IEEE Asian Solid-State Circuits Conference
November 10 - 12, 2014/Kaohsiung, Taiwan

A 4.5 to 13 Times Energy-Efficient Embedded Microprocessor with Mainly-Static/Partially-Dynamic Reconfigurable Array Accelerator

Itaru Hida, Dahoo Kim, Tetsuya Asai, Masato Motomura
Graduate School of Information Science and Technology, Hokkaido University
Sapporo, Hokkaido, Japan
E-mail: hida@lalsie@ist.hokudai.ac.jp

***Abstract*—Conventional processors are energy in-efficient in that they fail to utilize the fact that most of their time and energy are spent on heavily-recursively executed small code segments. A DYNaSTA accelerator, proposed and implemented, is an architectural solution to such a problem. It is an reconfigurable array accelerator featuring an hybrid architecture: only a limited portion is reconfigured dynamically (*i.e.*, frequently) while the rest is reconfigured statically (*i.e.*, only occasionally). This way, the DYNaSTA accelerator tries to achieve both flexibility and energy-efficiency at the same time. Results of power simulation and fabricated chip measurements have been quite encouraging: 4.5 to 13 times energy efficiency will be made possible by this accelerator when compared with a conventional embedded microprocessor.**

I. Introduction

Overwhelming trends towards Internet of Things explain why low energy embedded microprocessors (EMPs) are getting more important than ever. Sources of energy inefficiency in EMP architectures are fairly well understood: the needs 1) to fetch/decode every instruction from memory, 2) to write/read register files to acquire/store operands per every instruction, 3) and to clock numerous numbers of F/Fs for pipelining multiple instructions on a datapath [1]. Given this insight, we may choose to "statically" map those instructions in heavily executed "recursive codes" to array of ALUs prior to their execution. By running the codes just as combinatory datapath with no registers, 1) to 3) redundancies can be drastically reduced. Though this "reconfigurable accelerator" solution looks straightforward and attractive, there is an inherent drawback: it is hard to cope with complex control flows (*i.e.*, lots of branches) typical in embedded applications. It explains why previous such proposals have focused on simple code segments that do not have a branch [2][3].

II. Architecture

Based on this observation, we have recently proposed abstract architecture for achieving both energy efficiency and versatility in controls-rich embedded applications [4]. Contribution of this paper is to materialize the concept into executable micro-architecture, design/verify it in real silicon chip, and evaluate its energy efficiency. Key innovation in our proposal, a DYNaSTA reconfigurable accelerator shown in Fig. 1, is to combine two distinctive array structures different in nature, namely dynamic operand forwarding matrix (DYN),

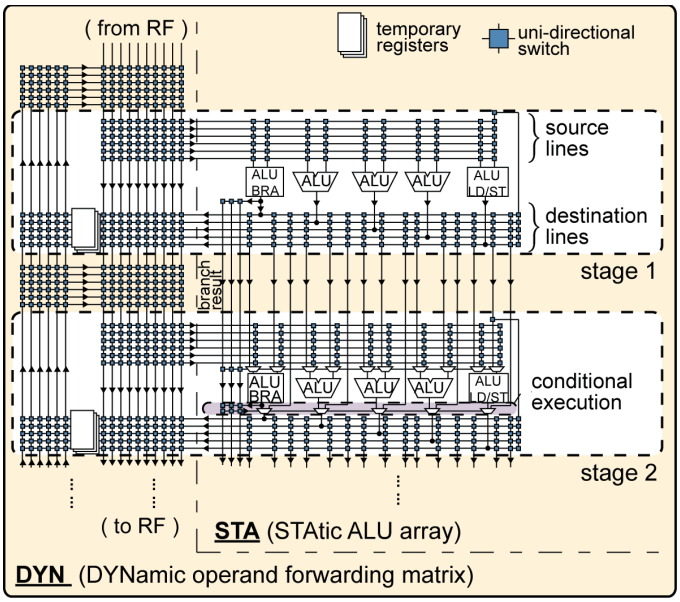

Fig. 1. DYNaSTA reconfigurable accelerator architecture.

