ADDRESS ACCELERATION MECHANISMS FOR AN ADAPTIVE CELLULAR TELEPHONY (ACT) COPROCESSOR

Ali Ibrahim, Al Davis

School of Computing, University of Utah, Salt Lake City, UT

ABSTRACT

The computational complexity of cellular telephone standards has increased faster than Moore's law. New architectural approaches will be needed in order to meet the performance needs while staying within acceptable energy budgets. The ACT coprocessor improves on the energy-delay characteristics of embedded systems by 2 to 3 orders of magnitude, and is within 1 to 2 orders of magnitude of an ASIC approach while retaining much of the generality of a general purpose processor. This paper summarizes the ACT architecture, details have been published elsewhere, and then presents the details of new architectural enhancements that have proven particularly effective in improving performance. The enhancements are a mode addressed register file (MARF) and separate address generation units (AGU's) for each SRAM port and a hardware loop unit (HLU). For the 9 DSP and telephony codes used to evaluate this architecture, 61% of the load on an Intel Xscale's execution pipeline can be removed by using the MARF, HLU, and AGU mechanisms.

1. INTRODUCTION

Moore's law scaling of current embedded processors will be insufficient to keep up with the energy and performance needs of rapidly evolving wireless communication applications [5]. Designers often achieve improvements in energy or delay at the expense of the other. The energy-delay product has been proposed [3] as a better metric for architectural merit. ASICs are typically employed in support of compute intensive kernels since their energy-delay characteristics are far superior to software implementations on energy efficient embedded processors. ASICs are expensive, incur lengthy design cycles, and need to be redesigned if the algorithms change or evolve. These 3 aspects are poor matches for the rapidly evolving cellular telephony arena. This motivated the design of ACT [7], which exhibits energy-delay characteristics that are 2 to 3 orders of magnitude better than the Intel XScale while retaining much of the generality of the software based approach. ACT is one to two orders of magnitude worse than an ASIC, but the generality and flexibility have a significant cost benefit.

Telephony and DSP algorithms are highly iterative, stream oriented, and exhibit relatively high levels of data and instruction level parallelism (DLP and ILP) which embedded architectures have been able to fully exploit. They exhibit high memory and register pressure and performance is often limited by address calculations. Increasing memory and register bandwidth by adding additional ports results in a large increase in power dissipation and still does not solve the address calculation problem. The ACT architecture exploits two additional sources of parallelism: loop control parallelism (LCP) and address generation parallelism (AGP). This paper presents architectural mechanisms which enhance AGP ald LCP in the ACT coprocessor. Previous research efforts [8, 9] have focused on less general strategies than those introduced here.

2. ARCHITECTURE

The ACT coprocessor architecture [6] is shown in Figure 1. A host processor loads frames of data into the input memory and pulls output frames from the output memory. The two scratch memories are local storage resources. The input memory is dual-ported and used in a double buffered fashion. Memories are 16-bits wide and have 1024 entries, except for the input memory which has 2048 entries. ACT handles the computationally intense real-time tasks on the four clusters of execution units. The multiple individually addressed SRAMs provides adequate bandwidth to avoid: XU starvation, cache control energy, and delay uncertainty. Caches are not particularly effective for stream based applications. ACT provides more precise program controlled data movement. These choices also increase the need to exploit AGP.

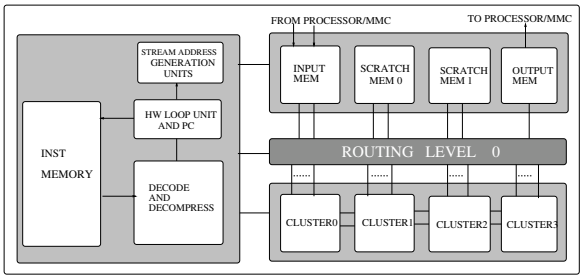

Fig. 1. ACT Coprocessor Organization

Each cluster in Figure 2, consists of three integer execution units (XUs), and two 32-entry register files, each with 1 read and 1 write port. Routing level 1 consists of a layer of 5:1 muxes with registered outputs and allows every XU in the cluster to communicate with routing level 0 and with the other 3 clusters. Routing level 0 also consists of a layer of 6:1 muxes and allows each cluster to access any of the memories. The 2-level mux strategy provides sufficient bandwidth for the execution resources and allows ACT to operate at 300 MHz in a .25 micron TSMC process.

