Region-Level Approximate Computation Reuse for Power Reduction in Multimedia Applications*

Xueqi Cheng and Michael S. Hsiao ({xcheng, hsiao}@vt.edu)
Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, 24061

ABSTRACT

Motivated by data value locality and quality tolerance present in multimedia applications, we propose a new micro-architecture, Region-level Approximate Computation Buffer (RACB), to reduce power consumption in such applications. The proposed RACB relaxes the exact matching into partial and approximate tag matching and applies it to regions of code in a program, thereby allowing for aggressive computation/execution reduction, in addition to reductions in memory accesses and pipeline activities. Our experiments demonstrate that a 64-entry RACB can yield up to 70% of region-level execution reduction without noticeable quality degradation in MPEG-2 video decoding, corresponding to 55.9% of system power savings with respect to the regions.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: *signal processing systems*

General Terms: Design, Algorithms, Performance

Keywords: Approximate computation reuse, Critical sequence, Low-power design

1. INTRODUCTION

With the increasing presence of multimedia functions such as audio, video playback and speech synthesis in battery-powered devices and smart sensors, power management has become a top design consideration due to the limited energy budget and battery capacity.

The motivation of our work comes from two characteristics common in multimedia applications: (1) Much computation involves highly regular operations with significant value locality, including large regions of arithmetic operations within loops, and (2) slight quality degradation may be unperceivable or tolerable to the human visual and auditory systems. For the first characteristic, some application-specific processors have reduced the precision on the on-chip arithmetic units to reduce power consumption; however, general-purpose processors found in laptops, PDAs, and future smart phones do not have such liberty. System-level variable bitwidth optimization was also proposed[3,4], but the actual execution of all the computations still needs to be performed, including floating-point arithmetic; For the second characteristic, some loss of QoMD (Quality of Multimedia Data) may be allowed, as the end user's perception and tolerance of media quality may be directly linked to the current multimedia service and device battery states. By combining both of the aforementioned characteristics, this work explores the tradeoff between QoMD and energy consumption.

In order to reduce the cost of large amount of arithmetic computation, computation reuse[1,2] is commonly applied with the aim of improving program performance as well as power consumption via reducing the computational redundancies in the programs. Through caching the operands and results of previous computations, a future computation with the same operands can use the corresponding cached results directly without having to actually execute the computation again via the functional units. Region-level computation reuse[5,6] extended the redundant execution elimination into computation sequences that having the same inputs values. In addition, fuzzy computation reuse[7,8] was proposed seeking for more computation skipping via reduced-precision tag-matching with tolerable quality degradation in multimedia files.

In this paper, we investigate the potential of fuzzy computation reuse at the granularity of regions. We further extend it to a two-dimensional approximation: region-level partial computation sequence matching and instruction-level approximate operand matching. A small cache-like structure called Region-level Approximate Computation Buffer(RACB) is proposed to store previous regions of computation instances. With this RACB, a large amount of arithmetic computations, memory accesses, and pipeline activities with respect to a region can be skipped. Intuitively, more skipping can be achieved with RACB compared with the conventional exact-match execution caches. As a result, significantly more power can be reduced with tolerable quality degradation. Our experiments on a real life application, MPEG-2 video decoder showed that with a 64-entry RACB, up to 70% of region-level computation/instruction executions can be skipped without noticeable quality degradation in output videos, which results in 55.9% of system power savings in the region.

2. REGION-LEVEL APPROXIMATE COMPUTATION BUFFER

2.1 Region-Level Approximate Computation Reuse

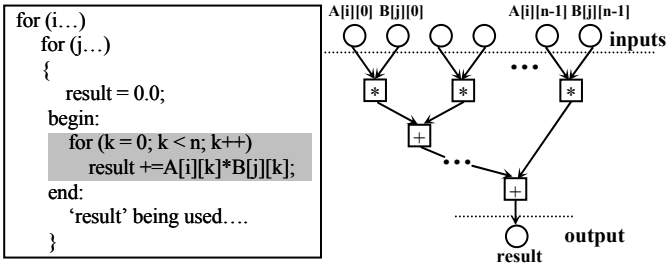

Figure 2.1 (a) Example Code Segment (b) Data Dependency Flow

Studies have shown that the execution of multimedia applications usually exhibit high data value locality. Figure 2.1(a) shows an example pseudo-code segment whose format can be commonly found in the IDCT (Inverse Discrete Cosine Transform) process in signal processing; the corresponding data dependency flow graph of the shaded code is shown in Figure 2.1(b), with 2*n* values from arrays A and B as inputs, and 'result' as output. The subsequent instructions after the shaded for-loop depend only on the value of 'result'. If the multimedia convolution filter process, e.g., IDCT, contains such shaded computation region as shown in Figure 2.1(a), the operands from arrays A and B, which are usually media source related, would exhibit significant data value locality due to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ISLPED'05, August 8-10, 2005, San Diego, California, USA.
Copyright 2005 ACM 1-59593-137-6/05/0008...$5.00.

