Low Power Electron.*, vol. 6, no. 2, pp. 263–279, 2010.
[16] Y. Wang *et al.*, "A Compression-based Area-efficient Recovery Architecture for Nonvolatile Processors," *Proc. Conf. Des. Autom. Test Eur.*, pp. 1519–1524, 2012.
[17] D. Balsamo *et al.*, "Hibernus : Sustaining Computation during Intermittent Supply for Energy-Harvesting Systems," *IEEE Embed. Syst. Lett.*, vol. 7, no. 1, pp. 1–4, 2015.
[18] S. Idgunji, "Case study of a low power MTCMOS based ARM926 SoC: Design, analysis and test challenges," *IEEE Int. Test Conf.*, pp. 1–10, 2007.
[19] S. Bhunia, H. Mahmoodi, D. Ghosh, S. Mukhopadhyay, and K. Roy, "Low-power scan design using first-level supply gating," *Very Large Scale Integr. Syst. IEEE Trans.*, vol. 13, no. 3, pp. 384–395, 2005.