Design-Effort Alloy: Boosting a Highly Tuned Primary Core with Untuned Alternate Cores

Elliott Forbes, Niket K. Choudhary†, Brandon H. Dwiel, Eric Rotenberg

Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, North Carolina 27695
{jeforbe2, bhdwiel, ericro}@ncsu.edu, niketc@qti.qualcomm.com

***Abstract*—**

A commercial flagship superscalar core is a highly tuned machine. Designers spend significant effort to tune the register-transfer-level (RTL) model, circuits, and layout to optimize performance and power. Nonetheless, the one-size-fits-all microarchitecture still suffers from suboptimal performance and power on individual applications. A single-ISA heterogeneous multi-core, with its multiple diverse core designs, has potential to exploit application diversity. However, tuning multiple core types will incur insurmountable design effort. This paper proposes a new class of single-ISA heterogeneous multi-core processor, called *design-effort alloy (DEA)*. Only one of the core types, called the *high-effort core (HEC)*, is tuned using a high-effort design flow. Much less effort is spent on tuning other core types, called *low-effort cores (LECs)*.

We begin with synthesizable RTL designs of a palette of out-of-order superscalar core types. A LEC and HEC is designed for each core type: the LEC is based on design automation and the HEC is derived from its LEC counterpart, using frequency and energy scaling factors that account for RTL, circuit, and layout optimizations. The resulting HECs have more than a 2x frequency advantage with only a 1.3x increase in energy consumption compared to their corresponding LECs. From the palette of core types, we find the best 4-core-type DEA processor for 179 SPEC SimPoints (program phases). Our study yielded the following key results:

1) **The DEA processor's HEC is the same core type in the best high-effort homogeneous multi-core, owing to most program phases demonstrating "average" instruction-level behavior and favoring this balanced core.**
2) **The DEA processor yields a speedup in BIPS^3/W of 1%-87%, and a geometric-mean speedup of 25%, on 20 out of 179 SimPoints over the best high-effort homogeneous multi-core. Thus, untuned LECs operating at less than half the frequency of the HEC nonetheless accelerate program phases with "outlier" instruction-level behavior.**

I. Introduction

A commercial flagship superscalar core is a highly tuned machine. Both its microarchitecture and physical design are carefully crafted, by hundreds of engineers over a period of three to five years, to achieve high single-thread performance on a wide range of applications. Nonetheless, a single microarchitecture leaves some performance on the table. Its design is tugged in different directions by diverse program characteristics. Opposing forces cause numerous compromises, for example, compromises between exploiting more instruction-level parallelism (ILP), more memory level-parallelism, and high frequency. As a result, the core is suboptimal in some program phases, especially those phases with outlier behaviors.

A single-ISA heterogeneous multi-core, with its multiple diverse core designs, can satisfy several of these competing forces [12] [17] [19]. They achieve this by better matching processor resources to different programs or phases within programs. For example, some phases may be able to take advantage of a wider pipeline, increasing performance and efficiency despite a potential decrease in frequency that width imposes. While it is conceptually easy to increase the pipeline width, the major problem with single-ISA heterogeneous multi-core processors is that crafting many different highly tuned core designs is prohibitively expensive. Recent work suggests automating the generation of register-transfer-level (RTL) designs of microarchitecturally diverse superscalar cores [7], but this strategy does not address the significant effort that goes into producing a high quality physical design, including tuning of the RTL, circuits, and layout.

In this paper, we propose a new class of single-ISA heterogeneous multi-core processor which we call Design-Effort Alloy, or DEA. In a DEA processor, only one of the core types is a high-effort core (HEC) and all other core types are low-effort cores (LECs). The design team spends most of its time tuning the RTL, circuits, and layout of the HEC. On the other hand, the LECs are not tuned: their RTL is unoptimized, yielding unbalanced pipeline stages; they are fully synthesized using standard cells, with the possible exception of some compiled SRAM memories; and their layouts are generated exclusively by automated place-and-route.

From a design space of eighteen superscalar core types in RTL form, we identify the best 4-core-type DEA processor for a workload of 179 SPEC SimPoints. Frequencies and energies of LECs are derived from automated synthesis. Frequencies and energies of HECs are derived from their LEC counterparts, using experimentally derived delay and energy scaling factors for tuning strategies that include pipeline stage balancing, custom transistor sizing, high speed flip-flop designs, and

†Author contributed to this paper while at North Carolina State University. Current affiliation is CPU Design Center, Qualcomm, Inc., Raleigh, NC.

978-1-4799-6492-5/14/$31.00 ©2014 IEEE

manual layout. The first key result is that the DEA processor's HEC ends up being the same as the core type in the best high-effort homogeneous multi-core. Thus, the role of the HEC is as workhorse for the majority of program phases that exhibit "average" instruction-level behavior. The HEC maintains robust leading-edge performance that can only be reliably achieved with a highly hand-optimized core. The second key result is that the DEA processor yields a speedup in $BIPS^3/W$ of 1%-87%, and a geometric-mean speedup of 25%, on 20 out of 179 SimPoints over the best high-effort homogeneous multi-core (hence, over the HEC). Thus, the LECs are able to accelerate "outlier" program phases despite their substantial frequency handicap.

An *overall* low-effort design flow for the LECs must also consider logic design and verification effort. Fortunately, there are several promising strategies for reducing logic design and verification effort of the LECs:

- IP reuse [10]: Leverage RTL of previous-generation core designs.
- Automatic core generation [2] [7] [9]: Leverage tools that generate RTL of whole core designs.
- Beta cores [25]: Consciously shorten the verification cycle and employ dynamic checkers in the field to correct residual bugs.
- ISA subsetting [15]: Implement a simple, high coverage subset of the full ISA in the LECs and fault to the HEC for unimplemented instructions.

