21

Baseband Processing Architectures for SDR

AQ1

AQ2 Seo

Chakrabarti

Lin

Woh

Mahlke

Mudge

21.1 Introduction **21**-1
21.2 SDR Overview **21**-3
21.3 Workload Profiling and Characterization **21**-4
W-CDMA Physical Layer Processing • W-CDMA Workload Profiling
21.4 Architecture Design Trade-Offs **21**-7
8 and 16 Bit Fixed-Point Operations • Vector-Based Arithmetic Computations • Control Plane versus Data Plane • Scratchpad Memory versus Cache • Algorithm-Specific ASIC Accelerators
21.5 Baseband Processor Architectures **21**-9
SODA Processor • ARM Ardbeg Processor • Other SIMD-Based Architectures • Reconfigurable Architectures
21.6 Cognitive Radio **21**-16
21.7 Conclusion **21**-16

21.1 Introduction

Wireless communication has become one of the dominating applications in today's world. Mobile communication devices are the largest consumer electronic group in terms of volume. In 2007, there was an estimated 3.3 billion mobile telephone subscriptions. This number is roughly half of the world's population. Applications like web browsing, video streaming, e-mail, and video conferencing have all become key applications for mobile devices. As technology becomes more advanced, users will require more functionality from their mobile devices and more bandwidth to support them. Furthermore, in recent years, we have seen the emergence of an increasing number of wireless protocols that are applicable to different types of networks. Figure 21.1 lists some of these wireless protocols and their application domains, ranging from the home and office WiFi network to the citywide cellular networks. The next-generation mobile devices are expected to enable users to connect to information ubiquitously from every corner of the world.

One of the key challenges in realizing ubiquitous communication is the seamless integration and utilization of multiple existing and future wireless communication networks. In many current wireless communication solutions, the physical layer of the protocols is implemented with nonprogrammable application-specific integrated circuit (ASIC) processors. The communication device consists of multiple processors, one for each wireless protocol. Such a solution is not scalable and is infeasible in the long run. Software-defined radio (SDR) promises to deliver a cost-effective and flexible solution by implementing a

AQ3

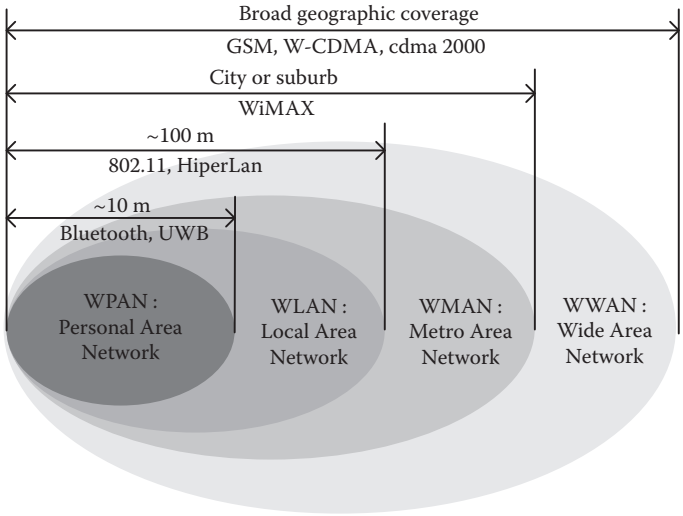

FIGURE 21.1 Categories of wireless networks.

wide variety of wireless protocols in software, and running them on the same hardware platform. A software solution offers many potential advantages, including but not limited to the following:

- A programmable SDR processor would allow multimode operation, running different protocols depending on the available wireless network, global system for mobile communications (GSM) in Europe, code division multiple access (CDMA) in the United States, and some parts of Asia, and 802.11 in coffee shops. This is possible with less hardware than custom implementations. AQ4
- A protocol implementation's time to market would be shorter because it would reuse the hardware. The hardware integration and software development tasks would progress in parallel.
- Prototyping and bug fixes would be possible for next-generation protocols on existing silicon through software changes. The use of a programmable solution would support the continuing evolution of specifications; after the chipset's manufacture, developers could deploy algorithmic improvements by changing the software without redesign.
- Chip volumes would be higher because the same chip would support multiple protocols without requiring hardware changes.

Designing a SDR processor for mobile communication devices must address two key challenges—meeting the computational requirements of wireless protocols while operating under the power budget of a mobile device. The operation throughput requirements of current third-generation (3G) wireless protocols are already an order of magnitude higher than the capabilities of modern digital signal processor (DSP) processors. This gap is likely to grow in the future. Figure 21.2 shows the computation and power demands of a typical 3G wireless protocol. Although most DSP processors operate at an efficiency of approximately 10 million operations per second (Mops) per milliwatt (mW), the typical wireless protocol requires 100 Mops/mW.

This chapter presents the challenges and trade-offs in designing architectures for baseband processing in mobile communication devices. It gives an overview of baseband processing in SDR, followed by workload and performance analysis of a representative protocol. Next it describes the architectural features of two low-power baseband architectures, signal processing on demand (SODA), and Ardbeg, followed by brief descriptions of other representative processor prototypes. It briefly introduces cognitive radio (CR) as the next challenge and concludes the chapter.

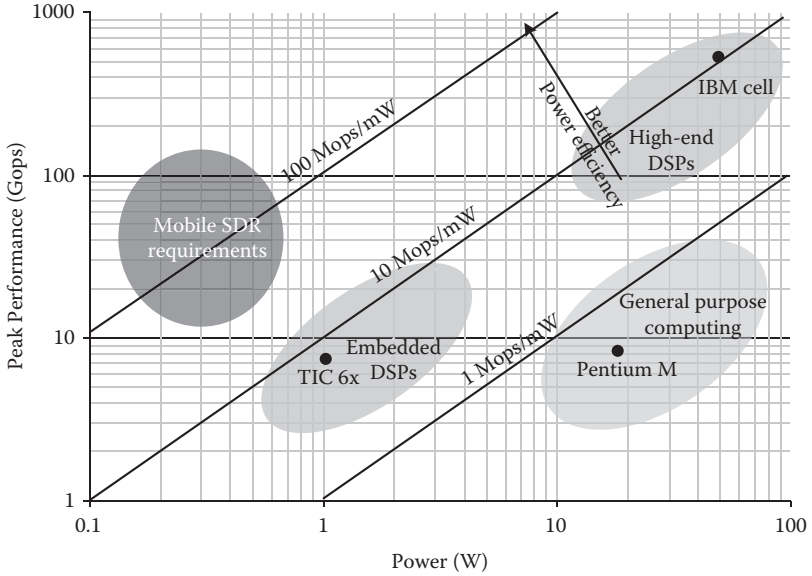

FIGURE 21.2 Throughput and power requirements of typical 3G wireless protocols. The results are calculated for 16 bit fixed-point operations.

21.2 SDR Overview

SDR promises to solve the problems of supporting multiple wireless protocols and addresses future challenges. The SDR forum, which is a consortium of service operators, designers, and system integrators, defines SDR as

> A collection of hardware and software technologies that enable reconfigurable system architectures for wireless networks and user terminals. SDR provides an efficient and comparatively inexpensive solution to the problem of building multimode, multiband, multifunctional wireless devices that can be enhanced using software upgrades. As such, SDR can really be considered an enabling technology that is applicable across a wide range of areas within the wireless industry.

