From SODA to Scotch: The Evolution of a Wireless Baseband Processor

Mark Woh*, Yuan Lin*, Sangwon Seo*, Scott Mahlke*, Trevor Mudge*, Chaitali Chakrabarti†, Richard Bruce‡, Danny Kershaw‡, Alastair Reid‡, Mladen Wilder‡ and Krisztian Flautner‡
*Advanced Computer Architecture Laboratory, University of Michigan - Ann Arbor, MI
{mwoh, linyz, swseo, mahlke, tnm}@umich.edu
†Department of Electrical Engineering, Arizona State University, Tempe, AZ
chaitali@asu.edu
‡ARM, Ltd., Cambridge, United Kingdom
{richard.bruce, danny.kershaw, alastair.reid, mladen.wilder, krisztian.flautner}@arm.com

Abstract

With the multitude of existing and upcoming wireless standards, it is becoming increasingly difficult for hardware-only baseband processing solutions to adapt to the rapidly changing wireless communication landscape. Software Defined Radio (SDR) promises to deliver a cost effective and flexible solution by implementing a wide variety of wireless protocols in software. In previous work, a fully programmable multi-core architecture, SODA, was proposed that was able to meet the real-time requirements of 3G wireless protocols. SODA consists of one ARM control processor and four wide single instruction multiple data (SIMD) processing elements. Each processing element consists of a scalar and a wide 512-bit 32-lane SIMD datapath. A commercial prototype based on the SODA architecture, Ardbeg (named after a brand of Scotch Whisky), has been developed. In this paper, we present the architectural evolution of going from a research design to a commercial prototype, including the goals, trade-offs, and final design choices.

Ardbeg's redesign process can be grouped into the following three major areas: optimizing the wide SIMD datapath, providing long instruction word (LIW) support for SIMD operations, and adding application-specific hardware accelerators. Because SODA was originally designed with 180nm technology, the wide SIMD datapath is re-optimized in Ardbeg for 90nm technology. This includes re-evaluating the most efficient SIMD width, designing a wider SIMD shuffle network, and implementing faster SIMD arithmetic units. Ardbeg also provides modest LIW support by allowing two SIMD operations to issue in the same cycle. This LIW execution supports SDR algorithms' most common parallel SIMD execution patterns with minimal hardware overhead. A viable commercial SDR solution must be competitive with existing ASIC solutions. Therefore, algorithm-specific hardware is added for performance bottleneck algorithms while still maintaining enough flexibility to support multiple wireless protocols. The combination of these architectural improvements allows Ardbeg to achieve 1.5-7x speedup over SODA across multiple wireless algorithms while consuming less power.

1. Introduction

In recent years, we have seen an increase in the number of wireless protocols that are applicable to different types of communication networks. Traditionally, the physical layer of these wireless protocols is implemented with fixed function ASICs. Software Defined Radio (SDR) promises to deliver a cost effective and flexible solution by implementing a wide variety of wireless protocols in software. Such solutions have many potential advantages: 1) Multiple protocols can be supported simultaneously on the same hardware, allowing users to automatically adapt to the available wireless networks; 2) Lower engineering and verification efforts are required for software solutions over hardware solutions; 3) Higher chip volumes because the same chip can be used for multiple protocols, which lowers the cost; and 4) Better support for future protocol changes. With the tremendous benefits of SDR, it is likely that many mobile communication devices are going to be supported by SDR technologies in the foreseeable future. Recently, Samsung was the first to announce a mobile phone that supports TD-SCDMA/HSDPA/GSM/GPRS/EDGE standards using a SDR baseband processor [1].

Wireless Protocol Workloads. The computational requirements of current generation wireless protocols are orders of magnitude higher than the capabilities of modern general-purpose processors. A wireless protocol processor must sustain this high computation throughput while meeting the strict power budget of an embedded mobile terminal. This is the reason why many wireless protocols to date are implemented with custom hardware. The challenge of SDR is to meet these performance and power requirements while maintaining the flexibility of a programmable processor. Previous work on workload characterization of 3G wireless and other wireless baseband processing protocols showed that there exists large amount of data-level parallelism (DLP), with the majority of the operations being long vector arithmetic computations [2].

SODA Processor Architecture. The SODA multi-core architecture was proposed for supporting 3G wireless baseband processing [3]. SODA consists of an ARM control processor, four data processing elements (PEs), and a shared global scratchpad memory. Designed for long vector arithmetic operations, each SODA PE includes a wide 512-bit SIMD

unit that is capable of operating on 32 16-bit elements concurrently. In addition, each PE also has a scalar datapath, local scratchpad memories, address generation unit (AGU), and direct memory access (DMA) support.

Ardbeg Processor Architecture. A commercial prototype, Ardbeg, based on SODA has been developed by ARM Ltd. Ardbeg shares many features with SODA. It is a multicore architecture, with one control processor and multiple data PEs. Each data PE contains a 512-bit wide SIMD datapath. Ardbeg adds algorithm-specific hardware and optimizes the architecture specifically for wireless applications. In contrast, SODA was designed to test the feasibility of a fully programmable wireless baseband solution and purposely avoided algorithm-specific designs. While SODA was focused on supporting 3G wireless protocols, Ardbeg is also designed to scale for future protocols. Overall, Ardbeg achieves between 1.5-7x speedup over SODA while operating at a lower clock frequency.

The evolution of SODA to Ardbeg was a process with many design choices. The major design choices can be grouped into the following three categories:

- **Optimized Wide SIMD Design.** SODA was originally designed in 180nm technology. In 90nm technology, the SIMD datapath choices need to be re-examined. We re-evaluated the SIMD width and found that SODA's original 32-lane 512-bit SIMD datapath is still the best SIMD design point in 90nm. On the other hand, the SIMD shuffle network redesigned to support faster vector permutation operations. Compared with SODA's two cycle multiplier, 90nm technology enables a single cycle multiplier, which provides significant speedup for several key SDR algorithms.
- **LIW Support for Wide SIMD.** For W-CDMA and 802.11a, the SODA SIMD ALU unit is utilized around 30% of the total execution cycles. LIW execution on the SODA SIMD pipeline was considered a poor choice due to the low utilization of the SIMD units and was abandoned due to the concern about the extra power and area costs of adding more SIMD register file ports. We revisited this concern when designing Ardbeg in order to improve the computational efficiency. The result was Ardbeg issuing two SIMD operations each cycle. Not all combinations of SIMD instructions are allowed. Ardbeg implements a restricted LIW designed to support the most common parallel execution patterns found in SDR algorithms with minimal hardware overhead. Our analysis shows that having this restricted LIW support would provide better performance and power efficiency over single-issue SIMD datapath, but also that having larger issue widths does not provide any additional performance benefit over a simple two-issue LIW.
- **Algorithm Specific Hardware Acceleration.** A set of algorithm specific hardware is also added to the Ardbeg architecture. These include an ASIC accelerator for Turbo decoder, block floating point support, and fused permute and arithmetic operations. This set of algorithm specific hardware was chosen to achieve higher computational efficiency while maintaining enough flexibility to support multiple protocols.

The rest of the paper is organized as follows. Section 2 gives a brief description of the overall architectures of SODA and Ardbeg. Section 3 presents the architectural evolution from SODA to Ardbeg. We provide experimental results and analysis to explain the rationale behind the major Ardbeg architectural design decisions. Section 4 presents the performance results of the two architectures for various wireless protocols. Section 5 provides a survey of the current SDR processor solutions.

2. Architecture Overview

Because the majority of the SDR computations are based on vector arithmetic, previous work on SODA has demonstrated that having a wide SIMD datapath can achieve significant speedup while maintaining low power consumption [3]. With a 32-lane SIMD datapath, SODA was able to achieve an average of 47x speedup for W-CDMA DSP algorithms over a general purpose Alpha processor. However, as an initial research prototype, many architectural optimizations were overlooked. Ardbeg has improved upon the base SODA architecture, as will be illustrated in the subsequent sections. This section provides an overview of the SODA and Ardbeg architectures and summarizes the differences.