The intent of this work is not to show the potential benefits of heterogeneity in general – that has been well explored in prior work [10] [12] [13] [14] [17] [19] [26]. Rather, we focus on the following *new* contributions:

- The idea of coupling a core designed using a traditional high-effort approach with cores designed using a fully-automated (low-effort) approach, to deliver a lower NRE cost heterogeneous multi-core. Our low-effort approach includes automated RTL generation of superscalar cores enabled by the FabScalar toolset and automated synthesis, placement, and routing (SPR) enabled by EDA tools.
- A detailed demonstration and evaluation of high and low-effort design methodology.
- An analysis of performance and efficiency gains achieved on application phases by a Design-Effort Alloy processor.

The big impact of these contributions is to answer the question of whether or not cores designed using a low-effort flow can have performance (BIPS) or efficiency ($BIPS^3/W$) that is better than a core designed using a high-effort flow. The answer is "yes" for both performance and efficiency even though the LEC sacrifices over half of the frequency possible with physical design tuning. Our results show that 20 out of 179 phases achieve an efficiency improvement of up to 87% over a high-effort homogeneous multi-core. Also, we find that 9 of the phases benefit in performance, up to about 33%, over a high-effort homogeneous multi-core.

TABLE I. EDA TOOLS USED IN THIS WORK.

Name and Version
Cadence NC-Verilog, vers. 06.20-s006
Synopsys Design Compiler, vers. E-2010.12-SP2
Synopsys PrimeTime, vers. E-2010.12-SP3-1
Cadence SoC Encounter, vers. 9.1
HSPICE, vers. C-2009.03-SP1
FreePDK BSIM4 45nm process technology library [24]

II. CORE MODELS

In order to evaluate Design-Effort Alloy CMPs, we need performance and power models for a wide range of core configurations. Both our low-effort cores (LECs) and high-effort cores (HECs) are based on the FabScalar toolset [7]. FabScalar enables generating synthesizable RTL designs of diverse superscalar cores, that differ in their superscalar widths, pipeline depths, and structure sizes.

Section II-A describes how we derive frequencies and energy consumption for LECs using FabScalar.

Frequencies and energy consumption of HECs are scaled from their corresponding LECs. Per-configuration scaling factors are derived through a series of techniques that include balancing pipeline stage delays, modeling key circuits in SPICE, and extrapolating placement and routing trends from standard cell estimates. HEC scaling factors are described in Section II-B.

A. Modeling Low-Effort Cores (LECs)

For generating RTL designs of whole cores, FabScalar is comprised of a Canonical Superscalar Template, a Canonical Pipeline Stage Library (CPSL), and a Core Generator [6]. The Template specifies composable interfaces of canonical pipeline stages. The CPSL provides many different RTL designs of each canonical pipeline stage that differ in their superscalar width and depth of sub-pipelining. Structure sizes are parameterized. The designer specifies a configuration to the Core Generator, which references the CPSL and Template to assemble a core of that configuration.

In addition to the pipeline RTL, the FabScalar toolset includes two means for supporting memories [7]. The first is FabMem, a memory compiler for generating full layouts of highly-ported RAMs and CAMs for the FreePDK 45nm technology. The other is a modified CACTI [22] technology file, adjusted for FreePDK 45nm, for modeling large memories such as caches and predictors.

To facilitate our analysis, we developed the FabScalar Performance-Power-Area (FabScalar-PPA) framework, shown in Figure 1. At the heart of FabScalar-PPA is a database that has cycle time, power, and area for every pipeline stage design in the CPSL. The EDA tools used to generate this database, and used for other purposes in this paper, are shown in Table I. The data are based on synthesis and, in some cases, post-synthesis place-and-route (e.g., bypass network). Synthesis used cells (.lib files) from FabMem for highly-ported RAMs and CAMs. For each stage design in the database, synthesis was performed multiple times with successively tighter cycle time targets until the tightest cycle time was achieved.

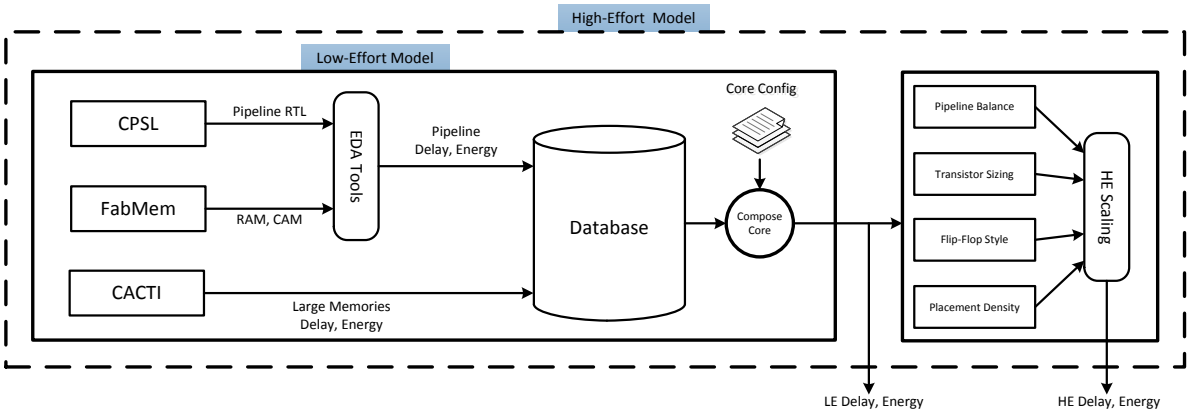


Fig. 1. Models for estimating low and high-effort design.

B. Modeling High-Effort Cores (HECs)

It is beyond the scope of this paper to carry out a high-effort design flow for each of the core configurations. Using corroborative sources [4] [5], we identify four key techniques that are used in the design of flagship cores. In the following sections, we address each of these techniques and then combine their effects into a model that is used to scale the frequency and energy of LECs to characterize HECs.