Figure 21.3 shows the architecture for a typical 3G cellular phone. The architecture includes four major blocks: analog front-end, digital baseband, protocol processor, and application processor. The physical

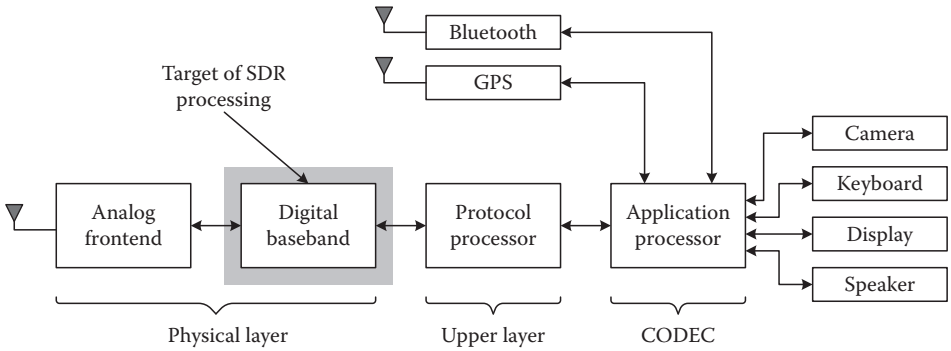

FIGURE 21.3 Architecture of a 3G cellular phone.

layer of wireless protocols includes both the analog front-end and the digital baseband. The analog front-end is usually implemented with analog ASICs. The digital baseband block performs the remaining physical layer operations and is also typically implemented with ASICs. The upper layers are implemented by the protocol processor and application processor, which are usually SoCs and consist of general purpose embedded DSP processors.

The objective of SDR is to replace the baseband ASICs with a programmable hardware platform, and implement the baseband processing in software. Designing programmable analog front-ends is quite a challenge and is beyond the scope of this chapter. Here, we focus on design of programmable digital baseband processing engines for SDR.

21.3 Workload Profiling and Characterization

21.3.1 W-CDMA Physical Layer Processing

We select the wide-band code division multiple access (W-CDMA) protocol as a representative wireless workload case study for designing the SDR processor. This section provides a brief summary of its algorithms and characteristics. A more detailed analysis can be found in [14].

The W-CDMA system is one of the dominant 3G wireless communication networks where the goal is multimedia service including video telephony on a wireless link [12]. It improves over prior cellular protocols by increasing the data rate from 64 Kbps to 2 Mbps. The protocol stack of the W-CDMA system consists of several layers. At the bottom of the stack is the physical layer which is responsible for overcoming errors induced by an unreliable wireless link. The next layer is the medium access control (MAC) layer which resolves contention in shared radio resources. The upper-layer protocols including MAC are implemented on a general purpose processor due to their relatively low computation requirements. In this section, we focus on the computation model of the W-CDMA physical layer. AQ5

Figure 21.4 shows a high-level block diagram of the digital processing in the W-CDMA physical layer. It shows that the physical layer contains a set of disparate DSP kernels that work together as one system. There are four major components: filtering, modulation, channel estimation, and error correction.

21.3.1.1 Filtering

Filtering algorithms are used to suppress signals transmitted outside of the allowed frequency band so that interference with other frequency bands are minimized. The finite impulse response (FIR) filter operations in W-CDMA can be very easily parallelized.

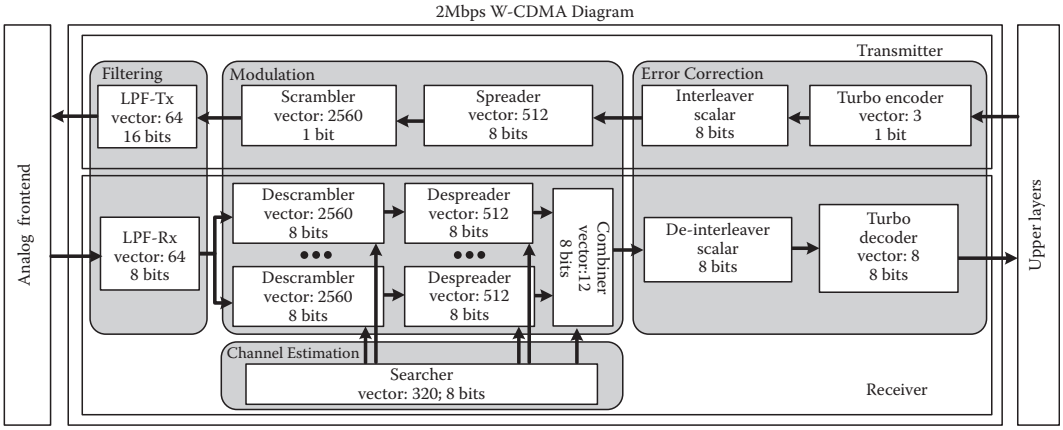

FIGURE 21.4 Physical layer operation of W-CDMA wireless protocol. Each block includes the algorithm's name, vector or scalar computation, vector width, and the data precision. The algorithms are grouped into four categories, shown in shaded boxes: filtering, modulation, channel estimation, and error correction.

21.3.1.2 Modulation

Modulation algorithms map source information onto the signal waveforms of the transmitter, and the receiver demodulates the signal waveforms back into source information. Two sets of codes are used for modulation: channelization codes and scrambling codes. Channelization codes are used so that the same radio medium can be used to transmit multiple different signal streams. Scrambling codes are used to extract the signal of a specific terminal among many transmitting terminals. On the receiver side, despreader is used to decode the channelization codes and descrambler is used to decode the scrambling codes. Demodulation requires the transmitter and the receiver to be perfectly synchronized. However, radio transmission suffers from multipath fading effect, where multiple delayed versions of the same signal stream are received due to environment interference. A searcher is used to find the synchronization point of each of the delayed signal streams and each of these delayed signals is decoded with its own despreader and descrambler. The decoded output of the despreader/descrambler pairs are then combined together as the demodulated output.

21.3.1.3 Channel Estimation

Channel estimation algorithms calculate the channel conditions to synchronize the two communicating terminals to ensure lockstep communication between the sender and the receiver. W-CDMA uses a searcher as its channel estimation algorithm. Searcher is called once per W-CDMA frame. There are two types of searchers—full searcher and quick searcher. In a group of eight frames, the full searcher is used for the first frame, and the quick searcher is used for the remaining seven frames. Both types of searchers consist of four steps: correlation, filtering out high-frequency noise, detecting peaks, and global peak detection.

21.3.1.4 Error Control Coding Algorithms

Error control coding (ECC) algorithms are used to combat noisy channel conditions. The sender encodes the original data sequence with a coding scheme that inserts systematic redundancies into the output, which is decoded by the receiver to find the most likely original data sequence. Two types of ECC algorithms are used—convolutional coding and Turbo coding. Turbo coding is used for the 2 Mbps data channel, while convolutional coding is used for all the other channels. For decoding, the Viterbi decoder is used for convolutional codes and the Turbo decoder is used for Turbo codes. Turbo decoding is usually the most computationally intensive algorithm in baseband processing. The corresponding decoder consists of two component decoders that are typically of type software output Viterbi algorithm (SOVA) or based on maximum a posteriori (MAP) and connected together by interleavers. The interleaving pattern is specified by the W-CDMA standard, but the choice of component decoder is left to the designers.