2.1. SODA Architectural Overview

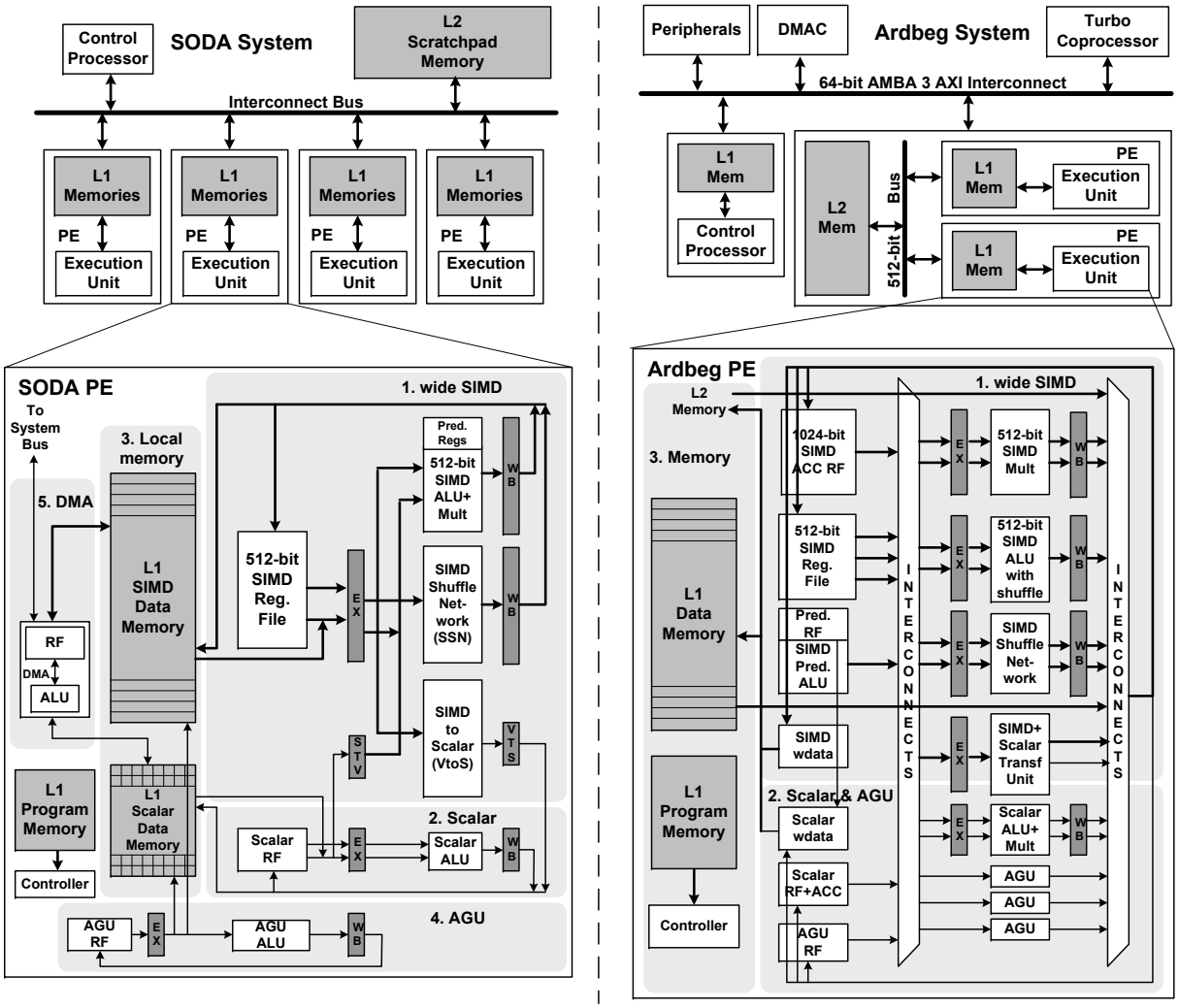
The SODA multicore system is shown on the left in Figure 1. It consists of four data PEs, a scalar control processor, and a global L2 scratchpad memory, all connected through a shared bus. Each SODA PE consists of five major components: 1) an SIMD datapath for supporting vector operations; 2) a scalar datapath for sequential operations; 3) two local L1 scratchpad memories for the SIMD pipeline and the scalar pipeline; 4) an AGU pipeline for providing the addresses for local memory access; and 5) a programmable DMA unit to transfer data between memories. The SIMD, scalar, and AGU datapaths execute in lock-step, controlled with one program counter.

The SIMD datapath consists of a 32-lane, 16-bit datapath, with 32 arithmetic units working in lock-step. It is designed to handle computationally intensive DSP algorithms. Each datapath includes a 2 read-port, 1 write-port 16 entry register file, and one 16-bit ALU with multiplier. Synthesized in 180nm technology, the multiplier takes two execution cycles when running at the targeted 400 MHz. Intra-processor data movements are supported through the SSN (SIMD Shuffle Network). The SSN consists of a shuffle exchange (SE) network, an inverse shuffle exchange (ISE) network, and a feedback path. Various SIMD permutation patterns require multiple iterations of the SSN network. SIMD-to-scalar (VTS) and scalar-to-SIMD (STV) units are used to transfer data between the SIMD and scalar datapath.

2.2. Ardbeg Architecture

The Ardbeg system architecture is shown on the right in Figure 1. Similar to the SODA architecture, it consists of multiple PEs, an ARM general purpose controller, and a global scratchpad memory. The overall architecture of the Ardbeg PE is also very similar to the SODA PE, with a 512-bit SIMD pipeline, scalar and AGU pipelines, and local memory. Ardbeg was designed using the OptimoDE framework [4]. The framework allowed the creation of custom VLIW-style architectures and evaluating many architectural design trade-offs quickly. These trade-offs will be discussed in the next section. The instruction set for Ardbeg was derived from the ARM NEON extensions [5]. The bottom portion of Figure 1 also provides a side-by-side comparison between the two architectures.

The Ardbeg system has two PEs, each running at 350 MHz in 90nm technology. In addition, it includes an accel-



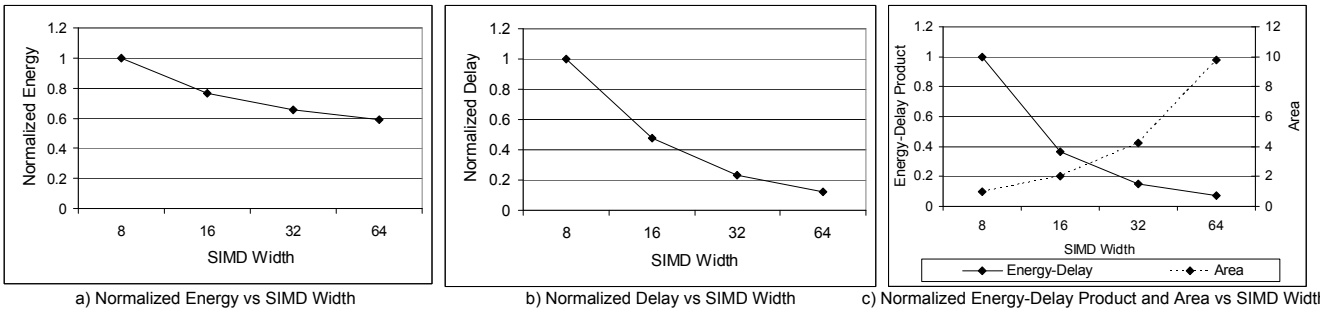
Comparison summary of the architectural features of SODA and Ardbeg

	SODA	Ardbeg
PE Architecture		
Organization	SIMD + scalar + AGU	SIMD + scalar + AGU
Execution Model	SIMD/Scalar LIW	SIMD/Scalar and SIMD/SIMD LIW
PE Frequency	400MHz (180nm)	350MHz (90nm)
SIMD Architecture		
SIMD Datapath	single issue	ALU + memory + SSN
SIMD Width	512 bits	512 bits
Data Precision	16-bit FXP	8/16/32-bit FXP
Block Floating Point	no	yes
SIMD Predication	yes	yes
SIMD Mult Latency	2 cycles	1 cycle
SIMD Shuffle Network	32-lane 1-stage iterative perfect shuffle	128-lane 7-stage Banyan network
Reduction Network	reduction tree	pair-wise operation/reduction tree
SIMD Reg File	2 read/1 write ports, 16 entries	3 read/2 write ports, 15 entries
L1 Memory	8KB	32KB~128KB
L2 Memory	64KB	256KB~1MB
Others		
Coprocessor	no	Turbo coprocessor
Compiler Opti.	no	software pipelining

Figure 1: SODA and Ardbeg architectural diagrams, and a summary of the key architectural features of the two designs.

erator dedicated to Turbo decoding. In comparison, in the SODA system, Turbo decoding is allocated to one of the four PEs. Both the Ardbeg and SODA PEs have three major functional blocks: SIMD, scalar, and AGU.