1) Balancing Pipeline Stages: One main area for improvement is balancing the delays of all pipeline stages. Pipeline stages in the CPSL begin and end at canonical pipeline stage boundaries. While this makes it possible to compose arbitrary cores within the canonical template, it comes at the expense of some stages having longer delays than others. The delay of the longest stage dictates the clock period of the entire pipeline. In a high-effort design, a team of designers would move logic from stages with longer delays to stages with shorter delays, reducing the overall clock period.

To model this effect in our hypothetical HECs, we take a very simple approach: we sum all of the delays of the individual pipeline stages and divide that total delay by the number of pipeline stages. This models a perfectly balanced pipeline. The factor of improvement in frequency is between 1.4x and 1.7x, which is in line with the 1.3x to 1.8x factor estimated by Chinnery and Keutzer [4].

Estimating the energy impact of pipeline balancing is difficult, however. Without balancing, pipeline registers hold final values of the previous stage. If the stages are evenly balanced, then pipeline registers may instead hold intermediate values, and it is highly likely that more flip-flops are needed. From this rationale, we posit that energy will increase, but the magnitude of this increase is still unknown without actually balancing the pipeline. Thus, we do not apply this energy tax to the HEC, which gives the $BIPS^3/W$ advantage to the HEC and, in turn, reduces the estimated benefit of DEA (since the key idea of DEA is that LECs can outperform the HEC for certain outlier program phases).

2) Transistor Sizing: Standard cell libraries are limited to only a handful of drive strengths. Custom circuit designers can reduce the delays of critical paths by increasing transistor sizes. Increasing transistor sizes reduces delay, but comes at the cost of increased capacitance, which increases energy. To find the frequency and energy impact that can be expected,

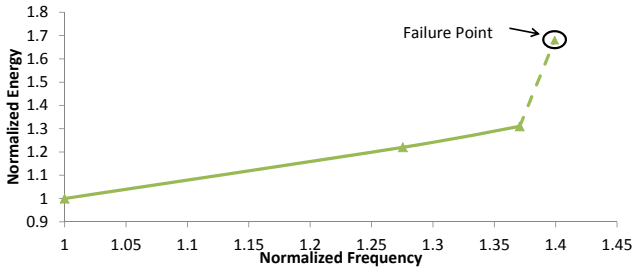


Fig. 2. Frequency and energy trend for transistor sizing.

we extracted timing-critical paths from the SPICE netlist of example 4-wide Issue and Register Read stages.

These paths were then manually optimized, by adjusting transistor sizes, to operate at a higher frequency. The results are plotted in Figure 2 (energy includes both dynamic and static energy). The "failure point" refers to the point at which we were no longer able to increase frequency without a substantial increase in energy. For the frequencies that we did meet, we find that energy increases linearly, evident in the solid part of the line. We can see that transistor sizing alone can increase frequency by about 37%, but that comes with a 31% increase in energy. Again, this is within the estimate made by Chinnery and Keutzer [4], which was in the range of 1.1x to 1.45x.

Another approach for shortening the critical path is to reduce the threshold voltages of transistors on the critical path, at the price of higher leakage. This alternative is not explored in this paper.

3) Pulse Latches: Both Intel [11] and IBM [27] [28] used pulse latches instead of flip-flops in their flagship designs. Pulse latches rely on a very short clock pulse to minimize the window during which values flow through the latch transparently. Due to the narrow window of time, the pulse latch behaves similar to a flip-flop and comes with the benefit of a negative setup time (i.e., the D input can change for a slight time after the rising clock edge). This property of pulse latches reduces the total delay of the pulse latch, determined by the setup delay plus the clock-to-Q delay. The lower delay comes at the cost of higher power consumption, however.

We implemented SPICE models of a pulse latch and a flip-flop to characterize the frequency and power increases afforded by pulse latches. We used the HLFF pulse latch [20], which has been used in several flagship designs. The NanGate standard

TABLE II. PULSE LATCH AND FLIP-FLOP CHARACTERIZATION.

Characteristic	Flip-Flop	Pulse Latch
T_{setup}	15ps	-4ps
$T_{clk-to-q}$	37.2ps	31.2ps
Total delay	52.2ps	27.2ps
Power	56.7μW	94.5μW

cell library [1] uses transmission gate-based flip-flops, which we designed using the techniques from Rabaey, et al. [21]. The delay and power results of the SPICE simulations are shown in Table II. The pulse latch is faster than the flip-flop by a factor of 1.9x, at a cost of 1.7x higher power consumption.

4) Layout: Finally, we consider the effect of careful layout of the chip, encompassing all three facets of layout: floorplanning, place-and-route, and custom layout of critical units. Our low-effort model is based on basic floorplanning, automatic place-and-route, and standard cells except for RAMs/CAMs. Automation reduces the time it takes to generate a layout, but the layout is of lower quality than a manual layout. A manual layout can achieve higher transistor density, hence, shorter wires, which reduces both delay and energy.

To explore this aspect, we manually placed-and-routed a simple module (synthesized to standard cells) numerous times at different densities, and plotted normalized frequency and energy as a function of transistor density, in Figures 3a and 3b, respectively. The measured data and corresponding trend lines are shown with solid lines. As this experiment is based on standard cells, it only accounts for the effect of floorplanning and place-and-route on layout quality. To also account for custom layout of critical units, we extrapolated the trend lines using transistor densities inferred from published die photos and transistor counts for the Intel Nehalem [11] and IBM POWER7 [27] [28] processors, both 45nm designs. (Caches were excluded, i.e., we inferred transistor density of logic only.) The extrapolated points are shown with dashed lines.

