An Ultra Low Power SIMD Processor for Wireless Devices

Mark Woh[1], Sangwon Seo[1], Chaitali Chakrabarti[2], Scott Mahlke[1] and Trevor Mudge[1]
[1]Advanced Computer Architecture Laboratory
University of Michigan, Ann Arbor, MI
{mwoh,swseo,mahlke,tnm}@umich.edu
[2]School of Electrical, Computer and Energy Engineering
Arizona State University, Tempe, AZ
chaitali@asu.edu

ABSTRACT

This paper presents an ultra low power programmable processor architecture for wireless devices that support 4G wireless communications and video decoding. To derive such an architecture, first we analyzed the kernel algorithms that constitute these applications. The characteristics of these algorithms helped define the wide-SIMD architecture, where the SIMD width can be configured at run time to the specifics of the algorithm being executed. For ultra low power operation, we advocate operating the processor at near threshold voltage. While a combination of near-threshold circuit techniques and parallel SIMD computations achieves excellent energy efficiency, near-threshold operations suffer from large delay variations due to increased process variability. The paper explores low overhead architectural techniques to tolerate and mitigate problems due to delay variations. The techniques include replication of SIMD functional units to replace faulty ones and use of an XRAM crossbar to efficiently set up the new error-free SIMD datapath.

1. INTRODUCTION

In the coming years, wireless devices will support high-bandwidth internet access, human-centric interfaces with voice recognition, high-definition video processing, and interactive conferencing. These devices are likely to be mobile, making throughput/watt the most critical design constraint.

Fourth generation wireless technology (4G) has been proposed to increase the bandwidth to maximum data rates of 100 Mbps for high mobility situations and 1 Gbps for stationary and low mobility scenarios like internet hot spots. This translates to an increase in the computational requirements of 10-1000x over previous third generation wireless technologies (3G) with a power envelope that can only increase by 2-5x [8]. Other forms of signal processing, such as high-definition video, are also 10-100x more compute intensive than current mobile video.

Figure 1 presents the demands of the 3G and 4G protocols in terms of the peak processing throughput and power budget. Conventional processors cannot meet the power-throughput requirements of these protocols. 3G protocols, such as W-CDMA, require approximately 100 Mops/mW. SODA [5] improved upon existing solutions such as VIRAM [2] and Imagine [1] and was able to meet both the power and throughput requirements for 3G wireless. For 4G wireless protocols, the computation efficiency must be increased to greater than 1000 Mops/mW. Mobile computing platforms will also need to perform high-definition video. Figure 1 also shows the performance requirements of video which exceed that of 3G wireless, but are not as high as 4G wireless. However the data access complexity in video is much higher than wireless.

To address the challenge os next generation mobile computing platforms that support 4G and video coding, we proposed AnySP[9]. The key feature of the architecture is that it has a configurable single-instruction multiple-data (SIMD) datapath. AnySP also attacks the traditional inefficiencies of SIMD computation: register file power, data shuffling, and reduction operators. In 90nm technology, its power consumption is 1.3W for 100Mbps 4G wireless.

For ultra low power applications, voltage scaling into the subthreshold regime (V_{dd} (supply voltage) $< V_{th}$ (threshold voltage)) has been shown to be very effective. However, the energy efficiency of subthreshold designs comes at the expense of significant performance degradation. Recent work by Zhai et al. [10] shows that in the near-threshold regime ($V_{dd} \sim V_{th}$), delay improves by 50-100x compared to operating in the subthreshold region with only a 2x increase in energy. However compared to operating in the super-threshold region, the delay is 10x larger. In applications with high degree of parallelism, such as in many kernel algorithms used in 4G and video decoding, a wide-SIMD architecture can compensate for the delay while maintaining energy efficiency.

Near-threshold designs also suffer from process variations and the variation-induced timing errors become much more critical in wide SIMD architectures. This is because in wide systems, the probability that all SIMD lanes are error-free decreases when variations are severe. We investigate the effect of process variations in wide-SIMD architectures, such as AnySP, operating at near-threshold voltages. We show that replication of SIMD lanes, where the replicated lanes serve as spares to replace the faulty ones, is a cost effective way to tolerate and mitigate problems due to timing variation.