The SODA and Ardbeg PEs both support 512-bit SIMD operations. The SODA PE only supports 16-bit fixed point operations, whereas the Ardbeg PE also supports 8-, 32-bit fixed point, as well as 16-bit block floating point operations. Support for 8-bit helped lower the power for many of the W-CDMA kernels that only needed 8-bit precision. Legacy wireless protocols like 802.11b have many kernels that operate on 8-bit data and do not require the 16-bit precision



a) Normalized Energy vs SIMD Width b) Normalized Delay vs SIMD Width c) Normalized Energy-Delay Product and Area vs SIMD Width

Figure 2: Plots of normalized energy, delay, and energy-delay product versus area plots for different Ardbeg SIMD width configurations running 3G wireless algorithms. The results are normalized to the 8-wide SIMD design.

that SODA supported. Support for 32-bit was added in order to accommodate future algorithms which may require higher precision.

One of the key differences between Ardbeg and SODA is that the Ardbeg PE supports LIW execution on its SIMD pipeline, allowing different SIMD units to execute in parallel. In the SODA PE, only one SIMD operation can be issued per cycle. Also, SODA's SIMD permutation network is a single stage, multi-cycle perfect shuffle network, whereas Ardbeg's SIMD permutation network is a modification of a 7-stage, single-cycle Banyan network. Detailed analysis of the Banyan network can be found in [6]. The shuffle instructions that are used in Ardbeg are an extended set of the ARM Neon permutation instructions. In terms of the number of registers, the Ardbeg PE has additional SIMD and scalar accumulators to hold the output of the multiplier. Ardbeg has a 1-cycle multiplier, whereas SODA's multiplier requires 2 cycles.

The memory hierarchy in Ardbeg is similar to the Cell processor [7] in that each PE has a local scratchpad memory and PEs share a global memory, which are all explicitly managed. The DMA can transfer data between each of the PE's local memories and also to and from the global memory. A write buffer to memory is also added to Ardbeg. Both Ardbeg's local and global memories are larger than SODA's memories. In addition, instead of the separate scalar and SIMD memories in SODA, Ardbeg has one unified local scratchpad memory. Because many DSP algorithms don't have much scalar code, it is more efficient to share the memory space between the SIMD and scalar datapath.

For system mapping in Ardbeg, the application is represented as a task graph and a set of filters (like StreamIt [8]). The compiler performs coarse-grain software pipelining to assign tasks to PEs and inserts DMA transfers to transfer data between PEs. Streaming dataflow is explicit, so data follows the task assignment and no special data partitioning is required. Oversubscription of the PE's local memory is handled by spilling sections to the global memory. More details about system mapping and scheduling can found in [9].

3. Architectural Evolution: SODA to Ardbeg

3.1. Optimized Wide SIMD Design

Since the majority of the SDR algorithms operate on wide vectors, SODA used a wide SIMD datapath, namely a 512-bit 32-lane SIMD datapath. Ardbeg has also adopted the 512-bit SIMD datapath, and extended it to support 64-lane 8-bit and 16-lane 32-bit SIMD arithmetics. The SIMD shuffle network (SSN) is redesigned to provide better performance at lower power. With a target frequency of 350 MHz, implementing Ardbeg in 90nm also allows for a single-cycle SIMD multiplication unit. The rest of this section explains our rationale for these architectural design decisions. For each of the studies, we synthesized in 90nm the different sizes and configurations of the functional units and calculated the number of cycles and energy to run the kernels.

SIMD Width Analysis. The SODA architecture was designed using a 180nm process technology. A 32-lane configuration was found to be the most energy efficient SIMD configuration. One of the first Ardbeg design considerations is to determine if SODA's proposed 32-lane SIMD is still the best configuration in 90nm. In this study, we examine SIMD configurations ranging from 8-lane to 64-lane. Figures 2a and 2b show the normalized energy and delay for different SIMD width Ardbeg processors synthesized for 350 MHz in 90nm for various key SDR algorithms like FFT, FIR, W-CDMA Searcher, and Viterbi. All values are normalized to the 8-wide SIMD configuration.

The figures show that as SIMD width increases, both delay and energy consumption decreases. The delay result is expected as wider SIMD configurations can perform more arithmetic operations per cycle. While power consumption of a wider SIMD is greater, because wider SIMD takes fewer cycles to perform the same number of arithmetic operations and the control overhead per instruction is amortized across the SIMD, the overall energy consumption is lower for wide SIMD. Figure 2c shows the energy-delay product and the area of these SIMD configurations. A 32-lane SIMD configuration has better energy and performance results compared to the 8-lane and 16-lane SIMD configurations. A 64-lane SIMD configuration has slightly better results than the 32-lane SIMD configuration. If energy and delay are the only determining factors, then implementing Ardbeg with a 64-lane SIMD configuration is probably the best design choice. However, in a commercial product, area is also a major design factor. As SIMD width increases, area increases at a higher rate than the decrease in either energy or delay. Taking area into account, Ardbeg chose to keep SODA's 32-lane SIMD datapath configuration.

SIMD Permutation Support. It is common for DSP algorithms to rearrange vector elements before computation. One of the central challenges in designing a wide SIMD architecture is the vector permutation support. A partially

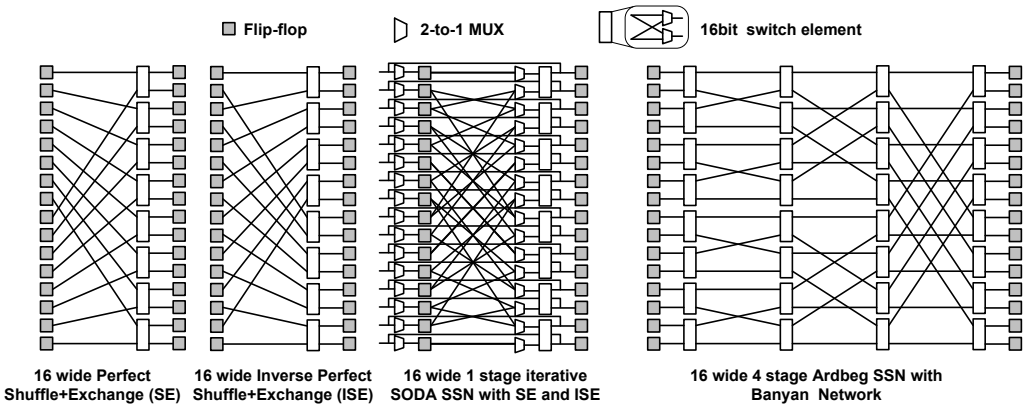


Figure 3: SIMD shuffle network for the SODA PE and the Ardbeg PE. For illustration clarity, these examples show 16-wide shuffle networks. The SODA PE has a 32-wide 16-bit 1-stage iterative shuffle network, and the Ardbeg PE has a 128-lane 8-bit 7-stage Banyan shuffle network.

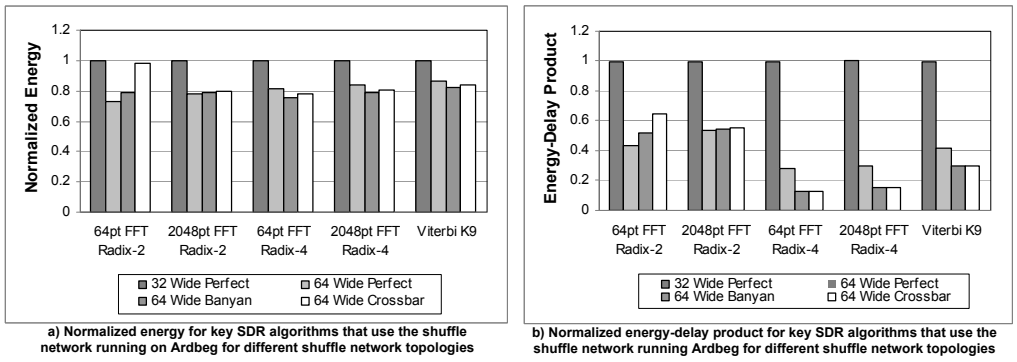


Figure 4: Normalized energy and energy-delay product for key SDR algorithms running on Ardbeg for different shuffle network topologies.

connected SIMD shuffle network (SSN) was employed in SODA as shown in Figure 3. It is a 32-lane single stage iterative shuffle network consisting of a perfect shuffle and exchange (SE) pattern, an inverse perfect shuffle and exchange (ISE), and a feedback path. Multi-stage networks were considered, but in 180nm technology the delay for the multi-stage network was more than one clock cycle running at 400 MHz. In addition, there were concerns that the area for a multi-stage network may be too large. Therefore, a multi-cycle iterative shuffle network was chosen for SODA. In designing Ardbeg's shuffle network in 90nm, several SIMD configurations and network topologies were revisited. We first examined the performance and energy trade-offs of a wider SSN. Figure 4a provides the normalized energy of key SDR algorithms for 32-lane and 64-lane SODA SSNs. The SIMD datapath is still 32-lane for both SSN configurations. The 64-lane SSN operates on two 32-lane SIMD vectors by reading from two SIMD register file ports. Filter algorithms are excluded from this study because their implementations do not use the SSN. Compared to the 32-lane network, a 64-lane network consumes approximately 20% less energy across all benchmarks, despite the fact that the 64-lane network consumes more power than the 32-lane network. This is because these DSP algorithms operate on long vectors, where the vector width is greater than the SIMD width. Because many long vector permutations require extra instructions to store intermediate permutation results, the number of instructions required to perform long vector permutations does not always scale linearly with the width of the SSN. A smaller SSN requires more instructions than a larger SSN, which results in more frequent SIMD register file accesses and other execution overhead.

