Low-Power Behavioral Synthesis Optimization Using Multiple Precision Arithmetic

Milos Ercegovac, Darko Kirovski, and Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles

Abstract

Many modern multimedia applications such as image and video processing are characterized by a unique combination of arithmetic and computational features: fixed-point arithmetic, a variety of short data types, high degree of instruction-level parallelism, strict timing constraints, and high computational requirements. Computationally intensive algorithms usually boost device's power dissipation which is often key to the efficiency of many communications and multimedia applications. Although recently virtually all general-purpose processors have been equipped with multiprecision operations, the current generation of behavioral synthesis tools for application-specific systems does not utilize this power/performance optimization paradigm.

In this paper, we explore the potential of using multiple precision arithmetic units to effectively support synthesis of low-power application-specific integrated circuits. We propose a new architectural scheme for collaborative addition of sets of variable precision data. We have developed a novel resource allocation and computation assignment methodology for a set of multiple precision arithmetic units. The optimization algorithms explore the trade-off of allocating low-width bus structures and executing multiple-cycle operations. Experimental results indicate strong advantages of the proposed approach.

1. Introduction

Recently, the fast growing communications and multimedia consumer market has redefined the relative importance of design metrics for modern signal processing application-specific integrated circuits (ASICs). The next generation of ASICs encapsulates a variety of arithmetic and computational features: fixed-point arithmetic, a variety of short data types, high degree of instruction-level parallelism, strict timing constraints, and computationally intensive algorithms. Since the efficacy of many applications such as communications, portable computing, and signal processing, is highly dependent upon their power dissipation characteristics, low-power silicon implementation of modern computationally intensive data processing algorithms becomes one of the most important design desiderata.

Another important issue in the design of an ASIC and currently ignored optimization potential, is synthesis support for variable-length data types. The importance of employing multiple precision arithmetic units in ASICs is well illustrated by the recent arithmetic and architectural trends in programmable platforms. The majority of the latest general purpose architectures provides support for multiple precision execution units. For example, both the Intel MMX multimedia extension to Pentium Pro [Pel96] and the SUN UltraSparc II architecture [Go96] provide instruction sets and adequate architectural support for execution of low-precision instructions in parallel on partitioned arithmetic logic units. Even further, a number of compilation techniques have been proposed to utilize these architectural features and improve performance of popular signal processing functions [Che97], [Jah97], [Bli97] and in particular, MPEG decoding [Zho96].

In this paper, for the first time, we combine arithmetic and behavioral synthesis techniques to explore the potential of multiple precision arithmetic units for power optimization of application-specific systems on silicon. A new simple, yet powerful, hardware scheme for multicycle addition of variable precision data is proposed. The scheme is supported with synthesis automation tools for multiple precision resource allocation and task assignment. For fixed circuit area constraint, the optimization algorithm aims at reducing the capacitance and/or switching activity on the datapath buses and arithmetic logic units at the cost of having as few as possible multicycle operations. The experimental results indicate high potential of the multiple precision paradigm as a viable low-power design methodology in the synthesis of application-specific systems.
1.1. Motivational Example

A flavor of advantages of multiple precision arithmetic is illustrated using the following motivational example. A set of 12 mutually independent additions is executed on two architectures, traditional (fixed) and new (multiprecision). The goal is to have a limited circuit area and computation deadline, allocate adders, assign computation, and find minimal operating voltage such that minimal energy is consumed. The first architecture is exclusively built out of monolithic arithmetic units (adders), while the second architecture is based on units which can be adjusted to the computation precision requirements. Relevant timing, area, and power dissipation data are given in Table 1.