The ACT processor is effectively a fine-grained VLIW architecture and the program consists of horizontal microcode instructions which control register load and output enables, mux select lines, XU opcodes, and a variety of internal control registers that manage the address modes, register file access, loop control, and compression strategy. The wide instruction memory is a potential energy consumption problem. However, due to the regularity of

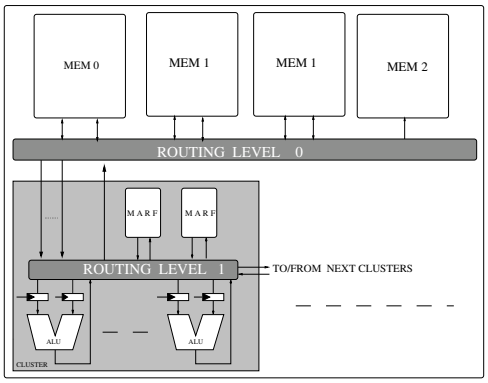

Fig. 2. Cluster Organization.

the instruction patterns, there is a significant opportunity for compression. These duties are handled by the decode and decompress unit.

3. AGP AND LCP MECHANISMS

3G telephony and DSP applications are dominated by regular inner loops which process streams of frame organized input signal data. Performing address calculations in separate units increases significantly the performance.

3.1. The hardware loop unit

The hardware loop unit (HLU) in Figure 3, controls up to four nested loops and avoids the normal loop overheads of loop index modification and conditional branching. Modulo scheduling [11] is supported using a modulo counter that counts from 0 to the initiation interval of the loop. Loop nests of size greater than 4 are rare in 3G baseband algorithms. Larger loop nests can't be fully accelerated since they will require compute support from the execution clusters. The HLU contains a dynamic table to track index variables, a static table that contains program constants, and a small function unit (FU) that supports the necessary arithmetic duties.

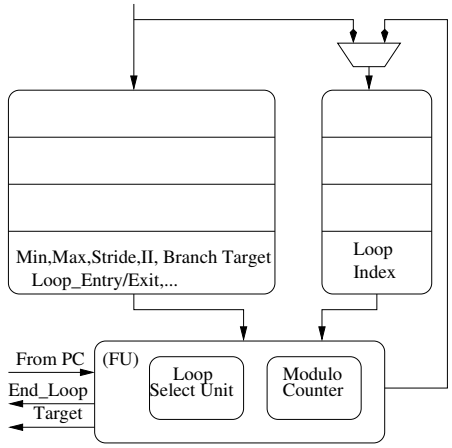

Fig. 3. Hardware loop unit.

3.2. Stream AGUs

The ACT AGUs are designed to be *semi-configurable* and support address generation for two types of addresses that we call *simple* (1D and unit stride 2D patterns) and *complex* (non-unit strided 2D and modulo patterns). Most 3G mobile baseband algorithms require simple addresses, but telephony standards are evolving to support image and video data which require complex addresses.

Each AGU, shown in Figure 4, has three integer units (ALU) connected via a mux. Each ALU receives inputs from the hardware loop unit, execution cluster, instruction memory, configuration table, or from the output of another ALU in the AGU. Each AGU also includes four general purpose addresses registers(for e.g, to store the base address, not shown in the figure) that can be accessed in both modes. Figure 5 illustrates two dimensional data accesses. Data is divided in sub-blocks (circle). The number of required strides to access these data is four: two for the data in the sub-blocks and two to access different sub-blocks. Switching sub-blocks is done via a base address. Hence, an access requires a base address, row size, column size, row stride, and column stride. Assuming that the number of elements in each row and column in each sub-block is power of 2, the address of each element in this sub-block will be calculated as: $(i << const1) + (j << const2) + base$.

