DarkMem: Fine-Grained Power Management of Local Memories for Accelerators in Embedded Systems

Christian Pilato

Università della Svizzera italiana
Lugano, Switzerland
e-mail: christian.pilato@usi.ch

Luca P. Carloni

Columbia University
New York, NY, USA
email: luca@cs.columbia.edu

Abstract— SRAM consumes a growing fraction of the static power in heterogeneous SoCs, as embedded memories take 70% to 90% of the area of specialized accelerators. We present DarkMem as a comprehensive solution for fine-grained power management of accelerator local memories. The DarkMem methodology optimizes at design time the bank configuration for each given accelerator to maximize power-gating opportunities. The DarkMem microarchitecture dynamically varies the operating mode of each memory bank according to the accelerator workload. In our experiments, DarkMem reduces the SRAM static power by more than 40% on average, which translates into a reduction of the total power by almost 18% on average with less than 1% overhead.

I. Introduction

The power-dissipation density and the utilization wall, two major concerns in integrated circuit design [9], have led to the emergence of heterogeneous system-on-chip (SoC) architectures. In these architectures, general-purpose processors and several *specialized accelerators* are combined through a bus or network-on-chip [6, 13], as shown in Fig. 1(a). In this work, we focus on configurable, non-programmable accelerators designed for high-throughput data processing [6, 7, 8], which feature a specialized microarchitecture that is optimized for one specific functionality. For example, *Debayer* is an accelerator for computer-vision applications [1]. It can be invoked by an application (by writing proper values onto its *configuration registers*) to process an image of thousands of pixels, which corresponds to megabytes of data. When unused, it can be turned off, thus alleviating the power density challenge.

A key component of each accelerator is its *private local memory* (PLM) [8, 12]. The PLM organization typically features many multi-ported SRAM banks to support fast, highly parallel, and fixed-latency access from the accelerator datapath logic. Even though the PLM takes most of the accelerator area [8, 12], it implements high-level data structures (e.g., arrays) that correspond only to a fraction of the data processed by the accelerator (e.g., only few pixel rows of the input image), while the rest is stored in DRAM. When the accelerator is active, the PLM continuously exchanges data with main memory (DRAM) through direct memory access (DMA) transfers, potentially creating congestion on the SoC interconnects in case of multiple components executing concurrently [4]. This situation creates opportunities for dynamic power savings with the help of voltage/frequency scaling (DVFS) techniques [13]. However, the SRAM static power can be responsible for up to 40-50% of the total power consumption of the system (Fig. 1(b)) and must be addressed with additional and specific solutions [10, 15, 17].

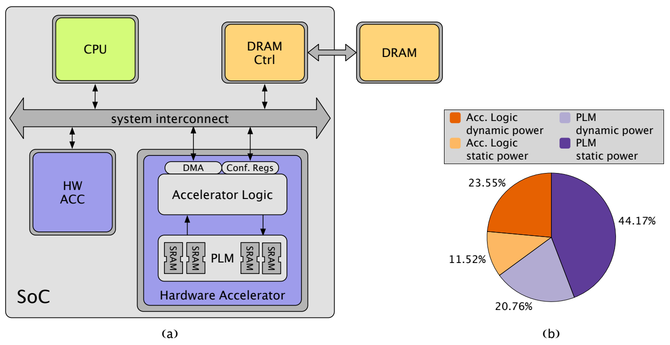


Fig. 1. Example of a heterogeneous architecture (a); each accelerator features a private local memory composed of many SRAM banks, which are responsible for a large fraction of its power dissipation (b).

DarkMem, our approach for reducing the PLM static power, is based on three main observations. First, SRAM memories are increasingly fabricated with the possibility of selectively turning off the peripheral circuitry (to be kept active only when effectively accessing the data) or also the memory cells (to be kept active to retain the data) [5]. These *dual-rail SRAMs* have been used in processors [14], GPUs [23], or application-specific systems [20] with only two additional sleep transistors between each SRAM bank and the ground [19]. Then, we notice that, even if an accelerator is meant to execute only one application, it is usually designed with a certain degree of configurability that broadens its applicability across a few different execution scenarios (e.g, processing images of different sizes and resolutions). When the accelerator is configured to operate on a subset of the data, part of the PLM is not used and we can eventually apply power gating to the unused SRAM banks (*scenario-based optimization*). At design time we thus determine the PLM organization that maximizes the number of its banks that can be power gated in each configuration. Finally, the computation and communication phases of any given accelerator are rarely perfectly balanced in every execution due, for example, to competition for shared resources [13]. So, the memory cells of its PLM can be idle for extended periods.

