# Media Architecture: General Purpose vs. Multiple Application-Specific Programmable Processor

Chunho Lee†, Johnson Kin‡, Miodrag Potkonjak† and William H. Mangione-Smith‡
**†Computer Science Department, UCLA, ‡Electrical Engineering Department, UCLA**

## Abstract

In this paper we report a framework that makes it possible for a designer to rapidly explore the application-specific programmable processor design space under area constraints. The framework uses a production-quality compiler and simulation tools to synthesize a high performance machine for an application. Using the framework we evaluate the validity of the fundamental assumption behind the development of application-specific programmable processors. Application-specific processors are based on the idea that applications differ from each other in key architectural parameters, such as the available instruction-level parallelism, demand on various hardware components (e.g. cache memory units, register files) and the need for different number of functional units. We found that the framework introduced in this paper can be valuable in making early design decisions such as area and architectural trade-off, cache and instruction issue width trade-off under area constraint, and the number of branch units and issue width.

## 1 Introduction

Since 1992, microprocessors account for $23\%$ of total semiconductor sales. By 1998, these chips will account for $27\%$ of total sales, estimated at $\$242$ billion by an industry expert [14]. The increasing share of microprocessors in semiconductor market is due to a new phase of silicon integration, thanks mainly to deep submicron fabrication technology. For example, SA-1100 from Digital Semiconductor [31] incorporates many system functions such as a memory controller, color LCD driver, PCMCIA interface, IrDA and USB communication channels, and extensive power management into a single chip. One major implication of this advanced technology is that almost all semiconductor manufacturers are moving into the microprocessor business. This shift will force manufacturers to focus on microprocessors, specifically media processors, that are cheaper and more aggressively optimized for specific applications. A challenge to microprocessor designers will be to design a microprocessor that executes a targeted application very well yet can achieve large economies of scale.

Recent advances in compiler technology and microprocessor architecture for instruction-level parallelism (ILP) have significantly increased the ability of microprocessors to exploit the opportunities for parallel execution that exist in many programs. Key ILP compiler technologies, such as trace scheduling [8], superblock scheduling [24], tree-scheduling [3], hyperblock scheduling [25] and software pipelining [15] are in the process of migrating from research labs to product groups. Commercial microprocessor manufacturers have introduced a number of products based on new architectures that present hardware structures that are well matched to ILP compilers. Architectural enhancements found in commercial products include predicated instruction execution, VLIW, split register files and multi-gauge arithmetic (or variable-width SIMD) [5, 12, 17, 32]. Most of the multimedia extensions of general-purpose processors also adopt similar architectural enhancements [20, 28].

The arrival of production quality ILP compilers and commercial DSPs with VLIW architecture stimulated the idea of so-called custom-fit processors [9]. Researchers have argued that applications differ from each other in, for example, the available ILP, demand on various hardware components such as cache memory units, register files, and the number of functional units. The presumption is that a microprocessor can be designed by adding hardware components tailored to a specific application so that it can execute the single application well.

Previous discussions and research efforts on multiple application mediaprocessors and single application processors have lacked a well-defined and agreed-upon set of benchmarks. No systematic study on programmable processor design space exploration using a comprehensive benchmark suite has been reported. We have developed a framework to rapidly explore the design space by focusing on the trade-off between cost and generality of possible designs under area constraints. We note that the advances in compiler and architecture technologies should be incorporated in a programmable processor design space exploration framework.

This paper is organized as follows. The next section briefly surveys related works and summarizes the contributions of this work. Machine model, benchmarks, tools and example set of results obtained using the tools are described in Section 3. Our approach in this work is explained in Section 4 in detail. Section 5 formulates the search problem defined in the previous section in precise terms. The search strategy and algorithm is described in Section 6. Experimental results of the tools and algorithms for the system-level synthesis are reported in Section 7. Finally, Section 8 draws conclusions.

## 2 Related Works and Our Contributions

The works on synthesis and evaluation of application-specific programmable processors has been conducted independently in two research communities, computer-aided design and architecture. There is, however, a strong converging trend of the two areas due to recent technological advances and application trends. In this section we survey the related works in these two fields.

