Power Efficient Mediaprocessors: Design Space Exploration

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

Abstract

We present a framework for rapidly exploring the design space of low power application-specific programmable processors (ASPP), in particular mediaprocessors. We focus on a category of processors that are programmable yet optimized to reduce power consumption for a specific set of applications.

The key components of the framework presented in this paper are a retargetable instruction level parallelism (ILP) compiler, processor simulators, a set of complete media applications written in a high level language and an architectural component selection algorithm. The fundamental idea behind the framework is that with the aid of a retargetable ILP compiler and simulators it is possible to arrange architectural parameters (e.g., the issue width, the size of cache memory units, the number of execution units, etc.) to meet low power design goals under area constraints.

1 Introduction

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. Recently, Hong and Potkonjak [14] presented a low power synthesis approach based on the minimization of the number of operations.

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 [1, 8, 15, 23, 24]. 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 [7, 31]. Multi-gauge arithmetic (or variable-width SIMD) is found in the family of MPACT architectures from Chromatic [17] and the designs from MicroUnity [12]. Most of the multimedia extensions of programmable processors also adopt this architectural enhancement [22, 26].

We investigate an approach to rapidly explore the design space of low power application-specific programmable processors (ASPP), in particular mediaprocessors. We focus on a category of processors that are general-purpose (programmable) but optimized to reduce power consumption for a specific set of applications.

The key components of the framework presented in this paper are a retargetable ILP compiler, instruction level simulators, a set of complete media applications written in a high level language and an architectural component selection algorithm. The fundamental idea behind the framework is that with the aid of a retargetable ILP compiler and simulators it is possible to arrange architectural parameters (e.g., the issue width, the size of cache memory units, the number of execution units, etc.) to meet low power design goals and satisfy area constraints.

In the following section we illustrate the key ideas on which this work is based using a simple example. We discuss the related works and our contribution in Section 3. Section 4 presents the preliminary materials including the power and area model, benchmarks, experiment platform such as tools and example set of results obtained using the tools. Our approach in this project is explained in Section 5 in detail. Section 6 formulates the search problem defined in the previous section in formal terms. The solution space exploration strategy and algorithm is described in Section 7. Extensive experimental results of the tools and algorithms we develop for the system-level synthesis of application-specific programmable processors are reported in Section 8. Finally, Section 9 draws conclusions.

2 A Motivational Example

Average power consumption in CMOS circuits is given by $P = \alpha C_L V_{dd}^2 f$, where f is the system clock frequency, V_{dd} is the supply voltage, C_L is the load capacitance and α is the probability of switching activity (the probability of a $0 \rightarrow 1$ transition during a clock cycle) [3]. The most effective means to reduce power consumption of a processor is to lower the supply voltage since the power consumption is quadratic function of the voltage [2]. The voltage reduction comes with the drawback that the circuit delay is increased requiring a longer cycle time. Another power optimization technique is system shutdown [29], which is usually less effective than voltage reduction.

We illustrate the key ideas of our approach using a simple example. Consider that we have a number of architectural choices in designing low power mediaprocessors such as the number of functional units, issue width, cache sizes, etc. A retargetable ILP compiler generates optimized codes for a machine configuration. Table 1 shows a baseline machine in the first row and an optimized machine in the second and third rows. The number of cycles to complete a task on each machine is shown in the second column of the table. The Energy savings of a processor configuration with respect to a baseline machine are in the sixth column. An application we are interested in has 600 operations on the baseline machine. The baseline machine is a single-issue machine with one branch unit, one execution unit, one memory unit, 0.5 KB of I-cache and D-cache units. The supply voltage of the baseline machine is 3V and cycle time is normalized to 1. The power consumption of the baseline machine to execute the application of interest is normalized to 1. The power consumption of a new machine with more hardware (namely, four-issue, four ALUs, one branch, four memory units, 2 KB of I-cache and D-cache units) is higher than that of the baseline machine. But since the new machine can execute the application much faster with fewer operations, we can use either shutdown or voltage scaling technique to reduce power consumption while meeting the performance level of the baseline machine. Assume that the power consumption of the new machine with shutdown technique is 0.8. Then the new machine can execute the application with a fraction of the baseline machine energy when the