Hence, we introduce some logic inside the PLM to vary the operating modes of the active banks based on workload conditions and save additional power (*workload-based optimization*). This logic turns off banks that are not containing valid data or selectively turns off the peripheral circuitry when data are valid but are not expected to be accessed for a long time. Voltage controllers can also reduce the supply voltage of the memory cells down to the standby limit [18]. The resulting microarchitecture can be synthesized by extending a traditional design flow for accelerators. This solution enables *fine-grained power management* of the accelerator's PLM with only minimal changes to its high-level specification.

Contributions. The three main contributions of this paper can be summarized as follows:

- a novel accelerator microarchitecture, which performs fine-grained power management of the accelerator private local memories (Section II);
- a design methodology based on high-level synthesis (HLS) to automatically determine the proper bank configuration of the different PLM units and generate such microarchitecture (Section III);
- a sensitivity analysis on the accelerator power consumption when varying the SRAM static power and the accelerators' execution modes (Section IV).

With our methodology we improved the design of eight complex accelerators for selected kernels from two benchmark suites (PERFECT [1] and CORTEXSUITE [22]). The experimental results show that DARKMEM can reduce the total power consumption of accelerators by almost 18% on average with a performance overhead less than 1%.

II. DARKMEM ARCHITECTURE

The DARKMEM microarchitecture aims at minimizing the SRAM static power by selectively applying power gating to the periphery circuitry and the memory cells of each PLM bank based on the accelerator execution. We assume that the memory library includes dual-rail SRAMs that can be used to find the proper implementation of each PLM data structure to be stored in the PLM. Each SRAM bank has thus two power-related pins (PGL and PGM) to control the power gating of the periphery circuit and memory cells, respectively. These are sufficient to support three operating modes:

- *active*: the entire SRAM is active (i.e., there are active memory operations);
- *idle*: both the peripheral circuitry and the memory cells are power gated (i.e., there are no valid data on the memory cells and no active operations on them);
- *deep-sleep*: the peripheral circuitry is power gated since there are no active memory operations, while the memory cells are active to retain the data.

Fig. 2 shows our proposed DARKMEM microarchitecture, where the accelerator logic includes the hardware blocks to perform the actual computation and manage the data transfers with DRAM, and the PLM contains multiple DARKMEM units, one for each data structure accessed by the accelerator logic. Each DARKMEM unit includes additional modules for power management of the SRAM banks based on the PLM configuration and commands received from the accelerator logic.

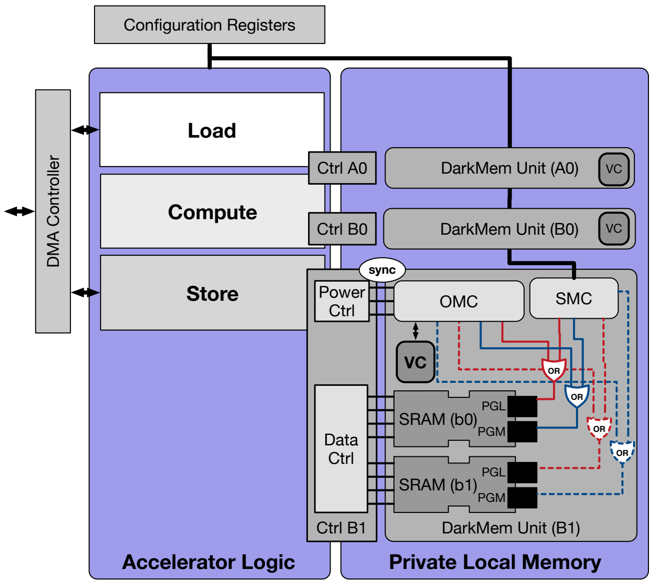


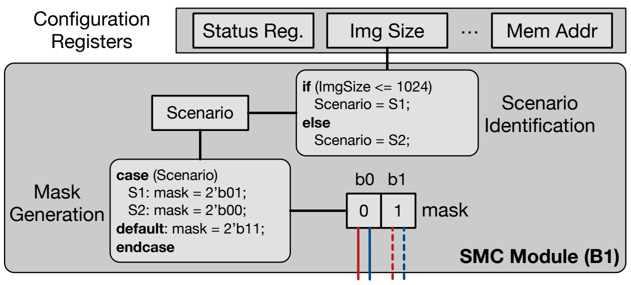
Fig. 2. High-level organization of the DARKMEM accelerator for *Debayer*; each DARKMEM unit implements a data structure. Each DARKMEM unit includes the proper SRAMs (black boxes represent the power-gating pins) and the logic for power management; it is also connected to a module for controlling the supply voltage (VC).