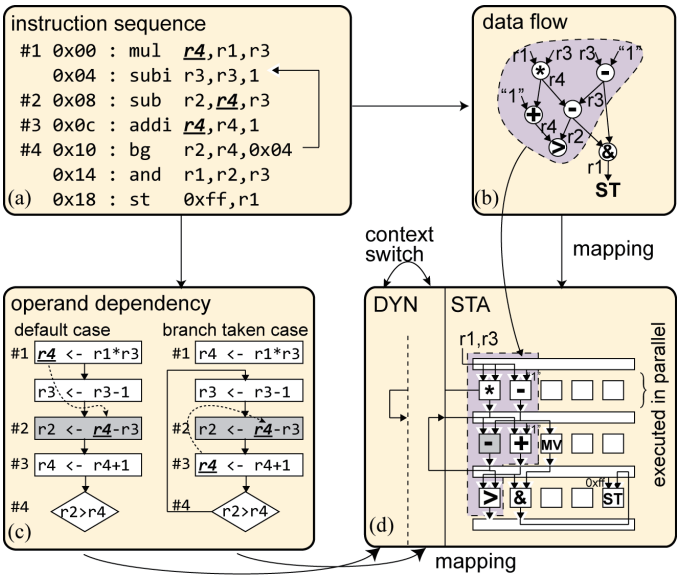

Fig. 2. Code mapping policy: (a) an example code, (b) extracted data flow, (c) extracted operand dependency, (d) mapping on DYN and STA.

978-1-4799-4089-9/14/$31.00 © 2014 IEEE

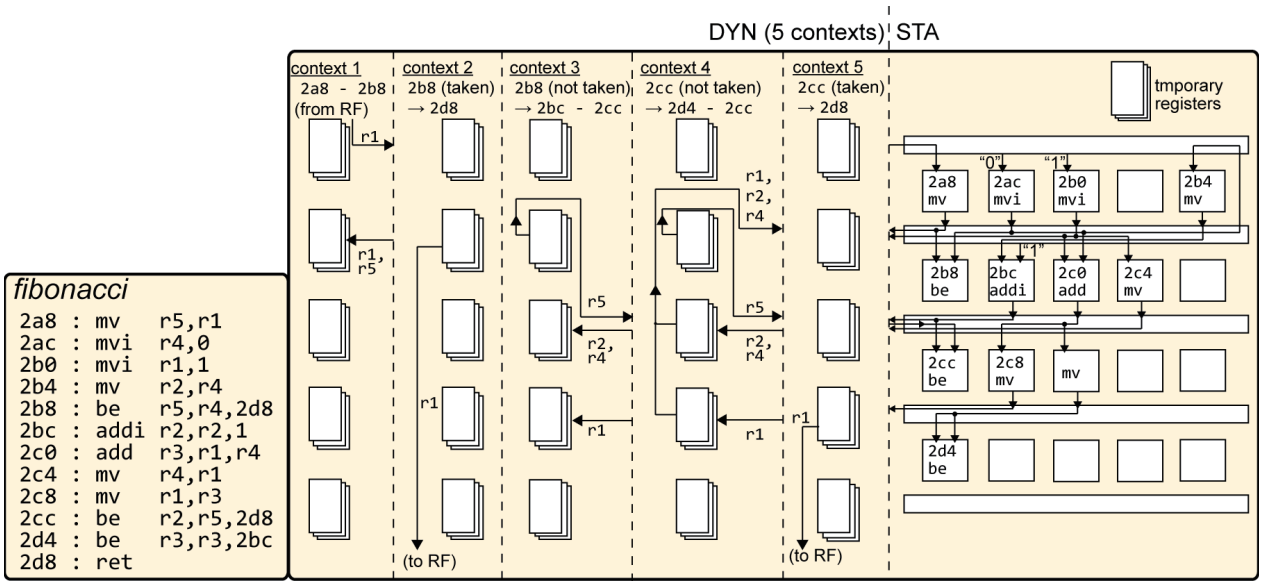

Fig. 3. *Fibonacci* assembly code and its mapping on DYNaSTA.