21.3.2 W-CDMA Workload Profiling

Table 21.1 shows the result of workload profiling for W-CDMA in active mode with a throughput of 2 Mbps [14]. The first column lists the W-CDMA algorithms. The second column lists the corresponding configurations for each of the algorithms. The third and fourth column lists the vector computation information for the algorithms. The fifth column lists the data precision width. The last column shows the peak workload of the algorithms. It is the minimum performance needed to sustain 2 Mbps throughput, under the worst wireless channel conditions. For this analysis the W-CDMA model was compiled with an Alpha gcc compiler, and executed on M5 architectural simulator [3]. The peak workload of each algorithm was calculated by dividing the maximum number of processing cycles by its execution time. AQ6

The analysis shows that there are a set of key DSP algorithms that are responsible for the majority of the computation. These algorithms include the FIR filter, searcher, Turbo decoder, descrambler, and despreader. The workloads of Viterbi and Turbo decoder require further verification because their

TABLE 21.1 Workload Analysis Result of W-CDMA Physical Layer Processing

Algorithms	Configurations	Vector Comp.	Vector Length	Bit Width	Comp. Mcycles/s
		W-CDMA (2 Mbps)			
Scrambler	Defined in W-CDMA standard	Yes	2,560	1, 1	240
Descrambler[a]	12 fingers, 3 base stations	Yes	2,560	1, 8	2,600
Spreader	Spreading factor = 4	Yes	512	8	300
Despreader[a]	12 fingers, 3 base stations	Yes	512	8	3,600
PN code (Rx)	3 base stations	No	1	8	30
PN code (Tx)	Defined in W-CDMA standard	No	1	8	10
Combiner[a]	2 Mbps data rate	Partial	12	8	100
FIR (Tx)	4 filters × 65 coeff × 3.84 Msps	Yes	64	1, 16	7,900
FIR (Rx)	2 filters × 65 coeff × 7.68 Msps	Yes	64	8, 8	3,900
Searcher[a]	3 base stations, 320 windows	No	320	1, 8	26,500
Interleaver	1 frame	No	1	8	10
Deinterleaver	1 frame	Partial	1	8	10
Turbo Enc.	$K = 4$	Yes	3	1, 1	100
Turbo Dec.[a]	$K = 4$, 5 iterations	Yes	8	8, 8	17,500

Note: "Vector Comp" indicates whether the algorithm contains vector-based arithmetic operations. "Vector Width" lists the native computation vector width. "Bit Width" lists the data precision width. "Comp Mcycle/s" lists the peak workload of running the algorithm on a general purpose processor.

[a] These algorithms have dynamically changing workloads that are dependent on channel conditions.

processing times are not fixed. They are based on estimates that are derived from the protocol standard specifications for the different transmission channels that use these decoders.

21.3.2.1 Instruction Type Breakdown

Figure 21.5 shows the instruction type breakdown for the W-CDMA physical layer and their weighted average for the case when the terminal is in the active state and has peak workload [14]. Instructions are

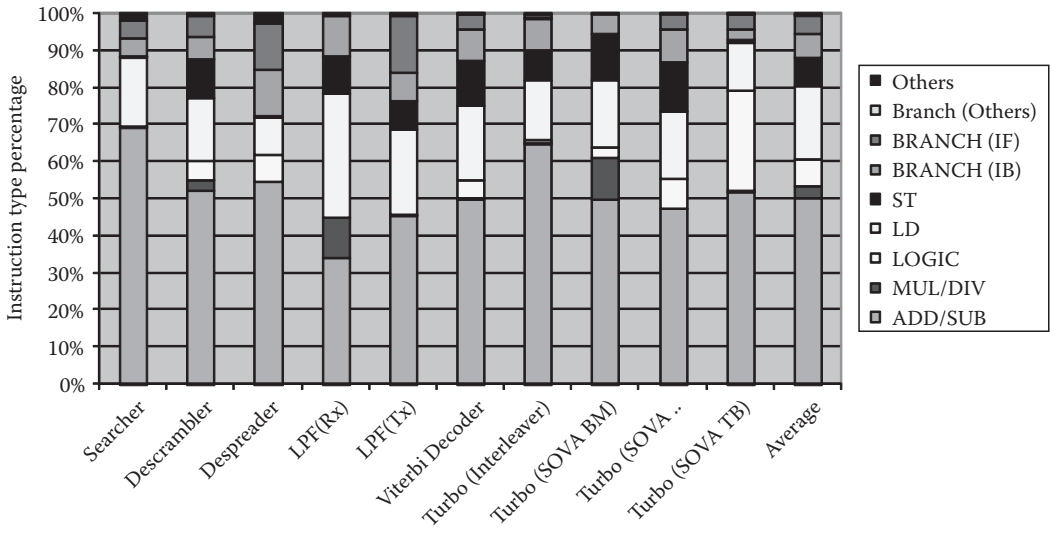

FIGURE 21.5 W-CDMA instruction type breakdown.

grouped into seven categories: add/sub; multiply/divide; logic; load; store; branches; and miscellaneous instructions. All computations are done using fixed-point arithmetic. The first thing to notice is the high percentage of add/subtract instructions. They account for almost 50% of all instructions, with the searcher at about 70%. This is because the searcher's core operation is the inner product of two vectors, and the multiplication here can be simplified into a conditional complement which can again be implemented by a subtraction. Other hot spots, like the Turbo decoder, also have a large number of addition operations. The second thing to notice is the lack of multiplications/divisions (<3% on average). This is because the multiplications of major algorithms are simplified into logical or arithmetic operations as discussed earlier. The multiplication of the combiner and Turbo encoder is not as significant because their workload is very small. One exception is multiplication in the LPF-Rx. Figure 21.5 also shows that the number of load/store operations is significant. This is because most algorithms consist of loading two operands and storing the operation result. Results also show that the portion of branch operations is about 10% on average. Most frequent branch patterns are loops with a fixed number of iterations corresponding to a vector size, and a conditional operation on vector variables. There are a few while or do-while loops, and most loops are 1–2-levels deep. AQ7

21.3.2.2 Parallelism in the Protocol

To meet the real-time W-CDMA performance requirement in software, the inherent algorithmic parallelism must be exploited. Columns 3 and 4 of Table 21.1 show the potential parallelism that can be exploited either through data-level parallelism (DLP) or thread-level parallelism (TLP). Most of the algorithms operate on one-dimensional vectors. Therefore, we define parallelism as the maximum vector width for these algorithms' arithmetic operations, which is shown in column 4.

From this result, we can see that the searcher, filter, scrambler, and descrambler all contain very wide vector operations. They also have a fair amount of TLP. For instance, the searcher operation can be executed as 5120 concurrent threads. For the case of scrambler and descrambler, TLP can be obtained by bisecting a wide vector into smaller ones. Although sliced vectors are not perfectly uncorrelated, we can execute the smaller vector operations with negligible dependency. Another example is the Turbo decoder, which when implemented in a straightforward manner, contains limited vector and TLP. This can be mitigated by use of sliding window techniques where multiple blocks of data can be processed at the same time.

21.3.2.3 Stream Computation

Another important characteristic of the W-CDMA protocol is that it is a streaming computation system. Here, multiple DSP kernel algorithms are connected together in feed-forward pipelines and data streams through these pipelines sequentially. There is very little data temporal locality, which means that data do not get reused. Thus, cache structures provide little additional benefits, in terms of power and performance, over software-controlled scratchpad memories.

21.4 Architecture Design Trade-Offs

This section describes some of the key challenges and trade-offs in designing processor architectures for supporting mobile SDR.

21.4.1 8 and 16 Bit Fixed-Point Operations

In W-CDMA wireless protocol, the majority of the algorithms operate on 1 to 8 bit data, with some algorithms operating on 16 bit data. In 802.11a wireless protocol, the majority of the DSP algorithms operate on 16 bit fixed-point data, with a few algorithms operating on 8 bit data. A few DSP algorithms, such as fast Fourier transform (FFT) for the next-generation 4G wireless protocols, may require 24 bit or 32 bit fixed-point precision. There are no wireless protocols that require floating-point support.