To further reduce the SRAM power consumption, the DARKMEM architecture can be combined with voltage controllers to lower the supply voltage of the memory cells down to the *data retention voltage* (DRV) value. To this end, each DARKMEM unit can feature a voltage controller (*VC* in Fig. 2), which can be implemented with a *bias generator* or an *integrated voltage regulator* (IVR) and integrated on the same die or with 2.5D chip stacking [11]. If technology or design constraints limit the number of voltage controllers per accelerator, the same component can be shared across multiple DARKMEM units as long as the power management logic keeps the highest voltage that is requested at any given time by the SRAMs.

Fig. 2 also shows the microarchitecture of each DARKMEM unit, which consists of three main modules:

- the *SRAM banks*: based on the technology information (e.g., available SRAM sizes, static power, etc.), we can implement the corresponding array with different dual-rail SRAMs transparently to the accelerator execution [16];
- the *scenario memory controller* (SMC): based on the values of the configuration registers, the SMC module determines at the beginning of the execution which banks are not necessary and, therefore, they can be power gated for the entire execution of the accelerator (i.e., until it is configured to start a new execution with different parameters);
- the *operating mode controller* (OMC): based on a set of command signals from the accelerator logic, the OMC module determines the SRAM operating modes (i.e., when to apply power gating to the periphery circuitry or also to the memory cells).

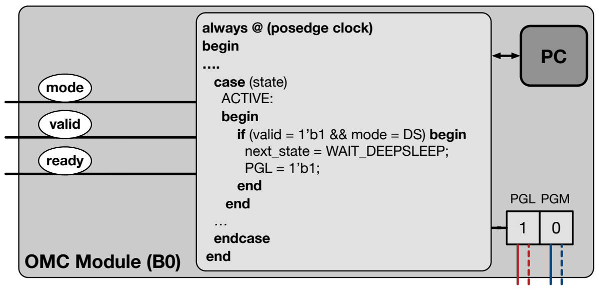
The results of these two controllers are then combined with OR gates to determine the actual power management for each single bank. The resulting design is highly modular and we can apply each of the optimizations selectively. The interface between the accelerator logic and each DARKMEM unit consists of two memory controllers: *Data Ctrl* is used to perform read-



```
if (ImgSize <= 1024)
  Scenario = S1;
else
  Scenario = S2;
```

```
case (Scenario)
  S1: mask = 2'b01;
  S2: mask = 2'b00;
default: mask = 2'b11;
endcase
```

Fig. 3. Microarchitecture of the SMC module.



```
always @ (posedge clock)
begin
....
  case (state)
   ACTIVE:
   begin
     if (valid = 1'b1 && mode = DS) begin
       next_state = WAIT_DEEPSLEEP;
       PGL = 1'b1;
     end
   end
  ...
  endcase
end
```

Fig. 4. Microarchitecture of the OMC module.

/write operations on the data and it is the same controller used in traditional microarchitectures, while *Power Ctrl* is our extension, which is used to provide information on the status of the execution phases and determine the SRAM operating modes.

Scenario Memory Controller. Once the accelerator is configured by the CPU, the DARKMEM controller identifies which is the current scenario (among the ones defined at design time) and determines which banks are not required during the corresponding execution. Hence these banks can be then power gated. As shown in Fig. 3, each SMC module receives the configuration registers as input to determine the current execution scenario (see component *Scenario Identification* in Fig. 3). To guarantee a correct computation in all cases, we assume the case where the entire PLM is used as the default scenario. The SMC module produces a corresponding mask as output to force power gating of specific SRAM banks. For example, one of the configuration registers of the *Debayer* accelerator stores the size of the image to elaborate and it is used to determine which scenario will be executed. When elaborating a smaller image (1,024×1,024 pixels), the SMC module determines that the accelerator is operating in the scenario s_1 thanks to the simple control logic shown in Fig. 3. When the accelerator operates in this scenario and two 1,024×32 SRAMs are used to implement array `B1`, the SMC module sets the mask to be "`01`". As shown in Fig. 2, the outputs of the SMC module are combined through `OR` gates with those of the OMC module so that the power pins of the first SRAM are controlled by the output signals of the OMC module (bit mask is `0`), while the second SRAM is always power gated (bit mask is `1`).