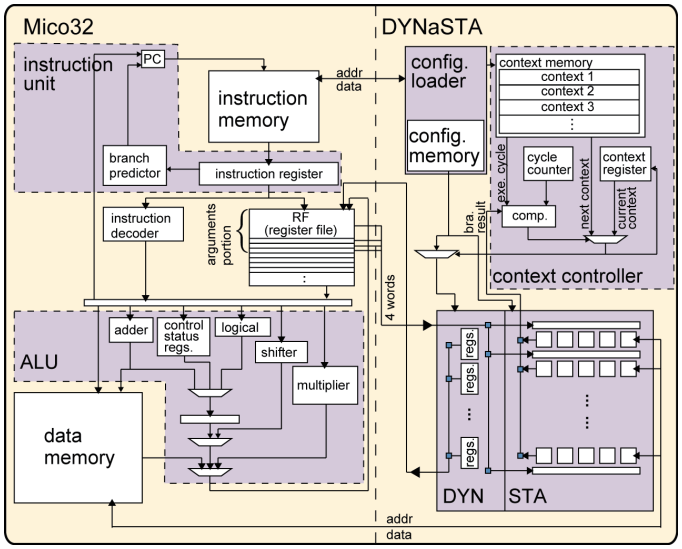

Fig. 4. Tight-integration of Mico32 (base EMP) and the DYNaSTA accelerator.

and static ALU array (STA). STA plays a key role in achieving energy efficiency, while DYN does so in versatility.

STA features non-fixed number of stages, where each stage has several ALUs sharing a set of source/destination lines (Fig. 1). For reducing the number of switches hence improving energy efficiency, only parallel instructions are mapped onto a same stage, where branch/jump and load/store instructions go to the 1st and last ALUs, respectively (Fig. 2(b) and (d)). The instructions dependent on preceding ones are mapped onto the next stage. Conditional execution is supported for discarding short forward branches. Appropriate number of STA stages is dependent on the sizes of target codes, while that of ALUs per stage will range from 2 to 8 as in superscalar/VLIW architectures. Note there is no registers hence no clocks in STA.

Difficulty in serving branches in a reconfigurable accelerator lies in that their outcome can never be known apriori: for example (Fig. 2(c)), the "r4" operand in #2 may be produced by #3 instead of #1 when #4 branch is taken. Efforts to accommodate such dynamic nature in the ALU array like STA unavoidably degrade its simplicity and regularity hence incurring energy inefficiency. DYN is a multi-context, bidirectional operand forwarding matrix for solving this difficulty: it is dynamically reconfigured only when operand dependencies amongst instructions are altered on a branch (Fig. 2 (c) and (d)). Operands that affected are memorized/forwarded in/from temporary registers (Fig. 1). Keeping power-consuming dynamic reconfiguration away from the massive ALU array (and leaving it static) is a key for achieving energy efficiency in DYNaSTA architecture.

We have designed an EMP with this DYNaSTA accelerator into silicon (Fig. 4). Base EMP is Mico32 [5], which has been chosen because of its typical RISC architecture and open source RTL code. The accelerator also includes a configuration loader, which sets whole DYNaSTA configuration prior to execution, and a context controller, which directs reconfiguration of DYN. By treating "recursive codes" that are mapped onto DYNaSTA as subroutines, the read/write path between Mico32's RF and DYN only needs to cover its arguments portion (4 registers, Fig. 4).

TABLE I. SUMMARY OF SAMPLE APPLICATIONS.