Configuration	Cycles	Supply Voltage	Cycle Time	Run Time	Power	Area (mm^2)
(1, 1, 1, 1, 0.5, 0.5)	600	3	1	1	1	14.01
(4, 4, 1, 4, 2, 2)	200	3	1	1/3	0.8	43.28
(4, 4, 1, 4, 2, 2)	200	1	3	1	0.2	43.28

Table 1: An illustrative power dissipation example: a machine configuration consists of (issue width, number of ALUs, number of branch units, number of memory units, size of instruction cache(KB), size of data cache(KB))

voltage scaling technique is applied.

3 Related Works and Our Contributions

Since power optimization has been separately pursued in both architecture and computer-aided design communities, we briefly summarize related works in both communities.

Kalavade and Lee [18] proposed a framework that addresses the need for managing complex system-level design environment. Singh et al. [28] presented a good survey on works on computer-aided design of low power systems across all level of design abstraction. Chandrakasan et al. [3] showed significant power optimization by using transformations on many computation-intensive DSP applications. Methods to explore trade-offs between voltage scaling, throughput, and power are reported in [11, 27]. A power reduction scheme targeted at fully hardwired design by minimizing switching activities is presented in [6]. Many power minimization techniques targeted at programmable platforms have been introduced [4, 30].

We adopt a methodology of system synthesis combining the key paradigms of computer aided design and architecture communities. Following the tradition of CAD field, we develop an accurate power estimate and aggressive optimization algorithms. We follow the tradition of the architecture field by using comprehensive real-life benchmarks and production quality compilation and simulation tools. This combination enables us to build a unique framework of system-level synthesis and to gain valuable insights about design and use of application-specific programmable processors for modern applications.

Unlike previous works, we use a set of complete applications written in a high-level language as benchmarks. The framework is based on the observation that ILP compilers can find significant parallelism in typical media applications written in a high level language. Combined with architectural enhancements such as VLIW and superscalar schemes, the ILP can be exploited to the benefit of designing power optimized ASPPs. We incorporate into the machine model the impact of cache memory units in low power design, which, with the realistic benchmarks, makes the model more realistic.

Using the developed framework we conduct an extensive exploration of the low power ASPP design space in the face of area limits. It is clear that there is useful ILP in the typical media and communication applications [21]. The framework addresses the need for the low power ASPP design by exploiting the ILP found in media applications by ILP compilers that target VLIW and superscalar machines.

The experiments presented in this paper are focused on how many machine configurations should be selected in order to reduce power consumption across all the benchmarks under various area constraints. The objective is minimization of selected machine configurations, thereby maximizing the number of benchmarks that can be run on a processor in a power optimized fashion as though it is optimized for each individual benchmark.

We find that the framework introduced in this paper can be 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 constraint, and the number of branch units and issue width.

4 Experiment Platform

We use StrongArm SA-110 as the baseline architecture for the analysis [25]. The device is a single-issue processor with the classic five stage pipeline. The SA-110 has an instruction issue unit, integer execution unit, integer multiplier, memory management unit for data and instructions, cache structures for data and instructions, and some other units such as phase locked loop (PLL). We simulate the power consumption model based on the power dissipation data given in [25]. It is fabricated in a 0.35-μm three-metal CMOS process with 0.35V thresholds and 2V nominal supply voltage. When it runs Dhrystone 2.1 at 2V nominal supply voltage, the total power is 3.3mW/MHz (528 mW at 160MHz).