Since the early 90's, there have been a number of efforts related to the design of application-specific programmable processors and application-specific instruction sets. Comprehensive surveys of the works on computer-aided design of application-specific programmable processors can be found in [11, 26, 27]. In particular, a great deal of effort has been made in combining retargetable compilation technologies and design of instruction sets [1, 21, 22, 23, 30]. Several research groups have published results on the topic of selecting and designing instruction set and processor architecture for

**35th Design Automation Conference ®**
Copyright ©1998 ACM
1-58113-049-x-98/0006/$3.50 DAC98 - 06/98 San Francisco, CA USA

particular application domains [18, 33].

Early works in the area of processor architecture synthesis tended to employ *ad hoc* methods on small code kernels, in large part due to the lack of good retargetable compiler technology. Conte and Mangione-Smith [6] presented one of the first efforts that focused on large application codes written in high-level languages (SPEC). While they had a similar goal to ours, i.e. evaluating performance efficiency by including hardware cost, their evaluation approach was substantially different. Fisher, Faraboschi and Desoli [9] studied the variability of applications-specific VLIW processors using a highly advanced and retargetable compiler. However, their study considered small program kernels rather than complete applications. They also focused on finding the best possible architecture for a specific application or workload, rather than understanding the difference between best architectures across a set of applications.

Unlike most previous works, we use a set of complete applications written in a high-level language as benchmarks. We incorporate the role of cache memory units in machine performance into the machine model which, is essential for realistic design.

The main contribution of the research presented in this paper is the framework that makes it possible for designers to rapidly explore programmable processor design space for a given set of applications under area constraints. It enables a designer to evaluate the trade-off between area and generality of possible designs.

| Configuration | Issue | IALU | Branch | Mem | Cache | Total |
|---|---|---|---|---|---|---|
| (1, 1, 1, 1, .5, .5) | 0.74 | 3.67 | 6.13 | 5.52 | 1.53 | 31.56 |
| (2, 2, 1, 2, 1, 1) | 2.94 | 7.33 | 6.13 | 11.03 | 2.54 | 43.95 |
| (4, 4, 1, 4, 2, 2) | 11.76 | 14.66 | 6.13 | 22.07 | 4.55 | 73.15 |
| (8, 8, 1, 8, 4, 4) | 47.04 | 29.32 | 6.13 | 44.14 | 8.56 | 149.17 |
| (4, 4, 2, 4, 8, 8) | 11.76 | 14.66 | 12.26 | 22.07 | 16.55 | 91.28 |
| (8, 8, 2, 8, 4, 4) | 47.04 | 29.32 | 12.26 | 44.14 | 8.56 | 155.30 |
| (8, 8, 4, 8, 8, 8) | 47.04 | 29.32 | 24.52 | 44.14 | 16.55 | 175.55 |

Table 1: 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)). The total area includes the overhead area of 13.98 $mm^2$.

| Benchmark | Instr. | Source | Description |
|---|---|---|---|
| JPEG encoder | 13.9 | Independent | JPEG image |
| JPEG decoder | 3.8 | JPEG Group | encoding/decoding |
| MPEG encoder | 1,121.3 | MPEG Simulation | MPEG-2 movie |
| MPEG decoder | 175.5 | Group | encoding/decoding |
| GSM encoder | 184.2 | Technische | European wireless |
| GSM decoder | 73.0 | Universität, Berlin | voice coding standard |
| G.721 encoder | 274.1 | Sun Microsystems, | CCITT voice |
| G.721 decoder | 511.7 | Inc. | coding standard |
| PGP encryption | 169.9 | Massachusetts Institute | encryption/ |
| PGP decryption | 155.3 | of Technology | decryption |
| Pegwit encryption | 34.0 | George Barwood | encryption/ |
| Pegwit decryption | 18.5 | | decryption |
| Mipmap | 47.6 | University of | 3-D rendering |
| OS-demo | 9.0 | Wisconsin | examples using Mesa |
| Texgen | 83.9 | | graphics library |
| Rasta | 24.4 | ICSI at UC Berkeley | Voice recognition |
| EPIC encoder | 50.3 | University of | Wavelet image |
| EPIC decoder | 7.2 | Pennsylvania | encoding/decoding |
| ADPCM encoder | 6.8 | Jack Jansen | speech compression |
| ADPCM decoder | 5.9 | | and decompression |

Table 2: Applications used in the experiment. Dynamic instruction counts were measured on a SPARC-5, and are reported in terms of millions of instructions

## 3 Experiment Platform

To estimate the cost of a machine configuration, we adopt a simple model described in [2]. Given the area of the issue unit, the cost of any machine configuration is a linear function of the number of function units for branch, memory, and ALU operations. For a superscalar machine, the issue unit area cannot be estimated using a simple linear model since it requires more complex logic for the on-the-fly code scheduling. We assume that the issue unit area will take $O(n^2)$ space since the complexity of dependency checking algorithm is $O(n^2)$. When a VLIW machine is considered, the issue unit area is of linear or sub-linear complexity.

