Communication Processors

Sridhar Rajagopal

WiQuest Communications, Inc.
sridhar.rajagopal@wiquest.com

and

Joseph R. Cavallaro

Department of Electrical and Computer Engineering
Rice University
cavallar@rice.edu

Abstract

Communication processors are processors with specific optimizations to support communication systems. Communication processors exist in a wide variety of forms and can be categorized based on the communication system, such as wired or wireless and based on the layer in the communication system, such as the physical layer, the medium access control layer or the network layer. Communication processors can be further categorized based on the application, such as audio, video or data and the end system requiring the communication system such as a laptop, a cell phone or a personal computer. As communication systems have evolved over the years, there has been an increase in data rates, increase in algorithm complexity, need for flexibility to adapt to different protocols and environments, need to optimize over varying constraints such as area, power, performance and the need for supporting multiple interfaces, devices and applications. In this chapter, we present a brief outline of the different types of communication processors and the need and requirements of each of these processors. We focus on the challenges in the physical layer design of communication processors with the evolution of communication systems.

Key words: wireless communications, application specific instruction processors, physical layer processors, MAC processors, network processors, embedded systems

1 Introduction

In this chapter, we define the term 'communications processor' as a device in a communication system that carries out operations on data in terms of either modifying or processing the data or transporting the data to other parts of the system. A communications processor has certain optimizations built inside its hardware and/or software that enables it to perform its task in an efficient manner.

The processing in a communications system is performed in multiple layers, according to the Open Systems Interconnection (OSI) model. When the communication is via a network of intermediate systems, only the lower three layers of the OSI protocols are used in the intermediate systems. In this chapter, we will focus on these lower 3 layers of the OSI model, shown in Figure 1. The bottom-most layer is called the physical layer (or layer 1 in the OSI model). This layer serializes the data to be transferred into bits and sends it across a communications circuit to the destination. The form of communication can be wired using a cable or wireless using a radio device. In a wireless system, the physical layer is also called as the baseband layer as all operations there are performed at the baseband frequency. Above this layer is the data link layer or more commonly known as the medium access control layer (MAC). The MAC layer is one of the two sub-layers in the data link layer of the OSI model. The MAC layer manages and maintains communication between multiple communication devices by coordinating access to a shared medium and utilizing protocols that enhance communications over that medium. The third layer in the OSI model is the network layer. The network layer knows the address of the neighboring nodes in the network, packages output with the correct network address information, selects routes and Quality of Service, and recognizes and forwards to the Transport layer incoming messages for local host domain. We define communication processors as processors having optimizations for the lower 3 layers of the OSI model. Depending on which layer has the most optimizations, communications processors are further classified into physical layer (or baseband) processors, medium access control processors or network processors. However, we will focus this chapter on the challenges in the digital baseband part of the physical layer processor design for wireless communication systems.

1.1 Evolution of wireless communication systems

Over the past several years, communication systems have evolved from low data-rate systems for voice and data (with data rates in several Kbps, such as dial-up modems, cellular systems, 802.11b local area networks etc..) to high data-rate systems supporting multimedia and video applications with data rates in several Mbps and going towards Gbps, such as DSL, cable modems, 802.11n local area networks (LANs) and ultra-wideband personal area networks (PANs) [1]. The first generation systems (1G) came in the 80's mostly for cellular analog voice using AMPS (Advanced Mobile Phone Service). This standard evolved into the second generation standard (2G) in the 90's to support digital voice and low bit rate data services. An example of such a cellular system is IS-54. At the same time, wireless local area networks started becoming in service starting at 1 Mbps for 802.11b standards and extending to 11 Mbps close to the year 2000. Local area networks are able to support higher data rates than cellular networks as the data is sent over a very short distance compared to cellular networks. In the current generation of the standards (3G), cellular services have progressed to higher data rates in terms of 100's of Kbps to support voice, data and multimedia and wireless LANs have evolved to 802.11a and 802.11g to support data rates around 100 Mbps. In the future, for the fourth generation systems (4G), the data rates are expected to continue to increase and will be able to provide IP-based services along with QoS (Quality of Service) [2]. Table 1 presents the evolution of wireless communication systems as they have gone from 1G to 4G systems. The data rates in the table

