Design Space Exploration of High-Performance Parallel Architectures

Enric Musoll and Mario Nemirovsky

ConSentry Networks, Inc., USA
email: enric@consentry.com

ABSTRACT

High-performance single-threaded processors achieve their performance goal partly by relying, among other architectural techniques, on speculation and large on-chip caches. The hardware to support these techniques is usually a large portion of the overall processor real state area, and therefore it consumes a significant amount of power that sometimes is not optimally used toward doing useful work. In this work, we study the intuitive fact that architectures with hardware support for threads are more power efficient than a more traditional single-threaded superscalar architecture. Toward this goal, we have created a model of the power, performance and area of several parallel architectures. This model shows that a parallel architecture can be designed so that (a) it requires less area and power (to reach the same performance), or (b) it achieves better power efficiency and less area (for the same power budget), or (c) it has higher performance and better power efficiency (for the same area constraint), when compared to a single-threaded superscalar architecture.

Index Terms: Parallel Architectures, Multi-core, Many-core, Processor Power Consumption

1. INTRODUCTION

At the architecture level, high-performance processors have used several techniques to increase the IPC. Several of these techniques (branch prediction, value prediction) heavily rely on speculation, while others (trace caches, out-of-order issue and execution, on-chip L2 caches) rely on a large amount of on-chip logic to obtain, in some scenarios, a small percentage increase in performance. These techniques come at an expense not only in design and verification time and die area, but also at a sometimes high cost in power dissipation.

Parallel architectures featuring either several hardware-supported threads and/or processors, have been used to extract more performance by exploiting the thread-level parallelism of the application. The more processors and threads the chip packs together, the less complex these cores are, for the simple reason that the chip area is limited. Intuitively the power budget is better utilized toward performing useful work rather than using it for speculative tasks. Traditional parallel architectures are chip-multiprocessing [17] and simultaneously multi-threading processors [7].

This intuition is more formally addressed in this work as we tackle the question of whether it would be better, from the power efficiency (ie power/performance) and power density (ie power/area) perspective, to have a single, high IPC processor in a die, or to have many of lower-IPC processors executing the application. Will several (smaller, more power-efficient) processors be better in terms of power and performance (with area limitation) than a single, power-hungry processor at a given area constraint?

Toward answering this question, the contribution of this work is twofold:

- first, the proposal of a high-level, first-order magnitude model that allows the comparison of three basic metrics of a design (area, performance and average power) for some architectures (multi-processor, multi-threaded and clustered multi-threaded) that exploit the thread-level parallelism of applications to increase the overall performance. The parameters of the model are explained and reasonable values are obtained for these parameters based on both simulations and reported values in the literature.
- second, the analysis of the results obtained by the model comparing a superscalar with architectures based on hardware multithreaded support. We take on the challenging goal of exploring the power, area and performance design space of heavily multi-threaded architectures.

2. PREVIOUS WORK

Manne [12] and Musoll [16] have tackled the reduction of useless power in a single, high-performance processor. The first work focus on reducing the useless power due to speculative instructions that are fetched by the processor but they are never committed. These instructions from the wrong path access different blocks of the processor before they are flushed out of the pipeline. The technique is based on a confidence estimation of the prediction performed by the branch prediction mechanism. The other work tackles even the useless power dissipated by instructions of the correct path when accessing specific power-hungry blocks, like for example the L2 cache. These two works demonstrate that the amount of speculative work in a high-performance processor draws enough power to be an area of concern by researches. As a consequence, recent multi-core architectures impose a design rule which states that any increase in core area has to come with a proportional increase in core performance (to maintain the power efficiency of the core), otherwise the area increase does not pay off [1].

One architecture that leverages the thread-level parallelization of an application is Simultaneous Multi-threading, or SMT [7]. Seng [18] has studied ways to minimize the power consumption in these architectures.

The two closest works to ours are [11,15]. In [11], the authors explore the multi-dimensional design space for chip multiprocessors, showing that in these architecture, thermal constraints may force the selection of simpler, lower-performance cores due to their better power efficiency. In [15], the authors investigate the performance, power and thermal characteristics of multi-core architectures. The authors also evaluate the effect that different floorplans have on the temperature distribution.

