DYNAMICALLY PARAMETERIZED ALGORITHMS AND ARCHITECTURES TO EXPLOIT SIGNAL VARIATIONS FOR IMPROVED PERFORMANCE AND REDUCED POWER

W. Burleson, R. Tessier, D. Goeckel, S. Swaminathan, P.Jain, J. Euh, S. Venkatraman, V. Thyagarajan

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst
burleson@ecs.umass.edu

ABSTRACT

Signal processing algorithms and architectures can use dynamic reconfiguration to exploit variations in signal statistics with the objectives of improved performance and reduced power consumption. Parameters provide a simple and formal way to characterize incremental changes to a computation and its computing mechanism. This paper examines five parameterized computations which are typically implemented in hardware for a wireless multimedia terminal: 1) motion estimation, 2) discrete cosine transform, 3) Lempel-Ziv lossless compression, 4) 3D graphics light rendering and 5) Viterbi decoding. Each computation is examined for the capability of dynamically adapting the algorithm and architecture parameters to variations in their respective input signals. Dynamically reconfigurable low-power implementations of each computation are currently underway.

1. INTRODUCTION

Reconfigurable computing has been proposed for signal processing with various objectives, including high-performance, flexibility, specialization and most recently adaptability. The actual reconfiguration can take many forms and is typically characterized by: 1) how fast the reconfiguration can occur, and 2) how much is actually reconfigured, and finally, 3) how many possible configurations are used.

In this work we propose that reconfiguration of algorithms and architectures can be used to improve performance and reduce power consumption. This objective can be achieved in at least two ways: 1) compromising the resulting quality of the algorithm, or 2) exploiting variations in the signals and necessary computations required to achieve a given level of quality. Rabaey [1] proposed a similar objective and has demonstrated it in a heterogeneous configurable baseband processor [4]. However, they have focused on reconfiguration for task specialization and standards specialization utilizing domain-specific processors with varying degrees of configuration granularity interconnected with hierarchical meshes. Others have shown the ability to use dynamic reconfiguration to implement virtual hardware at a number of levels, e.g. [2]. Our work focuses on dynamic reconfiguration of algorithms and architectures to adapt at various rates to variations in signals and their associated computations. Signal processing has a rich tradition of adaptive algorithms which estimate statistics of signals, channels and noise and then modify their computation accordingly. However this usually results in modification of coefficients rather than significant changes in the control flow of the programs. Although traditional signal processing algorithms are characterized by small programs and limited control flow changes, recent DSP trends have led to much more complex and data-dependent computation (e.g. MPEG4, adaptive wireless). This trend has been aided by the widespread use of software-based DSPs and compilers to implement complex heterogeneous applications. This paper takes a first step at control-flow adaptation and configuration by using signal statistics to dynamically modify parameters of domain-specific modules. We explore 5 modules that are important components in a larger heterogenous system for wireless multimedia.

2. CONFIGURABLE ARCHITECTURES

A variety of recent work has addressed the use of configurable and programmable architectures for video compression. This work avoids the completely reconfigurable approach using FPGAs. FPGAs do allow parallelism, pipelining, local memory and both functional and data specialization. They also allow generic prototyping and are well supported by design tools and libraries of pre-designed components. However FPGAs suffer from several disadvantages for low-power DSP [1]. FPGAs are relatively slow at sequential computations when compared to microprocessors or DSPs. FPGAs are not power efficient due to their high levels of programmability and lack of support for memory-intensive computation. FPGAs also have slow and inefficient reconfiguration mechanisms since they were not designed to support high-speed dynamic reconfiguration.

Recently it has been widely recognized that heterogeneous domain-specific reconfigurable architectures may be more appropriate for DSP than FPGAs [1]. [4] demonstrated functioning silicon for a heterogeneous configurable architecture for low-power video processing including on-chip low-power FPGA, ARM processor and dedicated circuitry for DCT. Numerous recent commercial offerings from FPGA vendors provide combinations of FPGA, DSP, microcontroller and memories.