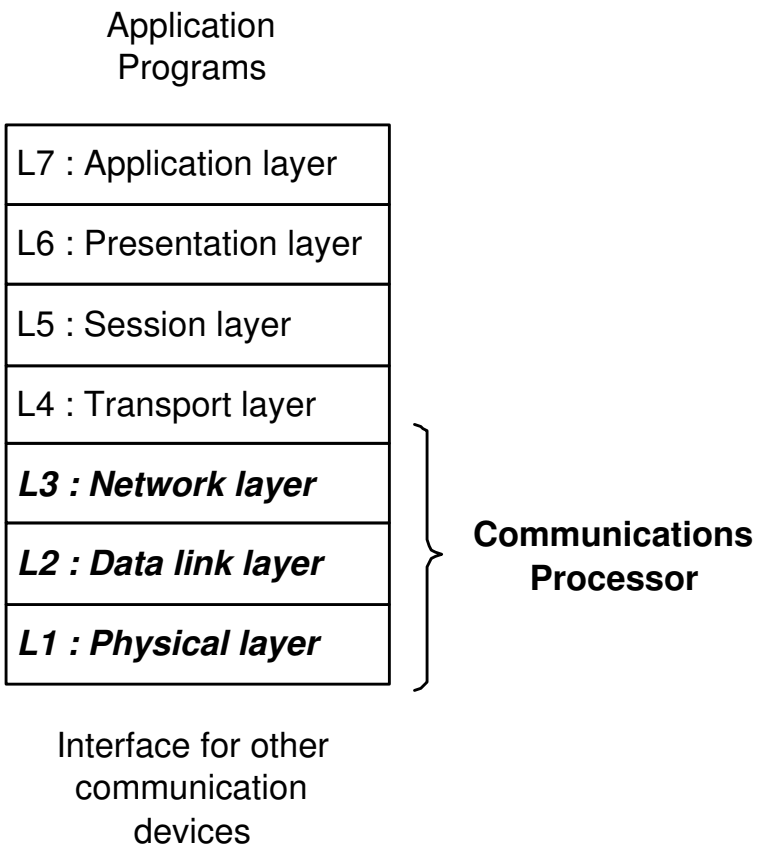

Fig. 1. Layers in a OSI model. The communication processors defined in this chapter are processors that have specific optimizations for the lower 3 layers.

Generation	Year	Function	Data Rates
1G	1980−1990	Analog Voice	Kbps
2G	1990−2000	Voice + low-rate data	10 Kbps − 10 Mbps
3G	2000−2010	Voice + data + multimedia	100 Kbps − 100 Mbps
4G	2010−2020	Voice + data + multimedia + QoS + IP	10 Mbps − Gbps

Table 1
Evolution of communication systems.

are shown to be variable since the data rates for wireless LANs are 2 orders-of-magnitude larger than data rates for cellular communication systems.

2 Challenges for communication processors

This evolution of communication systems has involved radical changes in processor designs for these systems for multiple reasons. First, the increase in data rates has come at the cost of

increased complexity in the system design. Secondly, the performance of communication systems have been consistently increasing with communication system designers coming up with sophisticated signal processing algorithms that enhance the performance of the system at the expense of increased computational complexity. Flexibility is also an important emerging characteristic needed in communication processors with the need to support multiple protocols and environments. Also, newer applications have become more complex and need to be backward-compatible with existing systems. As the number of standards and protocols increase, there is an increasing demand for new standards to be spectrum-efficient, avoid interference to other systems and also, be able to mitigate interference from other systems. The flexibility needed in the baseband and radio and regulatory requirements of spectrum and transmit power also make the design for testing these processors difficult. The interaction and integration between different layers of the communication system also presents interesting challenges since the communication layer is more signal processing based while the MAC layer is more data processing based. Finally, the range of consumer applications for communication systems has increased from small low-cost devices, such as RFID tags, to cellular phones, PDAs to laptops, personal computers to high-end network servers. Processors for different applications have different optimization constraints such as the workload characteristics, cost, power, area and data rate and require significant trade-off analysis. The above changes puts additional constraints on the processor design for communication systems.

2.1 Increasing Data Rates

Figure 2 shows the increase in data rates provided by communication systems over time. The figure shows that over the past decade, communication systems have had a 1000X increase in data rate requirements. Systems such as wireless LANs and PANs have gone from 1 Mbps systems such as 802.11b and Bluetooth to 100+ Mbps 802.11a LANs to now Gbps systems being proposed for ultra-wideband personal area networks. The same has been true even for wired communication systems, going from 10 Mbps ethernet cards to now Gbps ethernet systems. The increase in processor clock frequencies have not even kept up with the increase in raw data rate requirements. During the same period, the processor clock frequencies have only gone up by 1 order of magnitude. Also, applications such as multimedia are demanding more compute resources and more memory than previous processors. This implies that technology advances are insufficient to even meet the increase in raw data rate requirements and further architecture innovations such as exploiting parallelism, pipelining and algorithm complexity reduction are needed to meet the data rate requirements.