The set of scenarios is determined at design time, during the PLM configuration. The definition of these scenarios is based on the expected values of the configuration registers (known for non-programmable accelerators), the corresponding memory requirements, and the expected frequency of execution of these cases. It is worth noting that the configuration registers are modified only when the elaboration is terminated or the accelerator is reset. In both cases, a new execution starts with new values written into the configuration registers. So, each execution is independent from the others.

Operating Mode Controller. The OMC modules determine the operating modes of the SRAM banks based on the execution workload. Consider the example of Fig. 2: process COMPUTE terminates the execution of an iteration after elaborating some rows of the input image. A debayered image row is then contained into array `B1`, but no operations are performed by any other process. So the operating mode of the corresponding SRAM banks can be changed from *active* to *deep-sleep*. Then, process `Store` starts, but the system interconnection is not available (e.g., due to congestion). When the access to the system interconnect is granted, it wakes up the SRAM banks by changing the operating mode from *deep-sleep* to *active*, stalling until the memory becomes available. Finally, when the data transfer is completed and process COMPUTE is still producing data on array `B1`, the SRAM banks can be power gated by entering into mode *idle*. To perform the computation correctly, each OMC module is implemented as a finite state machine (FSM) that communicates with the accelerator logic through three explicit protocol signals (`mode`, `valid`, `ready`), as shown in Fig 4. These signals make sure that the accelerator can continue its execution only when these operations are completed, thus preventing any memory operations from executing during the SRAM transitions [3]. Each OMC module is then connected to the two power pins controlling the power gating of the peripheral circuitry and the memory cells, respectively. If the power management is configured to lower the supply voltage of the SRAMs, each OCM module is connected also to the corresponding voltage controller (i.e, bias generator [21] or IVR [13]). The FSM implements the transitions among the three operating modes considering also the mode-transition latency (e.g., the latency of the sleep transistors or the delay properties of the IVRs). Most of the operating-mode transitions have almost no impact on the performance because they overlap with the execution of the processes.

III. DESIGN METHODOLOGY

We developed a CAD flow for the automatic generation of the DARKMEM microarchitecture by extending the system-level design methodology for *Embedded Scalable Platforms* [4], which uses extensively high-level synthesis (HLS). First, we developed a DARKMEM Application Programming Interface (API), which is a lightweight, synthesizable SystemC library to augment the SystemC description of a given accelerator. With the DARKMEM API, the designer can extend the accelerator description by specifying when each given data structure is used or when it is not accessed but the data must be preserved. This specification corresponds to the operating mode transitions of the associated SRAM banks. Fig. 5 shows part of this specification.

This enhanced SystemC description is synthesized with HLS to produce the synthesizable Verilog description of the accelerator logic. In particular, the synthesis of our DARKMEM API generates the latency-insensitive protocol [3] signals to be interfaced with the OMC modules (see Fig. 4). We then extended MNEMOSYNE [16], a prototype CAD tool to generate

```
void Store() {
  int ping = 0;
  wait();
  while(true) {
     /// perform DMA request
     /// ...
     if (ping) DARKMEM_ACTIVATE(ctrl_B0);
     else DARKMEM_ACTIVATE(ctrl_B1);
     /// store produced results
     /// ...
     if (ping) DARKMEM_IDLE(ctrl_B0);
     else DARKMEM_IDLE(ctrl_B1);
     /// notify compute
     /// ...
     ping = !ping;
  }
}
```

Fig. 5. Example of the use of the DARKMEM API to specify the operating modes when using double buffering.

the synthesizable Verilog descriptions of the different DARKMEM units[1]. MNEMOSYNE takes as input the list of arrays to be stored in the PLM and the technology details of the available SRAMs. For each SRAM module, the designer provides details on its size (i.e., bitwidth and number of words), area, and static power consumption. This information is contained in the technology files given by the SRAM vendor (e.g., Liberty files). The new version also receives the list of execution scenarios and, eventually, technology details on the voltage controllers, such as their transition delays. Each execution scenario contains the corresponding memory requirements, the values for the configuration registers for its identification, and the frequency of execution, which must be estimated by the designer. For any possible scenario MNEMOSYNE can always generate a feasible implementation by executing these three steps: 1) it determines a PLM configuration that minimizes the average SRAM static power across all scenarios; 2) it generates the data controllers to interface the memory interfaces of the accelerator logic with the resulting banks; and 3) it generates the different power controllers, interconnecting them with the corresponding controlling signals of the accelerator logic.