In the previous two works, the number of cores are limited to only 8 ([15]) and 20 ([11]). Our goal is to evaluate multi-core architectures with a larger number of (simpler) cores and therefore we have developed a model based on empirical data obtained from published research. We have tried to compile these data in a coherent fashion toward building our model, but the lack of data and/or the large difference in the underlying architecture framework that led to the data has forced us to perform in some cases a rough approximation. We believe though that the study presented here is still a valid exercise toward the goal of analyzing the power efficiency of parallel architectures.

Table I. Building blocks of a core.

Alias	*Name*	*Alias*	*Name*
BPU	Branch Pred. Unit	SFA	Simple FP ALU
IC1	L1 Instruction Cache	**CFA**	Complex FP ALU
ITL	Instruction TLB	**LSU**	Load/Store Unit
DEU	Decode/Dispatch Unit	**DC1**	L1 Data Cache
IWU	Inst. Window Unit	**DC2**	L2 Unified Cache
RFU	Register File Unit	**DTL**	Data TLB
SIA	Simple Integer ALU	COU	Commit Unit
CIA	Complex Integer ALU		

3. NOMENCLATURE

In this work, we will use the following terms:

- **high-IPC processor (H)**: a typical high-performance, single-threaded processor that relies heavily in speculation, large caches, out of order issue, etc. to obtain the required performance. A high-IPC processor is the core of a *super-scalar architecture* (SS).
- **low-IPC processor (L)**: processor core intended to be replicated in a multi-processor. A low-IPC processor is smaller in area than a high-IPC processor because several of them need to fit in about the same chip real state as a single high-IPC processor. The low-IPC processor is the base of a *multi-processor architecture* (MP).
- **stream (S)**: set of hardware resources dedicated to run one of the software threads. A set of expensive resources are shared among the different streams. The streams are the main components of a *multi-threaded architecture* (MT).

Figure 1 depicts the different architectures. We will use the term core to refer to any of the above modules. A core is built out of several blocks of logic and/or memory circuitry. In this work, the blocks in Table I will be used. By choosing these blocks, some micro-architectural options have been already set (like having only one level of TLBs or split instruction and data first-level caches), but we believe that the block breakdown shown is generic enough for the purpose of this study.

All the shared blocks in the MT architecture (shown in bold in Table I) will be a subset of the blocks with the same characteristics as those in a low-IPC processor in the MP architecture. This is not necessarily a requirement since a MT chip may be designed for example with a higher performance fetching unit, larger caches and more functional units to better support the various streams. However, in this work it will be assumed that the blocks remain the same, and thus the streams will pay the performance penalty in terms of added latency to their operations. Each of the streams contains the rest of the blocks of a low-IPC processor that are not shared.

