Benchmarking Methodology for Embedded Scalable Platforms

Paolo Mantovani[†], Emilio G. Cota[†], Seongjong Kim[*], Kevin Tien[*],
Johnnie Chan[†], Giuseppe Di Guglielmo[†], Christian Pilato[†],
Martha A. Kim[†], Mingoo Seok[*], Kenneth Shepard[*], and Luca P. Carloni[†]
[†]Dept. of Computer Science, [*]Dept. of Electrical Engineering - Columbia University, New York, NY

Abstract—Embedded scalable platforms (ESP) are a novel generation of platform architectures that yield optimal energy-performance operations while supporting a diversity of embedded application workloads. A companion methodology combines full-system simulation, pre-designed HW/SW interface libraries, high-level synthesis and FPGA prototyping to enable an effective design-space exploration which is driven by the benchmarking of alternative ESP architectures through the execution of the actual software of the target workloads. As a result, ESP designers are guided in the choice and assembly of a heterogeneous set of programmable processors and hardware accelerators to balance regularity, flexibility, and specialization.

1 Heterogeneity: the Key to Power Efficiency

As semiconductor device scaling has pushed into nanometer technologies, power dissipation has become the biggest obstacle to embedded computing performance [2], [4]. Accelerators can offer 2 to 3 orders-of-magnitude higher efficiency than software for many computations [3], but reduces design regularity, increases system complexity, and prolongs design time.

To address these challenges we are developing *Embedded Scalable Platforms (ESP)*, which bring together a novel *platform architecture* and an innovative *companion methodology*. The architecture offers power efficiency through heterogeneity, while the methodology offers productivity by guiding software programmers and hardware engineers together to select the best mix of components. Early efficiency assessment of the actual target applications must drive the design efforts all the way to implementation.

An ESP instance features multiple embedded processors and accelerators. Processors follow the typical logic synthesis flow, starting from a pre-designed RTL specification (*soft-IP* block) and will run in most cases a complex operating system or just a simpler real time environment, whereas accelerators are realized using high-level synthesis (HLS). Reusing IP blocks and HLS allow the design effort to be focused on integration, evaluation and optimization.

The complexity of integrating heterogeneous components is mitigated by *Platform Services*, including data transfers (DMA, cache-line transfers, etc.), accelerator reservation, power policy management, performance counters and diagnostics. Power management is primarily based on dynamic-voltage frequency scaling (DVFS), which will be fine grained by leveraging our prior research on integrated voltage regulators [5]. The platform services are supported by (1) a *Scalable Communication and Control Infrastructure (SCCI)* that could be realized with a bus or network-on-chip, (2) a set of *templated wrappers*, to decouple components' design from the SCCI, and (3) a set of *software interfaces*, primarily consisting of device drivers, that convey to the programmer the illusion of a simpler homogeneous architecture.

2 Benchmarking: the Driver for ESP Design

As illustrated in Fig. 1, the ESP design methodology comprises three stages: (1) *System-Level Design (SLD)*, in which the co-development of hardware and software is supported by a virtual platform; (2) *Design-Space Exploration (DSE)*, in which critical kernels are identified, the local scratchpads and computational resources are allocated and sized; (3) *System Integration (SI)*, in which the product of the DSE is a specific ESP configuration.

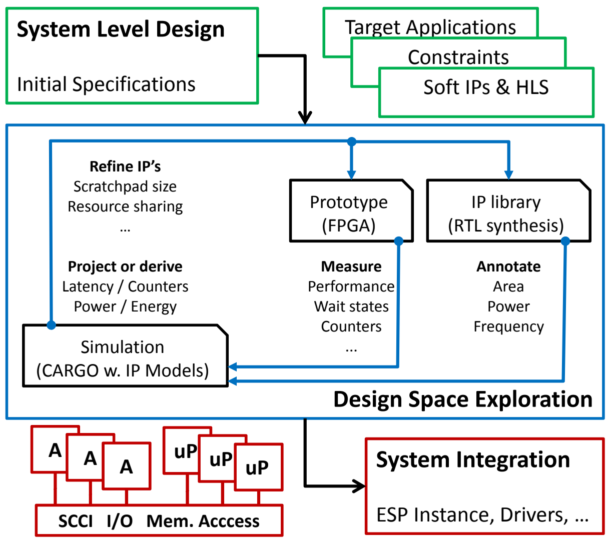

Fig. 1: The ESP methodology

To enable this methodology we employ three design and evaluation flows, shown in Fig. 1:

1) FPGA Emulation by which the an ESP instance is composed, tested and evaluated on FPGA. Fast emulation supports the execution of target applications with large input sets and full system software to produce reliable metrics. From this emulation, we collect total cycle counts as well as a breakdown of time spent on computation, communication, and waiting on DRAM, and use them to validate the simulator.
2) Component Based Synthesis: processors, accelerators and the SCCI are mapped separately to the target technology to build a library of IPs, each characterized with static and dynamic dissipated power for each of the available voltage frequency pairs.
3) Cargo Simulation: the full-system simulator leverages QSIM for the parallel execution of multiple QEMU instances modeling the embedded processors, whose power is estimated thanks to the integration of McPat. Cargo also supports the inclusion of functional models of the ac-