Determining the Bank Configuration. Each PLM data structure can be implemented with different SRAM configurations, provided that they have enough memory space. At design time we define the SRAM configurations to minimize the average SRAM static power with power gating of the unused banks (Eq. 1) based on the information provided by the designer. For the sake of simplicity in the generation of the controlling logic (see below), we use a homogeneous bank organization for each data structure and we model this optimization problem as:

$$min\,(PLM_{static}) = min\left(\sum_{s\in S} PLM^{s}_{static}\cdot freq(s)\right) \quad (1)$$

$$PLM^{s}_{static} = \sum_{b\in B} PLM^{b,s}_{static} \quad (2)$$

$$\sum_{m\in MEM} c^{b}_{m} = 1 \quad (3)$$

$$PLM^{b,s}_{static} = \sum_{m\in MEM} PWR^{b,s}_{m} * c^{b}_{m} \quad (4)$$

$$PWR^{b,s}_{m} = ON^{b,s}_{m} * p^{m}_{active} + (M_b - ON^{b,s}_{m}) * p^{m}_{gated} \quad (5)$$

$$\forall m\in MEM, b\in B : W_m \geq c^{b}_{m} * W^{b} \quad (6)$$

$$\forall m\in MEM, b\in B : H_m * M_b \geq c^{b}_{m} * H^{b} \quad (7)$$

[1]MNEMOSYNE is an open-source tool, available at: https://github.com/chrpilat/mnemosyne

TABLE I
CHARACTERIZATION OF THE *Reference* DESIGNS.

BENCHMARK	PLM UNITS	DATA SIZE (MB) PLM/DRAM	Total Area (mm^2) [PLM]	TOTAL POWER (MW) [PLM STATIC]
FFT-2D	2	0.128 / 64.00	0.83 [95.18%]	36.49 [38.84%]
Debayer	3	0.095 / 16.00	0.69 [98.64%]	33.68 [45.04%]
Lucas Kan.	11	0.020 / 32.00	0.36 [88.27%]	32.90 [20.17%]
Change Det.	10	0.062 / 320.00	1.40 [93.52%]	89.34 [26.18%]
Disparity	11	0.145 / 15.82	2.22 [98.92%]	75.57 [53.07%]
PCA	3	0.117 / 20.19	1.16 [98.23%]	42.48 [60.49%]
RBM	8	0.065 / 3.81	2.91 [99.15%]	52.67 [48.78%]
SRR	32	0.076 / 4.76	1.26 [97.09%]	82.53 [60.75%]

where S is the set of execution scenarios, each of them having frequency $freq(s)$; B is the set of PLM data structures, each of them having size $H^b \times W^b$; MEM is the set of available SRAMs, each of them capable of storing H_m data elements of W_m bits; c^b_m is a decision variable to represent whether the memory bank $m \in MEM$ is selected to implement the array b; M_b is the number of used banks; $ON^{b,s}_m$ is the number of active banks in scenario s; p_{active} and p_{gated} are the static power consumed by these SRAM banks when active or power gated, respectively. PLM^s_{static} is the SRAM static power in each scenario s, obtained by aggregating the power consumed by each data structure (Eq. 2). Eq. 3 is a constraint indicating that only one SRAM type m must be selected for each data structure b. Only the corresponding term $PWR^{b,s}_m$ (Eq. 5) contributes to $PLM^{b,s}_{static}$, that is the PLM static power of the data structure b in scenario s (Eq. 4). Finally, additional constraints are used to verify that, for each data structure b, the selected set of banks can store the entire amount of data (Eqs. 6 and 7) to guarantee feasible implementations.