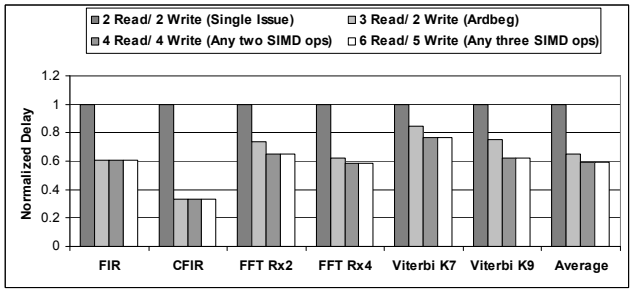
We then examined the performance and energy trade-offs of different network topologies. In addition to the iterative SE/ISE network, we also examined a 64-lane Banyan network and full crossbar. The SE/ISE and the Banyan networks are shown in Figure 3. The Banyan network is a flattened 7-stage network that can perform 64-lane 16-bit vector permutations in a single cycle. Energy and energy-delay products of these three networks are shown in Figure 4. For radix-2 FFT, a 64-lane iterative SE/ISE network is slightly better than a 64-lane Banyan network, because there exists an implementation of this algorithm that is optimized specifically for the SE/ISE network. However, if an algorithm requires more complex permutation patterns, such as the radix-4 FFT and Viterbi algorithms, the single-cycle Banyan network has shorter delays than the multi-cycle iterative shuffle network. Though the difference in energy consumption between the iterative SE/ISE network and 64-lane Banyan is not very large, Figure 4b shows that the single-cycle Banyan network has better energy-delay product than the iterative SE/ISE network. Overall, the Banyan network performs as well as the full crossbar, and with $\sim$17x area savings compared to the crossbar. Therefore, Ardbeg's SSN is implemented with the Banyan network. In addition to supporting 16-bit permutations, Ardbeg's Banyan network can also support 32-lane 32-bit and 128-lane 8-bit vector permutations.

Ardbeg Function Units	# of SIMD RF Ports Required
Memory Load/Store	1 read / 1 write
SIMD Arithmetic	2 read / 1 write
SIMD Multiply	2 read / No write (ACC RF)
SIMD Shuffle	2 read / 2 write
SIMD+Scalar Transfer Unit	1 read / 1 write
ACC-to-SIMD Move	1 read / 2 write
SIMD Comparison	2 read / 1 write (Pred. RF)

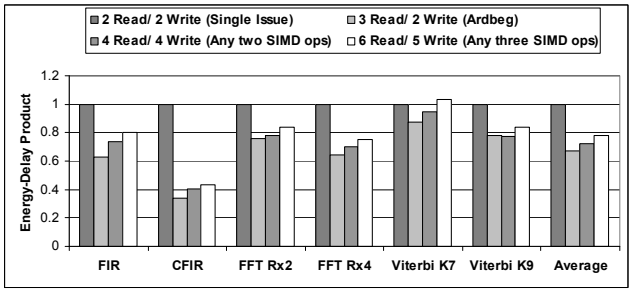
a) This table lists the function units in Ardbeg, and the number of SIMD register file ports required for each unit. At most two SIMD operations can be issued every cycle.

	Mem.	Arith.	Mult.	Shuffle	Trans.	Move	Comp.
Mem.	NA	--	--	--	--	--	--
Arith.	High	NA	--	--	--	--	--
Mult.	High	Mid	NA	--	--	--	--
Shuffle	Low	High	Mid	NA	--	--	--
Trans.	High	Mid	High	Mid	NA	--	--
Move	Low	Low	High	Low	Low	NA	--
Comp.	Low	Low	Low	Low	Low	Low	NA

b) Shaded box means Ardbeg can issue instructions on these two function units in the same cycle. "High/Mid/Low" represent the relative usage frequency for each pair of function units within wireless protocols.



c) Normalized delay for various key SDR kernels running on Ardbeg with different VLIW configurations.



d) Normalized energy-delay product for various key SDR kernels running on Ardbeg with different VLIW configurations

Figure 5: Ardbeg VLIW support. The results are shown for software pipelined Ardbeg assembly code. Ardbeg has 7 different function units, as listed in sub-figure a. These seven function units share 3 SIMD register file read and 2 write ports. At most two SIMD operations can be issued per cycle, and not all combinations of SIMD operations are supported. Different LIW configurations are evaluated in terms of delay and energy-delay product, as shown in sub-figures c and d.

Reduced Latency Functional Units. In SODA, the 180nm process technology put a constraint on the latency of the functional units. Because SODA's target frequency was set to 400 MHz, the multiplier had to be designed with a 2-cycle latency. For Ardbeg, the target frequency is set at 350 MHz due to the control latency for controlling the LIW pipeline. With 90nm process technology, Ardbeg implements power efficient multipliers with single cycle latency. Because many DSP algorithms require a large number of multiplication operations, the single-cycle multiplication results in up to 2x performance improvement (see Section 4).

3.2. LIW SIMD Execution

For W-CDMA and 802.11a, the SODA SIMD ALU unit is utilized around 30% of the total time. The poor utilization is mainly due to the fact that SODA's SIMD datapath is shared with the memory access unit and the SSN. Functional unit under-utilization not only increases register file accesses but also execution time. LIW execution on the SIMD pipeline was considered for the SODA architecture to reduce these problems, but was abandoned due to the concern about the extra power and area costs of adding more SIMD register file ports. In SODA, the SIMD register file was the largest power consumer, accounting for approximately 30% of the total power. When designing Ardbeg, we re-evaluated LIW execution to decrease execution time and to reduce register file power.

To determine the effectiveness of LIW, we analyzed different kernels within the set of wireless protocols and found how often functional units could be used in parallel. There are 7 SIMD function units in Ardbeg's SIMD datapath as listed in Figure 5a, along with their register port requirements. The values listed in Figure 5b represent the frequency that the functional units could execute instructions in parallel. We can see that there are few instruction combinations that occur in high frequency in the algorithms. This suggests that we could implement a LIW and minimize the number of register file ports to save power while increasing throughput.

We have studied the performance and energy efficiency trade-offs for supporting various LIW configurations in Ardbeg. We examined configurations with a different number of SIMD register file read and write ports: single issue with 2 read and 2 write ports, restricted 2-issue LIW support with 3 read and 2 write ports, full 2-issue LIW support with 4 read and 4 write ports, and full 3-issue LIW support with 6 read and 5 write ports. The performance and energy efficiency results of the synthesized implementations are shown in Figures 5c and 5d. The performance is normalized to the cycle count for a single issue Ardbeg. We found that LIW support is beneficial for many key SDR algorithms. This indicates that there is still instruction-level parallelism (ILP) within SIMDized Ardbeg assembly code. However, we also find that a 2-issue LIW configuration is enough to capture the majority of the ILP, as a 2-issue configuration results in a similar speedup as a 3-issue configuration. This is because a significant portion of the parallelism is already exploited through SIMD execution. Also, many SIMD operations cannot execute in parallel simply because of data dependencies.

LIW execution is supported in Ardbeg, but with restrictions on the combinations of instructions that can be issued in a cycle. This results in slower speedup than a full 2-issue LIW, but provides better energy-delay product due to a lesser number of SIMD register file ports. The set of valid Ardbeg LIW instruction combinations are shown in Figure 5b as shaded boxes. Among these LIW combinations, overlapping memory accesses with SIMD computation is the most beneficial because most DSP algorithms are streaming. The SIMD arithmetic/multiplication and SIMD-scalar transfer combination is the most beneficial for filter-based algorithms. And, the SIMD multiply and move combination

is the most beneficial for FFT-based algorithms. The responsibility is left to the compiler to produce valid instruction schedules that can utilize this capability. Overall, Ardbeg's SIMD datapath can achieve an average of 60% SIMD ALU utilization with restricted LIW execution.