Application	# of Instrs.	# of brs.	PE utilization [%]	# of contexts
fibonacci	12	3	24	5
sbox	25	2	50	5
crc32	18	2	36	5
sepia filter	22	1	44	3

III. EVALUATION

Before measuring our fabricated chip, performance and power consumption of the DYNaSTA accelerator are evaluated

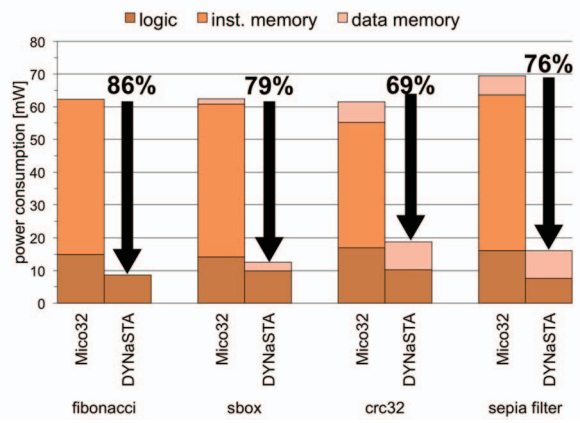

Fig. 5. Mico32 vs. DYNaSTA: total power consumption of sample applications.

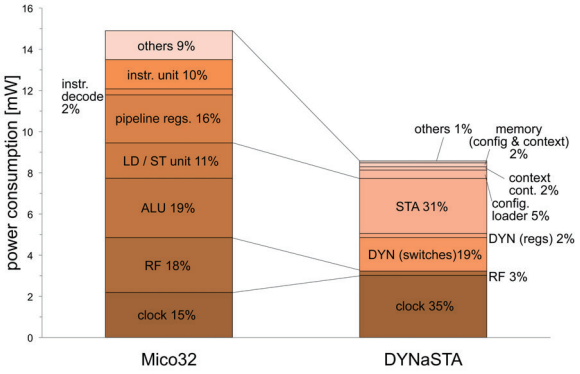

Fig. 6. Mico32 vs. DYNaSTA: logic power consumption (*fibonacci*).

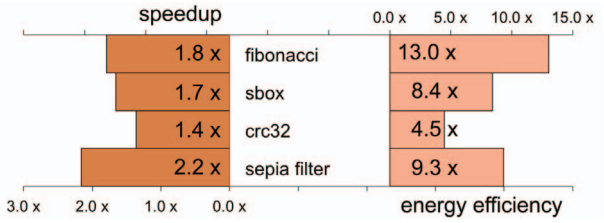

Fig. 7. DYNaSTA/Mico32 performance and energy efficiency improvements.

using sample applications (TABLE I) based on the synthesized netlist. For example, Fig. 3 shows disassembled *fibonacci* codes, a very control-rich one, along with its DYNaSTA configurations: all instructions are spatially mapped onto STA, while five contexts are allocated on DYN.

Fig. 5 compares power consumption of these codes running on Mico32 and the DYNaSTA. 69% to 86% reductions are observed due mainly to discarded instruction memory access. Logic power consumption is also reduced, as shown in Fig. 5, whose detailed breakdown is shown in Fig. 6 for the case of *fibonacci*. From Fig. 5 and Fig. 6, it is clear that 1) to 3) redundancies mentioned earlier have been successfully removed. Since instructions are executed in parallel in STA (Fig. 2(d)), the proposed architecture not only reduces the power but also enhances its performance (Fig. 7) at a same frequency (100MHz). Resultantly, energy efficiency is improved by 4.5

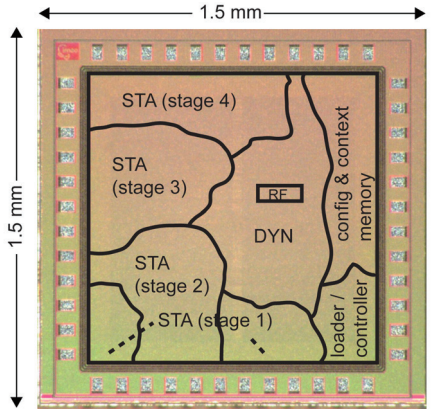

Fig. 8. Chip micrograph of proposed DYNaSTA (4 stages).