21.4.2 Vector-Based Arithmetic Computations

Computations in a typical DSP application consists of vector and matrix arithmetic operations. This is true for wireless protocols as well. However, it is important to note that wireless protocols' computations are mostly on one-dimensional vectors.

DSP computations on vectors and matrices can be broken into two key components: data computation and data rearrangement. Data computation includes addition, multiplication, and other arithmetic and logic operations. Data rearrangement reorders the elements in vectors and matrices. In most DSP computations, it is not the data computation, but the data rearrangement that is the real performance bottleneck. Data computation limitations can always be addressed by adding more hardware computation units (such as adders and multipliers). However, there is no single common hardware solution that can accelerate all rearrangement operations for any arbitrary-sized vectors and matrices. For example, FFT requires a butterfly operation, where pairs of data values in a vector are swapped. In comparison, the ECC decoder may require an interleaver that transposes a matrix by blocks. The optimal hardware accelerators for these two patterns are quite different. An efficient design must exploit the specific data rearrangement patterns of the kernel algorithms. Since most of the vector arrangement patterns do not change during run-time, the interconnects between the arithmetic and logic units (ALUs) and memories can be tailored to these algorithms.

21.4.3 Control Plane versus Data Plane

Complete software implementations of wireless protocols usually require two separate steps: (1) implementation of the DSP algorithms; and (2) implementation of the protocol system. The DSP algorithms are computational intensive kernels that have relatively simple data-independent control structures. The protocol system has relatively light computation load, but complicated data-dependent control behavior. Therefore, most commercial SDR solutions include a two-tiered architecture: a set of data processors that is responsible for heavy duty data processing; and a control processor that handles the system operations, and manages the data processors through remote-procedure-calls and DMA operations.

21.4.4 Scratchpad Memory versus Cache

In addition to data computation, wireless protocols must also handle the data communication between algorithms. These interalgorithm communications are usually implemented by data streaming buffers. While the processors operate on the current data, the next batch of data can be streamed between the memories and register files (RFs). There is very little data locality between the algorithms. This suggests that a scratchpad memory is better suited for SDR than a cache-based memory. Scratchpad memories are easier to implement and consume less power. However, they do not have the hardware support for managing data consistencies, which means that memory management must be performed explicitly by software. This adds extra burden and complexity for the programmers and compilers. In addition, executing memory management instructions prevents processors from doing DSP computations, which may significantly degrade the overall performance. Therefore, many scratchpad memory systems provide direct memory access (DMA) units. DMAs can move large blocks of data between the memories without interrupting the processors. However, these devices must also be explicitly managed by software.

21.4.5 Algorithm-Specific ASIC Accelerators

SDR is a solution where the entire wireless protocol is implemented in software. However, for a SDR solution to be competitive with existing ASIC solutions, both in terms of performance and power, ASIC accelerators are sometimes used for ECC and filtering. This is because these two groups of algorithms

are often the most computationally intensive algorithms in a wireless protocol, and different wireless protocols often use very similar algorithms for filtering and ECC. Previous studies have shown that there is an approximately 4–5× energy efficiency between an ASIC and a programmable solution for these two groups of algorithms. Consequently, there have been multiple efforts both in the academia and the industry on developing combined Viterbi and Turbo ASIC accelerators. Texas Instrument already sells a Viterbi accelerator, as a part of a SOC package, which can support configurations needed for multiple wireless protocols.

21.5 Baseband Processor Architectures

Many baseband processing architectures have been proposed in the last few years. They can be broadly categorized into single-instruction, multiple data (SIMD)-based architectures and reconfigurable architectures [20]. SIMD-based architectures usually consist of one or few high-performance DSP processors that are typically 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 SODA [15] and Ardbeg [23], that we will describe in details, fall under the SIMD-based architecture category. Reconfigurable architectures, on the other hand, are usually made up of many simpler processing elements (PEs). Depending on the particular design, these PEs range from fine-grain ALU units to coarse-grain ASICs. The PEs are usually connected through a reconfigurable fabric. Compared with SIMD-based design, reconfigurable architectures are more flexible at the cost of higher power. In this section, we first describe SODA and Ardbeg processors followed by descriptions of other SIMD and reconfigurable SDR architectures.

21.5.1 SODA Processor

SODA processor architecture is an academic research prototype for mobile SDR [15]. It is a SIMD-based DSP architecture designed to meet both the performance and power requirements for two representative protocols, W-CDMA and IEEE 802.11a. The SODA multiprocessor architecture is shown in Figure 21.6. It consists of multiple PEs, a scalar control processor, and global scratchpad memory, all connected through a shared bus. Each SODA PE consists of five major components: (1) an SIMD pipeline for supporting vector operations; (2) a scalar pipeline for sequential operations; (3) two local scratchpad memories for the SIMD pipeline and the scalar pipeline; (4) an address generation unit (AGU) pipeline for providing the addresses for local memory access; and (5) a programmable DMA unit to transfer data between memories and interface with the outside system. The SIMD pipeline, scalar pipeline, and the AGU pipeline execute in VLIW-styled lockstep, controlled with one program counter (PC). The DMA unit has its own PC, its main purpose is to perform memory transfers and data rearrangement. It is also the only unit that can initiate memory access with the global scratchpad memory. AQ8

The SIMD pipeline consists of a 32-lane 16 bit datapath, with 32 arithmetic units working in lockstep. 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. The multiplier takes two execution cycles when running at the targeted 400 MHz. Intraprocessor data movements are supported through the SIMD shuffle network (SSN), as shown in Figure 21.7. The SSN consists of a shuffle exchange (SE) network, an inverse shuffle exchange (ISE) network, and a feedback path. By including both the SE and ISE networks, the number of iterations can be reduced. In addition to the SSN network, a straight-through connection is also provided for data that does not need to be permutated.

The SIMD pipeline can take one of its source operands from the scalar pipeline. This is done through the scalar-to-vector (STV) registers, shown in the SIMD pipeline portion of Figure 21.6. The STV contains four 16 bit registers, which only the scalar pipeline can write, and only the SIMD pipeline can read. The SIMD pipeline can read 1, 2, or all 4 STV register values and replicate them into