The impact of layout quality on frequency is small, only a 0.5% increase from the lowest to the highest density. On the other hand, energy decreases by about 5%.

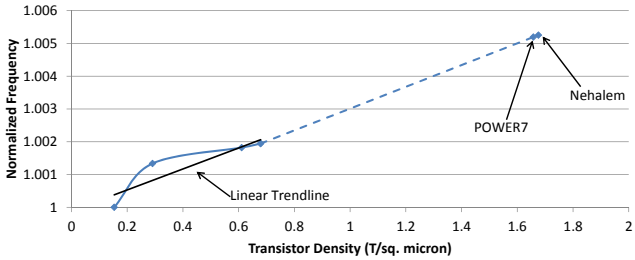
The linear extrapolations in Figure 3 assume a linear relationship between wire capacitance and wire length. Wire capacitance is proportional to area and fan-out [4]. If we assume a simple wire model, wherein capacitance is due only to parallel-plate capacitance, then capacitance is linear with wire length [21].

5) High-Effort Scaling Model: To derive full-pipeline estimates, we must combine the high-effort techniques into a single model for energy and frequency. We use the following equations:

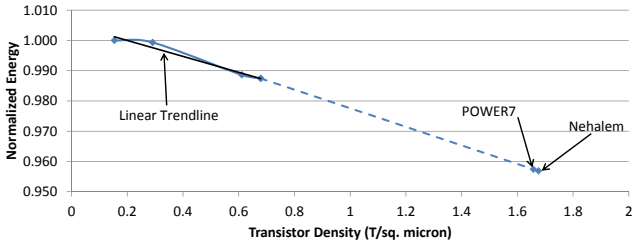
$$T_{HEC} = \alpha_{layout}[\alpha_{trans.size}(T_{balance} - T_{DFF}) + T_{PL}] \quad (1)$$

$$E_{HEC} = \beta_{layout}[\beta_{trans.size}(r_{logic}E_{LEC}) + \beta_{PL}(r_{FF}E_{LEC})] \quad (2)$$

The α coefficients, differentiated by subscripts, denote delay scaling factors given in previous sections. Similarly, the β coefficients denote corresponding energy scaling factors.



(a) Frequency trend for placement density.



(b) Energy trend for placement density.

Fig. 3. Estimated impact of layout on high-effort designs.

Equation (1) gives the clock period of the HEC, T_{HEC}. $T_{balance}$ is the clock period after balancing pipeline stages of the corresponding LEC (Section II-B1). T_{DFF} is the D flip-flop delay (Section II-B3). T_{DFF} is subtracted from $T_{balance}$ to get just the combinational logic delay, which is then scaled by the transistor sizing scaling factor, $\alpha_{trans.size}$ (Section II-B2). The pulse latch delay, T_{PL}, is added back to the scaled combinational logic. Finally, the layout scaling factor, α_{layout} (Section II-B4), is applied to the total delay.

Equation (2) gives the energy of the HEC, E_{HEC}. E_{LEC} is energy of the LEC. E_{LEC} is divided into two parts: energy contributed by logic cells and energy contributed by flip-flop cells. This breakout is necessary to separately apply the transistor sizing scaling factor, $\beta_{trans.size}$, to logic, and the pulse latch scaling factor, β_{PL}, to flip-flops. r_{logic} is the ratio of logic cells to total cells. r_{FF} is the ratio of flip-flop cells to total cells. Thus, $r_{logic} + r_{FF} = 1$. These ratios are obtained from the standard cell netlist of the LEC.[1] These ratios depend on the core configuration, but r_{FF} is in the range of 12% to 17%. Finally, the layout scaling factor, β_{layout}, is applied.

After applying these equations, the total frequency scaling factor is in the range of 2.1x to 2.5x and the total energy scaling factor is between 1.29x and 1.31x.

III. METHODOLOGY

This section describes our methodology for designing and comparing optimal homogeneous, heterogeneous, and design-effort-heterogeneous multi-core processors: core design space (III-A), metrics (III-B), workload (III-C), cycle-accurate processor simulator (III-D), and CMP design space exploration (DSE) tool (III-E). Figure 4 depicts the overall methodology.

[1]Since energy is proportional to capacitance, a more accurate model would base r_{FF} and r_{logic} on the total capacitances contributed by flip-flops and logic, respectively. This information is much harder to derive from the netlist and standard cell library, so cell counts are used as a simple proxy.

TABLE III. THE PALETTE OF 18 CORES CONSIDERED FOR EVALUATION.

Core Name	Period (ns) LE	HE	Depth	IQ, AL, LSQ	L1 Size(KB), Assoc Inst	Data
1W-S	0.4	0.167	15	8, 64, 16	8,1	8,1
1W-M	0.5	0.198	15	24, 128, 32	16,2	16,2
1W-L	0.6	0.246	14	64, 384, 128	32,4	32,4
2W-S	0.5	0.220	16	32, 96, 64	16,4	16,4
2W-M	0.6	0.273	14	48, 192, 128	32,4	32,4
2W-L	0.7	0.285	13	64, 384, 128	64,4	64,4
3W-S	0.5	0.201	18	16, 64, 32	16,4	16,4
3W-M	0.6	0.287	14	48, 128, 64	32,4	32,4
3W-L	0.7	0.308	15	64, 384, 128	64,4	64,4
4W-S	0.6	0.266	16	32, 128, 32	32,4	32,4
4W-M	0.7	0.306	15	48, 256, 128	64,4	64,4
4W-L	0.8	0.329	15	64, 384, 128	64,4	64,4
5W-S	0.6	0.258	16	24, 64, 32	32,4	32,4
5W-M	0.7	0.298	16	48, 192, 128	64,4	64,4
5W-L	0.8	0.318	16	64, 256, 128	64,4	32,4
6W-S	0.7	0.327	15	32, 128, 64	16,4	32,4
6W-M	0.8	0.325	16	48, 256, 128	32,4	64,4
6W-L	0.9	0.379	15	64, 384, 128	64,4	64,4
Common core parameters						
Shared L2, 2MB, 8-way assoc						
L2 stream prefetcher, 32 entry						