<table>
<thead>
<tr>
<th>Prec.</th>
<th>Inst.</th>
<th>Area</th>
<th>Delay</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>bits</td>
<td></td>
<td>10^6 \cdot \mu m^2</td>
<td>ns @3.3V</td>
<td>nW</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>1.7</td>
<td>54</td>
<td>9</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>4.0</td>
<td>72</td>
<td>15</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>6.6</td>
<td>90</td>
<td>26</td>
</tr>
</tbody>
</table>

Table 1: Computation and arithmetic units (CLA adders) [Deadline=360ns, Area=13.5 \cdot 10^6 \mu m^2]

The first architecture, in order to satisfy the computation deadline, requires one 32-bit, one 16-bit and one 8-bit adder resulting in chip area of 12.3 \cdot 10^6 \mu m^2. The delay of the system clock is set to 90ns and the computation requires 4 clock cycles. The overall power dissipation per computation cycle (360ns) totals

$$Power = 4 \cdot 26nW + 4 \cdot 15nW + 4 \cdot 9nW = 200nW.$$  

If multiple precision adders are used, the hardware allocated and computation assignment for minimal power dissipation result in a configuration and schedule as shown in Figure 1. The required area is equal to:

$$Area = 2 \cdot 4.0 \cdot 10^6 + 2 \cdot 1.7 \cdot 10^6 = 13.4 \cdot 10^6 \mu m^2 < 13.5 \cdot 10^6 \mu m^2$$

Since there are no 32-bit units and the computation on the allocated hardware still takes 4 cycles, the voltage supply can be scaled down to 2.8V from 3.3V according to the \( \frac{V_{dd}}{V_{il}} \) delay increase and with respect to the delay-voltage dependency formula \( Delay = K \cdot \frac{V_{dd}}{V_{il}} \) [Cha92]. This configuration and assignment result in power dissipation reduction per computation cycle (360ns) equal to 37%.

$$Power = (8 \cdot 15nW + 6 \cdot 9nW) \cdot \frac{2.8^2}{3.3^2} = 125nW.$$  

For an efficient power optimization methodology that uses multiple precision arithmetic units, the following two issues are crucial. First, operations of higher precision demand execution on as high as possible precision arithmetic units for minimum delay and energy overhead. Second, once multiprecision arithmetic support is provided, in order to minimize the system cost, efficient algorithms for resource allocation and operation assignment are required to create and utilize the computation system.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control step</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>32-bit adder</td>
<td>32 32 16 16</td>
</tr>
<tr>
<td>16-bit adder</td>
<td>16 16 8 8</td>
</tr>
<tr>
<td>8-bit adder</td>
<td>8 8 8 8</td>
</tr>
</tbody>
</table>

Architecture/Schedule ONE.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control step</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>16-bit adder</td>
<td>32 32</td>
</tr>
<tr>
<td>16-bit adder</td>
<td>16 16 16 16</td>
</tr>
<tr>
<td>8-bit adder</td>
<td>8 8 8</td>
</tr>
<tr>
<td>8-bit adder</td>
<td>8 8 8</td>
</tr>
</tbody>
</table>

Architecture/Schedule TWO.

Figure 1: Non-optimized and optimized allocation and scheduling

2. Related Work

The previous work can be traced along the following two research areas: low-power behavioral synthesis optimization techniques and architecture extensions for multiprecision arithmetic.

Although power optimization is often most effective when applied early in the design process at the algorithmic and architectural level, majority of power minimization techniques have been proposed for logic synthesis and physical design [Sin95]. Potkonjak and Rabaey presented a CAD system-level synthesis approach which targets low-power [Pot92]. However, the scope of this effort is restricted to a single task application specific system. Several system/behavioral level optimizations have been proposed in [Rag94], [Pot94], [Lak98].
Recently, multiple precision execution units have been augmented in new general purpose architectures (for example, the Intel MMX multimedia extension to Pentium Pro [Pel96] and the UltraSparc II Visual Instruction Set (VIS) architecture from SUN [Gol96]). Compiler support for these architectures has concentrated on various signal processing applications such as MPEG decoding [Zho95]. Efficient software utilization of the Pentium MMX and UltraSparc VIS architectures has been discussed [Bia97], [Mou96]. The difficulty of implementing and using efficiently multiprecision arithmetic on standard general purpose processors has been described in [Kar93].

Multiple precision has been a popular topic in computer arithmetic for a long time. Schwablander and Schulte have designed an interval arithmetic coprocessor [Sch95]. The interesting point they brought up is that arithmetic hardware was generally reusable, as long as some additional hardware was available to handle operand multiplexing and storage of results. Variable precision algorithms for multiplication, division and square-root have been developed and presented in [Sm96], [Tak95], [Lou95].