We develop a simple area model based on SA-110. The area of the chip is 49.92mm^2 (7.8mm $\times$ 6.4mm) an approximately 25% of the die area is devoted to the core (12.48mm^2). The issue unit and branch unit occupies approximately 20% of the die area (2.50mm^2). The integer ALU and load/store unit consume roughly 20% of the die area (2.50mm^2). The DMMU and IMMU (data and instruction MMU) occupies about 40% (5mm^2) of the area. The rest of the core area is used by other units such as the write buffers and bus interface controller. We assume that the area of miscellaneous units is relatively stable in the sense that it does not change as we increase the issue width or cache sizes. 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

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

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 did not include floating-point units in any machine configurations because the applications we used have mostly integer operations. Cache area is calculated using the Cache Design Tools [9]. A set of example area estimates for superscalar machines with different cache and core configurations are shown in Table 2. In the rest of the paper we describe a machine configuration by a 6-tuple as shown in Table 2.

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

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

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

Configuration	Issue	IALU	Branch	Mem	Cache	Total
(1, 1, 1, 1, .5, .5)	1.25	2.5	1.25	5.0	1.53	14.01
(2, 2, 1, 2, 1, 1)	5.0	5.0	1.25	10.0	2.54	26.27
(4, 4, 1, 4, 2, 2)	20.0	10.0	1.25	20.0	4.55	58.28
(8, 8, 1, 8, 4, 4)	80.0	20.0	1.25	40.0	8.56	152.29
(4, 4, 2, 4, 8, 8)	20.0	10.0	2.5	20.0	16.55	71.53
(8, 8, 2, 8, 4, 4)	80.0	20.0	2.5	40.0	8.56	153.54
(8, 8, 4, 8, 8, 8)	80.0	20.0	5.0	40.0	16.55	164.03

Table 2: Superscalar machine configuration examples and their area estimates (mm^2): a machine configuration consists of (issue width, number of ALUs, number of branch units, number of memory units, size of instruction cache(KB), size of data cache(KB))

units, the energy of single memory unit and miscellaneous energy, respectively.

The model for the cache power dissipation is given by

$$E_{cache} = E_{bit} + E_{word} + E_{output} + E_{ainput} \quad (3)$$

The terms E_{bit} is the energy dissipated by the bit lines due to precharging, readout and writes, E_{word} the energy dissipated in the word lines, E_{output} the energy dissipated in the data and address output lines, and E_{ainput} the energy dissipated on the address input lines. For further descriptions of each component of the model, refer to [19, 20].

The set of benchmarks used in this experiment is composed of complete applications which are publically available and coded in a high-level language. The collection has 21 applications culled from available image processing, communications, cryptography and DSP applications. Detailed descriptions of the benchmarks can be found in [21].

We use the IMPACT tool suit [5] to collect performance information of benchmarks 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.

IMPACT provides cycle-level simulation of both the processor architecture and implementation. The optimized code is consumed by the Lsim simulator. At simulation time, Lsim takes cache structure information provided by a user.

5 Approach

We collected run-times (expressed as a number of cycles) of the benchmarks on 175 different machine configurations (25 cache configurations for 7 processor configurations). First we build executables of the benchmarks on seven different architectures. They are machines with a single branch unit and one of the one-, two-, four-, and eight-issue units, machines with two branch units and one of the four- and eight-issue units, and machines with four branch units and a eight-issue unit. The IMPACT compiler generates aggressively optimized code to increase ILP for each architecture. The optimized code is consumed by the Lsim simulator. We simulate the benchmarks for a number of different cache configurations. For each executable of a benchmark, we simulate 25 combinations of instruction and data caches ranging from 512 bytes to 8 KB.

Measured run-times of benchmarks through simulations are used to compute the energy based on the power model described in Section 4. The power dissipation numbers are normalized with respect to a baseline machine power dissipation. We selected as a baseline configuration a machine with one branch unit, one-issue unit, 512 bytes of instruction cache, and 512 bytes of data cache. We take the inverse of the normalized energy to get the power savings factor. The power savings factor is used in the machine selection algorithm.