2. SIGNAL PROCESSING ALGORITHMS

2.1 4G Wireless Protocol

The major components of the 4G wireless physical layer consists of three blocks: a modulator/demodulator, a MIMO

Algorithm	SIMD Workload (%)	Scalar Workload (%)	Overhead Workload (%)	SIMD Width (Elements)	Amount of TLP
FFT/IFFT	75	5	20	1024	Low
STBC	81	5	14	4	High
LDPC	49	18	33	96	Low
Deblocking Filter	72	13	15	8	Medium
Intra-Prediction	85	5	10	16	Medium
Motion Compensation	75	5	10	8	High

Table 1: Data level parallelism analysis for different signal processing algorithms.

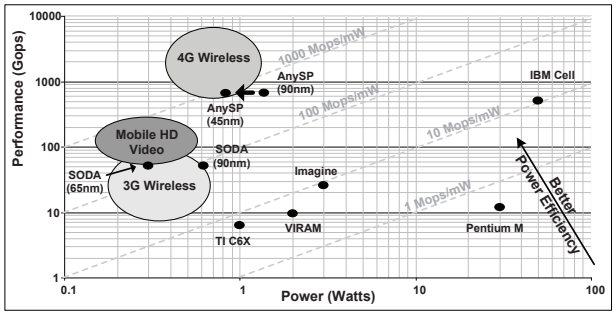

Figure 1: Performance verses power requirements for various mobile computing applications.

encoder/decoder, and a channel encoder/decode [8]. The modulator maps data sequences into symbols with certain amplitudes and phases, onto multiple orthogonal frequencies. This is done using inverse FFT. The demodulator performs the operations in reverse order to reconstruct the original data sequences. The MIMO encoder multiplexes many data signals over multiple antennae. The MIMO decoder receives all the signals from the antennae and either decodes all the streams for increased data rates or combines all the signals in order to increase the signal strength. The algorithm used to increase data rate is the vertical Bell Laboratories layered space-time (V-BLAST), and the algorithm used to increase the signal quality is the space time block coding (STBC). Finally, the channel encoder and decoder perform forward error correction (FEC) that enables receivers to correct errors in the data sequence without retransmission. Of the FEC algorithms, LDPC is widely used for high data rate applications. Our target for 4G wireless is the maximum data rate for high mobility, which is 100Mbps. This 4G configuration utilizes the FFT, STBC, and LDPC kernels.

2.2 H.264 Video Standard

H.264 is selected as the multimedia benchmark because it achieves better compression compared to previous standards and also contains most of the basic functional blocks (prediction, transform, quantization, and entropy decoding) of previous standards. We focused on the *Baseline* profile due to its potential application in videotelephony and videoconferencing.

The H.264 decoder receives a compressed bitstream from the network abstract layer (NAL). The first block is the entropy decoder which is used to decode the bitstream. After reordering the stream, the quantized coefficients are scaled and their inverse transform is taken to generate the residual block data. Using header information in the NAL, the decoder selects prediction values for motion compensation either from a previously decoded frame or from the filtered current frame (intra-prediction). According to the power profile of H.264, about 75% of the decoder power consumption is attributed to three algorithms: deblocking filter (34%), motion compensation (29%), and intra-prediction (10%) [4]. Therefore,we focus on these three H.264 kernel algorithms.

2.2.1 Algorithm Analysis

Table 1 presents our study analyzing the data level parallelism (DLP) of 4G and H.264 decoding algorithms. We calculate the available DLP within each of the algorithms and show the maximum natural vector width. The instructions are broken down into 3 categories: SIMD, overhead, and scalar. The SIMD workload consists of all the raw SIMD computations that use traditional arithmetic and logical functional units. The overhead workload consists of all the instructions that assist SIMD computations, for example loads, stores and shuffle operations. The scalar workload consists of all the instructions that are not parallelizable and must be run on a scalar unit or on an address generation unit (AGU).

From Table 1, we see that many of the algorithms have different natural vector widths–4, 8, 16, 96, 1024. Also, the algorithms with smaller SIMD widths exhibit a high level of TLP, which means that we can process multiple threads that work on separate data on a wide SIMD machine. For instance, 8 instances of STBC that have SIMD width of 4 can be processed on a 32-wide machine. Unlike most SIMD architectures that are designed with a fixed SIMD width to process all the algorithms, this study suggests that the best solution would be to support multiple SIMD widths and to exploit the available thread-level parallelism (TLP) when the SIMD width is small. By supporting multiple SIMD widths, the SIMD lane utilization can be maximized.