Generating the Data and Power Controllers. We use a simple data controller for each DARKMEM unit to translate the memory operations performed by the accelerator logic into the proper requests to the physical banks. To this end, we generate some lightweight data controllers that are sufficient to determine which physical bank is addressed based on the given address [16]. For example, when we implement a 2,048×32 data structure with four 512×32 SRAMs, the accelerator logic provides an 11-bit data address and only one of the four banks effectively contains the data. So we use the two most significant bits to select the target SRAM bank and the nine less significant bits as a physical address to effectively access the data inside the selected bank. After configuring the PLM organization at design time, we achieve additional power savings by controlling the operating modes of the active banks in each scenario. The additional DARKMEM modules (SMC and OMC) are generated in Verilog starting from high-level templates. The SMC modules are then connected to the configuration registers, while the OMC modules are connected between the accelerator logic and the power pins of the SRAM banks.

IV. EXPERIMENTAL RESULTS

To evaluate our DARKMEM approach, we designed eight accelerators for selected applications taken from the PERFECT [1] and CORTEXSUITE [22] benchmark suites.

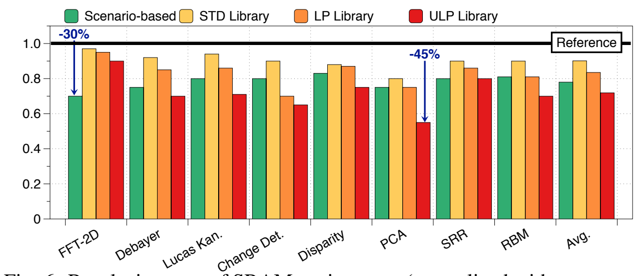


Fig. 6. Results in terms of SRAM static power (normalized with respect to the reference designs) when using only the scenario-based optimization (bars in green) and when using only the workload-based optimization with three different libraries (bars in the three shades of red).

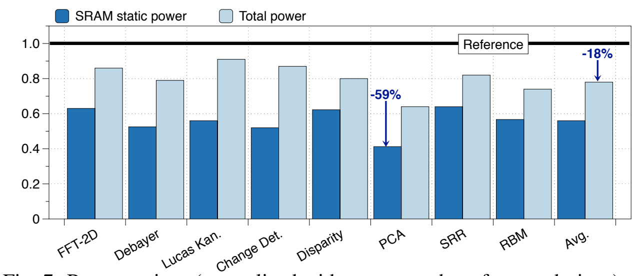


Fig. 7. Power savings (normalized with respect to the reference designs) when combining the two optimizations.

Experimental Setup. We designed each accelerator in synthesizable SystemC and we used Cadence C-to-Silicon (ver. 14.2) for synthesizing the Verilog description of the accelerator logic and our initial version of MNEMOSYNE [16] for determining the reference PLM configurations. We used an industrial 32nm CMOS technology with a memory library containing 18 distinct SRAMs, and we targeted a 1GHz clock frequency. Table I shows the characterization of the *Reference* designs (Figs. 6 and 7) in terms of PLM data structures (PLM UNITS), size of the data stored in PLM and DRAM, area occupation, total power consumption, and the fraction due to SRAM static power without using DARKMEM.

We then updated these SystemC designs with our DARKMEM API to specify the execution modes of each data structure. We used HLS for generating the updated accelerator logic and the DARKMEM-extended version of MNEMOSYNE [16] for creating the DARKMEM units. For each accelerator, we created two scenarios with different memory requirements for the data structures. We then tested three representative cases for the memory library:

- STD: we use only voltage controllers, reducing the deep-sleep leakage by 40%;
- LP: the memory library has dual-rail SRAMs; the deep-sleep mode reduces the static power by 70%;
- ULP: we use dual-rail SRAMs and voltage controllers, reducing the deep-sleep leakage by 85% [18].

In dual-rail SRAMs, the power gating of the entire SRAM (*idle* state) reduces the static power by 95% in all cases. These alternative cases are modeled by creating SRAM library files with different static power characterizations based on the given values of the power pins PGL and PGM.

We performed logic synthesis of the resulting accelerators with Synopsys Design Compiler (ver. J-2014.09-SP2) and simulation with Cadence Incisive (ver. 15.10) to extract gate-level SAIF back-annotations for the given workloads and obtain accurate power estimations with Synopsys Power Compiler (ver. J-2014.09-SP2) when using the different versions of the memory library.

Performance and Area Overhead. We performed different simulations varying the mode-transition latency of the SRAMs: we varied the power-gating latency between 2 and 10 cycles [23] and the voltage controller latency between 64 and 2,000 cycles [13]. The performance overhead is generally less than 1%. Only FFT-2D has a bigger overhead (i.e., 3.5% when using the voltage controller with maximum latency) because it does not use double buffering and, therefore, the transitions are not masked by the execution of other processes. The area occupation of the data controllers is proportional to the number of banks used to implement each data structure, while the area of the OMC and SMC modules depends on the number of scenarios. In any case, the total area overhead of the DARKMEM units with respect to the corresponding PLM units in the reference designs is much less than 1% in all cases. The area overhead introduced by the voltage controllers is fixed and depends only on the number of DARKMEM units and the technology used.