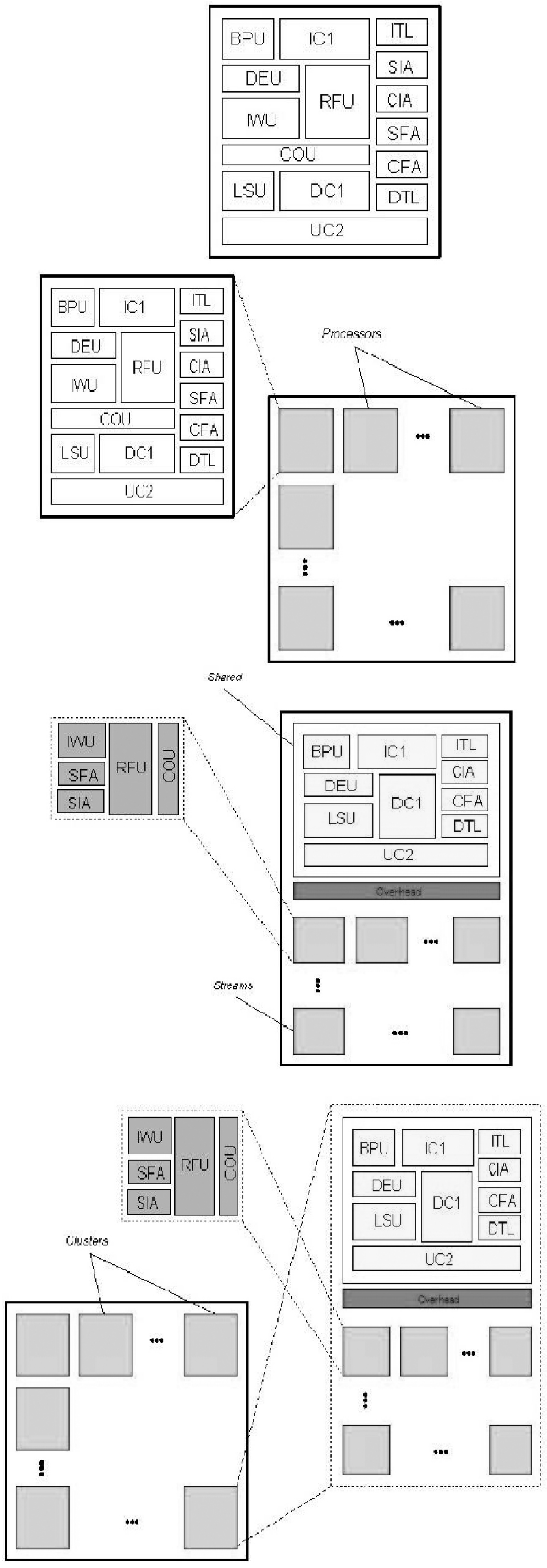

Figure 1. Architectures: Single-stream (SS); Multi-processor (MP); Multi-threaded (MT), and Clustered multi-threaded (CMT).

4. AREA, PERFORMANCE AND POWER MODEL

To compare the different architectures (SS, MP, MT and CMT) we will use a high-level model of their area, performance and power consumption. This model is intended to be used at the first stage of the design of a processor-based system. The model is parameterized, and the results are given relative to the single-threaded super-scalar architecture. The smallest block granularity in terms of parameterization of the model is the stream, ie the set of blocks that are replicated for each hardware-supported thread in the MT architecture.

The model provides therefore relative estimations of the area, performance and power consumption, not absolute values. It is thus intended as a comparison tool only. The parameters of the model are divided into three sections: (a) parameters that compare the area, performance and power of a high-IPC processor with a low-IPC processor, (b) parameters that compare the same metrics between a low-IPC processor and a stream, and (c) degradation factors in MP, MT and CMT architectures due to the hardware synchronization and resource sharing among the different processors/streams/clusters.

- *High-IPC vs. Low-IPC parameters*:
  - $A_{LH} = A_L/A_H$: ratio between the area of a low-IPC processor and a high-IPC processor.
  - $W_{LH} = W_L/W_H$: ratio between the average power consumption of a low-IPC processor and a high-IPC processor.
  - $P_{LH} = P_L/P_H$: ratio between the performance of a low-IPC processor and a high-IPC processor. Performance is the product of the IPC of the processor and its core frequency. In this work, all the architectures are compared at the same frequency, so IPC and performance are equivalent.
- *Low-IPC vs. Stream parameters* (s is the number of streams in a MT on in a CMT cluster):
  - $A_{SL}(s) = (A_S/A_L)$ * Overhead_A(s): ratio between the area of a stream and a low-IPC processor. A_L is the area of a low-IPC processor, and A_S is the area of a single stream without any area overhead that exists in a MT architecture or within a cluster in CMT to support several sources/destinations to/from the shared resources (namely, wider muxes, more control logic and more routing area). Overhead_A(s) is then > 1 for $s > 1$, and 1 for $s = 1$.
  - $W_{SL}(s) = (W_S/W_L)$ * Overhead_W(s): ratio between the average power consumption of a stream and a low-IPC processor. W_L is the power of a low-IPC processor, and W_S is the power of a single stream without any power overhead for the additional logic required to support several streams. Overhead_W(s) is always > 1 for $s > 1$, and 1 for $s = 1$.
  - $P_{SL} = P_S/P_L$: ratio between the performance of a single stream and a low-IPC processor. This ratio

is always 1 because a single-stream processor is the same as a low-IPC processor in this model. Of course, the overall performance does not necessarily scale linearly with the number of streams. This performance degradation is accounted for in the next set of parameters. We assume that the application is fully parallelizable so no overhead will be considered on the software side when the application is parallelized.

