Designing Power Efficient Hypermedia Processors

Chunho Lee†, Johnson Kin‡, Miodrag Potkonjak† and William H. Mangione-Smith‡
†Department of Computer Science and ‡Department of Electrical Engineering
University of California, Los Angeles, CA, USA
Email: {leec, miodrag}@cs.ucla.edu, {johnsonk, billms}@icsl.ucla.edu

Abstract

Distributed hypermedia system that supports collaboration is an emerging platform for creation, discovery, management and delivery of information. We present an approach to low power system design space exploration for distributed hypermedia applications. Traditionally, low power design and synthesis of application specific programmable processors has been done in the context of a given number of operations required to complete a task. Our approach utilizes the modern advances in compiler technology and architectural enhancements that are well matched to the compiler technology. This work is, to the best of our knowledge, the first attempt to address the need for synthesis of low power hypermedia processors. Also, this is the first work to address the power efficiency through exploiting instruction level parallelism (ILP) found in hypermedia tasks by an production quality ILP compiler.

Using the developed framework we conduct an extensive exploration of low power system design space for a hypermedia application under area and throughput constraints. The framework introduced in this paper is very valuable in making early low power design decisions such as architectural configuration trade-offs including the cache and issue width trade-off under area and throughput constraint, and the number of branch units and issue width.

1 Introduction

Hypermedia represents a combination of hypertext and multimedia technologies [4]. The concept of hypertext was proposed more than 50 years ago by V. Bush but T. Nelson is generally credited as the first to coin the term "hypertext" [5].

In recent years there have been significant advances in mobile computing which resulted in increasingly small and inexpensive computers and wireless networking. The roles mobile hypermedia systems must support present a unique challenge to system designers since most media tasks are computation-intensive and, by definition, they run in parallel side by side. Yet they are required to operate reliably and predictably while consuming as low power as possible. For example, tasks such as video and audio encoding/decoding, text processing, image processing, authentication and encryption/decryption should run in parallel to support hypermedia systems for collaboration. At the same time, due to the fact that the system must be mobile, they cannot afford to lavishly use power as their desktop counterparts can.

We present an approach to low power system design space exploration for distributed hypermedia applications that support. Traditionally, low power design and synthesis of application specific programmable processors has been done in the context of a given number of operations required to complete a task. Our approach utilizes the modern advances in compiler technology and architectural enhancements that are well matched to the compiler technology. Advances in compiler technology for instruction-level parallelism (ILP) have significantly increased the ability of a microprocessor to exploit the opportunities for parallel execution that exist in various programs written in high-level languages. State-of-the-art ILP compiler technologies are in the process of migrating from research labs to product groups [3, 9]. At the same time, a number of new microprocessor architectures having hardware structures that are well matched to most ILP compilers have been introduced. Architectural enhancements found in commercial products include predicated instruction execution, VLIW execution and split register files [2].

ISLPED99, San Diego, CA, USA

2 Target Architecture

The target architecture we use resembles a multiprocessor system with shared memory except that all the processors for a given hypermedia application scenario is assumed to be laid out on a single die. A media task is assigned to one and only one of the processors. More than one media application can be assigned to a processor if the given performance constraints are guaranteed to be met, i.e, all media tasks on the processor must be finished within a given time limit. Shared memory is used for data communication between tasks. Each processor maintains its own cache. When more than one media application are assigned to a single processor, flushing and refilling of the cache is needed and the run time measurement platform takes it into account.

We divide tasks into *quanta*. One of the benefits of using the notion of quantum is that it simplifies synchronization of several applications running on multiple processors. Aside from optimal use of allocated resources, running given tasks faster than required will provide no benefit to users. Hence, the use of quanta equivalent to the longest time frame with which tasks should be synchronized gives a convenient task assignment unit to allocate resources.

We develop a simple area model based on SA-110. We use different core models for the VLIW and superscalar machine configurations for the experiment. The difference comes mainly from the different complexity of the issue units found in superscalar machines and VLIW machines. We estimate the area of superscalar issue units based on the area complexity $O(n^2)$ since the complexity of dependency checking algorithm is $O(n^2)$. When a VLIW machine is considered, the issue unit area is generally of complexity $O(n)$ or sub-linear. The area of an arbitrarily configured superscalar machine is given by

$$Area = n_{issue}^2 A_{issue} + n_{ALU} A_{ALU} + n_{branch} A_{branch} + n_{mem} A_{mem} + A_{misc} \quad (1)$$

The terms n_{issue}, A_{issue}, n_{ALU}, A_{ALU}, n_{branch}, A_{branch}, n_{mem}, A_{mem} and A_{misc} are the issue width, the baseline issue unit area, the number of ALUs, the area of a single ALU, the number of branch units, the branch unit area, the number of memory units, the area of single memory unit and miscellaneous area, respectively.