Though the scalar and overhead workloads are not the majority, they still contribute 20-30% of the total computation. For instance, in LDPC, data shuffling and memory operations dominate the majority of the workload. This suggests that we cannot simply improve the SIMD performance, but also must reduce the overhead workload. This can be accomplished by introducing better support for data reorganization or by increasing the scalar and AGU performance.

Analysis of the algorithms provided us with four key insights that need to be exploited in order to achieve an efficient high-performance architecture these applications: (i) The vector width varies widely across the algorithms from 4 to 1024; (ii) Algorithms with small vector width frequently

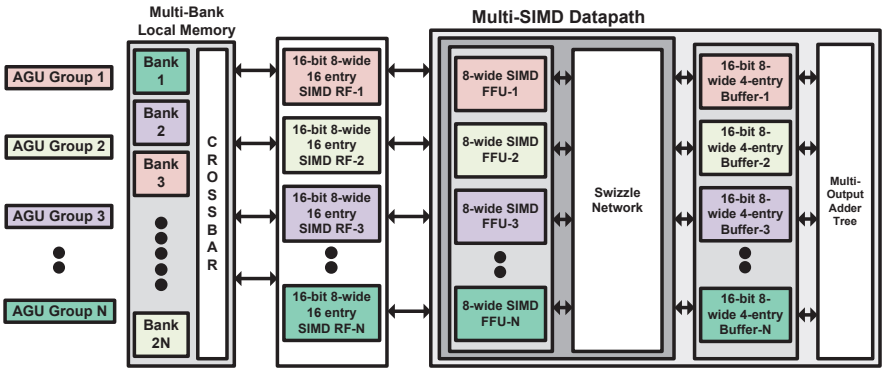

Figure 2: Block diagram of AnySP architecture

contain high TLP; (iii) A large percentage of register values are short-lived and do not need to be written to the register file; (iv) A small set of instruction pairs is used a large percentage of the time; (v) Each algorithm uses a small set of predetermined swizzle patterns.

3. ANYSP ARCHITECTURE

3.1 Overview

In [9] we presented AnySP, a wide SIMD low power architecture that was optimized for 4G and video decoding algorithms. Figure 2 shows the AnySP architecture, which consists of SIMD and scalar data paths. The SIMD data path consists of multiple groups of 8-wide SIMD units, which can configure to create SIMD widths from 8 to 128. Each of the 8-wide SIMD units are composed of groups of Flexible Functional Units (FFUs). The datapath supports three execution scenarios: wide vector computation (multiple groups combined), multiple independent narrow vector computation threads (8 threads x 8 lanes), and 2-deep subgraphs on moderate wide vector computation (32 lanes x depth 2 computations). This inherent flexibility allows the datapath to be customized to the application, but still retain high execution efficiency. Multiple SIMD register files feed the SIMD data path. Each register file is 8 wide and has 16 entries. The swizzle network aligns data for the FFUs. It can support a fixed number of swizzle patterns of 8-, 16-, 32-, 64-, and 128-wide elements. Finally, a multiple output adder tree can sum groups of 4, 8, 16, 32, or 64 elements and store the results in a temporary buffer.

The local memory consists of $2N$ memory banks, where N is the number of SIMD groups; each bank is an 8-wide SIMD containing 256 16-bit entries. Each group of 8-wide SIMD units has a dedicated AGU. When not in use, the AGU can run sequential code to assist the dedicated scalar pipeline. The AGU and scalar unit share the same memory space as the SIMD data path. To accomplish this, we use a scalar memory buffer that can store 8-wide SIMD locations. Because many algorithms access data sequentially, the buffer acts as a small cache that helps avoid multiple accesses to the vector banks.

The main hardware components of the AnySP processor were implemented as RTL Verilog and synthesized in TSMC 90 nm using Synopsys physical compiler. The timing and power numbers were extracted from the synthesis and used by our in-house architecture emulator tool to calculate the timing and power values for each of the kernels. AnySP was able to meet the throughput requirement of 100 Mbps 4G wireless while consuming 1.3 W at 90 nm. It can also achieve high quality H.264 4CIF video at 30 fps with 60 mW at 90 nm, meeting the requirements for mobile HD video.