The architecture chosen for the analysis, the PowerPC 604 [29], is a four-issue processor. The 604 has two simple integer ALUs and one complex integer ALU, one floating-point unit, one branch unit, and one memory unit. We assume that machine configurations with an issue width smaller than the baseline machine have at least one complex integer ALU. The area of the complex integer unit is assumed to be half of the baseline integer unit (two simple integer units and one complex integer unit). The area of issue unit is scaled down based on the area complexity ($O(n^2)$). We did not include floating-point units in any machine configurations as the benchmarks have mostly integer operations. Finally, we scaled the area to 0.35$\mu$ technology rather than the original 0.5$\mu$ technology. A set of example machine configurations and their respective area estimates are shown in Table 1.

The set of benchmarks used in this experimentation is composed of complete applications which are publically available and coded in a high-level language. The collection, known as MediaBench is composed of 21 applications culled from available image processing, communications, cryptography and DSP applications. A brief summary of benchmarks used is shown in Table 2. More detailed descriptions of benchmarks can be found in [19].

We use the IMPACT tool suit [4] to collect performance measurements 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 HMDES file 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. Figure 1 shows the flow of simulation using IMPACT tools.

## 4 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

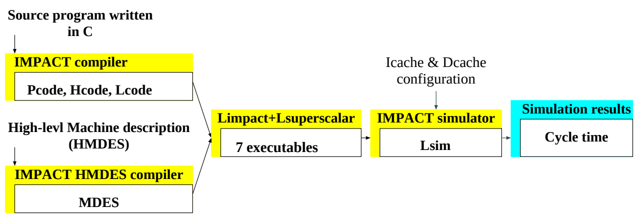


Figure 1: Performance measurement flow using IMPACT tools

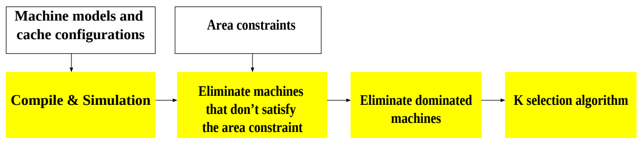

Machine models and cache configurations
Area constraints
Compile & Simulation
Eliminate machines that don't satisfy the area constraint
Eliminate dominated machines
K selection algorithm

Figure 2: Global design flow

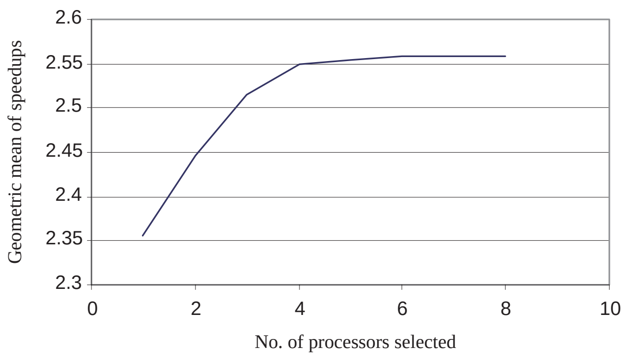


Figure 3: Performance vs. number of selected processors: Area constraint $84mm^2$

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 normalized with respect to a baseline machine with one branch unit, one-issue unit, 512 bytes of instruction cache, and 512 bytes of data cache.

Simulation results feed the search engine 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, i.e. a machine that runs slower than or equal to the speed of another machine for *all* benchmarks. Finally, we perform a $K$-selection algorithm (see Section 5) to select a set of machine configurations that run the benchmark set best. Figure 2 shows the global flow of design process.

The search space is relatively small due to the area constraints and the number of dominated implementations. Nevertheless, there is a possibility that the search space explodes for alternative hardware models. We use an efficient branch-and-bound based algorithm to avoid this problem.