In this work we take a longer term approach leveraging trends in VLSI which will allow much larger systems to be integrated on a chip [3]. We target our modules to be integrated into a novel adaptive system on a chip (aSOC) architecture which allows diverse computing modules to be implemented in a tiled structure with a statically scheduled interconnect fabric. [5] (Figure 1). Some tiles are general-purpose like RISC, DSP, RAM and FPGA, but this paper focuses on more application-specific blocks which are needed to achieve high performance levels for Motion Estimation, DCT and Viterbi decoding. These blocks have typically been implemented in hardware in previous ASICs hence a large literature of efficient architectures and optimizations exists [11, 8]. However these blocks were typically designed for worst-case performance and traded off flexibility for performance and efficiency. Instead, we use dynamic reconfiguration for both general-purpose and special-purpose tiles where the specific configuration mechanism depends on the local architecture of the tile.

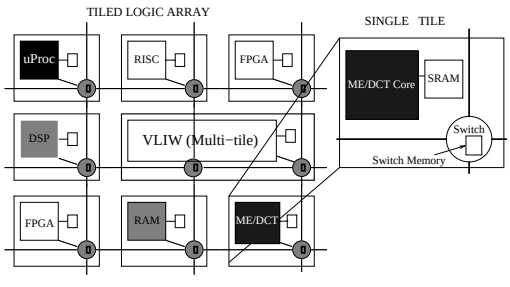

Fig. 1. Tiled Architecture for aSOC

3. CONTENT VARIATION

Video content and its associated processing are highly non-uniform in both space and time. Figure 2 [6] shows the distributions of the horizontal component of the motion vector over 100 frames of two different video sequences. It is clear that motion vectors are highly variable but still correlated in both space and time. The 'table tennis' sequence shows how the motion vector distribution differs in shape due to the changing picture content (content variation in time). Similarly, Figure 3 shows non-uniform content variation in space.

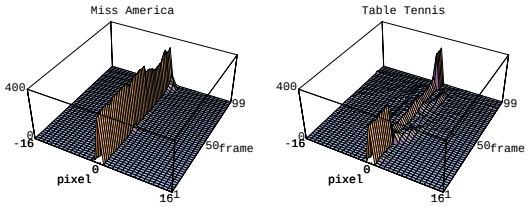

Fig. 2. Motion Vector Distribution over time in 'Miss America' and 'table tennis' video sequences.

Fig. 3. Content Variation in Space. Left - Full search, Right - Logarithmic search

4. DYNAMICALLY PARAMETERIZED ALGORITHMS AND ARCHITECTURES

Parameters provide a simple and formal way to characterize incremental changes to a computation and its computing mechanism. Examples of functional parameters include filter and transform lengths, search spaces, wordlengths, iteration number and quantization levels. Architectural parameters do not modify the output bits of the computing mechanism but allow tradeoffs in area, performance, power and reliability. Functional and architectural parameters can be bound at varying stages in the design of a video computation (Figure 4).

The basic idea behind dynamic parameter adjustment is shown in Figure 5. Functional parameters and architectural parameters are dynamically adjusted to tune the actual processing to the content variation and changing user/system requirements. A controller takes as inputs
1) system requirements and constraints (power, performance, etc.),
2) signal statistics from the input signals, and
3) algorithm statistics from post processing of the input signals (e.g. motion vectors being fed back).

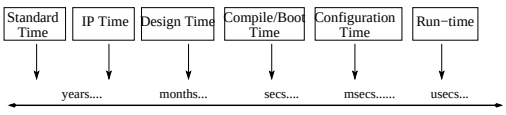

Fig. 4. The Spectrum of Parameter Binding Times.