TABLE II. CHIP SPECIFICATION.

Specification	
Technology	UMC 0.18um 1P6M CMOS
Package	48-pin DIL
Die area	1.5 mm x 1.5 mm
Gate count	86.5K
Supply voltage	1.8V core / 3.3V IO
Clock frequency	100 MHz
# of Stages	4
Register file	32 bit x 4 word
# of ALUs / stage	5
# of Contexts	6

to 13 times from Mico32 for these sample codes.

We fabricated the proposed DYNaSTA using a UMC 0.18um process (see Fig. 8 and TABLE II). Due to the area constraint, 4 STA stages have been implemented. Though the size of DYNaSTA is very small, extending it is quite straightforward due to its regular array structure.

We measured power consumptions of the fabricated chip during the configuration and the running phases of the DYNaSTA with *fibonacci* exhibited in Section II. The experimental setup is shown in Figs. 9 and 10. Fig. 11 shows the measured power consumption versus clock frequencies for both the configuration and the running phases. Due to the limitation of our FPGA-based power-measurement workbench, the maximum frequency for the measurement was 80 MHz. We then predicted the power consumption at 100 MHz by linear interpolation of the measured data. TABLE III shows comparison of simulated and measured (and interpolated)

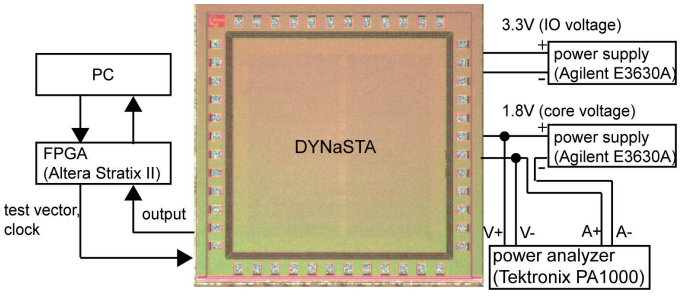

Fig. 9. Experimental configurations.

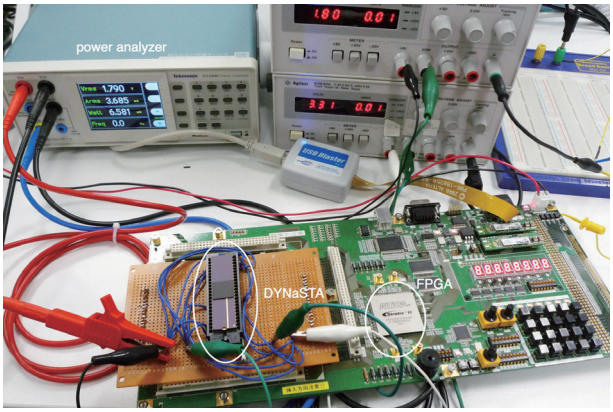

Fig. 10. Photograph of our power-measurement workbench.

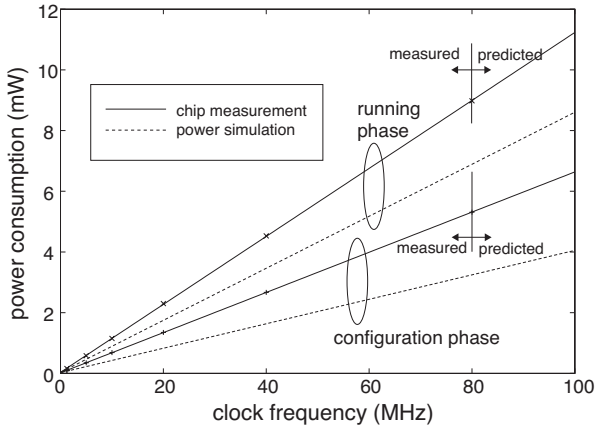

Fig. 11. Measured power consumption vs. clock frequency for configuration and running phases.

power consumptions for both the phases at 100 MHz. We observed slight mismatch (of about 2.6 mW) for both phases between the simulated and measured data. It resulted from circuit elements in the fabricated chip that are not included in the power simulation model, such as Mico32 register file (Fig. 8).