After all the simulations are completed and all the power savings factors for all applications on all machine configurations are obtained, we run a search algorithm to select machine configuration sets under various area constraints. For each area constraint, we eliminate all machines that do not satisfy the area requirement. From the machines that satisfy the area requirement, we eliminate all the dominated machines. By dominated, we mean a machine dissipates power more than or equal to that of another machine for *all* benchmarks. Finally, we perform K-selection algorithm (see Section 6) to select a set of machine configurations that runs the benchmark set with minimum power dissipation.

6 Selection Problem Formulation

Informally, the problem can be stated as follows. Given an area constraint and power savings factors of benchmarks on machines that fit into the given area, we select a subset of the machines in such a way that the geometric mean of power savings factors across all the benchmark is maximized and the subset size is kept small.

We normailze the energy with respect to a baseline since we are not interested in the sum of energy [16]. The sum of energy does not reflect the power dissipation effect of smaller benchmarks (ones that inherently dissipate less power) in the presence of bigger benchmarks (ones that consumes more power than others). In some cases, a benchmark that takes a long time to complete due to its big data, thus consumes much energy, dominates the sum of energy.

We use geometric mean to summarize the power savings factors of the selected machines since we normalize the measurements [13].

We define the problem using more formal Garey-Johnson format [10].

Selection problem

Instance: Given a set of n benchmarks, a_i, $i = 1, 2, ..., n$, k machine configurations, m_j, $j = 1, 2, ..., k$, the power savings factors E_{ij} of the benchmarks a_i, $i = 1, 2, ..., n$ on the machines m_j, $j = 1, 2, ..., k$ with respect to a baseline machine and constants K and C,

Question: Is there a set M of K machine configurations, c_p, $p = 1, 2, ..., K$, such that $\sqrt[n]{\prod_{i=1}^{n} \max_{j \in M} E_{ij}} \leq C$?

To determine the constant K we divide the problem into two sub-problems, namely, w-selection problem and K-selection problem. Starting from $w = 1$ we iteratively increase w until the benefit of increasing w is less than a given threshold ρ. Formally the sub-problems are stated as follows.

w-selection problem: Given a set of n benchmarks, $a_i, i = 1, 2, ..., n$, k machine configurations, m_j, $j = 1, 2, ..., k$, the power savings factors E_{ij} of the benchmarks a_i, $i = 1, 2, ..., n$ on the machines m_j, $j = 1, 2, ..., k$ with respect to a baseline machine and constants w,

$$\textbf{Maximize}: \quad D_w = \sqrt[n]{\prod_{i=1}^{n} \max_{j \in P} E_{ij}}, \quad (4)$$

where P is the selected machine set of size w.

The size of the machine set is determined by an iterative test of comparing D_w and D_{w+1}. Since the D is monotonic, we continue to evaluate D and compare them using Equation 5

```
for a set of machine configurations {
  for a set of benchmarks {
    generate an executable of a benchmark for the target machine;
    measure speed-up factors with respect to the baseline machine;
  }
}

eliminate all machines that don't satisfy area constraint;
eliminate all machines that are dominated by at least one other machine;
R = 1;
i = 1;
D_{i-1} = 1;
while ( (R ≥ ρ) && (i ≤ k) ) {
  D_i = branch-and-bound( D_{i-1}, i );
  R = (D_i - D_{i-1}) / D_i;
  D_{i-1} = D_i;
  i++;
}
```

```
branch-and-bound( D_{i-1}, w ) {
Stack s;
Node c;
  D_best = D_{i-1};
  Generate_new_nodes( c, D_best, s );
  while( s is not empty ) {
    c = take a node out of s;
   if( bound() == FALSE ) then continue;
   else {
    if the number of selected machines is less than w then
      Generate_new_nodes( c, s );
    else {
      D = nth_root( Π_{i=1}^{n} max_{j∈P} E_{ij} );
      if( D > D_best ) then D_best = D;
    }
   }
  }
  return D_best;
}
```

```
Generate_new_nodes( c, s ) {
  for a set of possible choices {
    est = estimate( c, choice );
    insert the new choice and sort s based on est;
  }
}
```

Figure 1: System-level synthesis of power optimized application-specific programmable machines

until we reach a point where the benefit of the set size increase diminishes to a certain degree.