3.2 Key features

Configurable multi-SIMD width support

The kernel algorithms in 4G and H.264 have different SIMD widths and mapping them all onto a fixed-width SIMD architecture is inefficient. In addition, most of these algorithms are able to exploit TLP for the same task because each task is independent of the others, runs the exact same code, and follows almost the same control path. To support the different types of kernel algorithms, we designed a multi-SIMD-width architecture. Each group of 8-wide SIMD units has its own AGU to access different data. We can also combine the 8-wide groups to create SIMD widths of 16, 32, or 64. Such a feature lets us exploit the DLP and TLP together for large and small SIMD width algorithms.

Swizzle network

Since the kernel algorithms use a small set of swizzle patterns, we use an SRAM-based swizzle network that adds flexibility while maintaining a customized crossbar's performance [6]. The swizzle patterns are stored in the SRAM-based swizzle network configuration memory at initialization. This network has lower power and also provides more functionality than permutation networks found in typical SIMD architectures. For instance, a 128x128 SRAM-based swizzle network consumes less than 30 percent of the power consumed by an equivalent mux-based crossbar.

Temporary buffer and bypass network

We implemented temporary register buffers and a bypass network to lower register file power. We implemented the temporary register buffers as a partitioned register file where the main register file contains 16 registers and additional second partition contains four registers. This small partitioned register file shields the main register file from accesses by storing values that have very short lifetimes. The bypass network

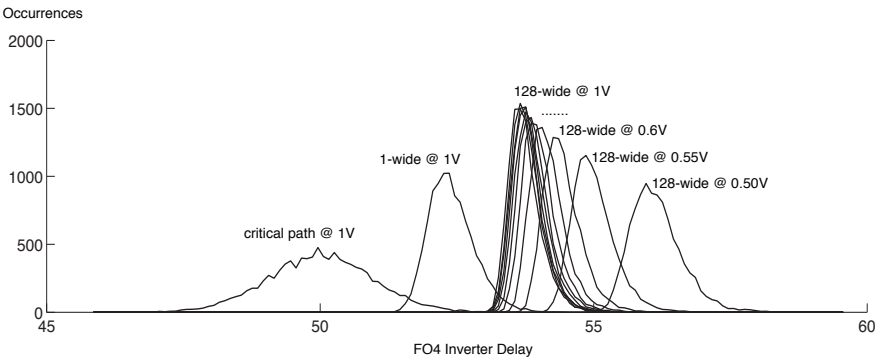

Figure 3: Delay distributions of a critical path (50-long FO4 inverter chain) for 90nm GP model at V_{dd}=1V, 1-wide SIMD system at V_{dd}=1V, and 128-wide SIMD system at different supply voltages from 0.50V to 1V.

modifies the write-back stage and forwarding logic. The programmer explicitly manages the forwarding and writing to the register file. This eliminates register file writes for values that are immediately consumed, reducing register file power.

Multiple output adder tree support

In many video decoding algorithms, we need sums of less than the SIMD width. So the AnySP architecture has an adder tree to allow for partial summations of 4, 8, 16, 32, or 64 elements, which are written back to the temporary buffer unit. The summed values have short lifetimes, so writing into the temporary buffer unit helps reduce the number of read and write accesses to the register file.

4. NEAR-THRESHOLD OPERATION

For ultra-low power applications, subthreshold design is a compelling approach. However, the large performance loss in subthreshold operations makes it unsuitable for mobile computing platforms. By using slightly higher V_{dd}, near-threshold operation significantly increases the performance by 50-100x compared to subthreshold with 2x increase in energy.

While near-threshold operation reduces the energy consumption by 10x compared to super-threshold operation, it has a performance degradation of 10x [10]. In cases where the application can be parallelized, such as in many of the kernel algorithms, using more near-threshold processing elements can be used to compensate for the degraded performance. Another drawback of near-threshold designs is that they suffer from delay variations due to increased process variability. This is because the driving current in the near-threshold voltage region is highly sensitive to the variations in threshold voltage. In this section, we examine architecture-level delay variations and ways to mitigate them.