![Diagram of Multiplexed 4x8-bit input network](image)

Figure 2: Multiplexed 4x8-bit input network.

### 3. The New Approach

#### 3.1. Arithmetic

In order to evaluate the effectiveness of multiprecision logic for signal processing applications, we have developed and extensively simulated a number of hardware implementations of simple multiprecision adders. Operand multiplexing was recognized as the main delay overhead in multiprecision hardware schemes. We present a simple, and yet effective, hardware scheme for multiple precision addition. The strategy does not use multiplexing elements on the critical path of the addition operation. The scheme is illustrated in Figure 2. The 8-bit operands are forwarded to the 8-bit addition logic by shifting within the 32-bit input registers. Similarly, the result is shifted in the 32-bit register. Note that usage of shift registers enables simultaneous occurrence of addition and shift operations.

The above hardware structure has been implemented using the Berkeley 0.5 µm CMOS Low Power Standard Cell Library. A detailed SPICE v3.13 [Ber97] simulation has resulted in area and delay estimations as presented in Table 1. Power dissipation of particular adder structures has been estimated using the IR-SIM tool (an incremental MOS switch-level simulator) [Sal89]. Although relatively simple addition algorithms were considered, the resulting implementation characteristics are comparable to real-life designs. The experimentally determined model experimented was further used in the evaluation of the optimization algorithm suite.

### 3.2. Optimization Problems

We now formulate the optimization problems associated with low-power behavioral-level synthesis of multiprecision application specific designs and establish their computational complexity. Our synthesis approach has two optimization intensive phases: resource allocation and operation assignment. Due to the high level of available operation-level parallelism, the standard scheduling algorithm provides high quality scheduling. The targeted synthesis subproblems are defined in the standard Garey-Johnson [Gar79] format.

**Problem: Assignment of computations to allocated hardware resources for power-optimal execution under a timing constraint.**

**Instance:** Given a set of $A$ arithmetic units with corresponding operation precisions $K_i, i = 1,\ldots, A$ and power dissipation costs $A^\text{power}_i, i = 1,\ldots, A$, and a set of $N$ independent computation tasks with corresponding precisions $L_i, i = 1,\ldots, N$ and time lengths $T_i, i = 1,\ldots, N$, and positive numbers $\text{DeadLine}$ and $\text{MaxPower}$.

**Question:** Is there an assignment which assigns each computation task to exactly one arithmetic unit in such a way that the required time for execution of all $N$ operations does not exceed $\text{DeadLine}$ and the overall power dissipation per computation cycle is less than $\text{MaxPower}$?

**Problem:** Multiple precision arithmetic unit allocation for synthesis of a low-power ASIC datapath under an area and timing constraint.

**Instance:** Given a set of $A$ arithmetic units with corresponding operation precisions $K_i, i = 1,\ldots, A$ and associated area $A^{\text{area}}_i, i = 1,\ldots, A$ and power dissipation costs $A^\text{power}_i, i = 1,\ldots, A$, and a set of $N$ independent equal-precision computation tasks with corresponding precisions $L_i, i = 1,\ldots, N$ and time lengths $T_i, i = 1,\ldots, N$, and positive real numbers $\text{AreaLimit}$, $\text{DeadLine}$, and $\text{MaxPower}$.

**Question:** Is there a multishubset of arithmetic units
(subset where some arithmetic units can be included more than once) such that each computation task is assigned to exactly one arithmetic unit, the sum of costs of selected arithmetic units is at most \textit{AreaLimit}, the maximum workload on a particular unit is less than \textit{DeadLine}, and the overall power dissipation per computation cycle is less than \textit{MaxPower}?

The specified resource allocation and operation assignment subproblems for power optimization of variable precision application-specific systems have been proven to be computationally intractable [Erc96].