We use the model for the cache power dissipation given in [6, 7]. There are five components in the core power dissipation

model we used: power dissipation by the issue unit, integer ALU, branch unit, memory unit, and other miscellaneous power consumption such as clock generator. The power dissipation model for superscalar machines is given by

$$E_{core} = n_{issue}^2 E_{issue} + n_{ALU} E_{ALU} + n_{branch} E_{branch} + n_{mem} E_{mem} + E_{misc} \quad (2)$$

The terms n_{issue}, E_{issue}, n_{ALU}, E_{ALU}, n_{branch}, E_{branch}, n_{mem}, E_{mem} and E_{misc} are the issue width, the baseline issue unit energy, the number of ALUs, the energy of a single ALU, the number of branch units, the branch unit energy, the number of memory units, the energy of single memory unit and miscellaneous energy, respectively.

The benchmark set used in this experiment is composed of complete media applications which are publically available and coded in a high-level language. We use 8 applications culled from available image processing, communications, cryptography and DSP applications. Detailed descriptions of the applications can be found in [8].

We use the IMPACT tool suit [1] to measure run times of media applications on various machine configurations. The IMPACT C compiler is a retargetable compiler with code optimization components especially developed for multiple-instruction-issue processors. The target machine for the IMPACT C can be described using the high-level machine description language (HMDES). A high-level machine description supplied by a user is compiled by the IMPACT machine description language compiler.

3 Problem Formulation

We collect run-times (expressed as a number of cycles) and cache performance of the benchmarks on 175 different machine configurations (25 cache configurations for 7 processor configurations).

Measured run-times and cache performance of benchmarks through simulations are used to compute the energy based on the power model. The simulated power consumption measurements we use are made with two levels of the shutdown technique: component level and system level. We assume that when a component (e.g., ALU, branch unit, etc.) is idle, it can be shutdown (lower level shutdown). Also, when there is no task to execute, a processor can be shutdown (higher level shutdown).

Low Power Hypermedia Synthesis Problem

Instance: Given a set T of n media applications, $a_i, i = 1, 2, ..., n$, a set of m processors, $p_j, j = 1, 2, ..., m$, the run times e_{ij} and the cache performance f_{ij} of the media applications a_i, $i = 1, 2, ..., n$ on the machines $c_j, j = 1, 2, ..., m$ and constants C, E and P,

Question: Is there a multisubset (subset in which more than one instance of a processor can be included) M of k processors, $p_l, l = 1, 2, ..., k, 1 \leq k \leq n$, such that $\sum_{j \in M} A_{p_j} \leq C$, $\max_{j \in M} \sum_{i \in t_j} e_{ij} \leq E$ and $\sum_{j \in M} w_{t_j j} \leq P$? A_{p_j} is the area of the machine p_j, t_j is the set of tasks that are assigned to the machine j and $w_{t_j j}$ is the power consumption of the processor p_j when a task set t_j is assigned to the processor.

Theorem. The Low Power Hypermedia Synthesis Problem is NP-complete.

Proof. We transform the Equal Subset Problem (ESP) to a special case of the Low Power Hypermedia Synthesis Problem. For a given task set T, $|T| = n$ and a set M of 2 identical processors, each integer number in the ESP is mapped to the power consumption measurement, which is obtained by applying shutdown technique, of each media task in T. In order to minimize the power consumption by applying the voltage scaling, we need to assign the tasks to the processors in such a way that the sums of the assigned tasks' power consumption measurements obtained using the shutdown technique should be as close to each other as possible.

4 Synthesis Algorithm

We develop heuristic algorithms for the low power hypermedia processor. We use a simulated annealing based algorithm to allocate resources given area and performance constraints. Allocated processors are then used by a force-directed heuristic [10] to assign the set of hypermedia tasks.

We supply five sets of inputs to the algorithm: a set of hypermedia tasks, a set of available processors, area constraints, timing constraints and a set of power consumption measurements of each task on each available processor. The initial power consumption measurements is obtained by measuring power consumption of each task on each processor with the shutdown technique. The algorithm returns two sets of outputs: a set of allocated processors and their assigned tasks and the total power consumption after voltage scaling.