4.1 Architecture-level Delay Variations

To examine the variation effects in a wide SIMD architecture, we emulate a critical path of an AnySP-type SIMD architecture with a 50-long FO4 inverter chain. We assume that the delay distribution of a critical path is Gaussian and that a hundred critical paths exist in one SIMD lane. Furthermore, we assume that the delay of one SIMD lane (1-wide) is determined by the slowest critical path in the lane and that the delay of an N-wide SIMD datapath is determined by the slowest of the N SIMD lanes.

Figure 3 shows the delay distributions of a critical path, 1-wide system operating at 1V and 128-wide systems operating at different supply voltages. The delay distribution of the 1-wide SIMD system is shifted to the right compared to that of one critical path because the delay of 1-wide system is determined by the maximum delay of 100 critical paths. For the same reason, the delay distribution of a 128-wide SIMD system is shifted to the right of the 1-wide SIMD system. Another characteristic is that the delay distributions of 128-wide systems operating at lower supply voltages drift to the right. This shift is due to the fact that the delay distribution of a critical path at near-threshold voltage has a wider spread than the distribution @ 1V. Also note that at lower voltages, the chip distribution has a long tail, which means that the variation-induced timing errors in near-threshold operations more adversely impact wide SIMD architectures than a scalar datapath (1-wide datapath).

4.2 Techniques to Control Effect of Variations

There are two mechanisms to tolerate variation-induced timing errors in a scalar pipeline: 1) flushing the pipeline and re-executing the instruction with relaxed timing, or 2) waiting one more cycle for the pipeline to generate the correct output. However, applying these approaches to wide SIMD architectures is problematic because the power penalty of the flush-rollback process in the SIMD pipeline is much larger than that of a scalar pipeline. Recent work also shows that there is a significant performance drop in SIMD architectures as single-stage-error probabilities increase [3]. To prevent variation-induced timing errors in near-threshold operation, we analyzed the effect of structural duplication for mitigating variation in SIMD architectures.

In structural duplication, redundant structures are added to the processor and designated as spares [7]. When some architectural modules fail in time, the spare structures replace the failed ones. This structural duplication idea can be used to handle slower SIMD lanes that fail to operate within a given clock period. If the faulty SIMD lanes can be identified at test time, spare SIMD lanes can be used to replace them.

We studied a 128-wide machine and analyzed how many

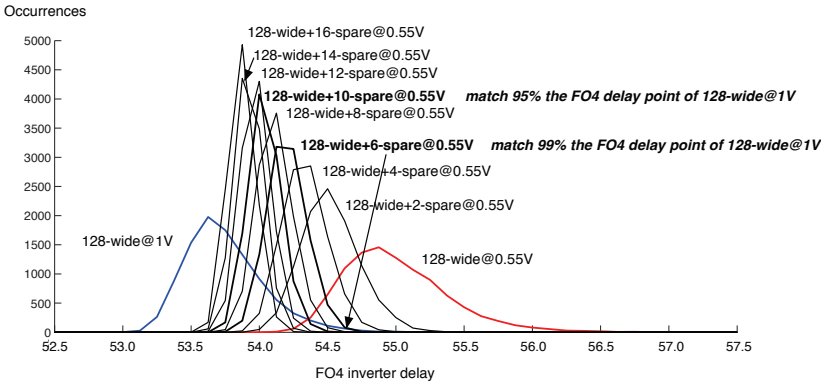

Figure 4: Chip delay distribution for various SIMD width and supply voltage configurations in 90nm GP node. 10000 samples for each curve are simulated.

SIMD lane replications (α spares) are required to tolerate variation-induced timing errors. The delay distribution of a 128-wide machine @ 1V is used as the baseline and the delay distribution of 128-wide+α-spares system @ 0.55V is used to demonstrate the effect of SIMD lane duplication.