4. The Synthesis Algorithms

In this section, we elucidate the algorithms for resource allocation and operation assignment. The resource allocation algorithm is based on a multi-gradient search, while the operation assignment algorithm relies on a novel generalized and modified Karmarkar-Karp’s number partitioning heuristic. While searching for the power-minimal arithmetic unit configuration and operation assignment, the resource allocation algorithm iteratively invokes the assignment procedure. The latter procedure decides whether the generated current arithmetic logic unit configuration can produce an operation assignment which satisfies the real-time constraint under predetermined area bounds.

4.1. Resource Allocation

The resource allocation search is initiated by selecting a starting configuration of all “highest precision” units. The maximal current system power dissipation (MCSP) is defined as the difference of the initial solution power dissipation (cost) and the minimal cost differential ($\delta$) among all available units. Then, the search for the solution with the lowest cost is performed based on the steepest descent gradient search algorithm. The pseudo-code and illustration of the search algorithm are presented in Figure 3.

The gradient search for the configuration which results in the lowest energy consumption is performed along the subset of solutions with total area smaller than the MCSP. Upon each consideration of a new resource configuration, a simplified procedure for operation assignment is invoked. The procedure checks whether the current allocation is able to perform the required computation. When, for a particular MCSP energy consumption bound, a satisfiable hardware configuration is found, the MCSP \textit{MaxPower} bound is decreased by $\delta$. The search ends when no better solution is found in $W$ successive iterations (in our experiments we used $W = 30000$).

4.2. Operation Assignment

The operation assignment algorithm is described in detail using the pseudo-code in Figure 4. In order to provide an efficient algorithmic solution, we have developed a novel constructive algorithm by modifying the original Karmarkar-Karp’s number partitioning algorithm. The modifications answer the need for additional optimization requirements.

For a set of single-precision arithmetic units, the problem of task assignment can be reduced to the number ($N$ numbers) partitioning ($K$ partitions) problem. The additional optimization requirements are a direct consequence of available degrees of freedom for operation assignment due to the available multiple precision arithmetic units. When several different multiple precision units are used, an alternative solution technique is applied. The solution uses standard simulated annealing algorithm as the basic algorithmic approach. The quality of each current solution is adjusted by applying the generalized number partitioning heuristic to each subset of units with equivalent precision and their ass-

\begin{figure}
\centering
\includegraphics[width=\textwidth]{resource_allocation_diagram.png}
\caption{Resource allocation algorithm}
\end{figure}
signed operations. Note that the ultimate goal of the simulated annealing outer optimization shell is not to balance the workload on all units, nor to assign tasks to units with corresponding equivalent precision, but to explore the tradeoff and find the intermediate solution which results in the least power dissipation.

5. Experimental Results

In this section, we report the results of a set of experiments that have been conducted in order to evaluate the quality of our synthesis paradigm and developed optimization tools. A set of test cases has been developed using the DSP Quant benchmark suite [Lee97] and a set of real-life examples extracted from various multimedia and communications applications.

Preprocessing:
Sort the starting set S and group consecutive K elements into groups G_i.
Sort all G_i in decreasing difference G_i^d between the largest and smallest element in G_i.
For each G_i and its smallest E_i^{min},
For each E_j of S smaller than E_i^{min}:
Calculate the \( \sum_{all~groups} (G_i^d)^2 \) if S is rearranged in a way that E_i is added to \( E_i^{min} \).
If there is a non-empty set of elements such that they have a sum of squares of group differences smaller than current sum:
Select E_j from this set that has the minimal sum and add it to \( E_i^{min} \).
Sort S and generate G_i, G_i^d.

Processing:
For each group G_i:
Sort elements in each group in increasing order.
Sort partitions in decreasing order of sums.
For each \( E_i \) and partition \( P_i \) assign \( E_i \) to \( P_i \).

Postprocessing:
Repeat
For each group pair that has the min and max sum of elements, search for two elements that, when swapped, lessen the difference between the sums of groups.
while there are pairs that improve the result.

Figure 4: Generalized Karmarkar-Karp’s number partitioning heuristic.