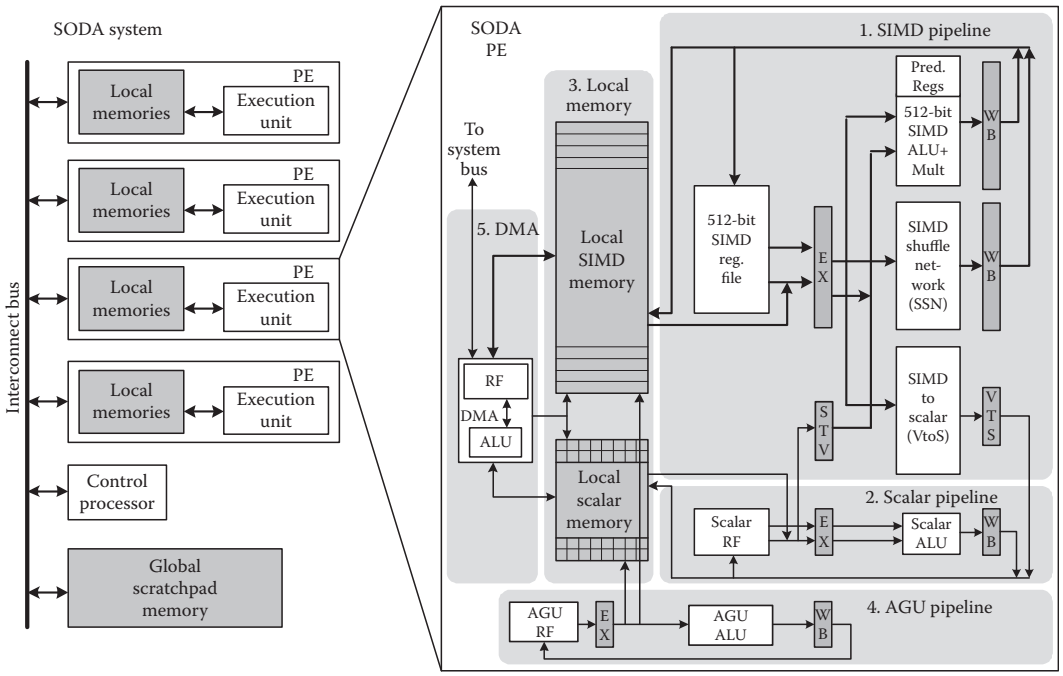

FIGURE 21.6 SODA architecture. The system consists of four data PEs, one control processor, and global scratchpad memory, all connected through a shared bus. Each PE consists of a 32-lane 16 bit SIMD pipeline, a 16 bit scalar pipeline, two local scratchpad memories, an AGU for calculating memory addresses, and a DMA unit for interprocessor data transfer.

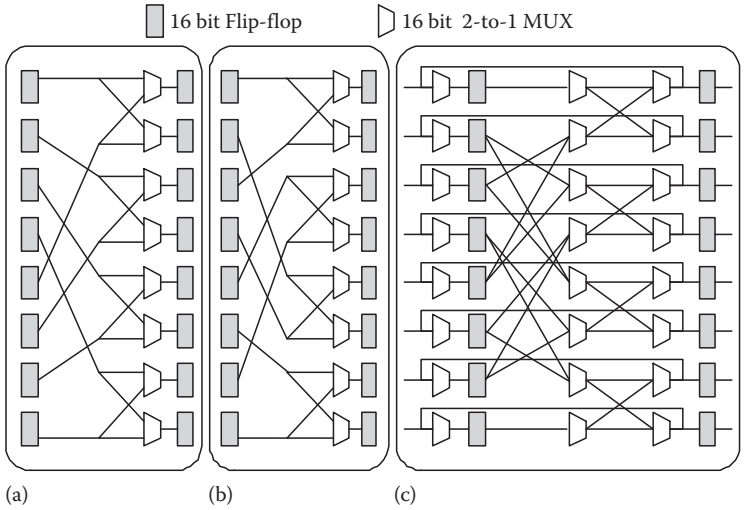

FIGURE 21.7 8-Wide SSN. (a) 8-Wide shuffle exchange network; (b) 8-wide inverse shuffle exchange network; (c) 8-wide SSN with shuffle exchange, inverse shuffle exchange and feedback path.

32-element SIMD vectors. SIMD-to-scalar operations transfer values from the SIMD pipeline into the scalar pipeline. This is done through the vector-to-scalar (VTS) registers, shown in Figure 21.6. There are several SIMD reduction operations that are supported in SODA, including vector summation, finding the minimum and the maximum.

The DMA controller is responsible for transferring data between memories. It is the only component in the processor that can access the SIMD, scalar and global memories. The DMA is also implemented as a slave device controlled by the scalar pipeline. However, it can also execute its own instructions on its internal register file and ALU, similar to the scalar pipeline. This gives the DMA the ability to access the memory in a wide variety of application-specific patterns without assistance from the master processor.

21.5.1.1 Arithmetic Data Precision

SODA PE only provides support for 8 and 16 bit fixed-point operations. This is because for both W-CDMA and 802.11a wireless protocols, the majority of the DSP algorithms, operate on 8 bit or 16 bit fixed-point data. Each lane in the SODA PE supports 16 bit fixed-point arithmetic operations. The AGU registers are 12 bit, but only support 8 bit addition and subtraction. This is because AGU is used for software management of data buffers, in which 8 bit are sufficient. The higher 4 bit are used to address different PEs, as well as different memory buffers within PEs.

21.5.1.2 Vector Permutation Operations

With SODA's SSN network (shown in Figure 21.7), any 32-wide vector permutation can be performed with a maximum of nine cycles. Combining with predicated move operations, the SSN network can support any vector length permutation. However, for every 32-wide vector permutation, one additional instruction must be used to setup the permutation network. This is because each MUX within the shuffle network requires its own control bit. Each iteration through the SSN requires 64 bit. For a maximum of nine iterations, this requires 576 bit of control information. The SSN is not very efficient if there are many permutation patterns that are used frequently. However, the majority of the algorithms in SDR only use one or a few shuffle patterns, which makes the network setup overhead not significant.

21.5.1.3 Performance

For W-CDMA and 802.11a, the SODA architecture achieves large speedups over a general purpose Alpha processor. For example, W-CDMA's searcher algorithm requires 26.5 giga operations per second (Gops) on the general purpose processor and requires only 200 Mops on SODA. The performance improvements are mainly due to SODA's wide-SIMD execution. AQ9

The RTL Verilog model of SODA was synthesized in TSMC 180 nm technology. The results show that for a clock frequency of 400 MHz; SODA consumes 2.95 W for W-CDMA 2 Mbps system and 3.2 W for 802.11a 24 Mbps system. However, with technology scaling, the power numbers are expected to reduce to acceptable levels such as 450 mW for 90 nm technology and 250 mW for 65 nm technology.

21.5.2 ARM Ardbeg Processor

Ardbeg is a commercial prototype based on the SODA architecture [23]. The Ardbeg system architecture is shown in Figure 21.8. It consists of two PEs, each running at 350 MHz in 90 nm technology, an ARM general purpose controller, and a global scratchpad memory. In addition, it includes an accelerator dedicated to Turbo decoding. Each Ardbeg PE supports 512 bit SIMD data, and can operate with 8, 16, and 32 bit fixed-point and 16 bit block floating-point (BFP) precision. It supports VLIW execution on its SIMD pipeline, allowing a maximum of two different SIMD instructions to be issued in parallel. The instruction set for Ardbeg is derived from NEON [1].

21.5.2.1 SIMD Permutation Support

The Ardbeg PE employs a 128-lane 8 bit 7-stage Banyan shuffle network. This network is designed to support the most common vector permutation operations in a single cycle. In addition, it also provides a few key permutation patterns designed for interleaving. 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 shuffle network can only permute vector patterns of SIMD width. However, by designing the network