Figure 4 shows the chip delay distribution of a 128-wide SIMD architecture with varying number of duplicated spares. For example, the graph of 128-wide+6-spares@0.55V shows the delay distribution of 128 *good* SIMD lanes out of 134 (128+6) SIMD lanes; i.e. six slowest SIMD lanes are dropped. As can be seen, extra SIMD lanes help shift delay distributions to the left and make the spread smaller. We match the 95% and 99% FO4 delay points of the duplicated systems (128-wide+α-spare @ 0.55V) with those of the baseline (128-wide @ 1V). We see that the 95% FO4 delay point of the baseline system distribution matches with that of the 128-wide+10-spare system. Similarly, the 99% FO4 delay point of the baseline system distribution matches with that of 128-wide+6-spare system. The smaller number of additional SIMD lanes required to match the 99% FO4 delay point is because of the heavy tailed distribution of the baseline system.

In order to support structural duplication, there are some mild modifications that have to be done to AnySP. First, an additional XRAM crossbar has to be added to the output of the register file. The sizes of the XRAM crossbars are now increased to 128x134 and 134x128. The corresponding increase in area and power is not substantial. Thus in 90nm technology, use of replication and XRAM crossbars can mitigate the timing variability problems of wide SIMD architectures.

5. CONCLUSION

Future uses for mobile devices will require more connectivity at higher data rates, support of high quality audio and video, as well as interactive applications. This increase in application diversity can be addressed by combining different processor types each tailored to a specific application. Such a solution is costly in terms of time, silicon area, and power. We proposed AnySP, a wide SIMD low power architecture, where the SIMD width can be configured to the algorithm specifics at run-time. For next generation wireless devices with an even more stringent energy budget, we advocate operating the processor at near threshold voltage. Unfortunately, near-threshold operations suffer from large delay variations due to increased process variability. In this work, we analyze variation issues on near-threshold wide SIMD architectures and explore use of structural duplication to minimize the variation impact. Results show that replicating SIMD lanes and using an enlarged XRAM crossbar significantly reduces variability while maintaining high energy efficiency.

6. ACKNOWLEDGEMENTS

This work was supported in part by NSF grants CSR 0910699 and CSR 0910851 and by ARM Ltd.

7. REFERENCES

[1] J. H. Ahn, W. Dally, B. Khailany, U. Kapasi, and A. Das. Evaluating the imagine stream architecture. In *Proc. of the 31st Intl. Symposium on Computer Architecture*, pages 14–24, Jun. 2004.

[2] C. Kozyrakis and C. Patterson. Vector vs. superscalar and VLIW architectures for embedded multimedia benchmarks. In *Proc. of the 35th Intl. Symposium on Microarchitecture*, pages 283–293, Nov. 2002.

[3] E. Krimer, R. Pawlowski, M. Erez, and P. Chiang. Synctium: a near-threshold stream processor for energy-constrained parallel applications. *IEEE Computer Architecture Letters*, 9:21–24, 2010.

[4] T. A. Lin, T. M. Liu, and C. Y. Lee. A low-power H.264/AVC decoder. *International Symposium on VLSI Design, Automation and Test, 2005.*, pages 283–286, April 2005.

[5] Y. Lin et al. SODA: A low-power architecture for software radio. In *Proc. of the 33rd Annual International Symposium on Computer Architecture*, pages 89–101, 2006.

[6] S. Satpathy, Z. Foo, B. Giridhar, D. Sylvester, T. Mudge, and D. Blaauw. A 1.07 tbit/s 128x128 swizzle network for simd processors. In *IEEE Symposium on VLSI Circuits*, 2010.

[7] J. Srinivasan, S. Adve, P. Bose, and J. Rivers. Exploiting structural duplication for lifetime reliability enhancement. In *Proc. of 32nd Intl. Symp. on Computer Architecture, ISCA '05*, pages 520 – 531.

[8] M. Woh et al. The next generation challenge for software defined radio. In *Proc. 7th Intl. Conference on Systems, Architectures, Modelling, and Simulation*, pages 343–354, Jul. 2007.

[9] M. Woh, S. Seo, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner. AnySP: anytime anywhere anyway signal processing. In *Proc. of the 36th Intl. Symp. on Computer Architecture*, ISCA '09, pages 128–139, 2009.

[10] B. Zhai, R. Dreslinski, D. Blaauw, T. Mudge, and D. Sylvester. Energy efficient near-threshold chip multi-processing. In *Proc. of the 2007 Intl. Symp. on Low Power Electronics and Design*, ISLPED '07, pages 32–37, 2007.