Figure 3 shows an example measurement on performance vs. number of selected processors. Empirical results suggests that there are break points of diminishing returns for adding architectures. This will be elaborated in more detail in the next section.

## 5 Selection Problem Formulation

Given an area constraint and the performance 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 speed-ups across all the benchmark is maximized and the subset size is kept small.

We normalize the run time with respect to a baseline since we are not interested in the sum of run-times [16]. We use geometric mean to summarize 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 speed-up 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 $\sqrt[n]{\prod_{i=1}^{n} \max_{j \in M} E_{ij}} \geq C$?

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 $\rho$. Formally the subproblems 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 speed-up 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 $w$,

**Maximize** :

$$D_w = \sqrt[n]{\prod_{i=1}^{n} \max_{j \in P} E_{ij}}, \qquad (1)$$

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

$K$-selection problem:

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

where $D_w$ is given by Equation 1 and $\rho$ is a cut-off ratio.

## 6 Solution Space Exploration: Strategy and Algorithms

Due to space limitation we present only the top level algorithm for system-level synthesis in Figure 4. Considering that the run-time of simulations for 20 benchmarks on 175 machine configurations is about a week, we can tolerate a longer search time to find an optimal selection result. Generally, the size of the search problem can be 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.

The search for optimum solution is organized using an implicit enumeration method. In particular, we adopt a branch-and-bound algorithm shown in Figure 4 to speed up the selection.

The algorithm consists of branching and evaluation. The branching step takes the current state of search and generates a number of new nodes by adding an available machine (i.e., one that has not been considered in particular search path of the search tree). It determines if adding a machine to the current state of selection can

```
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 ( ($R \geq \rho$) && ($i \leq k$) ) {
  $D_i$ = branch-and-bound( $D_{i-1}$, i );
  $R = \frac{D_i - D_{i-1}}{D_i}$;
  $D_{i-1} = D_i$;
  $i++$;
}
```

Figure 4: System-level synthesis of application-specific programmable machines

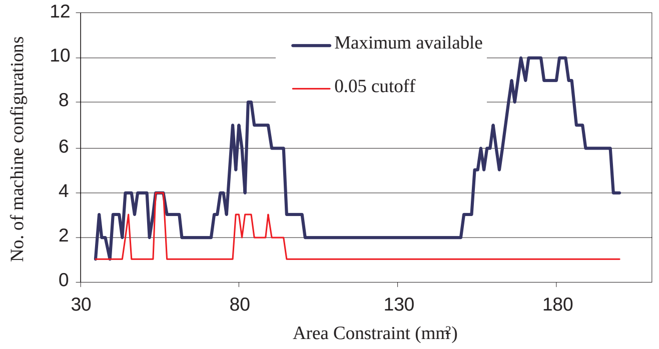
result in a better solution than the current best solution found. Initially, the current best solution is set to the previous best solution. The previous best solution is the best solution found for the machine set size less than the current search size by one. The bounding function compares the current 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 bound the 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 as to increase the bounding rate.

## 7 Experimental Results

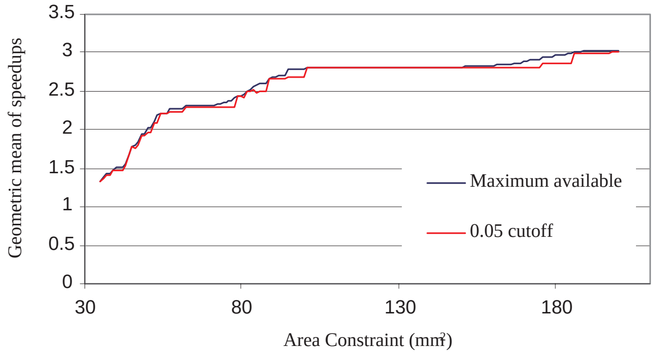
We evaluate the tools and algorithms by running experiments from 30$mm^2$ to 200$mm^2$. The implementation technology is assumed to be 0.35$\mu$. For each area constraint, we obtain the optimum set of machine configurations for cut-off values ($\rho$'s as defined in Equation 2) 0.1, 0.05, 0.01, and 0.005.