similarities in pixels within a local region. This in turn implies high value locality in the region output 'result'. Thus, we predict that significant potential exists in computation reuse in this region.

Given a computation region R with N inputs and M outputs, the conventional region-level computation reuse techniques would skip the actual computation in R when *all* the incoming N inputs match a cached computation instance in a buffer, and the corresponding stored M results will be used directly. However, requiring all operands available would decrease the potential savings brought by execution skipping especially in deeply pipelined architectures because a large number of instructions within the region that are responsible for obtaining the values of input operands have to be executed, including expensive memory accesses. Furthermore, execution cache hitrate will decrease significantly when all the input operands need to be compared, so that the power overhead of the cache may not be compensated by the savings brought by the skipped computation. Finally, if we wish to retain the high cache hit rate by increasing the cache size, both the area and power overhead will increase. In order to solve this dilemma, we introduce the concept of region-level *partially* matched region reuse combined with instruction-level approximate matching to obtain higher rate of computation skipping, together with reduction in memory and pipeline activity with tolerable quality degradation.

The main idea is: Use the region-level approximate computation buffer to store previous computation results of regions; then, for a given computation region with N inputs $\{a_1,\ldots,a_N\}$ and M outputs $\{o_1,\ldots,o_M\}$, we search the computation buffer to see whether a bypass is possible using approximate matching for the first k operands a_i, where $k<<N$. Approximate matching means that we try to find another computation sequence stored in the buffer with operands c_i that satisfy the condition: $a_i \approx c_i$ (where $1 \le i \le k$) within a range of error, then use the stored results $\{o'_1,\ldots,o'_M\}$ that obtained from $\{c_1,\ldots,c_N\}$ as the approximate result of $\{o_1,\ldots,o_M\}$. Subsequently, the actual computation on $\{a_1,\ldots,a_N\}$ can be skipped even if no region with such input values have been computed before. A computation region is considered as a candidate for region reuse satisfies following conditions: 1) the outputs $\{o_1,\ldots,o_M\}$ can be computed from the inputs $\{a_1,\ldots,a_N\}$ through a series of arithmetic operations; 2) the subsequent instructions after the region depend only on the values of $\{o_1,\ldots,o_M\}$;

In this paper, we assume that the programmer has knowledge of the parts of computation region that can accept approximate results and identifies them with *reuse_flag*, which is set/unset right before/after the region(e.g. at the 'begin'/'end' label in Figure 2.1(a)). In order to simplify the hardware implementation, we also assume that the target region(s) contains no conditional control instructions.

2.2 The Proposed Architecture

Figure 2.2 depicts the basic model of the caching structure: Region-Level Approximate Computation Buffer (RACB). A RACB consists of an array of computation entries. Each entry represents a sequence of computation instances, which is defined as the values of all the input operands (floating-point or fixed-point) in a given region, together with the corresponding output results. Each entry contains three fields: 1) a valid bit indicating the availability of the entry; 2) a set of N input operands, part of their values form the computation tags and are used for identifying the computation instances; 3) the computation results that were stored after last execution. The RACB is indexed by portions of input values of the computation sequence. A tag checking is performed before a target region is decided either being skipped or executed normally.

Two levels of approximation are provided: At the region-level, only the first k operands are being compared, the value of k is called Valid Comparison Sequence (**VCS**). For each individual operand within VCS, the bits included in partial tag matching comprise both sign and exponent (for floating-point) fields, together with part of the fractional fields, whose low-precision bits are masked out by a Comparison Mask. The number of consecutive non-zero bits in the Comparison Mask is called Valid Comparison Bitwidth (**VCB**). Take single-precision floating-point for example, if VCS is k and VCB is w, then the total tag checking includes only $k*(1+8+w)$ bits instead of $N*(1+8+23)$ bits of comparison, where $k<N$, $w<23$. Experiments showed that k and w could be very small without influencing the output media quality much. As a result, significant more power can be reduced with this narrow-tag checking compared to wide-tag checking in conventional exact matching computation reuse buffer.