Impact of Optimizations. First, we performed experiments with only one optimization active (either the scenario-based or the workload-based one). These results are shown in Fig. 6. For each benchmark, the green column *Scenario-based* reports the average power consumption in the two scenarios of each accelerator, while the other columns (in three shades of red) report the results obtained when performing the workload-based optimization with the three different memory libraries. Specifically, in case of scenario-based optimization, we removed the OMC modules. This corresponds to the case where the designer does not take advantage of the DarkMem API to annotate the execution modes of the data structures in the accelerator's SystemC. So, in each scenario, the used banks are always kept in the *active* state, while the unused ones are power gated for the entire execution of the accelerator. In this case, there is no performance overhead due to the operating-mode transitions during the execution. On the contrary, when we performed only the workload-based optimization, we removed the SMC modules. This corresponds to having no information on the scenarios. In the scenario-based optimization, *FFT-2D* achieves the best results since its requirements in terms of SRAM banks grow exponentially with the input parameters. So, executing a smaller scenario with power gating has significant advantages ($\sim$30% of SRAM static power saving). In general, to power gate the unused banks in the SMALL scenario enables quite significant SRAM static power savings ($\sim$20% on average), which correspond to a total power reduction of about 7%. In the workload-based optimization, the results depend on the characteristics of the memory library. The ULP library achieves, as expected, the best results ($\sim$28% of SRAM static power reduction on average); in this case, the SRAM static power of *PCA* is reduced by up to 45% thanks to an efficient power management of the PLM data structures. On average, this reduces the total power consumption of the accelerators by 10%, where the additional power spent in the SRAM transitions mitigates the overall benefits.

Finally, Fig. 7 shows the results obtained by combining the two optimizations. This reduces the SRAM static power by up to 60%, which on average translates in a reduction of the total power consumption by almost 18%.

V. Related Work

Power consumption is the main reason behind the emergence of heterogeneous architectures [6, 9, 12]. Such architectures contain several SRAMs for implementing local storage (e.g., processor caches or accelerator local memories). While power gating and DVFS have been widely applied to processor cores [2], GPUs [23], and hardware accelerators [12, 13], the design of individual accelerators with SRAM-specific techniques has not been studied before. Cong *et al.* [6] focused on creating power-efficient communications rather than optimizing the SRAM power consumption. Wang *et al.* [23] used power gating to reduce the static power of GPU caches. We implemented an extension to this approach for accelerator private local memories. In [12], multiple accelerators share a non-specialized set of small memory banks (2KB and 4KB SRAMs), which can be dynamically assigned to the accelerators or power gated when unused. The bank configuration, however, is not tailored to any specific accelerator. Mantovani *et al.* [13] applied DVFS to reduce the power consumption of accelerators, especially in case of congestion on the system interconnect. However, they use the same supply voltage for both the accelerator logic and the PLM and keeping the SRAM always active and above the standby voltage. Our DarkMem architecture is complementary and can further increase the energy efficiency of the accelerators' PLM with a more fine-grained power management. However, this requires the integration of multiple IVRs per accelerator [11]. Additional power savings can then be obtained by combining these approaches. The reduction of SRAM static power has not been specifically addressed in accelerator design based on HLS. Deep-sleep modes have been considered only for processor caches [14] or application-specific systems [20]. In contrast, we support the design of specialized accelerators where the PLM microarchitecture is automatically determined (at design time) and managed (at run time) to optimize the SRAM power consumption.

VI. Conclusions

We proposed an accelerator microarchitecture and a companion design methodology to control the operation mode of each SRAM bank in heterogeneous SoCs, enabling fine-grained power management of such architectures. Results on a set of complex accelerators show that the accelerator and total static power can be reduced on average by 40% and 18%, respectively, with negligible performance overhead. Future work includes the study of the interaction with the processor cores and their cache hierarchies.