K-selection problem:

$$\min\{w|\rho \leq \frac{D_{w+1} - D_w}{D_w}, w = 1, 2, ...k-1\}, \qquad (5)$$

where D_w is given by Equation 4 and ρ is a cut-off ratio.

7 Solution Space Exploration: Strategy and Algorithms

The algorithm for system-level synthesis of application-specific programmable processor is given in Figure 1. In general, the size of the search problem is dramatically reduced by eliminating machine configurations that do not satisfy a given area constraint and those that are dominated by at least one other machine. Consequently, a smaller number of machines needs to be considered.

Since the simulation takes very long time (more than a week on a SPARC5 machine), we try to come up with optimum solutions. Unless the optimal algorithm takes unreasonably long time, it is worth to obtain optimum solutions. The search for optimum solution is organized using an implicit enumeration method. In particular, we adopt a branch-and-bound algorithm shown in Figure 1 to speed up the selection. The algorithm completes less than 3 hours on a SPARC5 for a range of area constraints.

The branch-and-bound algorithm consists of two major components: branching and evaluation. The branching step takes the current state of selection (a node in the search tree) and generates a number of new nodes by adding an available (still not considered in particular search path of the search tree) machine to the current state of selection. As shown in Figure 1, it examines to see if adding a machine to the current state of selection can result in a better solution than the current best solution found. Initially, the current best solution is set to the previous best solution. The previous best solution is the best solution found for the machine set size less than the current search size by one. The branching is bounded by the bounding function. The bounding function compares the current node and a candidate processor with the best node of the same size found. The node size is the number of processors. If the current node and the candidate are dominated by the best node then we cut the path off from search. We compute the lower bound of the geometric mean of the maximum speed-up factors of each benchmark. The lower bound is obtained by using a steepest descent algorithm. The steepest descent algorithm selects machines in the order that the biggest improvement can be achieved. If the estimate is greater or equal to the current best solution, we have an opportunity to find a better solution than the current best solution by exploring the search path. Otherwise, there is less chance of obtaining a better solution. We sort the search order based on the lower bound so to increase the bounding rate.

8 Experimental Results

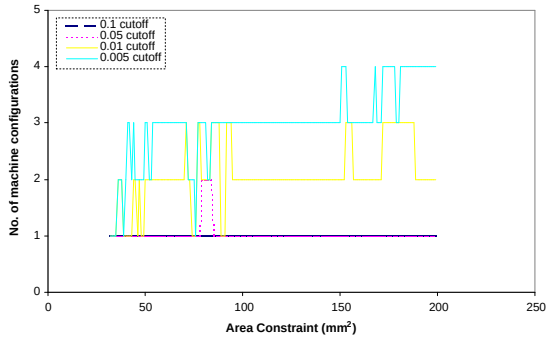
We evaluate the tools and algorithms by running extensive experiments ranging from the area constraint of $30mm^2$ to $200mm^2$. The implementation technology is assumed to be 0.35μ. For each area constraint, we obtain the optimum set of machine configurations for cut-off values 0.1, 0.05, 0.01, and 0.005. The cut-off values are ρ's as defined in Equation 5.

We conducted two sets of experiments using machine shutdown and voltage scaling. For each set of experiments, we use two machine area and power models, namely, superscalar machines with quadratic complexity issue units and VLIW machines with linear complexity issue units. We found that the two models exhibit quite different behaviors in terms of power consumption.