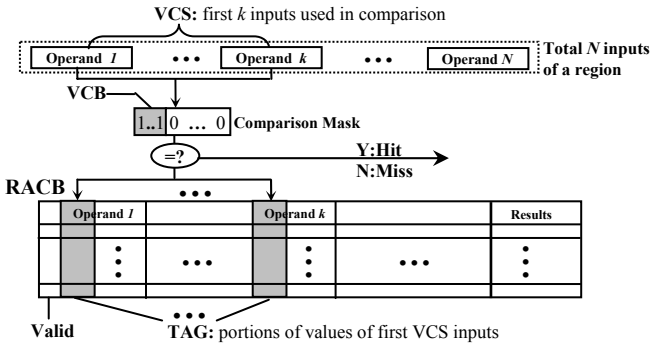

Figure 2.2 Model of Region-Level Approximate Computation Buffer

Both the value of VCS and VCB can be customized by the end users to adjust the final output QoMD according to user requirement and current battery level. The access to RACB will be disabled when k and w reaches certain values and the execution skipping drops below a threshold so that applying RACB causes more power than normal executions without extra cache access overhead for each computation region. All the computations will thus follow the original data path.

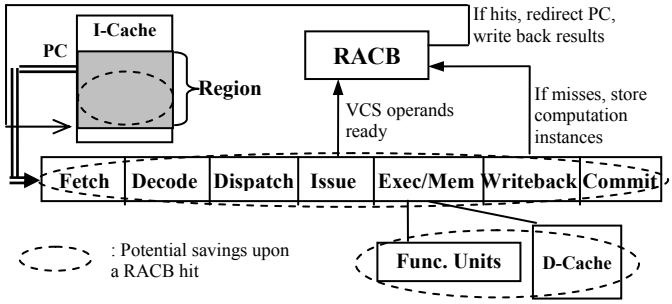

Figure 2.3 Pipeline Architecture with RACB

Figure 2.3 shows a pipeline architecture with RACB. When the region seeking for approximate reuse enters the pipeline, that is, the instruction that sets the *reuse_flag* is fetched, the following executions related to obtaining values of the first VCS input operands will be executed normally, otherwise instructions will be held from issuing even if they are ready. Then RACB is accessed to find whether a skip is possible, upon a hit, the values stored in the result fields of the cache are directly written back to the target registers, and at the same time, all the subsequent instructions in the region that have entered pipeline will be squashed immediately, the next PC will be redirected to the *reuse_flag* reset instruction of this region to completely skip all the instructions within the region that haven't been fetched in. When a miss in RACB occurs, all the held instructions that are ready will be issued and the execution will be performed normally. At the end of the region, the entire instance of computation sequence together with the result values will be stored into an entry of the RACB for future retrieval.

Take the region in Figure 2.1 for example, Figure 2.4 shows its corresponding pseudo code and instruction states when the RACB scheme is applied. We can see that when VCS=2, only the instructions that load the first 2 operands (i.e. A[i][0] and B[j][0]) from the data cache will be executed; all subsequent multiplications, adds, and other instructions (including memory loads of other operands e.g. A[i][1] and B[j][1]) are held, even if they are ready in pipeline. When a RACB hit occurs, not only the held instructions within the pipeline will be squashed, all the rest instructions in the region that are currently outside the pipeline will be skipped completely. Therefore, all the arithmetic computations in the region as well as all the memory accesses within the shaded sequence in Figure 2.4 can be skipped upon a RACB hit. In addition, the pipeline activities related to the skipped sequence such as, fetching, issuing, register file access etc. can all be reduced, which significantly increases the potential savings in system power compared to conventional full region computation reuse.