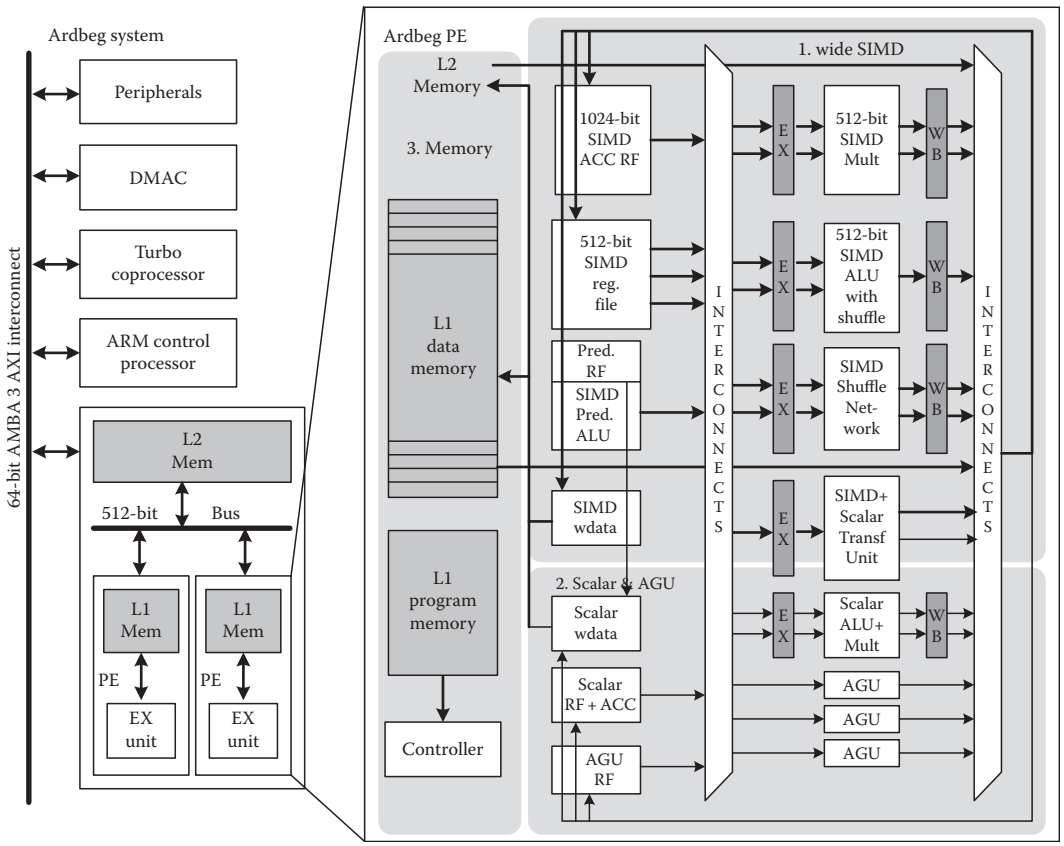

FIGURE 21.8 Ardbeg System Architecture. It consists of two data PEs, a Turbo coprocessor, an ARM control processor and peripherals. Each PE consists of a 512 bit SIMD pipeline and a 16 bit scalar pipeline.

to support a small set of permutation patterns, the permutation time can be significantly reduced even in these cases.

21.5.2.2 Partial VLIW SIMD Execution

VLIW 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 VLIW, but provides better energy efficiency due to lesser number of SIMD register file ports. For instance, combining overlapping memory accesses with SIMD computation is beneficial because most DSP algorithms are streaming. Similarly, combining SIMD arithmetic/multiplication with SIMD-scalar transfer is beneficial for filter-based algorithms and combining SIMD multiply with move operation is beneficial for FFT-based algorithms.

21.5.2.3 Block Floating-Point Support

Contemporary wireless protocols require large-sized FFTs. BFP provides near floating-point precision without its high power and area costs. A key operation in BFP is finding the maximum value among a set of numbers. While DSP processors support this operation in software, in Ardbeg, a special instruction is implemented that finds the maximum value in a 32-lane 16 bit SIMD vector. Though BFP is currently used only for FFT computations, any algorithm which requires higher precision can utilize the BFP instruction extensions.

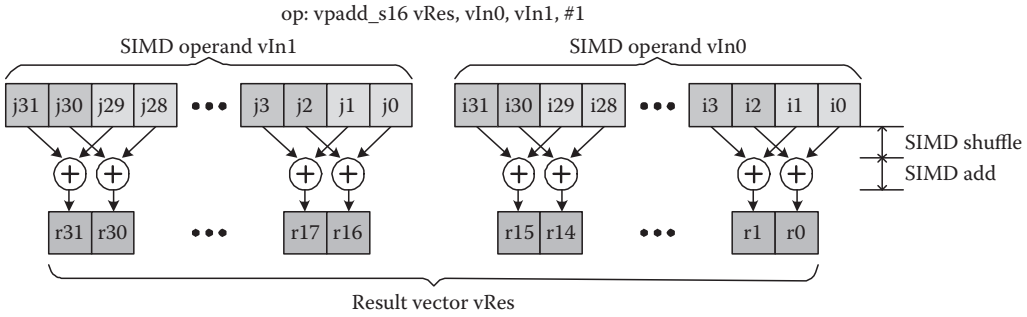

FIGURE 21.9 Ardbeg's pair-wise butterfly SIMD operation implemented using a fused permute and ALU operation. The figure shows pairs of a 2-element butterfly operation. Ardbeg supports pairs of 1-, 2-, 4-, 8-, and 16-element butterfly of 8 and 16 bit. This butterfly operation uses the inverse perfect shuffle pattern.

21.5.2.4 Fused Permute-and-ALU Operations

In many DSP algorithms it is common to first 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. The Ardbeg PE is optimized for these operations 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 front of the SIMD ALU. This 1-stage shuffle network only supports inverse perfect shuffle patterns between different groups of lanes. The different pair-wise butterfly operations are shown in Figure 21.9. The shuffle and the add operations are performed in the same cycle. A 2048-point FFT is able to gain 25% speedup using fused butterfly operations.

21.5.2.5 Performance

Three different wireless communication protocols, W-CDMA, 802.11a, DVB-T/H, are implemented on the Ardbeg architecture and the performance compared to SODA. The results show that Ardbeg's three enhancements—optimized wide-SIMD design such as wider SSN and reduced latency functional units (FUs), LIW support for wide-SIMD, and algorithm-specific hardware accelerator—achieve large speedup ranging from 1.5× to 7× over SODA. In addition, the evaluation shows that the Ardbeg processor synthesized in 90 nm technology running at 350 MHz is able to support the 3G wireless processing within the 500 mW power budget of typical mobile devices.

21.5.3 Other SIMD-Based Architectures

In this section, we describe several other SIMD-based architectures proposed in the industry and the academia. Like SODA and Ardbeg, most of these architectures consist of one or more high-performance SIMD DSP processors which are connected via a shared bus. These architectures usually use software-managed scratchpad data memories to meet the real-time constraints. Many of them support VLIW execution by allowing concurrent memory and SIMD arithmetic operations. Some solutions also choose to incorporate accelerators for ECC algorithms, such as those based on convolutional and Turbo codes.

21.5.3.1 EVP

The embedded vector processor (EVP) [22] from NXP consists of a 16-lane 16 bit SIMD datapath, one scalar datapath, programmable memory, and VLIW controller. It allows five vector operations, four

scalar operations, three address operations, and one loop control to be executed at once. The SIMD datapath consists of vector memory, 16 vector registers, load/store unit, ALU, MAC/shift unit, intravector unit, and code generation unit. The 16 bit datapath supports 8 and 32 bit data to allow word-level parallelism. It also supports data types such as complex numbers. In the SIMD datapath, the shuffle unit rearranges the elements of a single vector according to any pattern; the intravector unit supports summation, maximum, and split operations; and the code generation unit supports different CDMA-codes and CRC checks.

21.5.3.2 DXP

The Deep eXecution Processor (DXP) [13] from ICERA is a 2-LIW 4-lane SIMD architecture. Its key features are deeply pipelined execution units and a programmable configuration map which holds pseudostatic configurations and constants for the execution units. In the SIMD execution datapath, SIMD ALUs are chained to exploit the characteristics of streaming data. The chained operation saves register file access power at the cost of a less flexible SIMD datapath. Icera's processors do not use any hardware accelerators.