- *Performance Degradation parameters*: two sets of performance degradation factors exist: (a) due to the hardware synchronization of the different processors/streams/clusters, and (b) due to the sharing of resources by the processors/streams/ clusters:
  - $\mathrm{Dsync}_{MP}(p)$: degradation in an MP/CMT architecture due to the synchronization among the p processors/clusters. $\mathrm{Dsync}_{MP}(p)$ is always less or equal than 1 for a given p; if the application run by the cores were fully parallelizable requiring no synchronization, this factor would be 1.
  - $\mathrm{Dshare}_{MP}(p)$: degradation in an MP/CMT architecture due to the sharing of resources among the p processors/clusters (for example, when several processors/clusters access an external memory through a limited number of memory ports). $\mathrm{Dshare}_{MP}(p)$ is always less or equal than 1, being 1 for the case when no resource sharing occurs.
  - $\mathrm{Dsync}_{MT}(s)$: degradation in a MT architecture due to the synchronization among the s streams. $\mathrm{Dsync}_{MT}(s)$ is always less or equan than1.
  - $\mathrm{Dshare}_{MT}(s)$: degradation in a MT architecture due to the sharing of resources among the s streams. $\mathrm{Dshare}_{MT}(s)$ less or equal than 1; $\mathrm{Dshare}_{MT}(s)$ can theoretically be 1 in the case that the shared blocks where poorly architected in the low-IPC processor and were underutilized (thus having headroom for several streams to fully utilize them).

The expressions for the performance degradation parameters will be derived later on.

The model then provides comparisons relative to the SS architecture.

Table II. High-IPC and Low-IPC core configurations.

Parameter	High-IPC	Low-IPC
	Branch Prediction	
Direction Predictor	Combined, Bimodal 4K table, 2-lev 1K table, 10-bit hist, 4K chooser	Always not taken
BTB	1024-entry, 2-way	N/A
RAS	32 entries	8 entries
Mispred. penalty	7 cycles	5 cycles
	Memory Hierarchy	
L1 Data Cache	64KB, 2-way (LRU), 32B line, 1 cycle lat., 4 ports	8KB, 2-way (LRU), 32B line, 1 cycle lat., 1 port
L1 Inst. Cache	64KB, 2-way (LRU), 32B line, 1 cycle lat., 1 port	8KB, 2-way (LRU), 32B line, 1 cycle lat., 1 port
L2 Unified Cache	2MB, 4-way (LRU), 32B line, 12 cycle lat., 1 port	256KB, 4-way (LRU), 32B line, 2 cycle lat., 1 port
Memory latency	100 cycles	100 cycles
Inst. TLB	64-entry FA, 30-cycle miss lat.	8-entry FA, 30-cycle miss lat.
	Execution	
Inst. Window size	64 inst.	2 inst.
L/S queue size	32 inst.	2 inst.
Fetch queue size	8 inst.	2 inst.
Fetch width	4 inst./cycle	1 inst./cycle
Decode width	4 inst./cycle	1 inst./cycle
Issue width	4 inst./cycle	1 inst./cycle
Commit width	4 inst./cycle	1 inst./cycle
OOO execution	yes	no
Integer ALUs	4 simple ALUs	
1 complex mult/div	1 simple ALU	
1 complex mult/div		
FP ALUs	1 simple add, 2 complex mult&div/sqrt	1 simple add, 1 complex mul&div/sqrt

5. METHODOLOGY

In this section, each of the parameters of the model defined in the previous section is discussed in more detail and a well-educated value for each of them is obtained.

Once an estimation of the parameters is completed, the values will be plugged into the model and the conclusions regarding the area, performance, power, power efficiency and power density will be drawn.

A. High-IPC vs. Low-IPC parameters

In this section the parameters of the model that compare a high-IPC with a low-IPC processor are studied. The parameters are (see Section 4) $A_{LH} = A_L/A_H$, $W_{LH} = W_L/W_H$ and $P_{LH} = P_L/P_H$.