Figure 5(a) shows experimental results using the cut-off value of 0.05. The thicker line shows the number of machines that are left after eliminating dominated machines. The thinner line indicates the number of machines required to cover all the benchmarks under area constraints. We see that more machine configurations are needed when less area is given. In other words, with less area, each processor configuration is more specialized. On the other hand, the more area we have, the more general the processor we can design. The results suggest that when more than 100$mm^2$ of area is available, there is little advantage in having more than one architecture. Moreover, for the given compiler technology and benchmarks, there is no need to have more than 100$mm^2$ of area since the speed-up increase achieved by machines greater than 100$mm^2$ are minimal.

The overall performance comparison between all configurations and selected configurations are shown in 5(b). There are three distinctive points where the speed-up increase rate changes. Up to the area 57$mm^2$, we see rapid performance increase, which is mainly due to increased amount of cache memories. From 57$mm^2$ to 101$mm^2$, the measurement shows modest increase of performance due to increased issue width. For the processors larger than



(a) Available machines under area constraints and selected machines



(b) Performance of *all* available machines and selected machines

Figure 5: Quadratic complexity issue unit area model: Selected processor configurations and their performance (cut-off value: 0.05)

101$mm^2$, the performance increase is minimal. One of the underlying reasons that causes the phenomenon is that the ILP found by the compiler and hardware scheduler is fully exploited by having a certain amount of hardware, thereby performance increase possibility is exhausted. The limitation of performance increase in the face of increased area illustrates either the limitation of the current compiler technology or the inherent lack of ILP in the benchmarks, which we are not able to answer at this point. Note, however, that the measurement is not for a single processor. Smaller area cases tend to have more than one architectures which are more application specific.

Experimental results for the cut-off values 0.1, 0.05, 0.01, and 0.005 are given in Figure 6. Smaller cut-off values result in machine configuration sets that are more tuned to each application. In general, however, smaller cut-off values do not result in significant performance increase. This conclusion is valid when the available area is bigger than 100$mm^2$. In most cases, the cut-off value of 0.05 gives good trade-off between the number of machine configurations and performance.

A sample set of speed-up factors of each benchmark are shown in Table 3. They are snapshots of experimental results summarized by the line graphs in Figure 6. The table contains maximum speed-up factors for three cut-off values (0.05, 0.01, and 0.005) and three area constraints (85$mm^2$, 100$mm^2$, and 169$mm^2$). Note that the area constraints are not actual areas. We find machines under the given area constraints. Table 4 gives the number of machine configurations selected for the snapshot cases shown in Table 3. The actual areas of the selected machines are given in column 4 of the table. The combination of components for the selected machines are shown in column 3.

Figure 7 shows the results when the liner complexity issue unit

| Benchmark | cut-off:0.05 | | | cut-off:0.01 | | | cut-off:0.005 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 84 | 100 | 169 | 84 | 100 | 169 | 84 | 100 | 169 |
| JPEG enc | 3.02 | 3.51 | 3.87 | 3.02 | 3.65 | 3.87 | 3.02 | 3.65 | 3.87 |
| JPEG dec | 3.56 | 3.91 | 4.27 | 3.56 | 4.23 | 4.27 | 3.56 | 4.23 | 4.27 |
| MPEG enc | 2.45 | 2.56 | 2.57 | 2.45 | 2.56 | 2.57 | 2.45 | 2.42 | 2.57 |
| MPEG dec | 3.05 | 3.55 | 3.68 | 3.05 | 3.55 | 3.95 | 3.05 | 2.56 | 3.95 |
| GSM enc | 2.22 | 2.30 | 2.31 | 2.22 | 2.30 | 2.32 | 2.24 | 2.94 | 2.32 |
| GSM dec | 2.89 | 2.99 | 3.00 | 2.89 | 2.99 | 3.19 | 2.91 | 2.99 | 3.19 |
| G.721 enc | 2.68 | 3.38 | 3.38 | 2.88 | 3.38 | 3.38 | 2.88 | 3.38 | 3.38 |
| G.721 dec | 2.71 | 3.44 | 3.44 | 3.04 | 3.44 | 3.44 | 3.04 | 3.44 | 3.44 |
| PGP enc | 2.78 | 3.08 | 3.21 | 2.83 | 3.15 | 3.21 | 2.83 | 2.26 | 3.21 |
| PGP dec | 2.88 | 3.22 | 3.38 | 3.02 | 3.28 | 3.38 | 3.02 | 2.15 | 3.46 |
| Pegwit enc | 2.64 | 2.66 | 2.82 | 2.64 | 2.79 | 2.82 | 2.64 | 3.28 | 2.82 |
| Pegwit dec | 2.63 | 2.68 | 2.84 | 2.63 | 2.82 | 2.84 | 2.62 | 2.79 | 2.84 |
| Mipmap | 2.81 | 2.69 | 2.79 | 2.81 | 2.94 | 3.10 | 2.81 | 3.56 | 3.10 |
| Osdemo | 2.22 | 2.40 | 2.47 | 2.25 | 2.42 | 2.47 | 2.25 | 2.82 | 2.47 |
| Texgen | 2.11 | 2.31 | 2.40 | 2.11 | 2.40 | 2.40 | 2.11 | 2.42 | 2.40 |
| Rasta | 2.13 | 2.22 | 2.27 | 2.13 | 2.26 | 2.31 | 2.13 | 2.31 | 2.31 |
| EPIC enc | 1.93 | 1.78 | 2.17 | 1.93 | 2.06 | 2.17 | 2.04 | 2.17 | 2.30 |
| EPIC dec | 1.70 | 1.76 | 1.77 | 1.70 | 1.76 | 1.83 | 1.70 | 1.76 | 1.83 |
| ADPCM enc | 2.30 | 2.31 | 2.31 | 2.30 | 2.31 | 2.43 | 2.30 | 2.42 | 2.44 |
| ADPCM dec | 2.33 | 2.34 | 2.42 | 2.33 | 2.34 | 2.55 | 2.34 | 2.40 | 2.65 |