There are four major components in the simulated annealing based algorithm: the power consumption measurement, the neighbor solution generation, the temperature update function, the equilibrium criterion and the frozen criterion. Firstly, the power consumption measurement is made after the set of hypermedia tasks are assigned to a tentatively allocated processors. The tentative assignment is done by a force-directed heuristic. Secondly, a neighbor solution is generated by replacing one processor in the current tentative solution set by one from the available processor set including a null processor (i.e., not adding any). Thirdly, the temperature is updated by the standard geometric cooling schedule. Fourthly, the equilibrium criterion is specified by the number of iterations of the inner loop, which is set to 65. Lastly, the termination criterion is given by the temperature. If the temperature falls below 0.1, the simulated annealing algorithm stops.

The force-directed heuristic is a global optimizing algorithm in that it makes one decision at a time by comparing all the possible decision choices and selecting one that is least constraining to the future decisions. There are three major steps in each iteration in which one decision is made: construction of distribution graphs (DG), computation of force (F) and self force (SF) for each possible decision choice. For example, if we have a set of 8 tasks and a set of 3 processors, there are 24 possible decision choices. Thus 24 $SF's$ are computed and we choose the smallest SF. For example, $SF_{a_1 p_2}$ being the smallest means the assignment of the task a_1 to the processor p_2 is the choice selected by force-directed heuristic. When each and all tasks are assigned, voltage scaled power consumption measurements are made and returned.

5 Experimental Results

We evaluate the tools and algorithms by running extensive experiments ranging from the area constraint of $30mm^2$ to $155mm^2$ and the timing constraints 0.03 through 0.09 seconds. The timing constraint 0.03 seconds represents the time to process 1 *mpeg* frame at the rate of 30 frames per second. The implementation technology is assumed to be 0.35μ. For each area constraint and timing constraint combination, we obtain three different solutions using the optimizing algorithm (OA: the combination of simulated annealing based algorithm and force-directed heuristic), the simple-minded division algorithm (SA) and the largest possible processor selection (LA). SA simply divides a given area constraints into two or more processor areas and allocates identical processors. SA uses the same force-directed heuristic used by OA to assign tasks to allocated processors. LA allocates one largest processor that satisfies the area constraint.

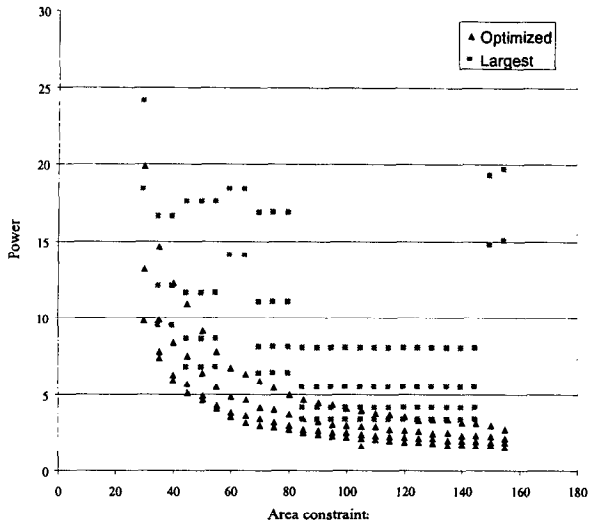

Figure 1: Power consumption results measurements made for a range of synthesized system: power consumption ranges are shown for each area and performance constraints

Figure 1 shows power consumption of synthesized systems by OA and LA for various area and timing constraints. Area constraints are represented by the horizontal axis and for each area constraint a number of results for various timing constraints are plotted along the vertical axis. The scatter plot clearly shows obvious strength of the OA. Figure 2 shows the same results but only for the timing constraint 0.03. It compares the OA, SA and LA. The results indicate that even a simple-minded collection of identical processors can perform better than one monolithic processor. The power improvements achieved by OA compared to LA range from 39% to 70% (average 54%). SA improves power consumption by 20% to 59% (average 41%) over LA. For some area constraint region (bigger than 85 mm^2), the LA performs slightly better than SA. The reason behind this generally impressive results of OA is that customized processors for each tasks provide more opportunity for power savings from two directions. The ILP compiler produces optimized code in terms of the number of execution cycles. The other is from the efficient use of allocated resources. If we can allocate resources in such a way that all of them are used all the time and the execution cycles are minimized given constraints, we can maximize the opportunity of voltage scaling. Although we shutdown individual components when they are not used, voltage scaling is in general more effective method to save power. When a large monolithic processor is used, it may offer more opportunity to shutdown individual components of the system since resource requirements of each individual task are different. On the other hand, since we need to meet the timing constraints, there are less opportunity to scale voltage.