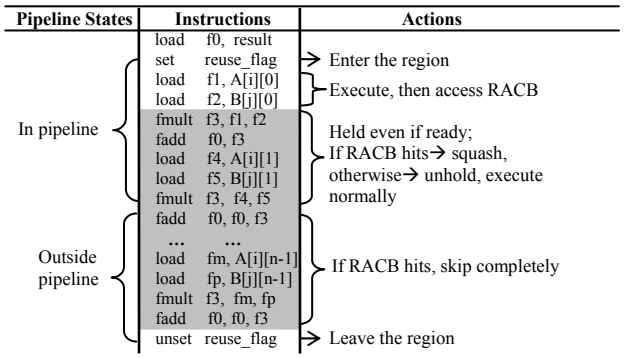

Figure 2.4 Example of RACB Applied on a Region(VCS=2)

2.3 Estimation of Power Savings

Suppose we define the power consumption of executing a computation region to consist of three parts: 1) P_{mem}, power consumed by memory accesses including load/store of data cache/memory; 2) P_{func}, power of functional units for arithmetic executions; 3) P_{pip}, power of all the pipeline activities other than 1) and 2) during instructions' lifetime within the pipeline from being fetched to commit. Then, the total power consumption for executing the flagged regions in the whole program under RACB scheme can be modeled as P_{region} in Equation (1):

$$P_{region} = P_{RACB} + r * k/N * P_{mem} + r * k'/N * P_{pip} + (1-r) * (P_{mem} + P_{pip} + P_{func}) \quad (1)$$

where the average power for every RACB access is denoted as P_{RACB}, which includes the power consumed by all the additional control logic, tag comparisons, execution redirection, etc.; r is the hit rate of RACB, k and N represent the VCS and total input operands within the region respectively; k' stands for the number of input operands that are being processed in the pipeline when RACB is accessed, where $k' \geq k$, to simplify the modeling, we will assume the ideal situation, that is, $k'=k$, then, the Equation (1) can be rewritten as Equation (1)':

$$P_{region} = P_{RACB} + r * k/N * (P_{mem} + P_{pip}) + (1-r) * (P_{mem} + P_{pip} + P_{func}) \quad (1)'$$

P_{reg_save} in Equation (2) is the percentage of power saved in the system with respect to the target regions by applying the partial and approximate computation reuse compared to normal executions.

$$P_{reg\_save} = \left[(P_{mem} + P_{pip} + P_{func} - P_{region}) / (P_{mem} + P_{pip} + P_{func}) \right] * 100\% \quad (2)$$
$$= \left[r * (1 - \frac{k}{N} * \frac{P_{mem} + P_{pip}}{P_{mem} + P_{pip} + P_{func}}) - \frac{P_{RACB}}{P_{mem} + P_{pip} + P_{func}} \right] * 100\%$$

Let $P_{ALL} = P_{mem} + P_{pip} + P_{func}$, since given a region, the ratio $(P_{mem}+P_{pip})/P_{ALL}$ is defined for a given microprocessor architecture, according to the Equation (2), the extent of power savings thus depends on 1) the RACB hit rate r; 2) k/N, the ratio between the VCS and total number of input operands; and 3) P_{RACB}/P_{ALL}, the ratio between average power consumption of RACB and the pipeline for the computation region. Tradeoffs must be considered between the hardware design, RACB parameter settings as well as application characteristics in order to achieve the desired power reduction. To simplify the discussion, we assume $(P_{mem}+P_{pip})/P_{ALL}$ to be 3/4 in this paper; Based on Equation (2), we can estimate the system power savings with respect to the entire program in Equation (3) assuming power consumption being proportional to the execution time.

$$P_{sys\_save} = R_{region} * P_{reg\_save} \quad (3)$$

where R_{region} is the ratio between execution time of the regions and the entire program. Higher proportion of region execution indicates greater potential of power savings in system.

3. EXPERIMENTAL RESULTS

In this section, we present our results on a MPEG-2 video decoder (MSSG, transform from 'double' to 'float' for floating-point experiments) for a number of video clips. We implement the RACB model in SimpleScalar toolset 3.0[10], integrate it into the original out-of-order pipeline architecture, and use SimpleScalar annotation to flag the reuse regions on which RACB is applied. The baseline configuration of our experiments is shown in Table 1.

Table 1 Baseline Configuration

Parameter	Value
Instruction window	64-RUU, 32-LSQ
Fetch/Decode/Issue/Commit	4 instructions/cycle
Functional units	4 Int/4FPs ALUs, 1Int/1FP mul/div
L1 D-cache	16kB 4-way LRU, 1 cycle latency
L1 I-cache	16kB 2-way LRU, 1 cycle latency
L2 cache	256k 4-way LRU, 6 cycles latency
Memory latency	18 cycles

Two regions within IDCT process having the similar format as in Figure 2.1 where n=8, N=16 are picked as target regions, which occupy most of the computation in IDCT, the most power hungry function in the whole application. All results are obtained using a 64-entry direct-mapped RACB and are mean of results from several video clips under the same parameter configurations.