Table 3: Snapshots of speed-ups of benchmarks at various cut-off values for the given area constraint (84$mm^2$, 100$mm^2$, 169$mm^2$) (NOTE: Actual area of designs are given in Table 4)

area model is assumed. The results suggest that the machine configuration selection problem has no strong dependence to an issue area model used. Although we observe that there is shift to smaller areas, the results are essentially equivalent to the results based on the quadratic complexity issue unit area model.

In summary, we found that under the machine models and machine configuration choices described in this paper, when more than 100$mm^2$ of area is available, there is no advantage in having more than one architecture to be able to build application-specific systems for all the benchmarks. Moreover, for the given compiler technology and benchmarks, there is very little need to have more than 100$mm^2$ of area since the speed-up increase achieved by machines greater than 100$mm^2$ are minimal one. Notable exception is that for highly cost sensitive designs we observe a need for a small number of specialized architectures which occupy smaller areas.

## 8 Conclusion

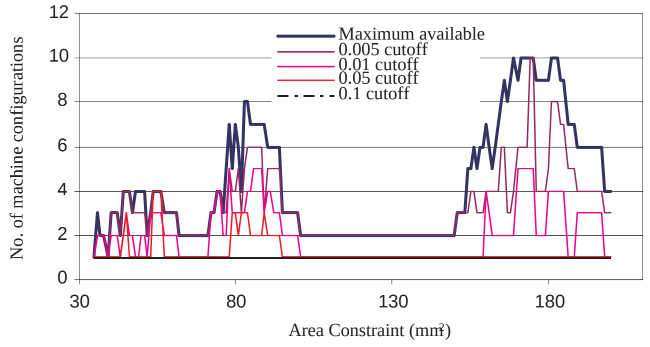
The arrival of production quality ILP compilers and commercial DSPs with VLIW architecture stimulated the idea of programmable processors that are aggressively tuned to specific applications. The assumption behind the idea is that there are ways of designing programmable processors that can exploit the run-time characteristics of specific applications. The run-time characteristics include the available ILP, demand on various hardware components such as cache memory units, register files, and the number of functional units. Under the assumption, a microprocessor can be designed by adding hardware components tailored to a specific application so that it can execute the single application extremely well.

We ran extensive experiments on a framework based on the key paradigms of CAD and architecture communities. This combination enabled us to gain valuable insights about design and use of application specific programmable processors for modern applications. We evaluated 175 machine configurations on 20 benchmarks under the area constraint ranging from 30$mm^2$ to 200$mm^2$. For each area constraint, we obtain the optimum set of machine configurations for a number of cut-off values. The run time of the entire synthesis process was about a week on a single HP workstation.