3.3. Application Specific Hardware Acceleration

Designing an application specific processor for SDR is a balancing act between programmability and performance. A processor must be flexible enough to support a multitude of wireless protocols. However, too much flexibility results in an inefficient architecture that is unable to meet the stringent performance and power requirements. SODA was designed to meet the throughput requirements of 3G wireless protocols, such as W-CDMA and 802.11a. In addition to these 3G protocols, Ardbeg was designed with future wireless protocols in mind. Hardware accelerators were added in Ardbeg to increase computational and energy efficiency.

3.3.1. Turbo Coprocessor

Turbo decoding is one of the error correction algorithms used in the W-CDMA wireless protocol for the 2 Mbps data communication channel. It is the most computationally intensive algorithm in W-CDMA. In addition, it is the most difficult algorithm to vectorize. Unlike the wide vector arithmetics of other SDR algorithms, Turbo decoder operates on narrow 8-wide vectors. Parallelization techniques can be applied to utilize the 32-lane SIMD datapath by processing four 8-wide vectors concurrently [10]. However, this requires concurrent memory accesses for the 4 vectors. Because the SODA and Ardbeg PEs only have one memory port, serialized memory accesses become the bottleneck of the algorithm. Software pipelining cannot help, because the main loop in the decoder has data dependencies between consecutive loop iterations. The combination of these factors makes Turbo decoder the slowest algorithm on the SODA and Ardbeg PEs. The SODA and Ardbeg PEs can sustain 50-400 Mbps of data throughput for various FIR and FFT algorithms, but only 2 Mbps for Turbo decoder. The SODA PE was targeted at 400 MHz because of the computational requirements of the Turbo decoder. Offloading the Turbo decoder to a coprocessor allows the Ardbeg PE to lower the target frequency to 350 MHz.

Because of the high computational requirements, one SODA PE is dedicated solely for Turbo decoding, accounting for roughly 25% of the total power consumption. In a 90nm implementation, a SODA PE would be able to maintain 2 Mbps while consuming an estimated power of 111mW. In contrast, in 130nm, an ASIC Turbo decoder is able to support 13.44 Mbps while consuming 262 mW [11]. In 90nm technology, this roughly translates to 21 mW for sustaining 2 Mbps throughput. Therefore, in the case of Turbo decoder, the cost of programmability is approximately 5x in terms of power consumption. Furthermore, since 2 Mbps is the maximum throughput for a SODA PE running at 400 MHz, higher decoding throughput, as required by future protocols, would require either higher frequencies or multiple PEs. Both these considerations led Ardbeg to offload Turbo decoding on a coprocessor. Other DSP systems aimed at wireless communications, such as the Phillips' EVP [12], have also taken a similar approach.

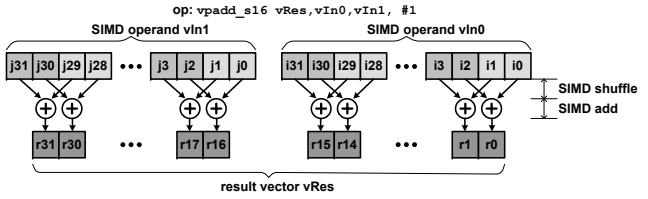


Figure 6: Ardbeg's pair-wise butterfly SIMD operation implemented using a fused permute and ALU operation. The figure shows pairs of a 2-element butterfly operation. Ardbeg supports pairs of 1-,2-,4-,8-,and 16-element butterfly of 8- and 16-bits. This butterfly operation uses the inverse perfect shuffle pattern because the input to each SIMD ALU lane must come from the same SIMD lane.

3.3.2. Application Specific Instruction Set Extensions

Many wireless protocols can share the same error correction ASIC accelerator, but the approach of using more ASIC accelerators is not viable due to the inherent differences in the protocols. However, while the algorithms are different, they share many commonalities within their basic computational blocks. This allows us to increase computational efficiency by adding re-usable algorithm-specific instructions.

Block Floating Point Support. Large point FFTs are used in many wireless protocols. Even though the input and output data are 16-bit numbers, the intermediate results often require higher precision. Block floating point (BFP) provides near floating point precision without its high power and area costs. In floating point, each number has its own mantissa and the exponent. In BFP, each number has its own mantissa, but the exponent is shared between a block of numbers. BFP is commonly used in ASIC design, but very few programmable processors have provided direct hardware support. A key operation in BFP is finding the maximum value among a block of numbers. Most DSP processors support this operation in software. However, for the 32-lane Ardbeg SIMD datapath, this is inefficient, as all lane values must be compared. In Ardbeg, BFP is supported through special hardware that finds the maximum value in a 32-lane 16-bit vector. Each instruction that supports BFP has special flags which, when enabled, automatically perform value tracking and store the result in a special register. BFP support allows the Ardbeg PE to operate in the 16-bit SIMD datapath mode for FFT computations, instead of the 32-bit SIMD datapath mode that would have been required to satisfy precision requirements. Though FFT is where BFP is currently used, any algorithm that requires higher precision can utilize the BFP instruction extensions.

Fused Permute-and-ALU Operations. It is common in DSP algorithms to permute the vectors before performing arithmetic operations. An example is the butterfly operation in FFT, where vectors are first shuffled in a butterfly pattern before vector adds and subtracts are performed. In an earlier design of the SODA PE, the SSN was placed in front of the SIMD ALU, so that permute-and-arithmetic operations could be performed in one instruction. However, arithmetic operations that do not require permutations always go through the SSN, increasing the number of pipeline stages and power consumption. So in the final SODA PE design, the SSN was taken out of the arithmetic pipeline, and placed as a separate unit, as shown in Figure 1. To support the permute-and-

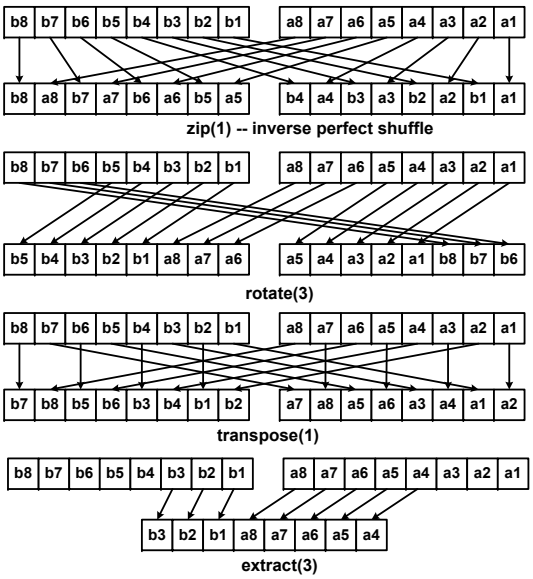


Figure 7: SSN shuffling patterns used for matrix transpose.

arithmetic operations, a separate permutation operation was needed. The result of this permutation operation is written back to the SIMD register file, only to be read out in the next cycle for the arithmetic operation, thereby increasing register file access power in SODA.

The Ardbeg PE addresses this problem by including two shuffle networks. The 128-lane SSN is a separate unit that can support many different permutation patterns. In addition, a smaller 1024-bit 1-stage shuffle network is included in the same pipeline stage in front of the SIMD ALU. This 1-stage shuffle network only supports inverse perfect shuffle patterns between different groups of lanes. This shuffle pattern implements the various pair-wise butterfly operations shown in Figure 6. In the figure, the shuffle and add operations are performed in the same cycle. This shuffle network is used to accelerate FFT and various other algorithms that use butterfly-and-addition operations. Because these fused butterfly operations are the majority of the permute-and-arithmetic patterns, Ardbeg is able to benefit from the best of both designs. A 2048-Point FFT is able to gain 25% speedup using fused butterfly operations.

SIMD Support for Interleaving. Interleavers are common in wireless protocols. They are used to protect the transmission against burst errors by rearranging the data sequence. Unlike most other DSP algorithms, there is no data processing or computations involved in interleaving; interleavers simply rearrange the data sequence in different patterns to account for varying types of transmission environments.

Interleaving is essentially a long vector permutation operation, where the vector width is far greater than the SIMD width. This is a challenge because the SODA and Ardbeg SSNs can only permute vector patterns of SIMD width. If we let N be the size of the vector, then a general purpose permutation algorithm would take $O(N)$ time. However, for certain permutation patterns, different types of SIMD shuffle patterns can be utilized to reduce the permutation latency. As mentioned in Section 3.1, the Ardbeg SSN supports a set of predefined permutation patterns for efficient implementation of certain interleaving patterns. For example, one commonly used pattern is the matrix transpose operation, where the input vector is organized as an $M \times N$ matrix, and the output vector is transposed into an $N \times M$ matrix. A $O(log(N))$ algorithm exists that uses the zip, transpose, extract, and rotate shuffling patterns [13] as shown in Figure 7. Using these predefined patterns, a 192 element vector can be transposed in just 37 cycles. This translates to an average speedup of 4x for interleaving kernels for Ardbeg in comparison to SODA.