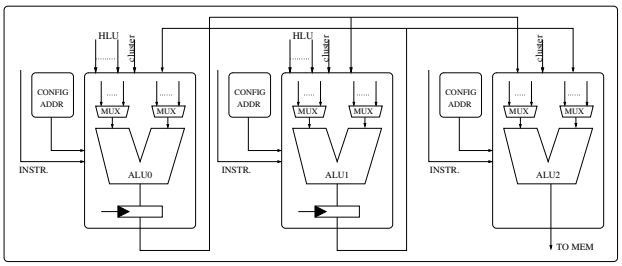

Fig. 4. Stream address generation unit (AGU)

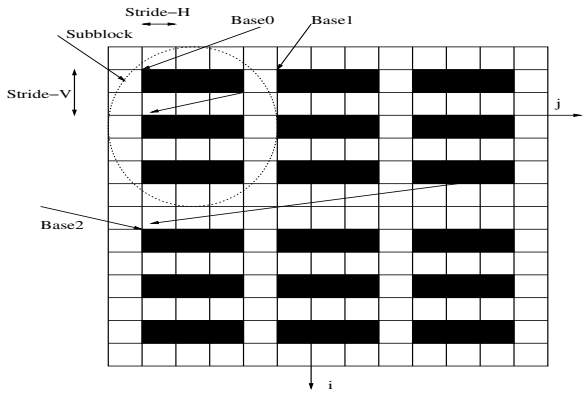

Fig. 5. 2D Stream access format processed in one sub-block at a time.

Opcodes, mux selects, constants, register load enables in the simple mode are part of the instruction. For complex addresses, the extra control bits are stored in a configuration table. Each AGU has a four entry configuration table. This supports an arbitrary number of simple address calculations and up to four complex patterns.

3.3. Data Reuse and Register Files

Data reuse opportunities can be seen by examining IIR/FIR filters, and MPEG2 motion estimation. FIR filtering is used in many DSP applications and performs the following operation:

$y(n) = \Sigma_{i=0}^{N-1} h(i)x(n-i)$

Where h is the coefficient array, x and y are the input and output sequences. Each loop iterates over all N coefficients and N-1 elements of array x are reused. The IIR filter used in VoIp, for instance, has a loop carried dependency. The filter is used to process multiple channels, each with different coefficients. For MPEG2 motion estimation encoding, the best matching block is searched by comparing it with a number of candidates in the reference frame. The new candidate block is formed by shifting by a few pixels from the current candidate. Most of the current and next block therefore overlap.

Exploiting data reuse reduces memory pressure. Previous approaches employ caches, loop transformations, and some hardware support to improve cache cache performance. Recently, software based mechanisms that exploit the data flow information available to the compiler have also been used in the elite processor [10]. The ACT approach is similar to elite but at a finer granularity. Register files are hot spots in modern processors since they are both the source and sink for most operations. In Imagine [12], it has been shown that the power for a centralized register file scales quadratically with the number of ports and linearly with the number of registers.

The ACT architecture employs two 32-entry register files in each of the 4 clusters. Each register file is split into two adjacent windows shown in Figure 6. The size of the windows are explicitly allocated by setting a tail pointer for the first window. The second window is addressed normally as a static register file, while the first window is accessed like a rotating register file using simple address modes supported by the register file address generation unit (AGURF). Each AGURF holds two pointers and performs modulo increment/decrement for circular addressing. If the data in a register file needs to be accessed with a different stride and base pointer, a new pointer can be loaded from the instruction or from the static portion of the register file. This organization allows significant register file specialization for different computation phases, provides hardware support for rotating registers. Since the registers are software controlled, values used in different computations can be kept and reused as long as necessary.

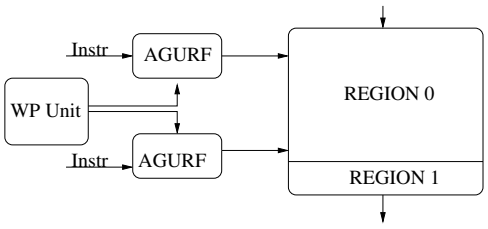

Fig. 6. MARF.