Table II shows the architectural configurations used in this work for the high-IPC and low-IPCcores. The high-IPC configuration roughly matches the Alpha 21264 processor. The low-IPC configuration is a scaled-down version of the high-IPC one.

- **Area Ratio A_{LH}**: The Chip-Space Estimator v1.0 [20] has been used to obtain, for each of the blocks and for each of the cores, the percentage of the block area with respect to the overall area. For the high-IPC core, we have seen that almost all the core area is used to implement the multi-ported L1 data cache and the big unified L2 cache (UC2). For the low-IPC configuration, the area is more equally distributed among the different blocks, but UC2 is still the most expensive block. Regarding the ratio between the low-IPC and high-IPC area, we have observed that the main differences are in the caches (because of the reduced size and single-ported L1 data cache in the low-IPC core), instruction window unit (because of the smaller number of entries, reduced number of ports and in-order issue in the low-IPC core) and branch predictor logic (because the low-IPC processor implements the straightforward always-not-taken scheme). On the contrary, no area difference occurs for the simple floating-point ALUs and for the complex integer ALU because both the high-IPC and low-IPC cores have the same number of these units. The A_{LH} ratio based on these results is 0.123.

- **Power Ratio W_{LH}**: The Wattch v1.0 [3] extension to the SimpleScalar simulator, running the integer and floating-point SPEC workloads, has been used to obtain the power numbers reported in this section. We have obtained, for each block and for each core, the percentage of power of that block with respect to the overall core. We have seen that for the high-IPC core, the power goes mainly to the caches, the multi-ported instruction window and the clock distribution. For the low-IPC core, the power goes mainly to the clock, the ALUs and the caches. Moreover, for each block, the ratio between the low-IPC and high-IPC power as been obtained. As expected, all the blocks in the high-IPC core are more power hungry than their low-IPC counterparts. In average the low-IPC core consumes about 17% of the high-IPC core. The W_{LH} ratio will use the average value of 0.171.
- **Performance Ratio P_{LH}**: The SimpleScalar Tool Set v2.0 [4], running the integer and floating-point SPEC workloads, has been used to gather the performance numbers. We have obtained that the average across all the benchmarks draws a value of 0.210 for the P_{LH} parameter.

B. Low-IPC vs. Stream parameters

Here the parameters that compare a low-IPC processor and a stream core are discussed. They are (refer to Section 4): $A_{SL}(s) = (A_S/A_L)$ * Overhead_A(s), $W_{SL}(s) = (W_S/W_L)$ * Overhead_W(s) and $P_{SL} = P_S/P_L$ (always 1).

- **Area Ratio $A_{SL}(s)$**: As it was shown in Table I, the blocks that will be shared among the streams are: BPU, IC1, ITL, DEU, CIA, CFA, LSU, DC1, UC2 and DTL, and the blocks that will be replicated for each of the streams are: IWU, RFU, SIA, SFA and COU. The ratio of the shared area and the area of a single stream, ie the A_S/A_L ratio, is 0.054 based of the results previously obtained. The Overhead_A(s) depends on the number of streams in the MT architecture, and usually quadratically (since the overhead takes the form of a cross-bar to interconnect the streams to the shared resources). There is scarce data in the literature to accurately model this parameter. The floorplan area overhead reported in [5] for an SMT architectures accounts for the area of the streams themselves. From the layout schematic in [8], it can be inferred that the area of the central resource and address arbiters is less than 3% of the overall chip area in a CMP architecture with four cores. Based on the work in [14] and complemented with a private communication by the same authors we conclude that an overhead of 10% of the overall chip area is needed in a MT architecture with 32 streams. In this work we will model the overhead based on this last data point as: Overhead_A(s) = O_A * s^2 + (1 – O_A). For $s = 1$ the overhead ratio is 1, ie no overhead; for $s > 1$, the overhead ratio is proportional to the square of the number of streams. The value for O_A is chosen so that the resulting curve roughly approximates to the empirical data reported in the previously mentioned works. For 32 streams, it provides an overhead of 9.2%, and for 4 streams, an overhead of 1.3%.
- **Power Ratio $W_{SL}(s)$** : The ratio between the shared power and the single stream power, ie the W_S/W_L ratio, is derived based on the power results for the low-IPC core obtained with the Wattch tool. However, the power due to the clock distribution needs also to be divided into the shared and stream components since the Wattch tool does not break down the clock power into the different blocks. We do this by distributing the clock power into each of the blocks at a rate that depends on the size of the block; the larger the block is, it is expected to consume more clock power. With this approximation of the clock power per block, the W_S/W_L ratio becomes 0.138. The Overhead_W(s) depends on the number of streams in the MT architecture, and it is modeled similarly as the area overhead in the previous section. The $W_{SL}(s)$ ratio is then 0.273 * Overhead_W(s).