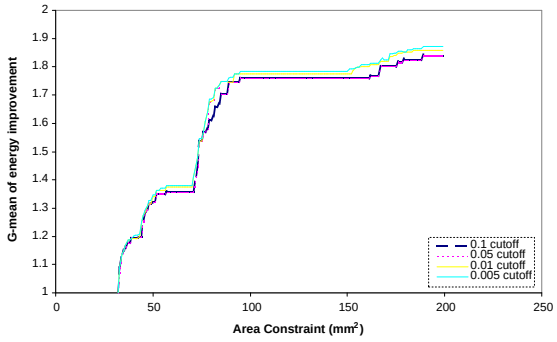
Figure 2 and Figure 3 show results from the power shutdown method. We can draw three conclusions from the results. First, empirically, the cut-off value 0.1 is good enough for most cases since smaller cut-off values do not result in more power savings as shown in Figure 2(b) and Figure 3(b). Second, when only the shutdown technique is available, there is no compelling reason to have more than one architecture from the power minimization point of view regardless of the area and power model used. Since we can use the cut-off value 0.1 without being too conservative, the number of selected machines at the cut-off value 0.1, shown in Figure 2(a) and Figure 3(a), is one throughout the entire area constraints. Finally, it is clear that the shutdown technique is ineffective to use with our framework. In other words, using this technique, it is impossible to exploit individual characteristics of applications to reduce power consumption.

We see no benefit at all of using superscalar machines as a power reduction scheme. Figure 3(b) indicates that as soon as we include multiple-issue issue units into the superscalar model (machines bigger than $45mm^2$), which possibly boost performance thus can reduce power, the power consumption of the issue unit overwhelms other benefits.

The results for voltage scaling are shown in Figure 4 and 5. In this set of results, we obtain greater power consumption improvement, which is consistent with the fact that voltage scaling is generally more effective than shutdown. The results agree with those of the shutdown case, i.e., the cut-off value of 0.1 is a good choice since the power improvement factors of different cut-off values are

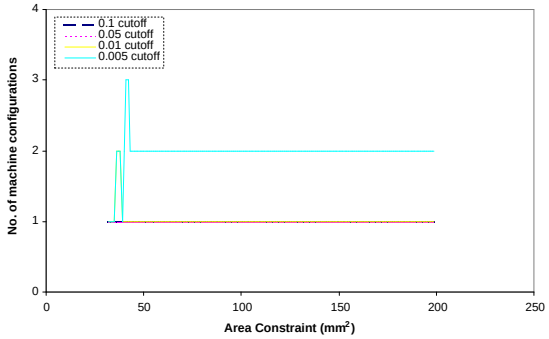


(a) Selection of machine configurations

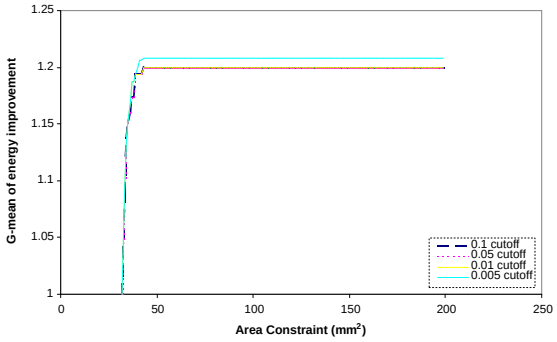


(b) Power improvement factors

Figure 2: VLIW (linear complexity issue unit area) model: Selected processor configurations and power improvement factors corresponding to selections when the shutdown technique is applied



(a) Selection of machine configurations



(b) Power improvement factors

Figure 3: Superscalar (quadratic complexity issue unit area) model: Selected processor configurations and power improvement factors corresponding to selections when the shutdown technique is applied

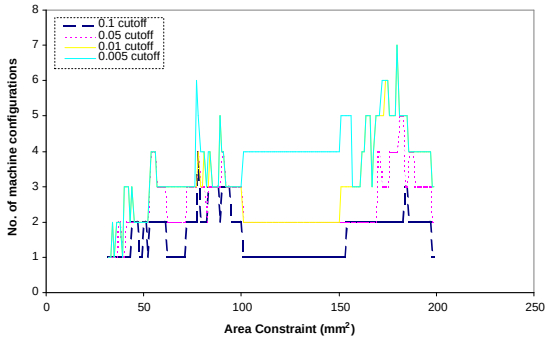
almost the same. Also, the results indicate that the VLIW machines with voltage scaling are best candidates that can exploit the capability of our framework. We observe that up to $100mm^2$, we get significant power savings by increasing hardware resources. It is evident that under the scheme, we can better utilize ILP in applications to the benefit of energy reduction to the extent that a ILP compiler can find. We found that there are more architectural choices and decisions to make for different applications when less area is available. On the other hand, although it performs better than the shutdown case, the superscalar machines with voltage scaling showed disappointing results. It provided moderate improvements up to the factor of 3 at around $60mm^2$ of area. The curve is identical to that of shutdown case except that it is a little shifted forward.