4. RESULTS

ACT power and performance measurements were obtained using Synopsys Nanosim on a fully synthesized and back-annotated .25μm Verilog- and Module Compiler based implementation. A full clock tree and worst case RC wire loads are included in the model. All algorithms were hand scheduled and mapped to the architecture. Conflicts on register file write ports or in any functional units were resolved by storing the data in the post-mux registers. This effectively creates a distributed set of single entry registers files. The circuit is then simulated in Nanosim for the duration of several input packets. The RMS current is used to calculate energy consumption. For the general purpose processor measurements, a low power 400MHz .18μm Intel XScale (StrongARM) PXA250 system has been used. Power numbers have been normalized to an .18μm process. These values were scaled by the feature-size λ using the method described by Gonzalez and Horowitz [3] at a conservative value of 2.

The power numbers for the ASIC implementations for FIR, TFIR, SAD, Dot Product, Vector Product and Square, SAD, matrix multiplication and strided matrix multiplication were obtained using .25μm Verilog implementations using Nanosim. The Rake implementation was taken from the literature [4]. Figure 7 illustrates the energy-delay product of the coprocessor and XScale, normalized to the ASIC.

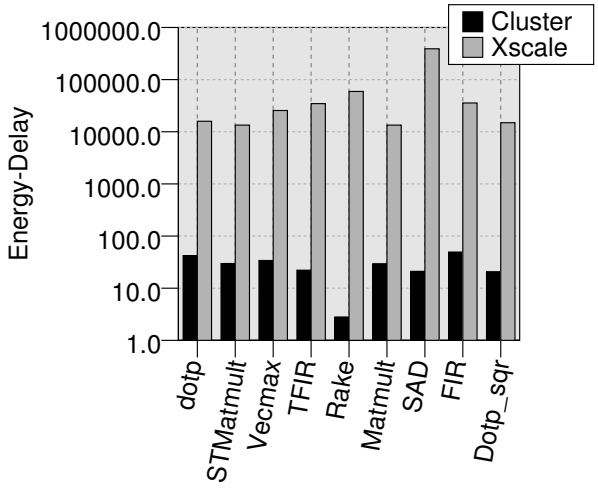

Fig. 7. Energy-delay with respect to ASIC (no compression)

Figure 7 shows that the ACT energy-delay product varies between one and two orders of magnitude when compared to the XScale. The data rate (shown in Figure 8) sustained by the coprocessor, on the other hand, varies. When compared to the TFIR ASIC, the coprocessor is able to achieve almost 1/4 the performance of the ASIC (Note: the ASIC has 4 times as many multipliers). In the FIR case, the coprocessor is capable of sustaining 1/6 the performance of the ASIC. The FIR implementation requires data transfers between the register files in different clusters. Such a transfer requires the use of an XU, since the register files are not directly connected The effective data transfer rate will decrease for filters with a higher number of taps. The excess performance of the coprocessor for Rake provides the opportunity to work on another task or it can be powered down. For SAD, the coprocessor and the ASIC have almost the same data rate since the ASIC was designed to perform four SAD operations per cycle.

The energy dissipation due to instruction memory ranges between 29% and 45% of the total dissipated power. In [6] we have developed decompression techniques that can reduce this power by up to 50%. For the Xscale, 44% of the application suite instructions are load/store, 39% are the real work, and 17% are addi-

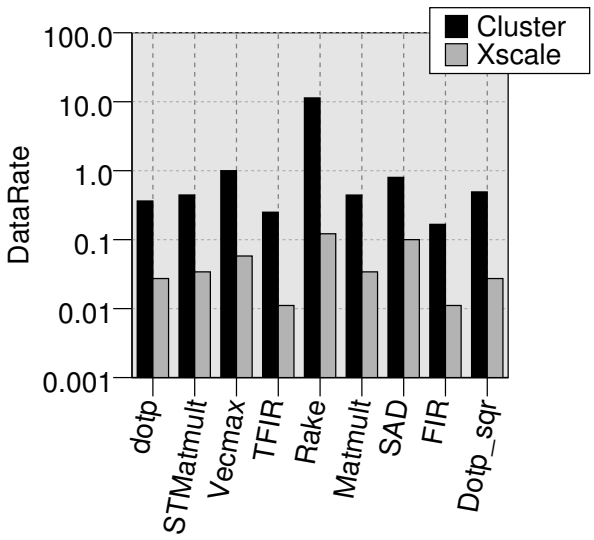

Fig. 8. Data Rate with respect to ASIC

tional address calculation instructions. A similar address calculation load was reported in [13]. However the load/store instructions also contain simple address calculations, and hence these instructions must first go through the Xscale execution pipeline before returning the effective address to the load/store pipeline. Using the MARF, HLU, and AGU units in ACT would reduce the load on the Xscale execution pipeline by 61%. For ACT the MARF and AGU mechanisms increase parallelism by exploiting AGP.