C. Performance Degradation parameters

Finally, the parameters that relate to the performance degradation are analyzed. The four parameters are (as explained in Section 4): $Dsync_{MP}(p)$, $Dshare_{MP}(p)$, $Dsync_{MT}(s)$ and $Dshare_{MT}(s)$.

In this work we will combine the MP degradation parameters into a single expression, and similarly for the two MT degradation parameters. Therefore, the two parameters will be:

- $D_{MP}(p) = Dsync_{MP}(p)$ * $Dshare_{MP}(p)$ (applies also to clusters)
- $D_{MT}(s) = Dsync_{MT}(s)$ * $Dshare_{MT}(s)$

Figure 2 shows several sets of data points for different MT, MP and CMP architectures regarding performance degradation obtained from different sources: (1) [21], (2) [6], (3) [19], (4) [7], (5,6) [2], (7) [9], (8,9) [13] and (10,11) [10]. As it can be observed, the degradation varies among the different data sets in part due to the different applications used in each data point to derive the performance results. We will model the MT and MP performance degrada-

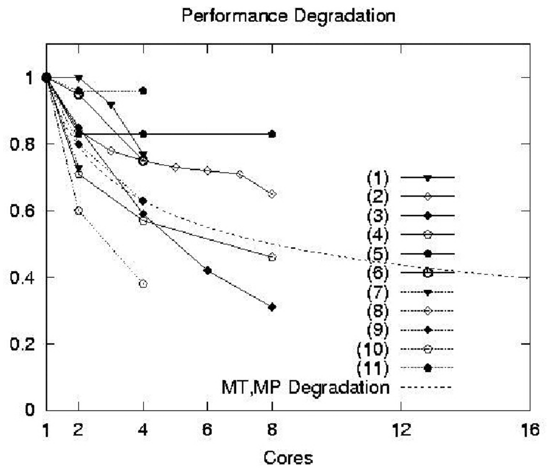

Figure 2. Performance degradation empirical data and model.

tion with the following equation:

$$\text{Degradation} = 1 / s^{(1/d)}$$

where d is a parameterized value that determines the degradation. Independently of d, the degradation is 1 (ie no degradation at all) when the number of streams is 1, and it is high when the number of streams is large. We will use a value d of 3 for our model; Figure 2 shows the modeling curve.

6. RESULTS

Figure 3 shows the area, performance and power metric comparisons of the MP, MT and CMT architectures with respect to the SS architecture. The power efficiency (power per unit of performance) and power density (power per unit of area) metrics are also shown. For the performance metric, the configurations with better performance than the SS are those above the 1.0 value; for the rest of the metrics, the best configurations are below the 1.0 line. Some conclusions can be derived:

- MP:
  - an MP with 6 or more cores will have higher power dissipation than the single high-IPC core in SS.
  - it takes about 8 cores or more to surpass the area of the single high-IPC core, and about 9 cores to beat the performance.
  - power efficiency is worse in the MP for 2 or more cores.
  - power density is constant and worse than in SS.
- MT:
  - MT favors the power and area, and maintains the performance, with respect to the MP.
  - the performance curve is the same as the MP one because the performance degradation penalty is the same for both MT and MP in this work.
  - power efficiency is always better than the MP and SS ones.
  - power density is always worse than in MP and SS.
- CMT:
  - power density remains the same as in MT.
  - as the number of clusters increase, the number of streams per cluster decrements to obtain better area and power than in the SS. Similarly, the minimum number of streams per cluster to obtain the same of better SS performance decreases.
  - power efficiency is still better than SS for two and four clusters (and for up to 128 and 256 streams respectively).