4. Results and Analysis

For the overall protocol performance evaluations, we have implemented three different wireless communication protocols that represent a wide spectrum of wireless communication applications. These are W-CDMA [14], 802.11a [15], and DVB-T/H [16][17]. W-CDMA is a widely used 3G cellular protocol. 802.11a is chosen to represent the workload of a typical Wi-Fi wireless protocol. DVB-H (Digital Video Broadcasting - Handheld) is a standard used for digital television broadcasting for handheld receivers and DVB-T (DVB - Terrestial) is used for stationary receivers. Beyond 3G, many of the protocols are OFDM based such as WiMAX. We analyzed DVB-H and 802.11a as representatives of OFDM-based systems. These protocols are chosen to stress the flexibility of the SODA and Ardbeg systems. Both SODA and Ardbeg are able to support real-time computations for these protocols.

The characteristics of these three protocols are listed in Figure 8. These protocols consist of the following four major algorithm categories: filtering, modulation, synchronization, and error correction. Filtering is used to suppress signals transmitted outside of the allowed frequency band so that interference with other frequency bands is minimized. Modulation algorithms translate digital signals into analog wave patterns consisting of orthogonal signals. Synchronization algorithms synchronize the two communicating terminals to ensure lock-step communication between the sender and receiver. Error correction algorithms are used to recover data from noisy communication channels.

The RTL Verilog model of the SODA processor was synthesized in TSMC 180nm technology. The estimated power and area results for 90nm technology were calculated using a quadratic scaling factor based on Predictive Technology Model [18]. The Ardbeg processor was developed as part of the OptimoDE framework [4]. The architectural model was written in OptimoDE's hardware description language. A Verilog RTL model, a cycle-accurate simulator, and a compiler are generated by OptimoDE. The Ardbeg processor was synthesized in TSMC 90nm using Synopsys physical compiler to place and Cadence Encounter to route with clock tree insertion. Ardbeg's PE area is 75% larger than SODA's estimated 90nm PE area. The total system area is comparable between the two systems because SODA contains 4 PEs compared to 2 PEs in Ardbeg. Ardbeg was targeted for 350 MHz, while SODA for 400 MHz.

4.1. Wireless Protocol Results

Evaluation results show that an Ardbeg multicore system synthesized in 90nm technology is able to support 3G wireless processing within the 500 mW power budget of a mobile device [19]. Figure 9 shows the power consumption re-

	W-CDMA	802.11a	DVB-T, DVB-H
Throughput	Voice: 12Kbps Data: 384Kbps/2Mbps	24Mbps, 54Mbps	5Mbps, 15Mbps
Filtering	Complex FIR 65-taps	FIR 33-taps	FIR 16-taps
Modulation	Scrambler/Descrambler Spreader/Despreader Combiner	FFT/IFFT 64 points QAM/IQAM 64 points	FFT 2048 points Scrambler/Descrambler QAM/IQAM 4/16/64 points
Synchronization	Searcher	Interpolator	Equalizer Channel Est.
Error Correction	Interleaver Viterbi K=9 Turbo Decoder K=4	Interleaver Viterbi K=7	Bit Interleaver Viterbi K=7

Figure 8: DSP algorithms that are used in W-CDMA, 802.11a and DVB, DVB-H wireless protocols.

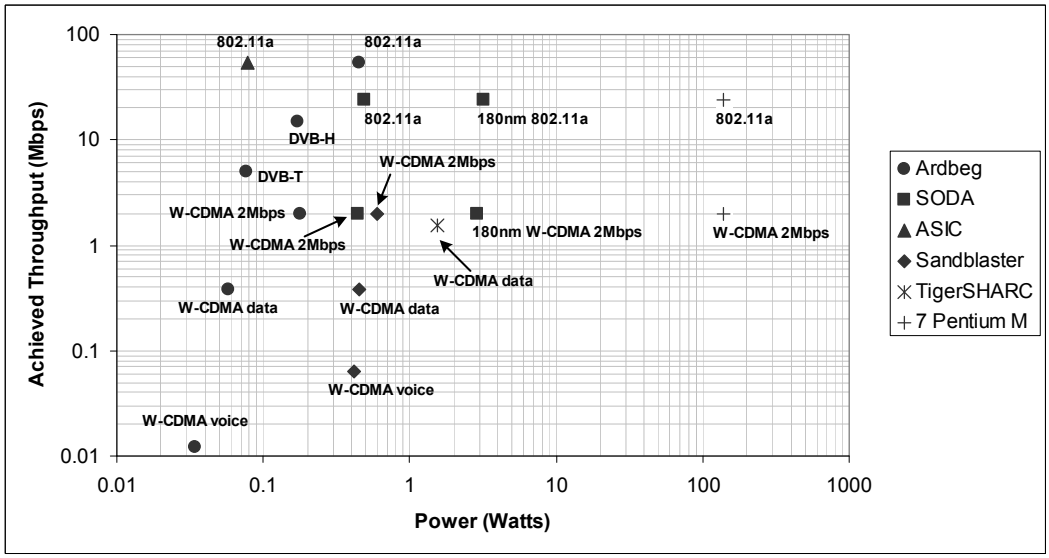


Figure 9: Throughput and power achieved for SODA and Ardbeg for W-CDMA, 802.11a and DVB-T/H. ASIC 802.11a, Pentium M, Sandblaster, and ADI TigerSharc results are also included for comparison purposes. Results are shown for processors implemented in 90nm, unless stated otherwise.

quired to achieve the throughput requirement of W-CDMA, 802.11a, and DVB-T/H. The graph includes the numbers for the SODA and Ardbeg systems, as well as an ASIC implementation for 802.11a [20], Sandbridge's Sandblaster, Analog Devices TigerSHARC, and Pentium M implementations. Data for the other processors was estimated using datasheets and publications. General purpose processors, such as Pentium M, require a power consumption two orders of magnitude greater than the 500 mW power budget. On the other end of the spectrum, an ASIC solution is still 5x more power efficient than any SDR solution. Overall, Ardbeg is more power efficient than SODA for all three wireless protocols. Because Ardbeg is designed to handle high-throughput wireless protocols, its performance for low-throughput W-CDMA voice channels is not as efficient. This is because the available vector parallelism is lower and the processing power of Ardbeg is not fully utilized. In these cases, the scalar datapath in Ardbeg would be utilized more frequently to save power. Both SODA and Ardbeg are very competitive compared to other SDR processors, including Sandbridge's Sandblaster and Analog Devices' TigerSHARC. The major sources of Ardbeg's efficiency are: the restricted LIW execution, application specific instruction set extensions, and larger shuffle network.

4.2. Wireless Algorithm Analysis

In this section, we present a performance analysis of the key DSP algorithms in each of the four algorithm categories. Details of the kernels can be found in [2]. The speedups are consolidated in Figure 10. The speedup analysis is further broken up into the Ardbeg architectural improvements that were highlighted in the Section 3. These improvements include: optimized SIMD ALU, wider single cycle SSN, and LIW execution. The OptimoDE framework used to design Ardbeg generates a compiler that performs optimizations like software pipelining and other compiler optimizations which we also report.

Filtering. Finite Impulse Response (FIR) filters are widely used in wireless communication protocols. Both the SODA and Ardbeg PEs can support the computation requirements of filters for real-time 3G wireless protocol processing. Figure 10 shows the Ardbeg PEs speedup over the SODA PE for various filter configurations. On average, Ardbeg achieved a 3.4x speedup over SODA.