5. CONCLUSIONS & RELATED WORK

Applications in the embedded domain require architectures that are flexible, high-performance, and consume minimal power. High performance and low power can be achieved by the use of ASICs but not flexibility. General purpose processors are flexible but have high power consumption and do not offer the required performance level. DSP processors improved their DSP performance by including application specific hardware support for algorithms that are computationally intensive such the turbo decoder and the rake receiver [1]. Other solution use FPGAs and ASICs as coprocessors for DSP processors [2]. While this approach provides more options for mapping the application onto the architecture, FPGAs have much lower performance and consume much higher power than an ASIC approach. Designing an embedded processor or a coprocessor that offers high performance, flexibility and low power relies on the opportunity for customization for a particular domain. ACT takes advantage of this opportunity and the benefit is enhanced by parallel address generation capabilites of the MARF and AGU mechanisms explored in this paper.

VLIW architectures [1] have been adopted for signal processing applications for their power and performance advantages. The register file size, number of ports, and limitations in the communication interconnect and memory bandwidth have a major effect on performance and power dissipation. Memory bandwidth has a big effect on performance. In [12], a bandwidth efficient stream processor was presented, however the design was targeted for high performance rather than low power. In [10], a vector pointer unit has been used to for efficient register file accesses similar to the MARF approach.

In this paper, a high performance, low power, and flexible coprocessor architecture has been presented. The AGU and MARF units exploit AGP to improve performance, allowing 61% of the Xscale workload to be processed in parallel.

6. REFERENCES

[1] Texas Instruments. http://www.ti.com.

[2] H. Blume, H. Hubert, H. T. Feldkamper, and T. Noll. Model-based exploration of the design space for heterogeneous systems on chip. In *IEEE International Conference on Application-specific Systems, Architectures and Processors*, pages 29–40, July 2002.

[3] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose processors. *IEEE Journal of Solid State Circuits*, pages 1277–1284, Sept 1996.

[4] L. Harju, M. Kuulusa, and J. Nurmi. A flexible Rake Receiver Architecture for WCDMA mobile terminals. *IEEE Workshop on Signal Processing Systems*, pages 177–182, Oct 2002.

[5] J. Hausner. Integrated circuits for next generation wireless systems. *European Solid-State Circuits Conference*, pages 26–29, 2001.

[6] A. Ibrahim, A. Davis, and M. Parker. ACT: A low power VLIW cluster coprocessor for DSP applications. *Submitted to EuroMicro Conference on Real-Time Systems (ECRTS05).*, 2005.

[7] A. Ibrahim, M. Parker, and A. Davis. Energy efficient cluster coprocessors. *International Conference on Acoustics, Speech, and Signal Processing*, May 2004.

[8] B. K. Mattew. The perception processor. *PhD. Thesis*, 2004.

[9] M. Miranda, F. Catthoor, M. Janssen, and H. D. Man. ADOPT: Efficient hardware address generation in distributed memory architectures. *Proc. IEEE 9th International Symposium on System Synthesis*, pages 20–25, November 1996.

[10] D. Naishlos, M. Biberstein, S. Ben-David, and A. Zaks. Vectorizing for a SIMdD DSP architecture. *International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pages 2–11, 2003.

[11] B. R. Rau. Iterative modulo scheduling: an algorithm for software pipelining loops. *Proceedings of the 27th annual international symposium on Microarchitecture*, pages 63–74, 1994.

[12] S. Rixner, W. J. Dally, U. J. Kapasi, B. Khailany, A. Lopez-Lagunas, P. R. Mattson, and J. D. Owens. A bandwidth-efficient architecture for media processing. In *International Symposium on Microarchitecture*, pages 3–13, 1998.

[13] S. Udayanarayanan and C. Chakrabarti. Address code generation for digital signal processors. *38th Conference on Design Automation (DAC'01)*, pages 353–358, 2001.