IV. Discussion and Conclusion

TABLE IV reveals reasons of this energy efficiency: though DYNaSTA consumes x18.5 more gates than Mico32, its average toggle rate is as low as x0.06 of Mico32. Specifically, gate-consuming STA features only 1.8% toggle rate, which accounts for its relatively low power occupation in Fig. 6.

Filling a chip with simple, regular and energy-efficient array like DYNaSTA can become an interesting solution in "Dark Silicon" [6] era (Fig. 12). Here, existing domain-oriented low power circuit techniques such as DVFS and power

TABLE III. Comparison between simulation results and measurement results at 100 MHz.

	configuration [mW]	run [mW]
simulation	4.03	8.57
measurement (interpolated)	6.63	11.20

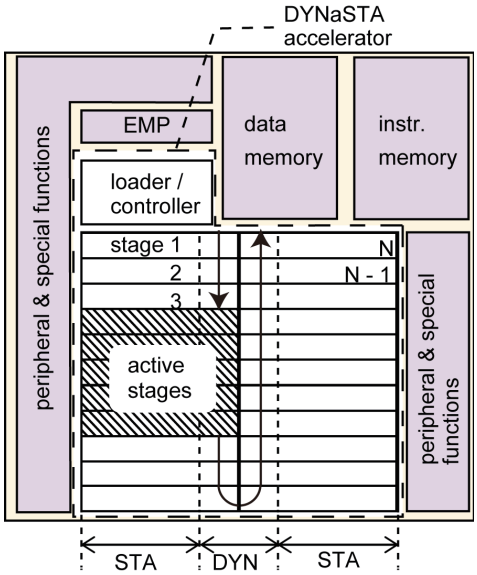

Fig. 12. DYNaSTA SoC concept toward "Dark Silicon" era.

gating can augment the architecture quite nicely. For instance, since only a few active stages propagate like a "wave" on the array, remaining numerous "silent" stages can be powered-off systematically for minimizing leak current (Fig. 12). Our next challenges include enhancing DYNaSTA with such low-power circuit techniques as well as establishing code mapping SW.

TABLE IV. Mico32 vs. DYNaSTA: gate counts and average toggle rates (*fibonacci*).

		Gate count [k gates]	Average toggle rate [%]
DYNaSTA	DYN	43.6	20.0
	STA	322.9	1.8
	Others	80.2	9.1
	All	446.8	4.9
Mico32		24.1	76.8
Ratio (DYNaSTA / Mico32)		18.48	0.06

References

[1] R. Hammed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B. C. Lee, S. Richardson, C. Kozyrakis, and M. Horowitz, "Understanding sources of inefficiency in general purpose chips," *ACM SIGARCH Computer Architecture News - ISCA '10*, vol.38, no.3, June 2010.

[2] N. Ozaki, Y. Yasuda, Y. Saito, D. Ikebeuchi, M. Kimura, H. Amano, H. Nakamura, K. Usami, M. Namiki, and M. Kondo, "Cool mega arrays: Ultralow-power reconfigurable accelerator chips," *IEEE Micro*, vol.31, no.6, pp.6-18, Nov.-Dec. 2011.

[3] Francisco-Javier Veredas, Michael Scheppler, Will Moffat, Bingfeng Mei, "Custom implementation of the coarse-grained reconfigurable address architecture for multimedia purposes," *FPL 2005*, pp.106-111, Aug. 2005.

[4] T. Hirao, Dahoo Kim, I. Hida, T. Asai, and M. Motomura, "A restricted dynamically reconfigurable architecture for low power processors," *ReConFig 2013*, pp.1-7, Dec. 2013.

[5] http://en.wikipedia.org/wiki/LatticeMico32

[6] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *ISCA 2011*, pp.365-376, June 2011.