3.1 QoMD for Floating-Point Video Files

We conduct experiments to evaluate the impact different VCS and VCB can have on the resulting media quality, which is a measurement of how applicable our approach can be. Mean Opinion Score(MOS) is chosen as the quality metrics, i.e., the average of the quality scores on a scale of 1 to 5, assigned to it by a number of listeners. (where 4-5: 'good', 2-3: 'fair', 1: 'poor').

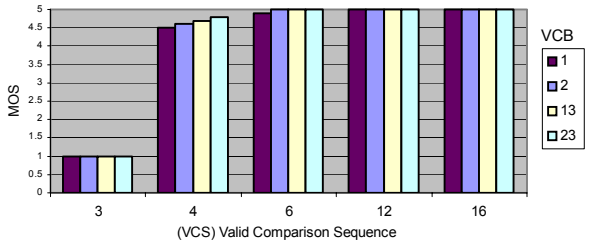

Figure 3.1 VCS/VCB vs. MOS

Figure 3.1 indicates that quality degradation is undetectable by humans until the VCS dropped from the original 16 to 4. Different value of VCB (i.e., the precision of floating-point fraction) has little impact on the output quality due to video's high value locality. Given that the quality of 'good' (MOS of 4-5) represents a level acceptable by humans, then the minimum VCS that can provide such output quality is called *critical sequence*. In this case, the critical sequence is 4. This experiment proves that a computation result can be approximated from a previous sequence that has the same critical

sequence for any value of VCB. Furthermore, if the critical sequence is applied, not only may regions be skipped much earlier before all the input operands are ready, RACB tag checking also becomes a narrower word comparison instead of the original 16*32 bit comparison, which will further decrease the P_{RACB}/P_{ALL} and k/N ratio, thereby achieving greater power savings.

3.2 RACB Hit Rate

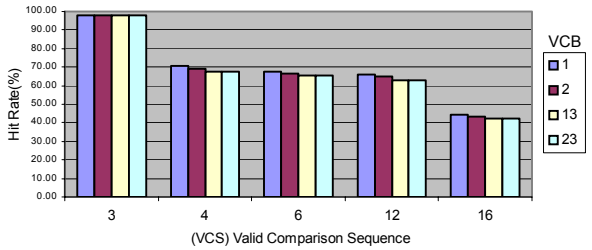

Figure 3.2 VCS/VCB vs. Cache Hit Rate

Figure 3.2 shows how VCS and VCB influence the RACB hitrate. We observe that VCS is the major factor for the hitrate, while VCB only influences the hitrate slightly under the same VCS. The hitrate reaches 70% for video decoder when VCS is set to be the critical sequence(=4). This result also demonstrates the effectiveness of our approach in expanding the data value locality. The hitrate for the RACB (when VCS=4, VCB=1) is around 30% higher than the fully and exact matching computation reuse technique (when VCS=16, VCB=23) without noticeable quality degradation. Therefore, partial and approximate matching has a higher energy savings potential than conventional approach even if they have the same ratio of P_{RACB}/P_{ALL}.

3.3 Power Savings Analysis

According to the preceding experimental results, for floating-point video decoder with acceptable output quality, the critical sequence is as low as 4. Suppose for each operand being compared, VCB=1, then, adding the bitwidth of the sign and exponent fields in VCS operands, the total bit number needed for an RACB tag comparison is only 4*(1+8+1)=40 (where exact match requires 16*32=512 bits). In order to estimate the power consumption ratio P_{RACB}/P_{ALL}, we implemented a direct-mapped RACB with 40-bit tags in VHDL and synthesized it using Synopsys Design Compiler under SunOS 5.8. For the pipeline execution power, we extracted the IDCT process from the MPEG-2 video decoder and use the Wattch 1.02[9] power model (assuming the aggressive ideal clock-gating cc2 model) to measure the power consumption for the target regions. The processor model used in Wattch is scaled to be the same as Synopsys, i.e., V_{dd}=5V, clock period=100ns and 1μm process technology. The results of gate-level power estimates of RACB using Synopsys Power Compiler and power per instruction of target region using Wattch are listed in Table 2.

Table 2 Power Evaluations of RACB and Pipeline Power Per Inst.