Functional parameter adjustment changes the quality of the system output. However processing can gracefully degrade in constrained environments by exploiting system and end-user tolerance. Figure 6 [6] shows the variation of compression with the search space size for the 'flower garden' sequence. A larger search area increases the probability of getting a good match during comparison, thus improving compression. Note that there is a point of diminishing return.

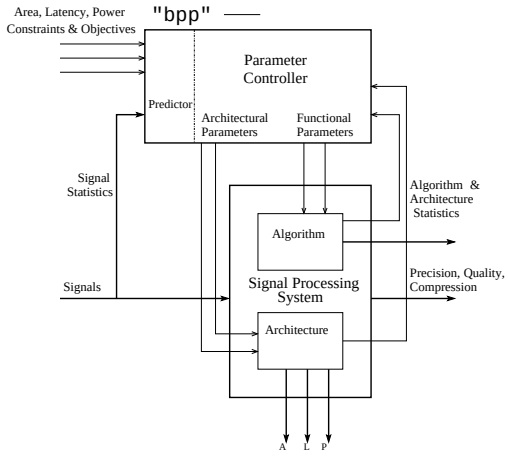

Fig. 5. Dynamic Parameter Adjustment

Figure 3 shows an example of dynamic parameter adjustment which chooses between two different algorithms. The Full-search algorithm results in a compression ratio of 70:1. The Logarithmic search algorithm compromises the compression ratio to 50:1 but uses 14 times less computation. This reduction can be translated directly into further power savings by allowing a reduced clock rate and power supply. The Logarithmic search samples the search area and performs computation on fewer blocks.

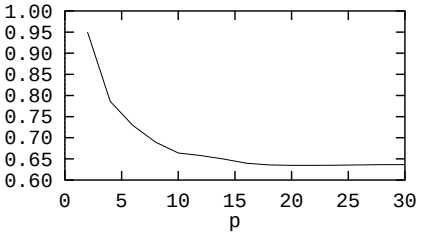

Fig. 6. Bits per pixel vs p (search window size) in 'flower garden' video sequence.

5. DOMAIN-SPECIFIC MODULES

We are currently implementing five modules using dynamic reconfiguration for low-power. Standard low-power optimizations are performed in each implementation. The implementations are all targeted to occupy one of the tiles in the Adaptive System on a Chip (aSOC) (Section 2). In this paper we focus on the characteristics of each computation which allow power/quality tradeoffs through rapid reconfiguration.

5.1. Motion Estimation

Motion Estimation compresses the temporal redundancies between consecutive frames. The most common algorithm is the Full search algorithm [8] which compares a macroblock (16x16 pixels) from the current frame to all the candidate blocks in a search window in the previous frame and the best match is used to estimate the motion. Figure 7 [6] shows Power Consumed v/s Search Area Size. The power due to both computations and I/O (memory accesses) is normalized since, depending on the memory architecture, the power of memory accesses can vary over several orders of magnitude.

In Figure 3 the background shows no variation over consecutive frames. The ball, hand and the paddle (with some camera zoom) are in motion and would do well in an exhaustive search. But, it would be wasteful to make a full search for the matching candidate block in the background, since it is likely that the first candidate block compared will be close enough. In this case a Spiral search gives very low latency values and conserves power. The Spiral search algorithm [8] involves dynamic adjustment of the search space size, with the search moving spirally around the vector predictor location until a threshold is passed. The vector predictor location and the threshold are set by the predictor in Figure 5 (Section 4).

Three different algorithms (chosen by a functional parameter) are implemented in our module architecture: Full search, Spiral search and the 3-Step search [8]. The predictor takes compression objectives, signal statistics and motion vectors (feedback) as inputs and decides on the algorithm to be implemented (Section 4). Details of the prediction algorithm are beyond the scope of this paper.