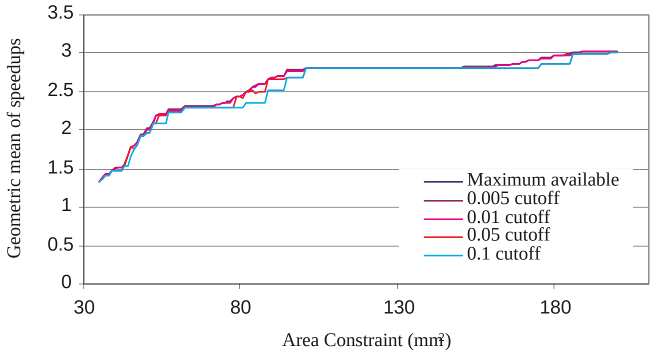
We found that the framework introduced in this paper can be

| Cut-off Value | Max area | Configuration | Actual area |
|---|---|---|---|
| 0.05 | 84 | (2, 2, 1, 2, 8, 8) | 61.11 |
| | | (4, 4, 1, 4, 4, 4) | 81.89 |
| | | (4, 4, 2, 4, 1, 4) | 83.24 |
| | 100 | (4, 4, 2, 4, 8, 4) | 94.36 |
| | 169 | (4, 4, 2, 4, 8, 8) | 100.71 |
| 0.01 | 84 | (2, 2, 1, 2, 8, 8) | 61.11 |
| | | (4, 4, 1, 4, 4, 4) | 81.89 |
| | | (4, 4, 1, 4, 8, 1) | 83.46 |
| | | (4, 4, 2, 4, 1, 4) | 83.24 |
| | 100 | (4, 4, 1, 4, 8, 8) | 94.58 |
| | | (4, 4, 2, 4, 8, 4) | 94.36 |
| | 169 | (4, 4, 2, 4, 8, 8) | 100.71 |
| | | (8, 8, 2, 8, 4, 4) | 166.33 |
| 0.005 | 84 | (2, 2, 1, 2, 8, 8) | 61.11 |
| | | (4, 4, 1, 4, 1, 8) | 83.46 |
| | | (4, 4, 1, 4, 4, 4) | 81.89 |
| | | (4, 4, 1, 4, 8, 1) | 83.46 |
| | | (4, 4, 2, 4, 1, 4) | 83.24 |
| | | (4, 4, 2, 4, 4, 1) | 83.24 |
| | 100 | (4, 4, 1, 4, 8, 8) | 94.58 |
| | | (4, 4, 2, 4, 4, 8) | 94.36 |
| | | (4, 4, 2, 4, 8, 4) | 94.36 |
| | 169 | (4, 4, 2, 4, 8, 8) | 100.71 |
| | | (8, 8, 2, 8, 1, 8) | 168.68 |
| | | (8, 8, 2, 8, 4, 4) | 166.33 |
| | | (8, 8, 4, 8, 1, 2) | 166.70 |

Table 4: Machine configuration and actual area of designs for the snapshot cases in Table 3: Column 3 shows machine configurations in the form of (issue width, number of ALUs, number of branch units, number of memory units, size of instruction cache(KB), size of data cache(KB)). Actual areas are given in the 4th column.

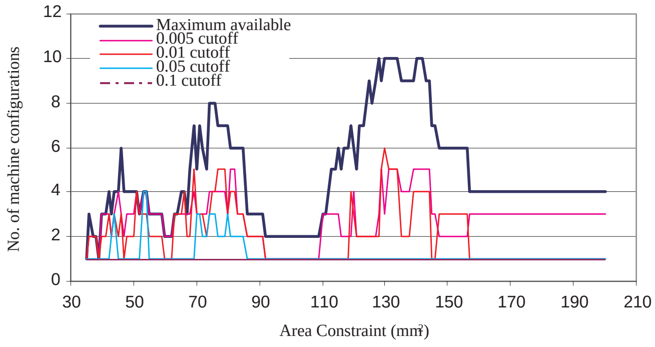


(a) Available machines under area constraints and selected machines

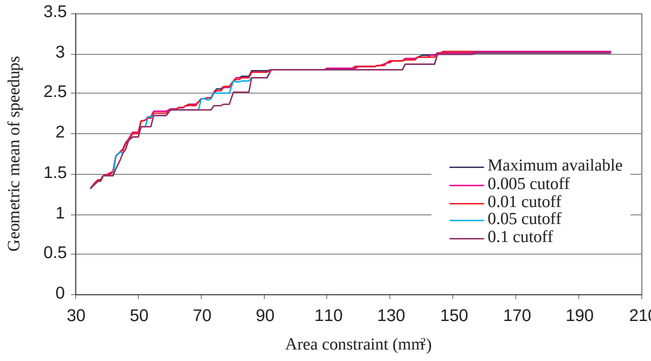


(b) Performance of *all* available machines and selected machines

Figure 6: Quadratic complexity issue unit area model: selected processor configurations and their performance for 4 cut-off values