Components	Power(mw)	P_{RACB}/P_{ALL}
64-entry direct-mapped RACB	0.054	<0.01
Power Per Inst. for the Region	980	

Based on the results, we can pessimistically approximate that our RACB consumes much power: $P_{RACB}/P_{ALL} \approx 0.01$ where P_{ALL} is the power for executing the whole region. According to our results, for such a RACB, the hit rate of region reuse can reach 70% without causing noticeable quality degradation. Therefore, with the $(P_{mem}+P_{pip})/P_{ALL}$ to be 3/4 and VCS to be the critical sequence, (so that k/N=4/16=0.25), based on Equation (2), we can obtain 55.9% of region execution power savings by applying the RACB technique.

The simulation results from SimpleScalar indicate that the execution time in the target regions occupies 26.29% of the total execution time of the video decoder. Then, by Equation (3), we estimate that 55.9%*26.29%=14.7% system power can be saved with respect to the *entire* application while retaining acceptable quality. Figure 3.3 illustrates the tradeoff between power savings and output quality degradation under different VCS values, with VCB set to 1. We can see that our approach can achieve significant savings in both region and system power consumption while still retaining acceptable output quality. It further demonstrates that the critical sequence provides an adjustable variable for balancing power savings and tolerable quality degradations.

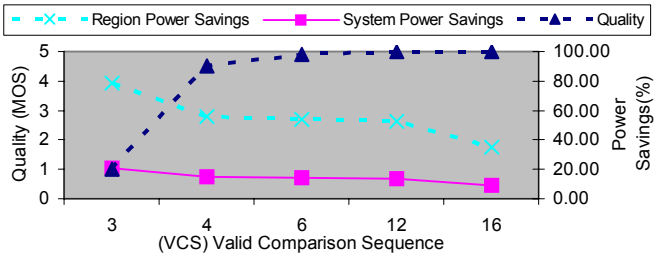

Figure 3.3 Power Savings vs. Output Quality

4. CONCLUSIONS

This paper presents a novel region-level partial and approximate computation reuse for computation-intensive multimedia applications. Energy savings can be achieved by skipping execution of arithmetic computations, memory accesses, as well as pipeline activities through the proposed Region-Level Approximate Computation Buffer. With tolerable degradation of output media quality, a 64-entry direct-mapped RACB can skip 70% of region-level computations/executions in video, almost 30% higher than the execution skipping achievable by the conventional fully and exact matching. Based on the Synopsys and Wattch power simulations, the corresponding power reduced for the region can be 55.9% and energy reduced for the entire system at 14.7%.

5. REFERENCES

[1] M. Azam, P. Franzon and W. Liu, "Low power data processing by elimination of redundant computations," in *Proc. of ISLPED*, 1997.

[2] S. E. Richardson, "Caching function results: faster arithmetic by avoiding unnecessary computation," *Technical Report SMLI TR-92-1, Sun Microsystems Laboratories*, 1992.

[3] Y. Cao and H. Yasuura. "Quality-driven design by bitwidth optimization for video applications," in *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference,* Jan 2003.

[4] F. Fang, T. Chen and R. A. Rutenbar. "Floating-point bit-width optimization for low power signal processing applications," in *Intl. Conf. on Acoustic, Speech and Signal Processing*, 2002.

[5] W. Wang, A. Raghunathan and N. K. Jha. "Profiling driven computation reuse: an embedded software synthesis technique for energy and performance optimization," in *Intl. Conf. on VLSI Design*, Jan 2004.

[6] D. A. Connors and W. W. Hwu. "Compiler-directed dynamic computation reuse: rationale and initial results," in *Proc. of the 32nd Annual Intl. Symposium on Microarchitecture* (MICRO), Nov 1999.

[7] C. Alvarez, J. Corbal, E. Salalmi and M. Valero. "Initial results on fuzzy floating point computation for multimedia processors," in *Computer Architecture Letters*, Vol. 1, Jan. 2002.

[8] C. Alvarez, J. Corbal, E. Salalmi and M. Valero. "On the potential of tolerant region reuse for multimedia applications," in *Proceedings of the 15th International Conference on Supercomputing,* 2001.

[9] D. Brooks, V. Tewari and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proc. of the 27th Annual Intl. Symposium on Computer Architecture*, Jun. 2000.

[10] D. Burger, T. Austin and S. Bennett. "Evaluating future microprocessors: the SimpleScalar tool set." *Technical Report CS-TR-96-1308, University of Wisconsin-Madison*, Jul 1996.

* This work is supported in part by NSF under grant 0121416.