21.5.3.3 TigerSHARC

The TigerSHARC [9] implementation, ADSP-TS001, from Analog Devices adopts several features found in general purpose processors such as a register-based load-store architecture with a static superscalar dispatch mechanism, a highly parallel short-vector-oriented memory architecture, support for multiple data types including 32 bit single-precision floating-point and 8 bit/16 bit fixed-point, parallel arithmetic instructions for two floating-point multiply-accumulate (MAC) operation or eight 16 bit MACs, 128-entry four-way set associative branch target buffer, and 128 architecturally visible, fully interlocked registers in four orthogonal RFs. The TigerSHARC architecture provides concurrent SIMD arithmetic operations by having two 4-lane SIMD computation blocks controlled with two instructions. It fetches four instructions and issues one to four instructions per clock cycle. The 128 32 bit registers are memory mapped and divided into four separate RFs of size 32×32 bit. The multiple data type supports subword parallelism in addition to inherent SIMD data parallelism.

21.5.3.4 MuSIC-1

The MuSIC-1[4] processor from Infineon consists of four SIMD core clusters, a general purpose processor, shared memory and dedicated programmable processors for FIR filter and Turbo/Viterbi decoder. Each SIMD core contains four PEs and supports special instructions and LIW features for arithmetic operations, local memory accesses, and inter-PE communications. The general purpose processor runs control programs to provide the PE controller with instruction addresses. The code and data are stored in an external memory, and the on-chip memory consists of multiple banks to support simultaneous accesses.

21.5.3.5 Sandblaster

The Sandblaster [11] architecture from Sandbridge Technology is an example of a multithreaded SIMD vector architecture. It consists of a RISC-based integer execution unit and multiple SIMD vector units. In addition, multiple copies of I-cache and data memory are available for each thread. Each instruction has four fields: load/store, ALU, integer multiplier, and vector multiplier. Therefore, it can issue up to four simultaneous instructions where one may be a data-parallel vector operation. The architecture also uses token-triggered threading, which consumes much less power than simultaneous multithreading because it issues instructions in round-robin fashion. The Sandblaster architecture supports up to eight threads.

21.5.3.6 SIMT

The single-instruction stream multiple task architecture (SIMT) [18] from Linkoping University consists of Complex MAC (CMAC) SIMD units, Complex ALU (CALU) SIMD units, multiple memory banks, on-chip network, accelerators, and a controller unit. The CMAC unit consists of four CMAC lanes, each of which uses 14×14 bit complex multipliers and has eight 2×40 bit accumulator registers. The CALU unit is similar to the CMAC unit except with simplified complex multiplier supporting only $0, +/-1, +/-i$ multiplications. The controller efficiently manages the two SIMD units and the memory so that several threads can be run simultaneously. The memory is organized into multiple banks and each bank contains its own AGU to minimize memory access conflicts. The programmable reconfigurable crossbar switch is used as the on-chip network.

21.5.4 Reconfigurable Architectures

Reconfigurable architectures usually consist of many small PEs, which are connected through an interconnection fabric. These architectures can be categorized as either homogeneous or heterogeneous based on the type of PE. In addition, these PEs range from fine-grain lookup tables (LUTs) to coarse-grain ALU units and even ASICs.

21.5.4.1 ADRES

An architecture for dynamically reconfigurable embedded system (ADRES) [16] from IMEC is an example of a coarse-grain reconfigurable tile architecture, which tightly couples a VLIW processor and a coarse-grain reconfigurable matrix. This tightly coupled system has the advantages of shared resources, reduced communication costs, improved performance, and simplified programming model. The VLIW processor and the reconfigurable matrix share FUs and RFs. For the reconfigurable matrix part, there are many reconfigurable cells (RCs) which comprise FUs, RFs, and configuration RAM. The RCs are connected to the nearest neighbor RCs and RCs within the same row or column in the tile. Therefore, kernels with a high level of DLP are assigned to the ADRES tiles, whereas sequential codes are run on the VLIW processor. In ADRES architecture, the data communication is performed through the shared RF and memory, which is more compiler-friendly than the message-passing method. In addition, the ADRES relies on modulo scheduling and traditional VLIW compiler support to exploit both DLP and ILP to maximize PE utilization.

21.5.4.2 Montium

The Montium [21] architecture from Delft University is a coarse-grained reconfigurable processor targeting 16 bit algorithms. Montium consists of two parts: (1) communication and configuration unit (CCU) and (2) Montium tile processor (TP). The CCU configures the Montium TP and parts of the CCU for either ''block-based communication'' mode or ''streaming communication'' mode. The TP consists of five Montium ALUs and ten local memories, which are vertically segmented into five processing part arrays (PPAs). A relatively simple sequencer controls the entire PPA and selects configurable PPA instructions that are stored in the decoders. Montium ALU consists of four FUs in level 1 followed by a multiplier and an adder in level 2. Neighboring ALUs can also communicate directly on level 2 in the TP. Finally, each local SRAM is 16 bit wide and accompanies each AGU, and the memory can be used for both integer and fixed-point LUTs.

21.5.4.3 Adapt2400 ACM

Adapt2400 adaptive computing machine (ACM) [19] from QuickSilver consists of two basic components: nodes and matrix interconnection network (MIN). Nodes are the computing resources in the ACM architecture that perform actual work. Each node has its own controller, memory, and computational resources, so that it can independently executes algorithms that are downloaded in the

form of binary files, called SilverWare. A node is capable of implementing a first come, first serve, non pre-emptive multitasking with the support of a hardware task manager. The MIN ties the heterogeneous nodes together and provides the backbone for carrying data, SilverWare, and control information. It is hierarchically structured, and data within the MIN is transported in single 32 bit word packets to other nodes or external interface. This heterogeneous coarse-grain reconfigurable architecture cooperates with InSpire SDK Tool Set to provide integrated scalable hardware/software platform.

21.5.4.4 XiRisc

XiRisc [8] from XiSystem is an example of a fine-grain reconfigurable architecture. It is a VLIW RISC processor with two concurrent execution datapaths and a set of DSP-like FUs that are shared between two datapaths. The concurrent execution path represented by a pipelined configurable gate array (PiCoGA) provides a run-time extension of the processor ISA for application-specific functions. The PiCoGA is a programmable pipelined datapath composed of an array or rows, which can function as customized pipeline stages. Each row is composed of 16 reconfigurable logic cells (RLCs) containing two 4-input 2-output LUTs, four registers, and dedicated logic for a fast carry chain. Each RLC is connected to the others through a programmable interconnection matrix with 2 bit granularity switches. XiRisc exploits the synergy of the different execution units, ranging from a 32 bit dedicated MAC unit to bit-level processing blocks on PiGoGA, thereby increasing the execution efficiency and saving energy.

21.6 Cognitive Radio

Wireless networks typically have to operate on a fixed spectrum assignment, as a result of which, the spectrum is severely underutilized. CR built on SDR has the ability to change the transmitter parameters and operating characteristics based on interaction with its environment. It reduces spectrum underutilization through dynamic and opportunistic access to licensed and unlicensed bands. In order to support dynamic spectrum management, CR has to provide (1) spectrum sensing, the ability to monitor available spectrum bands, (2) spectrum decision, the capability to select the optimum spectrum band, (3) spectrum sharing, the coordination of users' transmission, and (4) spectrum mobility, the ability to ensure smooth transmission during spectrum changes [2]. Standards such as IEEE 802.11 (WiFi), IEEE 802.15.4 (Zigbee), and IEEE 802.16 (Wimax) implement some level of cognitive functionality.

