BEHAVIORAL SYNTHESIS OPTIMIZATION USING MULTIPLE PRECISION ARITHMETIC

Milos Ercegovac, Darko Kirovski, George Mustafa, and Miodrag Potkonjak

Computer Science Department,
University of California, Los Angeles, CA 90095-1596 USA
{milos,darko,mustafa,miodrag}@cs.ucla.edu

ABSTRACT

Modern image and video processing applications 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, high computational requirements, and high cost sensitivity. The current generation of behavioral synthesis tools does not address well this type of application.

In this paper we explore the potential of using multiple precision arithmetic units to effectively support implementation of image and video processing applications as application specific integrated circuits. A new architectural scheme for collaborate addition of sets of variable precision data is proposed as well as an allocation and assignment methodology for multiple precision arithmetic units. Experimental results indicate strong advantages of the proposed approach.

1. INTRODUCTION

Recently, the fast growing multimedia consumer market redefined the relative importance of design metrics for modern image and video processing (IVP) application specific integrated circuits (ASICs). The next generation of IVP ASICs is 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, high computational requirements, and high cost sensitivity. The current academic and industrial synthesis tools do not address well this type of requirement.

One of the key problems and currently ignored optimization potential, is a support for variable-length data types. The importance of employing multiple precision arithmetic units in IVP ASICs is well illustrated by the recent arithmetic and architectural trends in IVP programmable platforms. The majority of the latest general purpose architectures provide support for multiple precision execution units. For example, both the Intel MMX multimedia extension to Pentium Pro [Pel96] and the SUN UltraSparc II architecture [Gol96] provide instruction sets and adequate architectural support for execution of low-precision instructions in parallel on partitioned arithmetic logic units.

In this paper, for the first time, we combine arithmetic and behavioral synthesis techniques to explore the potential of multiple precision arithmetic units for area optimization of IVP systems on silicon. A new simple, yet powerful, hardware scheme for multicycle addition of variable precision data is proposed. The scheme is supported with developed synthesis software for resource allocation and computation assignment of multiple precision arithmetic units. The experimental results indicate high potential of the multiple precision paradigm as a viable design methodology in the development of IVP ASIC systems.

1.1. Motivational Example

A flavor of advantages of multiple precision arithmetic is illustrated using the following motivational example. A set of 15 additions is implemented on two architectures, traditional (fixed) and new (multiprecision). 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 needs". Relevant timing and area data are given in Table 1.

Computation		Arithmetic Units		
Prec.	Instances	Prec.	Area(μm^2)	Delay
8b	5	8b	$1.69 \cdot 10^5$	53.8ns
16b	5	16b	$3.9 \cdot 10^5$	71.7ns
32b	5	32b	$6.6 \cdot 10^5$	89.6ns

Table 1: Motivational example: Computations and arithmetic units (CLA adders) [Deadline = 360ns]

The first architecture, in order to satisfy the computation deadline, requires two 32-bit, one 16-bit and one 8-bit adder resulting in chip area of $18.8 \cdot 10^5 \mu m^2$. The delay of the system clock is set to $90ns$ and the computation requires 4 clock cycles. If the hardware units are allocated and the computation is assigned and scheduled as shown in Figure 1, the required area is equal to:

$$\text{Area} = 2 \cdot 3.9 \cdot 10^5 + 3 \cdot 1.69 \cdot 10^5 = 12.8 \cdot 10^5 \mu m^2$$

Since there are no 32-bit units and the computation deadline allows 5 cycles, the system clock is set to $72ns$.

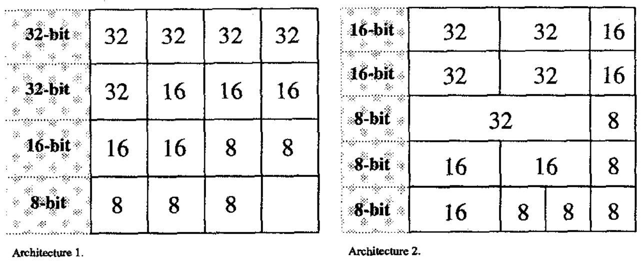

Figure 1: Motivational example: Optimized allocation and scheduling

For an efficient optimization methodology which uses multiple precision units, the following two issues are crucial. First, operations of higher precision demand execution on lower precision arithmetic units with minimum delay 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.

2. RELATED WORK

Recently, multiple precision execution units have been augmented in new general purpose architectures (for example, 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 IVP applications such as MPEG decoding [Zho95]. Efficient software utilization of the Pentium MMX and UltraSPARC VIS architectures has been discussed [Bli97], [Mou96]. The difficulty of implementing and using efficiently multiprecision arithmetic on standard general purpose processors has been described in [Kar93], [Rog95].