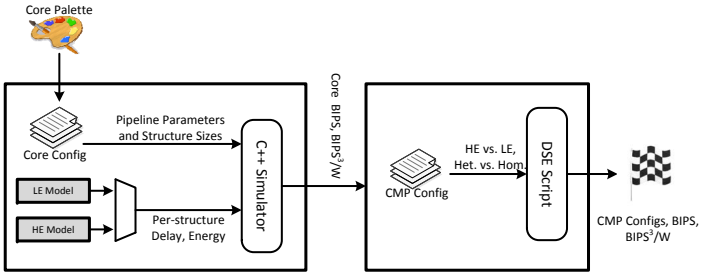


Fig. 4. Methodology for evaluating Design Effort Alloy.

A. Core Palette

The number of valid core configurations that can be modeled by FabScalar-PPA is about 38,000. It was previously observed that the diversity in this large design space can be reasonably captured by far fewer core types. We follow the same strategy as Choudhary et al. [7]. They considered three cycle time targets for each of superscalar widths 2 through 8, yielding 21 core types. As width increases, the three cycle time targets also shift to accommodate more logic complexity. The three cycle time targets for a given width accommodate small, medium, and large structure sizes for that width.

In this paper, we consider superscalar widths of 1 (scalar) through 6 and three cycle time targets for each, yielding the 18 core types listed in Table III. These 18 core types form the core palette for our evaluation. A core type is named by its width (e.g., 4W) and relative size – small (S), medium (M), or large (L) structures, corresponding to three cycle time targets.

B. Metrics

Performance is measured in billions-of-instructions-per-second (BIPS) to factor-in both instructions-per-cycle (IPC) and frequency. The goodness metric used by the DSE to rank cores, however, is BIPS^3/W. This metric captures both performance (BIPS) and power (W) which is our aim. Moreover, BIPS^3/W is a voltage-independent metric, hence, BIPS^3/W is considered a better metric than energy-delay-product [16]. In this paper, we refer to BIPS^3/W simply as *efficiency* because it emphasizes both time-efficiency and power-efficiency.

C. Workload

The workload for both the DSE and experiments consist of 179 program phases taken from the SPEC CPU2000 and CPU2006 suites. The benchmarks were compiled with gcc version 4.5.2 with the -O3 optimization level for the MIPS64r2 instruction set. Our toolchain does not include a Fortran compiler. The fortran-to-C converter, f2c, allows us to include Fortran-77 programs but not Fortran-90/95 programs, which were the only SPEC benchmarks omitted as a result. The 179 program phases were obtained using SimPoint [23], configured to select a maximum of ten 10-million-instruction SimPoints from each benchmark. The benchmark regions returned by SimPoint are hereafter referred to as *phases* and named by the combination of benchmark name and SimPoint interval number.

D. Cycle-Accurate Simulator

We developed a cycle-accurate (with respect to FabScalar) processor simulator that we interface with FabScalar-PPA to evaluate core configurations. As the simulator executes, it maintains activity counters. When the simulation completes, the simulator queries the FabScalar-PPA database for per-component energy costs and combines these costs with corresponding activity counters to estimate overall energy. The simulator also measures IPC which is multiplied by frequency to get BIPS.

E. CMP Design Space Exploration

The inputs to the CMP design space exploration (DSE) tool are 1) performance or efficiency of every phase on every core in the core design space, 2) whether the CMP should be homogeneous or heterogeneous, and 3) whether the CMP should be all high-effort, all low-effort, or mixed-effort (a DEA configuration).

For the homogeneous CMP, the DSE tool finds the one core type that gives the highest average performance or efficiency over all phases.

For heterogeneous CMPs, it is important to determine the number of core types, N. Figure 5 shows the percent-improvement in efficiency of the best N-core-type combination over the best single core type for increasing N. The knee in the curve for both high and low-effort heterogeneous configurations is at four core types. Additional core types yield marginal efficiency gains. Therefore, N is chosen to be four.

When evaluating a given combination of four core types, the DSE assumes each phase will execute on its best core among the four. The DSE explores all possible 4-core-type combinations and selects the one with the highest average performance or efficiency over all phases. For DEA, the search space quadruples relative to all low-effort core types, because one of the four core types is high-effort.

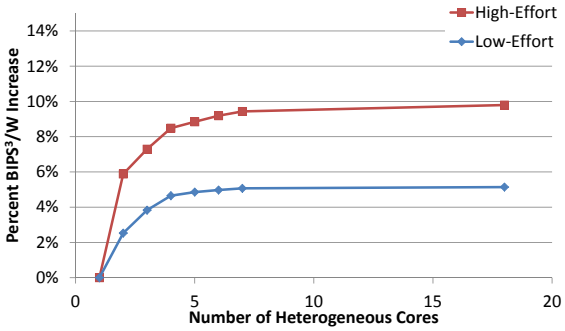


Fig. 5. Best average efficiency (BIPS^3/W) improvement for low and high-effort heterogeneous cores.

TABLE IV. CORE CONFIGURATIONS FOR EACH CMP TYPE.

CMP	Core Configs
Low-Effort Homogeneous	2W-S
Low-Effort Heterogeneous	1W-S, 2W-S, 3W-M, 4W-S
High-Effort Homogeneous	2W-S
High-Effort Heterogeneous	1W-S, 2W-S, 2W-L, 4W-S
Design-Effort Alloy	HE-2W-S, LE-1W-S, LE-1W-L, LE-2W-S

IV. RESULTS

In Section IV-A, we analyze the results of the design space exploration of five different CMP types, including DEA. We then compare efficiency and performance of DEA with the other four CMP types in Sections IV-B and IV-C, respectively.