As shown in Figure 4(b), the energy improvement diminishes after $100mm^2$. This can be due to one or both of the following limitations. The media applications we used is likely to have inherent ILP upper bounds. It is also possible that the current upper bound revealed in this experiment has a lot to do with the current ILP compiler technology.

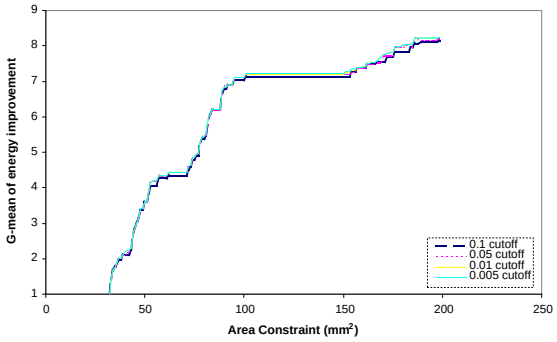
Interestingly, the results from VLIW machines shown in Figure 4 show that up to $100mm^2$ of area, more than one machines are required to maximize power savings. This is consistent with the fact that specifics of individual applications should be used to fully utilize limited amount of resources. In this experiment, we found that $100mm^2$ of area is enough to fit in a general-purpose machine to run all the benchmarks in a power efficient manner.

9 Conclusion

The availability of production quality ILP compilers and commercial DSPs with architectural enhancements (e.g., predicated instruction execution, VLIW execution, split register files and multi-gauge

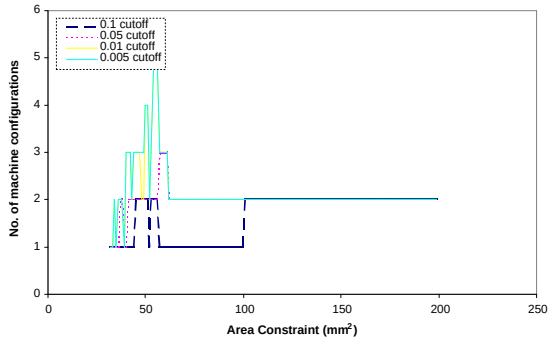


(a) Selection of machine configurations

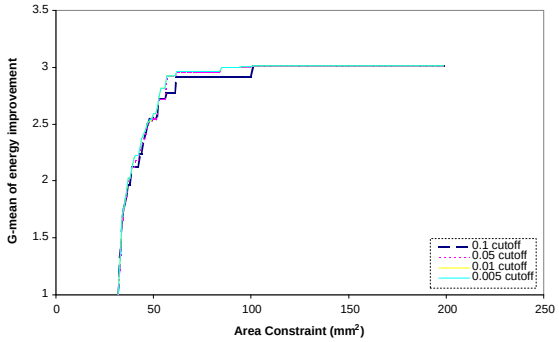


(b) Power improvement factors

Figure 4: VLIW (linear complexity issue unit area) model: Selected processor configurations and power improvement factors corresponding to selections when the voltage scaling is used



(a) Selection of machine configurations



(b) Power improvement factors

Figure 5: Superscalar (quadratic complexity issue unit area) model: Selected processor configurations and power improvement factors corresponding to selections when the voltage scaling is used

arithmetic (or variable-width SIMD)) stimulated the idea of programmable processors that are tuned to specific applications.

The key components of the framework presented in this paper are a retargetable instruction level parallelism (ILP) compiler, instruction level simulators, a set of complete media applications written in a high level language and an architectural component selection algorithm.