Table III is a summary of the results: for each bounded metric it shows the best architecture and its configuration in terms of processors/streams/clusters. The results in italics indicate that the equivalent SS architecture would perform better.

In particular, a MT architecture with 32 streams uses 64% less area than a superscalar architecture at the same power. With 2 clusters of 48 streams the performance can be 4.4 times at the same area. With 10 streams, the power can be reduced by as much as 61% at the same performance level. The power efficiency can be improved from 37% to 61% depending on the bounded metric. The power density metric, however, is always worse (higher) than the SS one, being MP better than both MT and CMT.

7. CONCLUSIONS

In this work, we study the intuitive fact that architectures with hardware support for threads are more power efficient than a more traditional single-threaded superscalar architecture. Our model of several parallel architectures shows that indeed these architectures outperform a single-threaded superscalar architecture when the application allows parallelization. In particular, a parallel architecture exists that (a) requires less area and power (to reach the same performance), or (b) achieves better power efficiency and less area (for the same power budget), or (c) has higher performance and better power efficiency (for the same area constraint), when compared to a single-threaded superscalar architecture.

Table III. Summary of the best MP, MT and CMP architectures and configurations (*p* processors, *s* streams/cluster, *c* clusters).

Scenario	***Area***		***Perf.***		***Power***	
	Conf	Value	Conf	Value	Conf	Value
Area-bound	–	–	2c,48s	4.4	***8p***	***1.37***
Perf.-bound	1c,10s	0.19	–	–	1c,10s	0.39
Power-bound	1c,32s	0.36	1c,34s	2.2	–	–

Scenario	***Power/Perf.***		***Power/Area***	
	Conf	Value	Conf	Value
Area-bound	3c,27s	0.63	***8p***	***1.39***
Perf.-bound	1c,11s	0.39	***10p***	***1.39***
Power-bound	1c,32s	0.46	***6p***	***1.39***

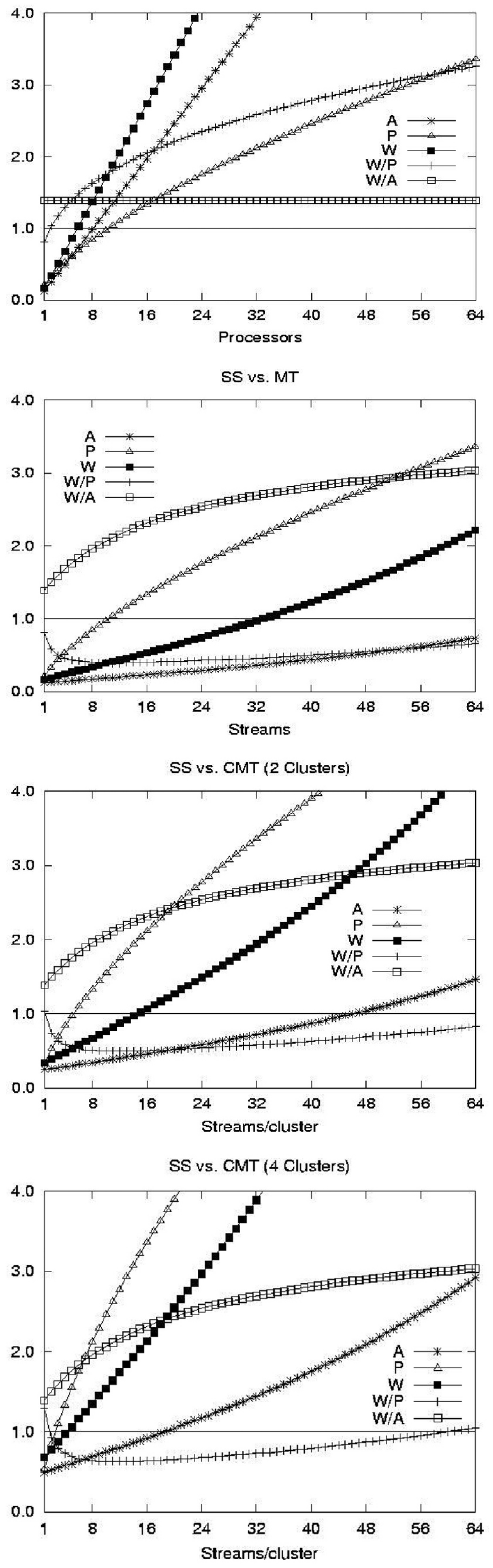

Figure 3. SS vs MP, SS vs MT, and SS vs CMT for 2 and 4 clusters (*A* area, *P* performance, *W* power, *W/P* power efficiency, *W/A* power density)