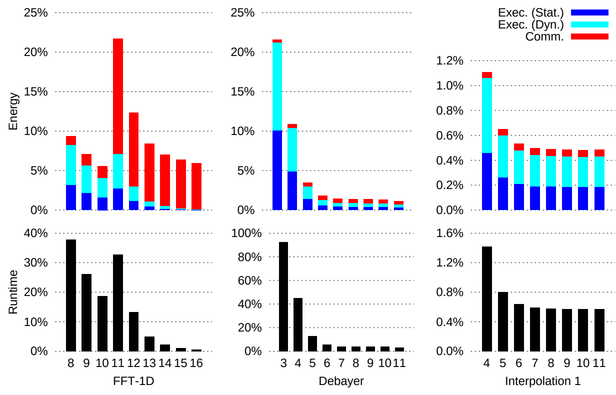

Fig. 2: HW vs. SW energy and performance

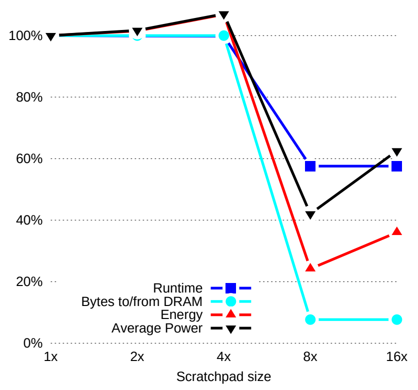

Fig. 3: FFT-1D: DSE on scratchpad size

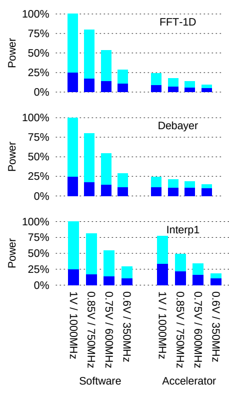

Fig. 4: Power w. VFS

celerators with annotation of data transactions and other metrics calibrated on the execution of their synthesized versions on FPGA and RTL design flows. The target applications run on top of the same operating system, middleware, and memory stack that will be part of the final ESP implementation, thus providing projections of latency and cycle counts for the target technology together with an estimate of power consumption. After calibration, the simulator enables rapid accurate evaluation of many ESP configurations and settings (e.g. the accelerators' scratchpad size).

3 Experimental Results

We apply this methodology to the ESP instance described in Table 1. The target applications are three kernels taken from The PERFECT benchmark suite [1]: Fast Fourier Transform (FFT), Debayer from Wide Area Motion Imagery and Interp1 from Synthetic Aperture Radar.

TABLE 1: ESP Instance Description

Parameter	Description
Processor	LEON 3 (Sparc)
Cache line size	16 bytes
L1d cache	Private 64KB, 4-way, write-through
L1i cache	Private 16KB, 2-way, write-through
Hardware Accelerators	{FFT, Debayer, Interpolation}
SCCI	Based on AMBA2 bus
Operating System	Linux 3.8.0

Accelerator Energy Savings and Speedup. Fig. 2 reports execution time and energy consumption breakdown for the three kernels executed on their respective accelerators, w.r.t. the software baseline. The x axis shows the $log2$ of the input size. Performance and energy are derived using the methodology described above and have been used to calibrate CARGO. The FPGA emulation can capture the overhead and the uncertainty due to the interaction with the OS, as well as with DRAM. Notice how FFT run-time, relative to software, does not decrease monotonically. When the input grows beyond the size of the accelerator's scratchpad, it suffers a huge penalty; however, as the input keeps growing, the accelerator's optimized memory access pattern outperforms the general purpose processor cache. The other two accelerators' major advantage is in maximizing computation per token read from memory. This also explains why the improvement quickly becomes almost constant as soon as the input does not fit in the processor cache.

Accelerator Tuning. Fig. 3 demonstrates how CARGO supports accelerator DSE. The size of the scratch pad can be easily modified in the FFT model and the calibrated simulator can quickly estimate benefits and overhead of instantiating more static RAM. Here we find the optimal scratchpad size for inputs of 2^{13} elements. Once the input fits in the local SRAM, any additional block of memory deteriorates efficiency by increasing power consumption.

Voltage Frequency Scaling Benefits. Finally Fig. 4 shows the average power for the three kernels when executed both in hardware and in software at different voltage frequency points. From the charts we can infer that the processor benefits more from scaling than specialized hardware, which leads to the conclusion that for these accelerators it is better to run the accelerators as fast as possible rather than slowing them down to reduce voltage. The relative similarity in software and accelerator power for interp1 is due to the deeply pipelined design of this accelerator which leads to higher power, compensated w.r.t energy savings by a very short run-time. Evaluating fine-grained DVFS is part of our ongoing research.

4 Concluding Remarks

This abstract describes a methodology for performance and efficiency evaluation of ESP. We believe that the complexity and heterogeneity of our architecture will be shared with other modern and future high performance embedded systems, and thus the proposed evaluation and co-design flow could be applied to a wider class of systems, as long as they share one key feature: optimization for a specific target workload.

Acknowledgments. This work is supported in part by the DARPA PERFECT program.

References

[1] K. Barker et al. *PERFECT (Power Efficiency Revolution For Embedded Computing Technologies) Benchmark Suite Manual*. Pacific Northwest National Laboratory and Georgia Tech Research Institute, 2013. http://hpc.pnnl.gov/projects/PERFECT/.

[2] H. Esmaeilzadeh et al. Dark silicon and the end of multicore scaling. pages 365–376, June 2011.

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

[4] O. Shacham et al. Rethinking digital design: Why design must change. *IEEE Micro*, 30(6):9–24, Nov. 2010.

[5] N. Sturcken, L. Carloni, K. Shepard, et al. A 2.5D integrated voltage regulator using coupled magnetic core inductors on silicon interposer delivering 10.8A/mm^2. In *ISSCC Digest of Technical Papers*, pages 400–402, Feb. 2011.