Pel subsampling [8] is another functional parameter which is adjusted to reduce computations during a block comparison. Pels within a block are correlated with each other and we can use a sub-set of the total number of pels to provide a good approximation for matching. This gives reduced computations and memory accesses at the cost of a small matching quality degradation. The algorithm can also be configured to use Half-Pel motion estimation, where the previous data block are filtered to half-pel [8] resolution by bilinear interpolation. Finally, the matching criteria computation, Sum of Absolute Differences, can also be configured to use shorter wordlengths.

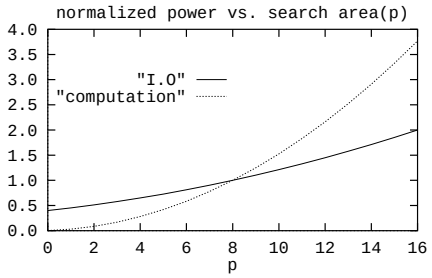

Fig. 7. Power vs Search Area

Since Motion Estimation is a highly parallel application it is implemented on a 16x16 pipelined array of processors. An array controller reconfigures the array elements to use either of the three algorithms, Pel Subsampling, Half-pel and match value wordlengths. This reconfiguration can occur at frame rate to track temporal variation, or even faster, to track intra-frame variations in motion.

A memory adjacent to the PE array is partitioned into blocks which each store complete macroblocks of current and previous frame data, All other blocks except the one being accessed are switched off by having their bitlines and wordlines disabled. This results in a substantial power savings of both dynamic and static power.

5.2. Discrete Cosine Transform

The Discrete Cosine Transform, particularly the 2D DCT [9], is an integral part of many image and video compression systems and is typically implemented as two 1D DCTs. The most power efficient DCT design we know of to date is [10] which uses a variety of mechanisms to exploit content variation to save power, however none of these involve dynamic reconfiguration.

We augment this architecture with two dynamically reconfigurable capabilities: 1) dynamic threshold and 2) dynamic parallelism. The threshold requires careful tracking of the DCT data, but is implemented quite simply as a comparator value. In contrast, the dynamic parallelism is used to tradeoff increased area for higher performance and involves large amounts of reconfiguration data at relatively slow rates. This can be useful to achieve an increased data rate, or can be coupled with voltage scaling to achieve low-power.

Several mechanisms are used to save power based on content variation. **Most Significant Bit Rejection** exploits the property that in spatially correlated images the pixel sums are likely to have a number of equal most significant bits, thus avoiding redundant Read-accumulate computations (RAC). **Row-Column Classification** seeks to reduce the overall arithmetic activity by imposing an upper bound on the number of clock cycles required for the distributed arithmetic operation. Every row or column input to the 1D DCT is assigned to a class based on user-specified thresholds. This categorization leads to different upper-bounds on the number of clock cycles for which the RAC will be computed and hence incurs an imprecision but results in reduced signal activity. The user-specified thresholds can be set dynamically based on dynamic power requirements and power/quality trade-offs. **Replication of RAC Units** allows the DCT unit to exploit further parallelism by replicating the ROM and accumulator units. The dot product of 8-bit vectors X and Y takes eight cycles to compute with one RAC unit. However, two RAC units can finish the computation of the dot product in only four clock cycles. This dynamic parallelism could also be exploited by scaling the supply voltage to achieve a fixed performance, leading to quadratically reduced power consumption. The techniques can be combined to provide a wide range of power/performance tradeoffs, with varying levels of configuration overhead. The $Synopsys\ Design\ Compiler^{TM}$ is being used to synthesize a low-power gate-level netlist from a parameterized RTL description of the DCT unit.

5.3. Lempel-Ziv Compression

Lempel-Ziv compression is a lossless compression technique used in a wide variety of data communication and storage applications and also represents a large class of computations which rely on variable length matching sequences (e.g. Bio-sequence matching, Data mining). Average file compression is 2:1, but for highly redundant data files, much higher levels can be attained. However, the power savings due to compression may be offset by the power consumed performing the string matching computation. Thus, the parameters for the LZ algorithm can be set depending on the statistics of the data as well as the tradeoffs between compression ratio and power/resources required to perform the compression. In some cases, LZ should be replaced with a completely different algorithm which is better suited to the statistics of the data.