Multiply-and-accumulate (MAC) operations are the central arithmetic operation for filtering. For complex filter arithmetics, multiplications are even more important as every complex multiplication requires four MAC operations. The SODA PE has a two cycle multiplier, whereas the Ardbeg

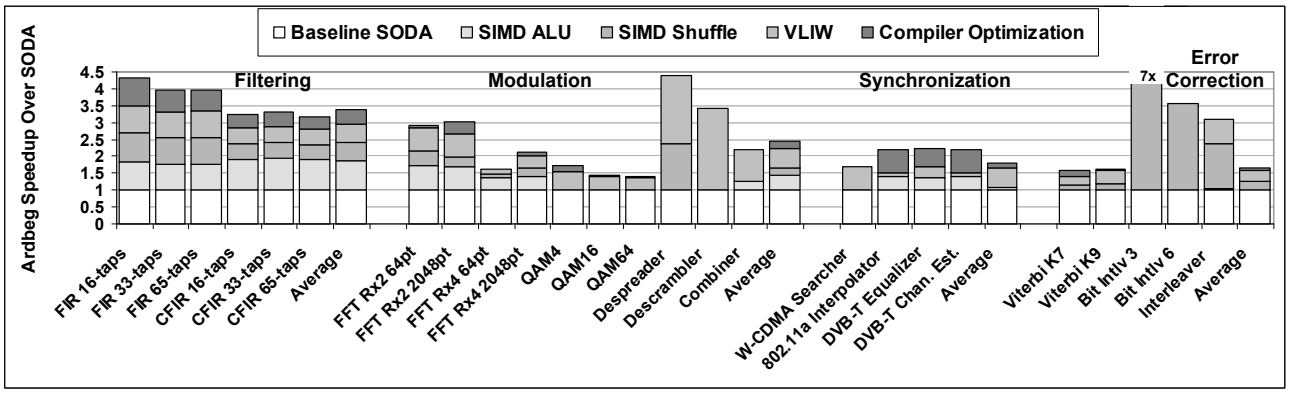


Figure 10: Ardbeg speedup over SODA for the key DSP algorithms used in our wireless protocol benchmarks. The speedup is broken down into the different architectural optimizations. These include optimized SIMD ALU, wider 1-cycle SIMD shuffle network, reduced SIMD memory latencies through LIW execution, and compiler optimizations with software pipelining.

PE has a single cycle multiplier. A significant portion of Ardbeg's speedup is due to the faster multiplier.

In this analysis, both SODA and Ardbeg implement a vectorized version that requires one 64-wide SIMD vector permutation operation for processing each sample point. The SODA PE only has a 32-wide SIMD permutation network, compared to Ardbeg's 64-wide network. The permutation operation takes 3 cycles on SODA, but only one cycle on Ardbeg. Because memory is accessed for each sample, LIW support on the Ardbeg PE is able to hide the multi-cycle memory latencies. Finally, software pipelining and other compiler optimizations help better utilize Ardbeg's LIW datapath.

Modulation. Fast Fourier Transform (FFT) is widely used in OFDM protocols like 802.11a/g and DVB-T. Figure 10 shows the Ardbeg PE speedup over the SODA PE for various FFT configurations. On average, Ardbeg achieves a 2.5x speedup over SODA. Like the filters, there is about a 50% speedup attributed to single cycle multiplies. This speedup is less for a Radix-4 implementation because multiplications are reduced by 25%. Another 50-100% speedup is attributed to the fused operations. The butterfly operation is implemented efficiently by fusing multiplication with add or subtract operations. Another benefit is that Ardbeg allows specialized shuffle operations, followed by ALU operations to be computed in one cycle. Finally, the LIW scheduling provides the remaining speedup. Ardbeg can overlap the memory loads of the next butterfly with the current butterfly's shuffle operation.

Modulation in W-CDMA consists of three kernels: descrambler, despreader, and combiner. The despreader gains significant speedup (almost half) by utilizing Ardbeg's wide shuffle network. The descrambler implementation on Ardbeg is a direct translation of the SODA version. Ardbeg gains, because in every cycle, it can overlap the memory and ALU operations. The combiner, like the despreader and descrambler, benefits from the LIW scheduling as well as the one cycle multiplication. All three kernels benefit greatly from LIW scheduling because each iteration of the inner-loop is small and independent. This allows the overlap of memory loads and stores, shuffle operations, and ALU operations in the same cycle.

Synchronization. Synchronization in W-CDMA is accomplished by the searcher, which achieves almost 1.5x speedup on Ardbeg. The gain in performance is due to Ardbeg's pipelined memories and LIW scheduling. However, these gains are offset by performance loss due to its SIMD predicate support. The number of instructions needed to calculate the predicate values on the Ardbeg PE is 4 cycles, whereas the SODA PE can perform the same task in 2 cycles. This is because SODA's predicate values are stored in the SIMD register file, whereas Ardbeg's predicate values are stored in a dedicated register file. Although Ardbeg's dedicated register file is able to compute different predicate patterns more quickly, it takes longer to load the predicate values into the SIMD datapath. Because all of searcher's predicate patterns can be pre-computed, SODA's faster predicate read latency proves to be more beneficial. This accounts for a 20% performance difference. The major benefit of Ardbeg's LIW scheduling is hiding the memory's multi-cycle access latencies. Because half of every loop iteration can be overlapped, the Ardbeg searcher still results in almost 2X speedup despite its inefficient predication support.

802.11a interpolator, DVB-T equalizer, and DVB-T channel estimation are all similar to the FIR operations, and their speedup rationales are similar to those of the FIR. The only difference is that these algorithms have intra-iteration data dependencies that cannot exploit the LIW datapath. Software pipelining is beneficial by scheduling different loop iterations onto the LIW datapath.

Error Correction. There are two commonly used error correction algorithms in wireless communication – Viterbi and Turbo decoding. As mentioned in the previous section, the Turbo decoder in Ardbeg is offloaded to an accelerator. However, the Viterbi decoder is still implemented by the Ardbeg PE. As shown in Figure 10, Ardbeg's Viterbi implementation has a speedup of only 1.2x to 1.6x compared to SODA. The small speedup is because the Viterbi computation does not have multiplication operations, so the optimized SIMD ALU does not help. In addition, there are data dependencies between consecutive loop iterations, so software pipelining techniques do not help. The majority of the speedup comes from hiding the memory access latency through LIW execution on the SIMD pipeline.

Interleavers are widely used in many wireless protocols. As mentioned in the last section, a few SIMD shuffle

patterns are added to accelerate these algorithms. As shown in Figure 10, the Ardbeg interleaver implementations gain a significant speedup, up to 7x speedup over SODA. The speedup is solely due to the Ardbeg's SSN. Because the majority of the interleaver instructions are SIMD permutation operations, Ardbeg's single cycle 64-wide SSN has a significant advantage over SODA's multi-cycle 32-wide SSN.

5. Wireless Baseband Processor Survey

There has been tremendous industrial interest in SDR, resulting in a wide range of proposed architectural solutions from many leading semiconductor companies. The proposed SDR solutions can be categorized into two different design philosophies – SIMD-based and reconfigurable architectures, as explained in [21]. SIMD-based architectures usually consist of one or few high-performance DSP processors. The processors are usually connected together through a shared bus, and managed through a general purpose control processor. Some SIMD-based architectures also have a shared global memory connected to the bus. Both Ardbeg and SODA fall under the SIMD-based architecture category. Reconfigurable architectures are usually made up of many simpler PEs. Depending on the particular design, these PEs range from the fine-grain ALU units to the coarse-grain ASICs. The PEs are usually connected together through a reconfigurable fabric. The rest of this section will present existing design solutions in these two categories.

SIMD-based SDR Architecture. In addition to Ardbeg and SODA, there are several other SIMD-based SDR architectures. These include Infineon's MuSIC [22], Analog Device's TigerSHARC [23], Icera's DXP [24], Phillips's EVP [12], and Sandbridge's Sandblaster [25]. A comparison between these architectures, SODA, and Ardbeg is shown in Figure 11. These are all embedded systems that consist of 1 to 8 high performance DSP processors. Because data are accessed in a regular pattern, all of the processors use software-managed scratchpad data memories instead of caches to reduce power. Even though most of these processors are designed in 90nm technology, they operate at relatively low frequencies to reduce power. The exception is the Icera DXP, which implements a deeply pipelined high frequency design. Its SIMD ALUs are chained so that a sequence of vector arithmetic operations are performed before the data are written back to the register file. This has the advantage of saving register file access power at the cost of a less flexible SIMD datapath.

Most SIMD-based SDR processors support VLIW execution by allowing concurrent memory and SIMD arithmetic operations. Analog Device's TigerSHARC goes one step further, and provides concurrent SIMD arithmetic operations by having two 4-lane SIMD ALU units that are controlled with two instructions. With 32 lanes, Ardbeg and SODA have the widest SIMD design. Wider SIMD datapaths have higher power efficiency, but also require higher levels of vector parallelism within the software applications. Because the majority of SDR's computation are on wide vector arithmetics, the 32-lane SIMD can be utilized fairly well. In addition, Ardbeg's execution stage is optimized so that any arithmetic operation can finish in one cycle. As we showed in the algorithm analysis, having single cycle ALU provides significant speedup for SDR algorithms. And finally, like Ardbeg, some other commercial solutions also chose to incorporate accelerators for error correction algorithms, including Viterbi and Turbo decoders.