REFERENCES

[1] A. Agarwal and M. Levy. The KILL rule for multicore. In Proceedings of f the Design Automation Conference, June 2007

[2] L. Barroso and et al. Piranha: A scalable architecture based on single chip multiprocessing. In ISCA'00, June 2000

[3] D. Brooks, V. Tiwari and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimization. In International Symposium on Computer Architecture, June 2000

[4] D. Burger and T. Austin. The SimpleScalar Tool Set, Version 2.0. Technical Report, Computer Sciences Department, Univ. of Wisconsin-Madison, June 1997

[5] J. Burns and J.-L. Gaudiot. SMT layout overhead and scalability. IEEE Transactions on Parallel and Distributed Systems, 13(2), February 2002

[6] P. Crowley, M. Fiuczynski, J. Baer and B. Bershad. Characterizing processor architectures for programmable network interfaces. In Proceedings of the International Conference on Supercomputing, May 2000

[7] S. Eggers, J. Elmer, H. Levy, J. Lo, R. Stamm and D. Tulsen. Simultaneous multithreading: a platform for next-generation processors. IEEE Micro, September 1997

[8] L. Hammond, B. Hubbert, M. Siu, M. Prabhu, M. Chen and K. Olukotun. The Stanford Hydra CMP. IEEE Micro, 2000

[9] S. Kapil. Gemini: a power-efficient chip multi-threaded (CMT) UltraSPARC processor. Hot Chips XV, August 2003

[10] S.-W. Lee and J.-L. Gaudiot. Clustered microarchitecture simultaneous multithreading. In Proceedings of the International Conference on Parallel and Distributed Computing, August 2003

[11] Y. Li, C. Li, D. Brooks, Z. Hu and K. Skadron. CMP design space exploration subject to physical constraints. In Proceedings of the International Symposium on High Performance Computer Architecture, Februeary 2006

[12] S. Manne, A. Klauser and D. Grunwald. Pipeline Gating: speculation control for energy reduction. In International Symposium on Computer Architecture, June 1998

[13] D. Marr. HyperThreading technology in the Netburst microarchitecture. Hot Chips XIV, August 2002

[14] S. Melvin, M. Nemirovsky, E. Musoll, J. Huynh, R. Milito, H. Urdaneta and K. Saraf. Network Processor Design: Issues and Practices. Chapter: A Massively Multithreaded Packet Processor. Morgan Kaufman, 2003

[15] M. Monchiero, R. Canal and A. Gonzalez. Design space exploration for multicore architectures: A power/performance/thermal view. In Proceedings of the International Conference on Supercomputing, 2006

[16] E, Musoll, Predicting the usefulness of a block result: a microarchitectural technique for high-performance low-power processors. In International Symposium on Microarchitecture, November 1999

[17] K. Olukotun. The case for a single-chip multiprocessor. In ASPLOS, October 1996

[18] J. Seng, D. Tullsen and G. Cai. Power-sensitive multithreaded architecture. In Proceedings of the Internaltional Conference on Computer Design, 2000

[19] U. Sigmund, M. Steinhouse and T. Ungerer. On performance, transistor count and chip space assessment of multimedia-enhanced simultaneous multithreaded processors. In Workshop on Multi-threaded Execution, Architecture and Compilation (MTEAC-4), December 2000

[20] M. Steinhaus, R. Kolla, J. Larriba, T. Ungerer and M. Valero. Transistor count and chip-space estimation of simplescalar-based microprocessor models. In Workshop on Complexity-Effective Design, June 2001

[21] W. Yamamoto and M. Nemirovsky. Increasing superscalar performance through multistreaming. In International Conference on Parallel Architectures and Compilation Techniques (PACT), June 1995