LZ compression has fine-grain parallelism which has been exploited in a variety of recent systolic array and CAM implementations [7]. The LZ algorithm has three main parameters that can be dynamically configured: 1) the data type, for example ASCII characters, 2) the longest matching length, and 3) the length of the dictionary or sliding window. The configuration task is similar to that for motion estimation except that the search occurs in only one dimension and that the search candidate is of variable length. However the statistics of the longest matching length can easily be tracked and used to modify the parameter in the compression hardware. The size of the dictionary can also be modified dynamically by tracking the LZ pointers to determine how frequently remote sections of the dictionary result in matches. We are currently developing a configurable LZ prototype for a radar telemetry application using the Annapolis Wildfire development system.

5.4. 3D Graphics Light Rendering

Real-time 3D graphics will be a major power consumer in future portable embedded systems. Fortunately, we can exploit content variation and human perception to significantly reduce the power consumption of many aspects of 3D graphics rendering. In [12] we study the impact on power consumption of novel adaptive versions of the Gouraud and Phong shading algorithms which consider both the graphics content (e.g. motion, scene change) and the perception of the user. Novel dynamically configurable architectures are proposed to efficiently implement the adaptive algorithms in power-aware systems with gracefully degradable quality.

In [12] we introduce an integrated algorithm and hardware solution based on human visual perceptual characteristics and dynamically reconfigurable hardware. Three approaches are explored which are based on human vision and loosely analogous to video coding techniques. The first approach is called distributed computation over frames and exploits the after image phenomenon of the human visual system. The second approach exploits visual sensitivity to motion. According to the speed and distance from camera to object, either the Gouraud or Phong shading algorithm is selected. The third approach is an adaptive computation of the specular term computation used in Phong. Using the same selection criteria as in adaptive shading, a reduced computational cost algorithm is used for fast moving objects. Results based on simulation indicate a power savings of up to 85% using short but realistic rendering sequences.

Figure 8 shows how the power consumption between Phong and Gouraud shading can vary by a factor of 20 for large triangles. If the graphics system can use the lower quality shading algorithm (Gouraud) in situations where human perception will allow it, significant power can be saved. We are currently developing dynamically configurable shading hardware to eventually be used in a complete low-power 3D graphics system.

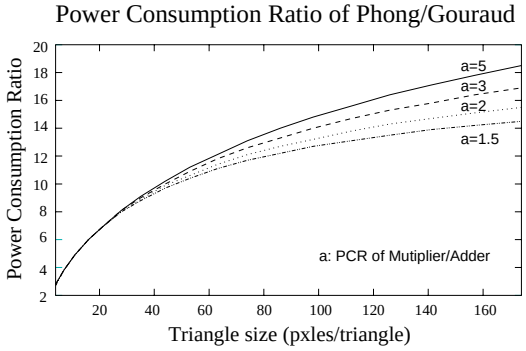

Fig. 8. Power consumption ratio of Phong shading and Gouraud shading: one triangle shading

5.5. Adaptive Viterbi Decoding

Convolutional codes which allow for efficient soft-decision decoding are widely employed in wireless communication systems. As convolutional codes become more powerful, the complexity of the corresponding decoders generally increases. The Viterbi algorithm (VA) [13, 14], which is the most extensively employed decoding algorithm for convolutional codes, works well for codes with short constraint length K. However, its memory requirement and number of computations poses an obstacle when decoding more powerful codes with large constraint lengths. In order to overcome this problem, the adaptive Viterbi algorithm (AVA), wherein the average number of computations per decoded information bit is reduced, has been developed [16, 15]. We look at AVA for reducing power consumption.