Reconfigurable SDR Architecture. Wireless protocols can be broken down into key computational patterns, which can be as fine-grained as a sequence of arithmetic operations, or as coarse-grained as DSP kernels. There have been numerous SDR solutions based on fine-grained computation fabrics. Examples of such solutions include picoArray [26], and the XiSystem's XiRisc [27]. The XiRisc, also includes a scalar/VLIW processor, with the reconfigurable logic acting as an accelerator. One of the major drawbacks of this approach is the high communication cost of data shuffling within the computation fabrics. The coarse-grained reconfigurable architectures contain a system of heterogeneous coarse-grained PEs, with each type of PE tailored to a specific DSP algorithm group. Examples include Intel RCA [28], QuickSilver [29] and IMEC ADRES [30]. Both RCA and QuickSilver have 3 or 4 different types of PEs, ranging from simple scalar processors to application specific instruction processors to serve as Viterbi and Turbo accelerators. These heterogeneous SDR systems provide a tradeoff between overall system flexibility and individual kernel computational efficiency. Different wireless protocols require very different types of DSP algorithms and a heterogeneous system is more-likely to under-utilize its hardware, resulting in less efficient overall system operation.

6. Conclusion

Software defined radio promises to revolutionize the wireless communication industry by delivering a low-cost multi-mode baseband processing solution. Previous work has proposed SODA, a multi-core wide SIMD DSP architecture. Ardbeg is a commercial prototype based on SODA designed by ARM Ltd. Aspects of the SODA design are kept intact, such as the wide 512-bit SIMD datapath and the coupled scalar and SIMD datapath. Application-specific design tradeoffs are made to achieve higher computational efficiency while maintaining enough flexibility to support multiple protocols. The evolution of SODA to Ardbeg happened due to optimization in three main areas: wide SIMD design, LIW support for wide SIMD, and algorithm specific hardware acceleration. The results show that Ardbeg's architectural optimizations achieve between 1.5-7x speedup over SODA across multiple wireless algorithms.

Acknowledgment

We thank the anonymous referees for their useful comments and suggestions. This research was supported by ARM Ltd. and the National Science Foundation under grants CSR-EHS 0615261, CSR-EHS 0615135, and CCR-0325761.

References

[1] *Samsung, NXP, and T3G Showcase World's First TD-SCDMA HSDPA/GSM Multi-mode Mobile Phone*, NXP Semiconductors, Nov. 2007. [Online]. Available: http://www.nxp.com/news/content/file_1377.html

[2] H. Lee, Y. Lin, Y. Harel, M. Woh, S. Mahlke, T. Mudge, and K. Flautner, "Software defined radio - a high performance embedded challenge," in *HiPEAC. Volume 3793 of Lecture*

	ARM Ardbeg	SODA	Infineon MuSIC	ADI TigerSHARC	Icera DXP	Phillips EVP	Sandbridge Sandblaster
# DSPs	2	4	4	8	—*	1	4
PE frequency in MHz	350	400	300	250	1000	300	600
# 16-bit SIMD lanes	32	32	4	2x4	4	16	4
VLIW support on SIMD	restricted	no	yes	yes	yes	yes	yes
Max # of EX stages	1	2	4	2	20	—*	4
Scalar datapath	yes	yes	no	no	no	yes	yes
Hardware coprocessor	yes	no	yes	no	no	yes	no
Scratchpad memory	yes	yes	yes	yes	yes	yes	yes
Shared global memory	yes	yes	yes	no	—*	no	no

Figure 11: Architectural comparison summary between proposed SIMD-based SDR processors. *For the Icera DXP and the Phillips EVP, some of the architectural details are not released to the public at this time.

Notes in Computer Science. Springer, Nov 2005, pp. 6–26.

[3] Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, and C. Chakrabarti, "Soda: A low-power architecture for software radio," in *In Proc. of the 33rd Annual International Symposium on Computer Architecture*, 2006, pp. 89–101.

[4] N. Clark *et al.*, "OptimoDE: Programmable Accelerator Engines Through Retargetable Customization," in *Proc. Hot Chips 6*, "Aug." 2004.

[5] *ARM Neon Technology*, ARM Ltd., Sep. 2004. [Online]. Available: http://www.arm.com/products/CPUs/NEON.html

[6] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," in *ISCA '73: Proceedings of the 1st annual symposium on Computer architecture*. New York, NY, USA: ACM, 1973, pp. 21–28.

[7] P. H. Hofstee, "All About the Cell Processor," in *IEEE Symposium on Low-Power and High-Speed Chips(COOL Chips VIII)*, April 2005.

[8] B. Thies, M. Karczmarek, and S. Amarasinghe, "Streamit: A language for streaming applications," in *In Proceedings of the International Conference on Compiler Construction*, June 2002, pp. 179–196.

[9] M. Kudlur and S. Mahlke, "Orchestrating the execution of stream programs on multicore platforms," in *PLDI '08: Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation*. New York, NY, USA: ACM, 2008, pp. 114–124.

[10] Y. Lin, S. Mahlke, T. Mudge, C. Chakrabarti, A. Reid, and K. Flautner, "Design and implementation of turbo decoders for software defined radio," Oct. 2006, pp. 22–27.

[11] M. Schneider, H. Blume, and T. G. Noll, "Power estimation on functional level for programmable processors," vol. 2, 2004, pp. 215–219. [Online]. Available: http://www.adv-radio-sci.net/2/215/2004/

[12] K. van Berkel, F. Heinle, P. P. E. Meuwissen, K. Moerman, and M. Weiss, "Vector processing as an enabler for software-defined radio in handheld devices," *EURASIP J. Appl. Signal Process.*, vol. 2005, no. 1, pp. 2613–2625, 2005.

[13] *RealView Compilation Tools Assembler Guide*, ARM Ltd., Mar. 2007. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.dui0204h/

[14] H. Holma and A. Toskala, *WCDMA for UMTS: Radio Access For Third Generation Mobile Communications*. New York, New York: John Wiley and Sons, LTD, 2001.

[15] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer in the 5 GHz Band*, IEEE Standard 802.11a-1999, Part 11, 1999.

[16] *Digital Video Broadcasting(DVB); Implementation guidelines for DVB terrestrial services; Transmission aspects*, ETSI TR 101 190 V1.2.1, Apr. 2004.

[17] *Digital Video Broadcasting(DVB); Transmission System for Handheld Terminals(DVB-H)*, ETSI EN 302 304 V1.1.1, Nov. 2004.

[18] *Predictive Technology Model*. [Online]. Available: http://www.eas.asu.edu/ ptm/

[19] Y. Neuvo, "Cellular phones as embedded systems," Feb. 2004, pp. 32–37 Vol.1.

[20] P. Ryan, T. Arivoli, L. De Souza, G. Foyster, R. Keaney, T. McDermott, A. Moini, S. Al-Sarawi, L. Parker, G. Smith, N. Weste, and G. Zyner, "A single chip phy cofdm modem for ieee 802.11a with integrated adcs and dacs," *Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. 2001 IEEE International*, pp. 338–339, 463, 2001.

[21] U. Ramacher, "Software-Defined Radio Prospects for Multistandard Mobile Phones," *Computer*, vol. 40, no. 10, pp. 62–69, 2007.

[22] H.-M. Bluethgen, C. Grassmann, W. Raab, and U. Ramacher, "A programmable platform for software-defined radio," Nov. 2003, pp. 15–.

[23] J. Fridman and Z. Greenfield, "The TigerSharc DSP architecture," in *IEEE Micro*, Jan. 2000, pp. 66–76.

[24] S. Knowles, *The SoC Future is Soft*, IEE Cambridge Branch Seminar 2005, Dec. 2005. [Online]. Available: http://www.ieee-cambridge.org.uk/arc/seminar05/slides/SimonKnowles.pdf

[25] J. Glossner, E. Hokenek, and M. Moudgill, "The Sandbridge Sandblaster Communications Processor," in *3rd Workshop on Application Specific Processors*, Sept. 2004, pp. 53–58.

[26] R. Baines and D. Pulley, "Software defined baseband processing for 3G base stations," in *4th International Conference on 3G Mobile Communication Technologies (Conf. Publ. No. 494)*, June 2003, pp. 123–127.

[27] A. Lodi *et al.*, "XiSystem: A XiRisc-Based SoC With Reconfigurable IO Module," in *IEEE Journal of Solid-State Circuits*, vol. 41, No. 1, Jan. 2006, pp. 85–96.

[28] I. Chen, A. Chun, E. Tsui, H. Honary, and V. Tsai, "Overview of Intel's Reconfigurable Communication Architecture," in *3rd Workshop on Application Specific Processors*, Sept. 2004, pp. 95–102.

[29] B. Plunkett and J. Watson, *Adapt2400 ACM Architecture Overview*, Quicksilver Technology, Jan. 2004. [Online]. Available: http://www.qstech.com

[30] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix." in *Proceedings of the Conference on Field Programmable Logic*, vol. 2778. Springer, 2003, pp. 61–70.