Acknowledgments

This research has received funding from the EU Commission's H2020 Programme under grant agreement N. 732105, the CERBERO project. This work was also supported by the National Science Foundation (A#: 1527821), and C-FAR (C#: 2013-MA-2384), one of the six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

[1] K. Barker et al. *PERFECT Benchmark Suite Manual*. PNNL and GeorgiaTech Research Institute, Dec. 2013. http://hpc.pnnl.gov/projects/PERFECT/.

[2] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Trans. on VLSI Systems*, 8(3):299–316, June 2000.

[3] L. P. Carloni. From latency-insensitive design to communication-based system-level design. *Proc. of the IEEE*, 103(11):2133–2151, Nov. 2015.

[4] L. P. Carloni. The case for embedded scalable platforms. In *Proc. of Design Automation Conference (DAC)*, pages 1–6, June 2016.

[5] Y. H. Chen et al. A 0.6V Dual-Rail Compiler SRAM Design on 45nm CMOS Technology With Adaptive SRAM Power for Lower VDD$_{min}$ VLSIs. *IEEE Journal of Solid-State Circuits*, 44(4):1209–1215, 2009.

[6] J. Cong, M. A. Ghodrat, M. Gill, B. Grigorian, and G. Reinman. Architecture support for accelerator-rich CMPs. In *Proc. of Design Automation Conference (DAC)*, pages 843–849, June 2012.

[7] F. Conti et al. An IoT endpoint system-on-chip for secure and energy-efficient near-sensor analytics. *IEEE Trans. on Circuits and Systems I: Regular Papers*, pages 1–14, May 2017.

[8] E. Cota, P. Mantovani, G. Di Guglielmo, and L. P. Carloni. An analysis of accelerator coupling in heterogeneous architectures. In *Proc. of Design Automation Conference (DAC)*, June 2015.

[9] M. Horowitz. Computing's energy problem (and what we can do about it). In *Proc. of ISSCC*, pages 10–14, Feb. 2014.

[10] N. S. Kim et al. Leakage Current: Moore's Law Meets Static Power. *Computer*, 36(12):68–75, Dec. 2003.

[11] W. Kim, M. S. Gupta, G. Y. Wei, and D. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *Proc. of Int.l Symp. on High Performance Computer Architecture (HPCA)*, Feb. 2008.

[12] M. J. Lyons, M. Hempstead, G.-Y. Wei, and D. Brooks. The Accelerator Store: A shared memory framework for accelerator-based systems. *ACM Trans. on Architecture and Code Optimization*, 8(4):48:1–48:22, Jan. 2012.

[13] P. Mantovani et al. An FPGA-based Infrastructure for Fine-grained DVFS Analysis in High-performance Embedded Systems. In *Proc. of Design Automation Conference (DAC)*, pages 1–6, June 2016.

[14] Y. Meng, T. Sherwood, and R. Kastner. On the limits of leakage power reduction in caches. In *Proc. of Int.l Symp. on High Performance Computer Architecture (HPCA)*, pages 154–165, 2005.

[15] A. Pedram, S. Galal, S. Kvatinsky, S. Richardson, and M. Horowitz. Dark memory and accelerator-rich system optimization in the dark silicon era. *IEEE Design & Test*, 34(2):39–50, Apr. 2017.

[16] C. Pilato, P. Mantovani, G. D. Guglielmo, and L. P. Carloni. System-level optimization of accelerator local memory for heterogeneous systems-on-chip. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 36(3):435–448, Mar. 2014.

[17] M. Qazi et al. Challenges and Directions for Low-Voltage SRAM. *IEEE Design & Test*, 28(1):32–43, Jan. 2011.

[18] H. Qin et al. Standby supply voltage minimization for deep sub-micron SRAM. *Microelectronics Journal*, 36(9):789 – 800, 2005.

[19] S. Rusu et al. Power reduction techniques for an 8-core Xeon processor. In *Proc. of ESSCIRC*, pages 340–343, Sept. 2009.

[20] M. Shafique et al. Adaptive power management of on-chip video memory for multiview video coding. In *Proc. of Design Automation Conference (DAC)*, 2012.

[21] H. Singh et al. Enhanced leakage reduction techniques using intermediate strength power gating. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 15(11):1215–1224, Nov. 2007.

[22] S. Thomas et al. CortexSuite: A synthetic brain benchmark suite. In *Proc. of IISWC*, pages 76–79, Oct 2014.

[23] Y. Wang et al. Run-time power-gating in caches of GPUs for leakage energy savings. In *Proc. of DATE*, pages 300–303, Mar. 2012.