In the adaptive Viterbi algorithm (AVA), the number of survivor paths retained at every trellis stage varies according to the current path costs themselves - a path is kept if its current cost is less than $d_m + T$, where d_m is the current cost of the best path, and T is a parameter [16, 15]. Thus, unlike the standard Viterbi algorithm, which always retains 2^{K-1} paths, the number of paths varies over time as shown in Figure 9. Because the variability in the number of paths can be significant, it is convenient to also set a maximum number of surviving paths to be $N_{\max}$. For a given constraint length, the average number of paths with current cost less than $d_m + T$ is generally *much* less than 2^{K-1}, implying a significant computational savings *on average* on a serial processor over the standard Viterbi algorithm. However, as can be seen from Figure 9, there is a significant variation in the instantaneous number of paths which can complicate parallelization to achieve real-time performance.

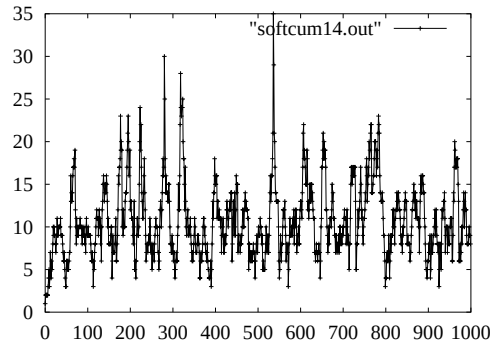

Fig. 9. Number of surviving paths versus decoded symbol number for the adaptive Viterbi algorithm for a $K = 14$,$r = \frac{1}{2}$ convolutional code with $N_{\max}$ set to 2^{K-1} (its maximum).

The threshold condition to determine whether a path is retained depends on the parameter T. If a small value of T is selected, it will result in an increased bit error rate (BER), but the *average* computational complexity will be reduced. On the other hand, if a large value of T is selected, the *average* number of survivor paths increases and will result in a smaller BER. Thus, the optimum value of T has to be selected so that the BER is within allowable limits while matching the size and reconfiguration capabilities of the hardware. Per above, $N_{\max}$ denotes the maximum number of survivor paths to be retained at any trellis stage. Whereas the choice of T selects an operating point on the *average* computational complexity versus BER tradeoff curve, the choice of $N_{\max}$ selects a point on the *maximum* computational complexity versus BER tradeoff curve in a similar manner.

The variation in instantaneous computations required makes the AVA well-suited to exploit the parallelism and dynamic reconfigurability of FPGAs. Since FPGA resources can be allocated to either memory or logic, configuration contents depend heavily on preset values of T and N_{max}. If additional survivor paths are required, the amount of path storage needed increases reducing available area for logic. As a result, fewer parallel logic functions can be created, potentially limiting decoder performance. In contrast, if too few survivor paths are retained, the quality of the decoded signal may be adversely affected. The variation in instantaneous computations required makes the AVA well-suited to exploit the dynamic reconfigurability of the FPGA. In particular, rapid dynamic reconfigurability should allow us to realize the promise of the significant reduction in average number of computations (and thus power) versus the standard Viterbi algorithm. However, it is clear from Figure 9 that reconfiguration must happen at a rapid rate. Methods of reconfiguration currently being considered are altering the amount of memory versus computation units and amortizing complexity over time.

6. CONCLUSIONS

This paper has described preliminary algorithmic and architectural aspects of a larger project in low-power multimedia. The most significant contribution is probably the outline of a methodology for design and experimentation in the general area of dynamic parameterization as a mechanism for adapting computing systems to varying computational loads. Remaining work to be done on this project includes:

1) Implementation at the C, HDL, netlist and physical levels,

2) Power estimation of the modules and the overall architectures including configuration mechanisms,

3) Techniques for statistically tracking content and channel variation,

4) Full prototyping based on actual workloads using a logic emulator from IKOS systems.

This work has been partially supported by NSF 9988238. Extended paper can be found at www.ecs.umass.edu/ece/vspgroup/icassp01