We conducted an extensive exploration of the low power ASPP design space under area constraints. It is clear that there is enough ILP in the typical media and communication applications. The framework addresses the need for the low power ASPP design by exploiting the ILP found in media applications by ILP compilers that target VLIW and superscalar machines.

We found that the framework introduced in this paper can be very valuable in making early design decisions such as area 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] S. Banerjia, W. A. Havanki, and T. M. Conte. Treegion scheduling for highly parallel processors. In *Euro-Par*, pages 1074–1078, Passau, Germany, 1997.

[2] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. Broderson. Optimizing power using transformations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(1):12, 1995.

[3] A. P. Chandrakasan, S. Sheng, and R. W. Broderson. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, 1992.

[4] A. P. Chandrakasan, M. Srivastava, and R. Broderson. Energy efficient programmable computation. In *VLSI Design Conference*, pages 261–264, 1994.

[5] 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.

[6] A. Chatterjee and R. Roy. Synthesis of low power DSP circuits using activity metrics. In *VLSI Design Conference*, pages 265–270, 1994.

[7] 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.

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

[9] M. J. Flynn. *Computer Architecture: Pipelined and Parallel Processor Design*. Jones and Bartlett, 1996.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, 1979.

[11] L. Goodby, A. Orailoglu, and P. Chau. Microarchitectural synthesis of performance-constrained low power VLSI designs. In *International Conference on Computer Design*, pages 323–326, 1994.

[12] C. Hansen. MicroUnity's MediaProcessor architecture. *IEEE Micro*, 17:34–41, 1997.

[13] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufman, San Francisco, CA, 1993.

[14] I. Hong and M. M. Potkonjak. Power optimization using divide-and-conquer techniques for minimization of the number of operations. In *ICCAD-97 IEEE/ACM International Conference on Computer-Aided Design*, 1997.

[15] P. Y. Hsu. Highly concurrent scalar processing. Technical Report CSG-49, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1986.

[16] R. Jain. *The Art of Computer Systems Performance Analysis*. Wiley, 1991.

[17] P. Kalapathy. Hardware-software interactions on MPACT. *IEEE Micro*, 17:20–26, 1997.

[18] A. Kalavade and E.A. Lee. Complexity management in system-level design. *Journal of VLSI Signal Processing*, 14(2):157–169, 1996.

[19] 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.

[20] 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.

[21] 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.

[22] R.B. Lee and M.D. Smith. Media processing: A new design target. *IEEE Micro*, 17:6–9, 1997.

[23] 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.

[24] S. A. Mahlke, D. C. Lin, W. Y. Chen, R. E. Hank, and R. A. Bringmann. Effective compiler support for predicated execution using the Hyperblock. In *International Symposium on Microarchitecture*, 1992.

[25] J. Montanaro et al. A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor. *IEEE Journal of Solid-State Circuits*, 31(11):1703–1714, November 1996.

[26] A. Peleg and U. Weiser. MMX technology extension to the Intel architecture. *IEEE Micro*, 16(4):42–50, August 1996.

[27] A. Raghunathan and N. Jha. Behavioral synthesis for low power. In *International Conference on Computer Design*, pages 318–322, 1994.

[28] D. Singh, J. Rabaey, M.Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, and T. Mozdzen. Power conscious CAD tools and methodologies: A perspective. *Proceedings of IEEE*, 83(4):570–594, 1995.

[29] M. Srivastava, A. P. Chadrakasan, and R. Broderson. Predictive system shutdown and other architectural techniques for energy efficient programmable computation. *IEEE Transactions on VLSI Systems*, 4(1):42–55, 1996.

[30] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: A first step towards software power minimization. *IEEE Transactions on VLSI Systems*, 2(4):437–445, 1994.

[31] J. Turley and H. Hakkarainen. TI's new 'C6x DSP screams at 1,600 MIPS. *The Microprocessor Report*, 11:14–17, 1997.