Most significant power reduction is observed area constraints between 30 and $65mm^2$. Bigger area than $65mm^2$ offers minimal power improvements. Interestingly, the results shown in Figure 2 show that more than one machines are required to maximize power savings. This is consistent with the fact that specifics of individual applications should be used to fully utilize limited amount of resources. In this experiment, we found that $65mm^2$ of area is enough to fit in a set of processors that provides ability to run the hypermedia application in a power efficient manner.

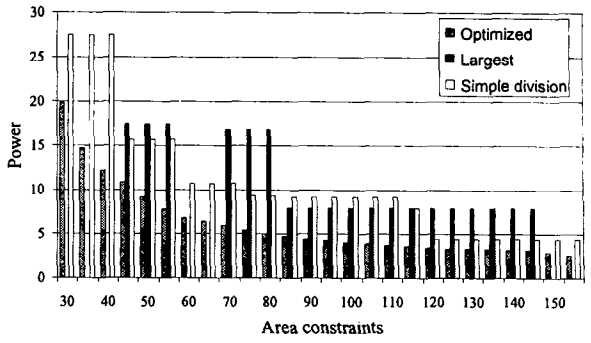

Figure 2: Power consumption measurements made for a range of synthesized system: power consumption ranges are shown for each area and one performance constraint (0.03)

6 Conclusion

The availability of production quality ILP compilers and commercial DSPs with architectural enhancements stimulated the idea of programmable processors that are tuned to specific applications.

We developed an effective framework and efficient algorithms and tools to rapidly explore low power hypermedia processor design space. We conducted an extensive exploration of the low power design space under area and timing constraints. The framework addresses the need for the low power hypermedia design by exploiting the ILP found in media applications by ILP compilers that target multiple-instruction-issue machines although we found that low power design of the hypermedia system does not benefit much from it. Nevertheless, the algorithms and tools we developed showed impressive results.

We found that the framework introduced in this paper can be very valuable in making early design decisions such as power and architectural configuration trade-off, cache and issue width trade-off under area constraint, and the number of branch units and issue width.

References

[1] P. P. Chang, S. A. Mahlke, W. Y. Chen, N. J. Warter, and W. m. W. Hwu. IMPACT: An architectural framework for multiple-instruction-issue processors. In *International Symposium on Computer Architecture*, 1991.

[2] R. P. Colwell, R. P. Nix, J. J. O'Donnell, D. B. Papworth, and P. K. Rodman. A VLIW architecture for a trace scheduling compiler. In *Proceedings of ASPLOS-II*, pages 180–192, 1982.

[3] J. A. Fisher. Trace scheduling: A technique for global microcode compaction. *IEEE Transactions on Computing*, C-30:478–490, 1981.

[4] K. Gronbaek and R. Trigg. Design issues for a Dexter-based hypermedia system. *Communications of ACM*, 37(2):41–49, February 1994.

[5] F. Halasz. Reflections on note-cards: Seven issues for the next generation of hypermedia systems. *Communications of ACM*, 31(7):836–852, July 1988.

[6] M. B. Kamble and K. Ghosse. Analytical energy dissipation models for low power caches. In *Proceedings 1997 International Symposium on Low Power Electronics and Design*, pages 143–148, 1997.

[7] J. Kin, M. Gupta, and W.H. Mangione-Smith. The filter cache: An energy efficient memory structure. In *Proceedings of 30th Annual International Symposium on Microarchitecture*, 1997.

[8] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. Mediabench: A tool for evaluating and synthesizing multimedia and communications systems. In *International Symposium on Microarchitectures*, 1997.

[9] W. m. W. Hwu, S. A. Mahlke, W. Y. Chen, P. P. Chang, N. J. Warter, R. A. Bringmann, R. G. Ouellette, R. E. Hank, T. Kiyohara, G. E. Haab, J. G. Holm, and D. M. Lavery. The superblock: An effective technique for VLIW and superscalar compilation. *Journal of Supercomputing*, 1993.

[10] P. G. Paulin and J. P. Knight. Force-directed scheduling for the behavioral synthesis of ASICS. *IEEE Transactions on CAD*, 8(6):661–679, June 1989.