A. DSE Results

The core configurations selected by the DSE tool are listed in Table IV for the five CMP types. Note that the DEA cores are labeled with HE- or LE- to denote which core is high-effort and which are low-effort, respectively.

The interesting aspect of these results is that the 2W-S core appears in all types of CMP. It is the well-balanced core that most phases prefer, owing to their "average" instruction-level behavior.

The 2W-S is the high-effort homogeneous CMP as well as the HEC in the DEA CMP. Because of this, it is not possible for phases to have an efficiency or performance that is worse on DEA than either the low or high-effort homogeneous CMPs.

Furthermore, the 2W-S configuration is selected as the high-effort core and one of the low-effort cores in the DEA CMP. It is possible to have both the high and low-effort versions of the same core configuration when exploring DEA. This is because the frequency and energy relationship changes when scaling from low-effort to high-effort.

B. Efficiency Results

Figure 6 shows the average improvement in efficiency of the DEA CMP over the baseline CMPs. The average is measured as the geometric mean speedup of BIPS^3/W. DEA outperforms the low-effort homogeneous and heterogeneous CMPs by 107% and 96%, respectively. This shows that the HEC of the DEA CMP buys a substantial increase in efficiency over simply relying on low-effort cores. We additionally show the upper-bound for high-effort heterogeneity in the right-most bar of Figure 6. The all high-effort heterogeneous CMP has

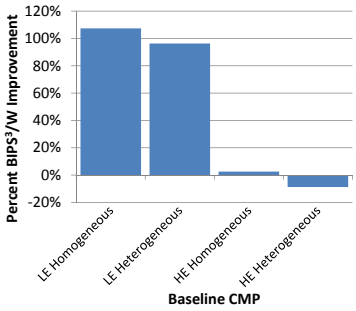


Fig. 6. Average efficiency increase of DEA CMP, relative to reference CMPs.

better efficiency, but it is the thesis of this paper that the design effort of such a configuration prohibits its implementation.

Thus, the key objective is for DEA to improve upon the efficiency of the high-effort homogeneous CMP. Indeed, the average improvement in efficiency of the DEA CMP over the high-effort homogeneous CMP is about 2.5%. While the average improvement seems modest, it belies the significant efficiency improvements enjoyed by individual phases and applications. We delve into this perspective, below.

We find that most phases – 159 out of 179 – are most efficiently executed by the HEC. Since the HEC in the DEA CMP is the same as the high-effort homogeneous core, these phases exhibit zero efficiency gains and consequently pull-down the average. Thus, the average improvement should necessarily be modest. This finding also underscores the continued importance of the flagship core – it is very efficient for a majority of phases.

The remaining 20 phases execute more efficiently on one of the LECs than on the HEC. Figure 7 plots the percent improvement in efficiency for these phases, rank-ordered by the magnitude of the percent improvement. (The phases that execute most efficiently on the HEC are not shown but would appear at 0% increase.) The point style indicates on which LEC each phase was most efficient. Several phases enjoy a significant improvement in efficiency, as high as 80%. A detailed analysis of an example phase that executes more efficiently on a LEC appears in Section V.

C. Performance Results

While efficiency is the primary metric and basis of our key findings, we find that some phases also run faster (higher BIPS) on their LEC than on the HEC. Figure 8 plots the percent performance increase for individual phases in a similar manner as was plotted for efficiency. While there are fewer phases that perform better on a LEC than on the HEC (9 out of 179) compared to efficiency improvement, there are enough phases that performance warrants consideration.

All phases in Figure 8 achieved their speedup on LE-1W-L. Although LE-1W-L has a clock period (0.6ns) that is nearly 3 times longer than that of HE-2W-S (the HEC) (0.22ns), and half the fetch width (1W vs. 2W), the microarchitecture of LE-1W-L is superior in its instruction window size (see Table III). This enables it to better tolerate long-latency cache misses, in addition to exposing and exploiting more memory-level parallelism (MLP). The phases in Figure 8 benefit more from these sources of performance (latency tolerance, MLP)

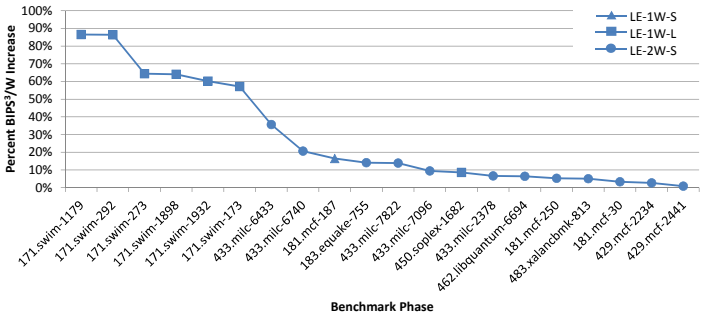


Fig. 7. Percent efficiency improvement of DEA over homogeneous CMP.