There are a handful of hardware architectures for CR. Most of them are heterogeneous computing platforms consisting of FPGAs, DSP processors along with the analog front-end. The FPGA-based BEE2 platform has been used for implementing matched filter, energy detection, and cyclostationary detection for spectrum sensing [6]. Cooperative sensing using ultra-wide-band transmission has also been mapped onto this platform [7]. MAC layer-based optimization has been shown to be effective for CRs, and a SDR-based implementation is described in [5]. Cog-Net, a FPGA-based platform for CR, has been developed for the SDR modem, MAC, and network engines [17]. Finally, a comprehensive design that includes FPGA-based universal software radio peripheral, GNU radio, and Cell Broadband Engine has been developed in [10]. All these designs are flexible but have high power and are not suitable for mobile devices. The challenge is to design a CR architecture that achieves a balance between programmability and low power.

21.7 Conclusion

Wireless communications will soon be ubiquitous. Users will be able to access high-speed Internet with their mobile devices in every corner of the globe. To support this, mobile communication devices will have to be able to connect to a multitude of different wireless networks. SDR promises to deliver a cost-effective communication solution by supporting different wireless protocols in software. In this chapter, we presented low-power digital baseband processors for SDR. We analyzed the characteristics of the

signal processing kernels, and described the design rationales behind SODA and its commercial prototype, Ardbeg. Both are multiprocessor wide-SIMD architectures that exploit the data-level parallelism, the thread-level parallelism and the streaming nature of the inter-kernel computations. We also described the salient features of other existing baseband architectures that are either SIMD-based or reconfigurable tile-based. We hope that the lessons learned from designing the SDR architectures will accelerate the development of low-power architectures for mobile devices that support CR and other emerging wireless technologies.

References

AQ10

1. ARM Advanced SIMD Extension (NEON Technology). [online]: http://www.arm.com/products/CPUs/NEON.html
2. I.F. Akyildiz, W.-Y. Lee, M.C. Vuran, and S. Mohanty. Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 50(13): 2127–2159, September 2006.
3. N.L. Binkert, R.G. Dreslinski, L.R. Hsu, K.T. Lim, A.G. Saidi, and S.K. Reinhardt. The M5 simulator: Modeling networked systems. *IEEE Micro*, 26(1): 52–60, July/August 2006.
4. H. Bluethgen, C. Grassmann, W. Raab, U. Ramacher, and J. Hausner. A programmable baseband platform for software defined radio. In *Proceedings of the 2004 SDR Technical Conference and Product Exposition*, Phoenix, AZ, November 2004.
5. B. Bougard, S. Pollin, J. Craninckx, A. Bourdoux, L. Ven der Perre, and F. Catthoor. Green reconfigurable radio system. *IEEE Signal Processing Magazine*, 24(3): 90–101, May 2007.
6. D. Cabric. Addressing the feasibility of cognitive radio. *IEEE Signal Processing Magazine*, 25(6): 85–93, November 2008.
7. D. Cabric, I.D. O'Donnell, M.S.-W Chen, and R.W. Brodersen. Spectrum sharing radios. *IEEE Circuits and Systems Magazine*, 6(2): 30–45, 2006.
8. A. Cappelli, A. Lodi, M. Bocchi, C. Mucci, M. Innocenti, C. De Bartolomeis, L. Ciccarelli, R. Giansante, A. Deledda, F. Campi, M. Toma, and R. Guerrieri. XiSystem: A XiRisc-based SoC with a reconfigurable IO module. In *IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, pp. 196–593, February 2005.
9. J. Fridman and Z. Greenfield. The TigerSharc DSP architecture. *IEEE Micro*, 20(1): 66–76, January 2000.
10. F. Ge, Q. Chen, Y. Wang, C.W. Bostian, T.W. Rondeau, and B. Le. Cognitive radio: From spectrum sharing to adaptive learning and reconfiguration. In *IEEE Aerospace Conference*, Big Sky, MT, pp. 1–10, March 2008.
11. J. Glossner, D. Iancu, M. Moudgill, G. Nacer, S. Jinturkar, S. Stanley, and M. Schulte. The sandbridge SB3011 platform. *EURASIP Journal on Embedded Systems*, 2007(1): 16–16, 2007. AQ11
12. H. Holma and A. Toskala. *WCDMA for UMTS: Radio Access For Third Generation Mobile Communications*. John Wiley & Sons, New York, 2001.
13. S. Knowles. Ths SoC Future is Soft. In *IEE Cambridge Processor Seminar*, December 2005. [online]: http://www.iee-cambridge.org.uk/arc/seminar05/slides/SimonKnowles.pdf
14. H. Lee, Y. Lin, Y. Harel, M. Woh, S. Mahlke, T. Mudge, and K. Flautner. Software defined radio—A high performance embedded challenge. In *Proceedings of the 2005 International Conference on High Performance Embedded Architectures and Compilers (HiPEAC)*, Barcelona, Spain, pp. 6–26, November 2005.
15. Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner. SODA: A low-power architecture for software radio. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture*, Boston, MA, pp. 89–101, July 2006.

16. B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins. ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable Matrix. In *13th International Conference on Field-Programmable Logic and Applications*, Berlin, Germany, pp. 61–70, January 2003.
17. Z. Miljanic, I. Seskar, K. Le, and D. Raychaudhuri. The WINLAB network centric cognitive radio hardware platform WiNC2R. In *Cognitive Radio Oriented Wireless Networks and Communications*, Orlando, FL, pp. 155–160, August 2007.
18. A. Nilsson and D. Liu. Area efficient fully programmable baseband processors. In *International Symposium on Systems, Architectures, Modeling and Simulation (SAMOS)*, Greece, pp. 333–342, July 2007.
19. B. Plunkett and J. Watson. *Adapt2400 ACM Architecture Overview*, QuickSilver Technology Inc., San Jose, CA, January 2004.
20. U. Ramacher. Software-defined radio prospects for multistandard mobile phones. *Computer*, 40(10): 62–69, 2007.
21. G.K. Rauwerda, P.M. Heysters, and G.J.M. Smit. Towards software defined radios using coarse-grained reconfigurable hardware. *IEEE Transactions Very Large Scale Integration (VLSI) Systems*, 16(1): 3–13, January 2008.
22. K. van Berkel, F. Heinle, P.P.E. Meuwissen, K. Moerman, and M. Weiss. Vector processing as an enabler for Software-Defined Radio in handsets from 3G+WLAN onwards. *EURASIP Journal on Applied Signal Processing*, 2005(1): 2613–2625, January 2005.
23. M. Woh, Y. Lin, S. Seo, S. Mahlke, T. Mudge, C. Chakrabarti, R. Bruce, D. Kershaw, A. Reid, M. Wilder, and K. Flautner. From soda to scotch: The evolution of a wireless baseband processor. *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on microarcheture*, Lake Como, Italy, pp. 152–163, November 2008.

AUTHOR QUERIES

[AQ1] Please check the Chapter title for correctness.
[AQ2] Please provide affiliations.
[AQ3] Please check the inserted expansion for ASIC.
[AQ4] Please provide the expansion for the following abbreviations: GSM, DSP, SoCs, MUX, RTL, TSMC, ARM, LIW, NXP, CRC, RISC, RAM, ILP, SRAM, FPGAs, SOC, DVB-T/H, ICERA, GNU, ISA.
[AQ5] For MAC, both medium access control and multiply-accumulate have been used. I have left author's usage as it is. Please check.
[AQ6] Since Figure 21.5 is given in table format we have changed it to Table 21.1 and also renumbered the remaining figures. Please check.
[AQ7] Kindly provide expansion for LPF-Rx.
[AQ8] Kindly provide expansion for VLIW.
[AQ9] Please check the inserted expansion for Gops.
[AQ10] Since all references are cited in the text. We have changed bibliography to references. Please check.
[AQ11] Please check the page range for Reference 11.