(a) Available machines under area constraints and selected machines



(b) Performance of *all* available machines and selected machines

Figure 7: Linear complexity issue unit area model: selected processor configurations and their performance for 4 cut-off values

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.

## Acknowledgments

This work was supported by DARPA under contract F04701-97-C-0010. The authors also greatly appreciate the assistance of Professor W. M. Hwu's IMPACT team at UIUC.

## REFERENCES

[1] G. Araujo, A. Sudarsanam, and S. Malik. Instruction set design and optimizations for address computation in DSP architectures. In *Proceedings of International Symposium on System Synthesis*, pages 102–107, 1996.

[2] D. C. Argyres. Performance and cost analysis of the execution stage of superscalar microprocessors. Master's thesis, Department of Computer Science, University of Illinois, Urbana IL, May 1995.

[3] S. Banerjia, W. A. Havanki, and T. M. Conte. Treegion scheduling for highly parallel processors. In *Euro-Par*, Passau, Germany, 1997.

[4] 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.

[5] 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.

[6] T. Conte and W. Mangione-Smith. Determining cost-effective multiple issue processor designs. In *International Conference on Computer Design*, 1993.

[7] T. M. Conte, K. N. P. Menezes, and S. W. Sathaye. A technique to determine power-efficient, high-performance superscalar processors. In *Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences*, volume 1, pages 324–333, 1995.

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

[9] J. A. Fisher, P. Faraboschi, and G. Desoli. Custom-fit processors: Letting applications define architectures. In *International Symposium on Microarchitectures*, Paris, France, 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] G. Goossens, J. Van Praet, D. Lanneer, W. Geurts, , et al. Embedded software in real-time signal processing systems: design technologies. *Proceedings of the IEEE*, 85(3):436–454, March 1997.

[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] R. D. Hof and O. Port. Silicon dreams. *Business Week International Edition*, May 13 1996.

[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] K. Kim, R. Karri, and M. Potkonjak. Heterogeneous built-in resiliency of application specific programmable processors. In *International Conference on Computer-Aided Design*, pages 406–411, 1996.

[19] 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.

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

[21] R. Leupers and P. Marwedel. Retargetable generation of code selectors from HDL processor models. In *European Design and Test Conference*, pages 140–144, 1997.

[22] S. Liao, S. Devadas, K. Keutzer, and S. Tjiang. Instruction selection using binate covering for code size optimization. In *International Conference on Computer-Aided Design*, pages 393–399, 1995.

[23] C. Liem, T. May, and P. Paulin. Instruction-set matching and selection for dsp and asip code generation. In *The European Design and Test Conference*, pages 31–37, 1994.

[24] 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.

[25] 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.

[26] P. Marwedel. Processor-core based design and test. In *Asia and South Pacific Design Automation Conference*, pages 499–502, 1997.

[27] P.G. Paulin, C. Liem, M. Cornero, F. Nacabal, , et al. Embedded software in real-time signal processing systems: application and architecture trends. *Proceedings of the IEEE*, 85(3):419–435, March 1997.

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

[29] S. P. Song and M. Denman. The PowerPC 604 RISC microprocessor. Technical report, Motorola Inc. and International Business Machines, 1994.

[30] A. Sudarsanam and S. Malik. Memory bank and register allocation in software synthesis for ASIPs. In *International Conference on Computer-Aided Design*, pages 388–392, 1995.

[31] J. Turley. SA-1100 puts PDA on a chip. *Microprocessor Report*, 11(12), September 1997.

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

[33] W. Zhao and C.A. Papachristou. An evolution programming approach on multiple behaviors for the design of application specific programmable processors. In *European Design and Test Conference ED&TC 96*, pages 144–150, 1996.