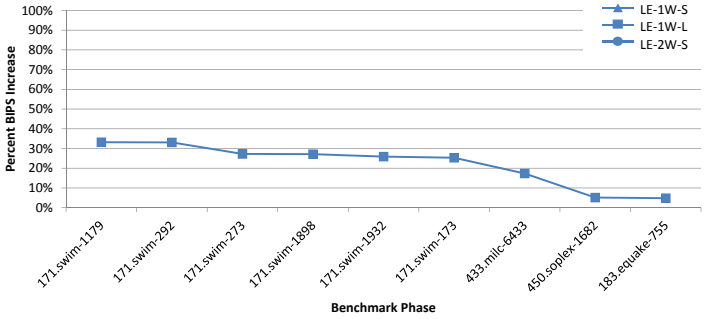
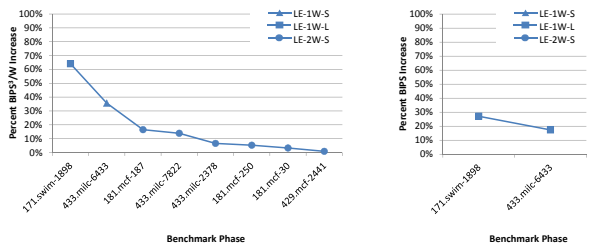


Fig. 8. Percent performance improvement of DEA over homogeneous CMP.

than from the other sources of performance exploited by the HEC (frequency, nearby ILP).

D. Sensitivity to Unseen Program Phases

While any processor design is susceptible to being over-designed for benchmarks used by the designers, heterogeneous multi-core processors, including DEA, are perhaps more susceptible. On the other hand, their diversity may actually make them more robust than homogeneous processors for unseen workloads. To evaluate this aspect, we used 70% of our program phases in a "training" group to design the DEA processor and 30% in a "test" group to evaluate the DEA processor, selected randomly. The resulting DEA processor is the same as before (same HEC and same three LECs). Because the test group is smaller, fewer program phases yield improvement in efficiency (BIPS^3/W) and performance (BIPS); but the percentage of program phases that are sped-up (efficiency or performance) increases as a percentage of the test group. Program phases that are sped-up in efficiency and performance are shown in Figures 9a and 9b, respectively.



(a) Efficiency. (b) Performance.

Fig. 9. Percent improvement of DEA over homogeneous CMP, using different training and test groups.

V. Program Phase Analysis

In this section, we dissect the source code of a phase from our benchmark suite. We seek to understand how it is possible that a phase can execute with higher efficiency and/or performance when executing on a LEC rather than the HEC.

Listing 1 shows a loop from the *mcf* benchmark. In this loop, there is a data structure traversal based on the variable named `arcin`. The lines of code that participate in the traversal are highlighted in red (lines 2, 4 and 17). The code blocks guarded by the `if`-statements at lines 3 and 9, highlighted in blue, are executed rarely (15% and 2.2% of the time, respectively). Therefore, essentially only lines 2, 7, and 17 are important for this loop. Performance is dominated by the pointer-chasing of lines 2 and 17 which is highly serial.

Thus, a core must simply execute the serial dependence chain as rapidly as possible. HE-2W-S has the highest frequency of all our modeled cores and is, indeed, the highest performing on this phase, as noted by the absence of *mcf.187* in the performance curve in Figure 8. For efficiency, however, the best core for this phase is LE-1W-S. Since the phase is mostly serial, there is little opportunity for ILP. So core configurations with additional instruction width cannot take advantage of that width, and the energy spent on this unused width is wasted. Similarly, cores with larger structure sizes do not benefit because they also come with a higher clock period.

```
1  while( arcin ) {
2    tail = arcin->tail;
3    if( tail->time + arcin->org_cost > latest ) {
4      arcin = (arc_t *)tail->mark;
5      continue;
6    }
7    red_cost = arc_cost - tail->potential
8                        + head_potential;
9    if( red_cost > 0 ) {
10     if( new_arcs < MAX_NEW_ARCS ) {
11       insert_new_arc( ... );
12       new_arcs++;
13     }
14     else if( (cost_t)arcnew[0].flow > red_cost )
15       replace_weaker_arc( ... );
16   }
17   arcin = (arc_t *)tail->mark;
18 }
```

Listing 1. Code example taken from mcf.187

VI. Related Work

Kumar et al. proposed single-ISA heterogeneous multi-core architectures for reducing energy of a single thread [10]. They extended the concept to increasing throughput of multi-programmed workloads within a constrained power and area budget [12] [13]. Suleman et al. [26] extended the concept to accelerating multithreaded applications by executing critical sections on a latency-optimized core and parallel tasks on throughput-optimized cores. ARM commercialized a heterogeneous design with two core types, "big" and "little" [8].

None of these works, nor the large body of inspired works, consider physical design effort as a parameter when doing core selection. Homogeneous physical design effort is assumed. Past work on superscalar design space exploration [14] [18] also assumes the same physical design effort for all cores.

Bazeghi et al. [3] proposed a methodology using regression models to quantify the design effort in the RTL implementation and verification phase in the processor development process. Using their methodology, the design effort of different components of a processor can be estimated, allowing design teams to allocate commensurate resources to the more effort-intensive components. However, their study does not account for physical design effort.

VII. Conclusion

This paper proposed a new class of heterogeneous multi-core processor comprised of a primary core type and multiple alternate core types, wherein only the primary core type is quality-crafted in terms of its physical design. The approach is referred to as *Design-Effort Alloy* (DEA). DEA balances a key tradeoff in the design of single-ISA heterogeneous multi-core processors, which is how to reap the benefits of having multiple microarchitecturally diverse core types without incurring the high NRE cost of crafting high quality physical designs for all of these core types. A key finding is that a significant number of "outlier" program phases execute more efficiently, and even faster in some cases, on an alternate core despite its severe frequency handicap relative to the primary core.

Acknowledgments

This project is supported by NSF grant CCF-1218608, relating to design for competitive automated layout, and a grant from Intel. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] NanGate Open Cell Library. http://www.nangate.com/?page_id=22.

[2] Xtensa Customizable Processors. http://www.tensilica.com/products/xtensa-customizable.

[3] C. Bazeghi, F. J. Mesa-Martinez, and J. Renau. uComplexity: Estimating Processor Design Effort. In *Proceedings of the 38th Annual IEEE/ACM Int'l Symposium on Microarchitecture*, pages 209–218, Nov. 2005.