Multiple precision has been a popular topic in computer arithmetic for a long time. Schwartzlander 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 [Smi96], [Tak95], [Lou95].

3. THE NEW APPROACH

3.1. Arithmetic

In order to evaluate the effectiveness of multiprecision logic in IVP 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, paradigm for multiple precision addition hardware. 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.

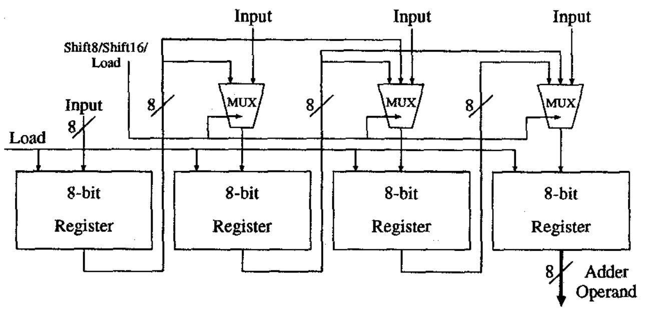

Figure 2: Multiplexed 4x8-bit input network.

The above hardware structure has been implemented using the Berkeley 0.5 μm CMOS Low Power Standard Cell Library. A detailed SPICE simulation has resulted in area and delay estimations as presented on Table 1. Although relatively simple addition algorithms were considered, the resulting performance is comparable to real-life designs. The model experimented is used in the optimization algorithm evaluation.

3.2. Optimization Problems

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

Problem: Multiple precision arithmetic unit allocation for synthesis of area-minimized ASIC datapath.

Instance: Given a set of A arithmetic units with corresponding operation precisions $K_i, i = 1,..,A$ and associated costs $A_i^{area}, i = 1,..,A$, and a set of N independent computations with corresponding precisions $L_i, i = 1,..,N$ and positive real number C.

Question: Is there a multisubset of arithmetic units (subset where some arithmetic units can be included more than once) such that each computation is assigned to exactly one arithmetic unit and the sum of costs of selected arithmetic units is at most C?

Problem: Assignment of computations to allocated hardware resources for optimal execution performance.

Instance: Given a set of A arithmetic units with corresponding operation precisions $K_i, i = 1,..,A$ and associated costs $A_i^{area}, i = 1,..,A$, and a set of N independent computations with corresponding precisions $L_i, i = 1,..,N$ and positive integer $Deadline$.

Question: Is there an assignment which assigns each computation to exactly one arithmetic unit in such a way that the required time for execution of all N operations does not exceed $Deadline$?

We proved that the allocation and assignment subproblems for IVP area and performance optimization are NP-complete. The proof is based on the classical Karp's polynomial time reduction technique and the number partitioning problem as the starting point [Gar79].

4. ALLOCATION AND ASSIGNMENT

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 area-minimal arithmetic unit configuration, the resource allocation algorithm iteratively invokes the assignment procedure. This procedure decides whether the generated current ALU configuration can produce an operation assignment whicih satisfies the real-time constraint.

4.1. Resource Allocation

The resource allocation search is initiated by selecting a starting configuration of all "highest precision" units. The maximal current system cost (MCSC) is defined as the difference of the initial solution cost and the minimal cost differential (δ) 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.

```
Define initial solution
Decrease MCSC;
    Subtract X Units of M-bits
    Add Y Units of N-bits
    The cost < COST_LINE
    ASSIGNMENT(CurrentConf)
    if (AssignTime < DEADLINE)
        break
while (NO_IMPROVEMENT < K)
```

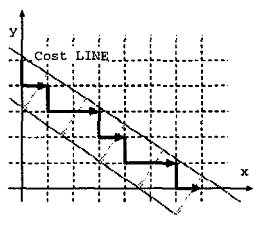

Figure 3: Resource allocation algorithm

The gradient search for the best configuration is performed along the subset of solutions with total area smaller than the MCSC. 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 MCSC area bound, a satisfiable hardware configuration is found, the MCSC area bound is decreased by δ. The search ends when no better solution is found in K successive iterations (in our experimentations we used $K = 10000$).