The experimental results which describe the advantages of the multiprecision datapath design paradigm are shown in Table 2. The first column shows two numbers, where the first quantifies the number of operations to be assigned, and the second is the ratio of the predetermined multiprecision system cost and the cost of a system which does not exploit multiple precision units. The next two columns show the power consumption resulting from a behavioral compilation of a particular task composition onto a traditional fixed precision arithmetic system (second column) and multiprecision platform (third column). The results for both hardware structures are generated using the developed simulated annealing allocation platform. In each row both systems have the equivalent area constraint. The fourth column presents the percent improvement of the multi- versus fixed-precision design paradigm. Note that consistent power improvements were obtained while using the multiprecision design paradigm and that the last row in the table shows the average power improvement over series of task compositions and hardware platforms.

The efficacy of the developed generalized Karmarkar-Karp’s algorithm for multiset number partitioning is shown in Table 3. The first column represents the number of randomly generated numbers. They are partitioned into a number of sets shown in the second column. In the third column, for a particular test case, the relative offset of the largest/smallest sum of numbers in a set with respect to a lower bound is presented. The lower bound is established as a sum of all numbers divided with the number of sets. The percentages in the last column present worst case results in \( N \cdot S \) different number sets. The average marginal offset for the set of test benchmarks is shown in the last row.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Fixed Cost (32b) ( \mu W )</th>
<th>Multi Cost (8-16-32b) ( \mu W )</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>25-4</td>
<td>147</td>
<td>94</td>
<td>36%</td>
</tr>
<tr>
<td>50-7</td>
<td>311</td>
<td>190</td>
<td>39%</td>
</tr>
<tr>
<td>100-12</td>
<td>705</td>
<td>490</td>
<td>30%</td>
</tr>
<tr>
<td>250-13</td>
<td>1011</td>
<td>627</td>
<td>39%</td>
</tr>
<tr>
<td>250-22</td>
<td>1739</td>
<td>1164</td>
<td>33%</td>
</tr>
<tr>
<td>500-12</td>
<td>1902</td>
<td>1173</td>
<td>38%</td>
</tr>
<tr>
<td>750-13</td>
<td>3006</td>
<td>1021</td>
<td>46%</td>
</tr>
<tr>
<td>1000-25</td>
<td>8359</td>
<td>6242</td>
<td>25%</td>
</tr>
</tbody>
</table>

| Average improvement | 35% |

Table 2: Comparison of fixed and multiprecision datapath synthesis approaches.

6. Conclusion

We have developed and quantified the efficacy of a new set of behavioral level power optimization synthesis tools which allocate variable precision addition logic and map an arbitrary set of computation tasks onto the variable precision unit configuration. While the core of the optimization engine, the assignment algorithm, is based on a developed generalized Karmarkar-Karp’s
<table>
<thead>
<tr>
<th>Numbers</th>
<th>Sets</th>
<th>Maximum offset with respect to lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>4</td>
<td>0.69%</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>0.86%</td>
</tr>
<tr>
<td>100</td>
<td>12</td>
<td>0.63%</td>
</tr>
<tr>
<td>250</td>
<td>13</td>
<td>0.12%</td>
</tr>
<tr>
<td>250</td>
<td>22</td>
<td>0.39%</td>
</tr>
<tr>
<td>500</td>
<td>12</td>
<td>0.013%</td>
</tr>
<tr>
<td>750</td>
<td>13</td>
<td>0.0048%</td>
</tr>
<tr>
<td>1000</td>
<td>25</td>
<td>0.0086%</td>
</tr>
<tr>
<td>1000</td>
<td>112</td>
<td>0.30%</td>
</tr>
<tr>
<td>Average maximal offset</td>
<td></td>
<td>0.26%</td>
</tr>
</tbody>
</table>

Table 3: Efficacy of the number partitioning heuristic.

heuristic for multiset number partitioning embedded into a simulated annealing search shell, steepest descent search algorithm is employed to allocate a configuration which accommodates the specified set of computations for minimal energy consumption. Experimental results showed significant reduction in power dissipation (33% on the average) with respect to the fixed precision design methodology.

References
[BER97] http://www-cad.eecs.berkeley.edu/Software/