[4] D. Chinnery and K. Keutzer. *Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design*. Kluwer Academic Publishers, Norwell, MA, 2002.

[5] D. Chinnery and K. Keutzer. *Closing the Power Gap Between ASIC & Custom: Tools and Techniques for Low Power Design*. Springer, New York, NY, 2007.

[6] N. K. Choudhary, S. V. Wadhavkar, T. A. Shah, H. Mayukh, J. Gandhi, B. H. Dwiel, S. Navada, H. H. Najaf-abadi, and E. Rotenberg. FabScalar: Automating Superscalar Core Design. *IEEE Micro*, 32(3):48–59, May-June 2012.

[7] N. K. Choudhary, S. V. Wadhavkar, T. A. Shah, H. Mayukh, J. Gandhi, B. H. Dwiel, S. Navada, H. H. Najaf-abadi, and E. Rotenberg. FabScalar: Composing Synthesizable RTL Designs of Arbitrary Cores Within a Canonical Superscalar Template. In *Proceedings of the 38th Annual Int'l Symposium on Computer Architecture*, pages 11–22, June 2011.

[8] P. Greenhalgh. Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7. Sept. 2011.

[9] V. Kathail, S. Aditya, R. Schreiber, B. R. Rau, D. C. Cronquist, and M. Sivaraman. PICO: Automatically Designing Custom Computers. *IEEE Computer*, 35(9):39–47, Sept. 2002.

[10] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction. In *Proceedings of the 36th Annual IEEE/ACM Int'l Symposium on Microarchitecture*, Dec. 2003.

[11] R. Kumar and G. Hinton. A Family of 45nm IA Processors. In *Proceedings of the 2009 IEEE International Solid-State Circuits Conference*, pages 58–59, Feb. 2009.

[12] R. Kumar, D. M. Tullsen, and N. P. Jouppi. Core Architecture Optimization for Heterogeneous Chip Multiprocessors. In *Proceedings of the 15th Int'l Conference on Parallel Architectures and Compilation Techniques*, pages 23–32, Sept. 2006.

[13] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance. In *Proceedings of the 31st Annual Int'l Symposium on Computer Architecture*, June 2004.

[14] B. C. Lee and D. M. Brooks. Illustrative Design Space Studies with Microarchitectural Regression Models. In *IEEE 13th Int'l Symposium on High Performance Computer Architecture*, pages 340–351, 2007.

[15] T. Li, P. Brett, R. Knauerhase, D. Koufaty, D. Reddy, and S. Hahn. Operating System Support for Overlapping-ISA Heterogeneous Multi-core Architectures. In *IEEE 16th Int'l Symposium on High Performance Computer Architecture*, pages 1–12, Jan. 2010.

[16] A. J. Martin and M. Nystroem. ET2: A Metric For Time and Energy Efficiency of Computation. In *Power-Aware Computing*, 2001.

[17] H. H. Najaf-abadi, N. K. Choudhary, and E. Rotenberg. Core-Selectability in Chip Multiprocessors. In *IEEE/ACM 18th Int'l Conference on Parallel Architectures and Compilation Techniques*, pages 113–122, Sept. 2009.

[18] S. Navada, N. K. Choudhary, and E. Rotenberg. Criticality-Driven Superscalar Design Space Exploration. In *Proceedings of the 19th Int'l Conference on Parallel Architectures and Compilation Techniques*, pages 261–272, Sept. 2010.

[19] S. Navada, N. K. Choudhary, S. V. Wadhavkar, and E. Rotenberg. A Unified View of Non-monotonic Core Selection and Application Steering in Heterogeneous Chip Multiprocessors. In *Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques*, Sept. 2013.

[20] H. Partovi, R. Burd, U. Salim, F. Weber, L. DiGregorio, and D. Draper. Flow-Through Latch and Edge-Triggered Flip-Flop Hybrid Elements. In *Proceedings of the 42nd IEEE Int'l Solid-State Circuits Conference*, pages 138–139, Feb. 1996.

[21] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits: A Design Perspective, 2nd Ed.* Prentice Hall, Upper Saddle River, NJ, 2003.

[22] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi. CACTI 5.1. *Tech. Report HPL-2008-20, HP Labs*, 2008.

[23] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically Characterizing Large Scale Program Behavior. In *Proceedings of the 10th Int'l Conference on Architectural Support for Programming Languages and Operating Systems*, pages 45–57, Oct. 2002.

[24] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal. FreePDK: An Open-Source Variation-Aware Design Kit. In *Proceedings of the 2007 IEEE Int'l Conference on Microelectronic Systems Education*, pages 173–174, June 2007.

[25] S. Sudhakrishnan, R. Dicochea, and J. Renau. Releasing Efficient Beta Cores to Market Early. In *Proceedings of the 38th Annual Int'l Symposium on Computer Architecture*, pages 213–222, June 2011.

[26] M. A. Suleman, O. Mutlu, M. K. Qureshi, and Y. N. Patt. Accelerating Critical Section Execution with Asymmetric Multi-core Architectures. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 253–264, Mar. 2009.

[27] D. F. Wendel, J. Barth, D. M. Dreps, S. Islam, J. Pille, and J. A. Tierno. IBM POWER7 Processor Circuit Design. *IBM Journal of Research and Development*, 55(3):6:1–6:8, May 2011.

[28] D. F. Wendel, R. Kalla, R. Cargoni, J. Clables, J. Friedrich, R. Frech, J. Kahle, B. Sinharoy, W. Starke, S. Taylor, S. Weitzel, and S. G. Chu. The Implementation of POWER7: A Highly Parallel and Scalable Multi-Core High-End Server Processor. In *Proceedings of the 2010 IEEE Int'l Solid-State Circuits Conference*, pages 102–104, Feb. 2010.