4.2. Operation Assignment

The operation assignment algorithm is described in detail using the pseudo-code in Figure 4. In order to provide efficient algorithmic solution, we developed a novel constructive algorithm by modifying the original Karmarkar-Karp's number partitioning algorithm. The modifications answer the need for additional optimization requirements. Namely, for a set of single-precision arithmetic units, the problem of computation 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 operation assignment freedom due to the available multiple precision arithmetic units. When several different multiple precision units are used, an alternative solution technique is used. 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 assigned operations.

```
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 $E_i$ 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 ¡
    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 can improve the result.
```

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

5. EXPERIMENTAL RESULTS

In this section we report the results of experimentations 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. 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 allocation results for the traditional fixed precision arithmetic system and multiprecision approach respectively. The results for both approaches are generated using the developed simulated annealing allocation platform. The fourth column presents the percent improvement of the multi- versus fixed precision design paradigm. Note that consistent area improvements were obtained while using the multiprecision design paradigm and that the last row in the table shows the average area improvement over series of design tasks.

Tasks - Cost	Precision		Impr.
	Fixed (32b) (μm^2)	Multi (8-16-32b) (μm^2)	
25 - 4	2.61e+06	1.41e+06	46.45%
50 - 7	4.62e+06	2.42e+06	47.52%
100 - 12	7.92e+06	4.84e+06	38.84%
250 - 13	8.58e+06	4.39e+06	48.75%
250 - 22	1.45e+07	0.99e+07	31.53%
500 - 12	7.92e+06	4.44e+06	43.89%
750 - 13	8.56e+06	4.44e+06	48.21%
1000 - 25	1.65e+07	1.02e+07	38.46%
Average improvement			42.97

Table 2: Comparison of fixed and multiprecision datapath synthesis approaches

Numbers N	Sets S	Maximum offset with respect to lower bound
25	4	0.694416%
50	7	0.868912%
100	12	0.63868%
250	13	0.125221%
250	22	0.396182%
500	12	0.0135149%
750	13	0.00484549%
1000	25	0.00868861%
1000	112	0.309386%
Average maximal offset		0.2627%

Table 3: Efficiency of the generalized Karmarkar-Karp's number partitioning heuristic.

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 maximal offset for the set of test benchmarks is shown in the last row.

6. CONCLUSION

We proposed and quantified the efficacy of a new arithmetic scheme for variable precision addition logic, and developed resource allocation and task assignment algorithms as a part of our behavioral-level design automation methodology for designing area-efficient IVP systems-on-silicon. The assignement algorithm is based on a developed heuristic for multiset number partitioning. Experimental results showed area savings with respect to the fixed precision design methodology in the range of 31-48 percents.

7. REFERENCES

[Bli97] Blinn, J.F. Fugue for MMX. IEEE Computer Graphics and Applications, vol.17, (no.2), pp.88-93, 1997.

[Gar79] Garey, M.R.; Johnson, D.S. Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman, San Francisco, CA, 1979.

[Gol96] Goldman, G.; Tirumalai, P. UltraSPARC-II: the advancement of ultracomputing. COMPCON '96.

[Kar93] Karmer, W. Multiple-precision computations with result verification. Scientific computing with automatic result verification, Academic Press, pp.325-56, 1993.

[Lee97] Lee, C. et. al. DSP Quant: Design, Validation, and Applications of DSP Hard Real-Time Benchmarking. ICCASP '97.

[Lou95] Louie, M.E.; Ercegovac, M.D. A variable-precision square root implementation for field programmable gate arrays. Journal of Supercomputing, vol.9, (no. 3): 315-36, 1995.

[Mou96] Mou, Z.J.A.; Rice, D.S.; Wei D. VIS-based native video processing on UltraSPARC. International Conference on Image Processing, pp.153-6, 1996.

[Pel96] Peleg, A.; Weiser, U. MMX technology extension to the Intel architecture. IEEE Micro, vol.16, (no. 4): 42-50, 1996.

[Rog95] Rogers, J. Using the multiple precision library. Dr. Dobb's Journal, vol.20, (no.1), p.36, 38, 40, 42, 86, 88-9, 1995.

[Sch95] Schulte, M.J.; Swartzlander, E.E., Jr. Hardware design and arithmetic algorithms for a variable-precision, interval arithmetic coprocessor. 12th Symposium on Computer Arithmetic, 1995.

[Smi96] Smith, D.M. A multiple-precision division algorithm. Mathematics of Computation, vol.65, (no. 213): 157-63, 1996.

[Tak95] Takagi, N. A multiple-precision modular multiplication algorithm with triangle additions. IEICE Transactions on Information and Systems, vol.E78, 1995.

[Zho95] Zhou C.G. et. al. MPEG video decoding with the UltraSPARC visual instruction set. COMPCON '95.