2.2 Increasing Algorithm Complexity

While the data rate requirements of communication processors are increasing, the processor design is exacerbated by the introduction of more sophisticated algorithms which give significant performance improvements for communication systems. Figure 3 shows the increase in compu-

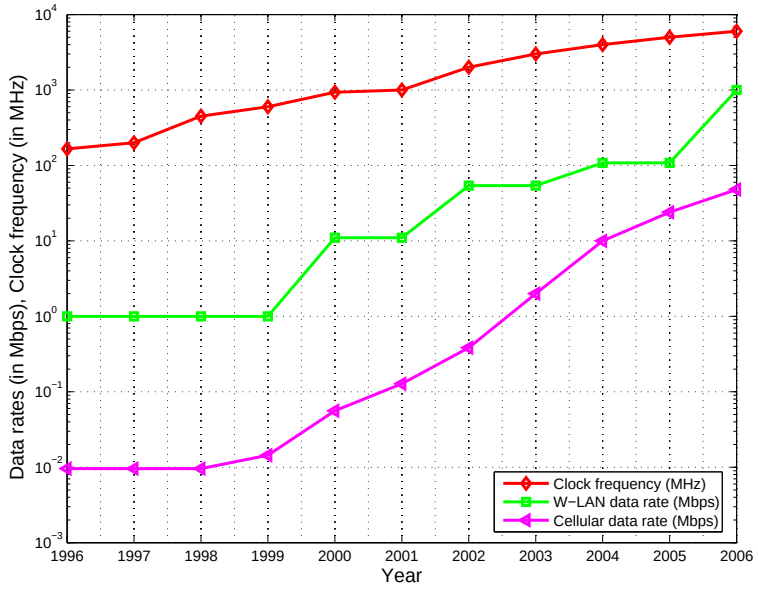

Fig. 2. Increase in data rates for communication systems. The data rates in communication systems are increasing at a much higher rate than clock frequencies, necessitating new processor designs for communication systems

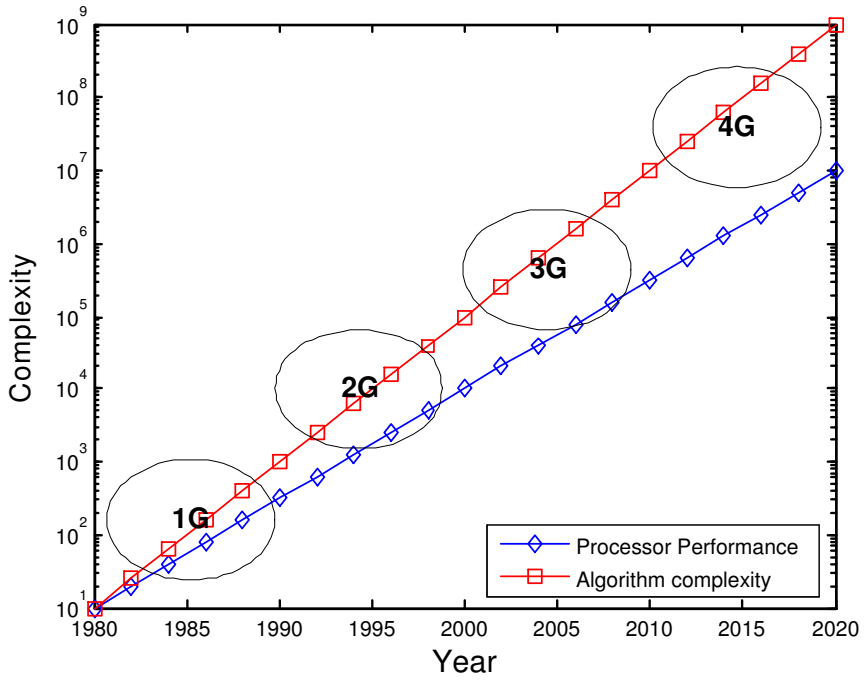

Fig. 3. Algorithm complexity increasing faster than technology advances. Source: Rabaey [3]

tational complexity as standards have progressed from first generation (1G) to second and third generations. The figure shows that even if the data rates are assumed constant, the increase in algorithmic complexity cannot be met up with just increase in technology.

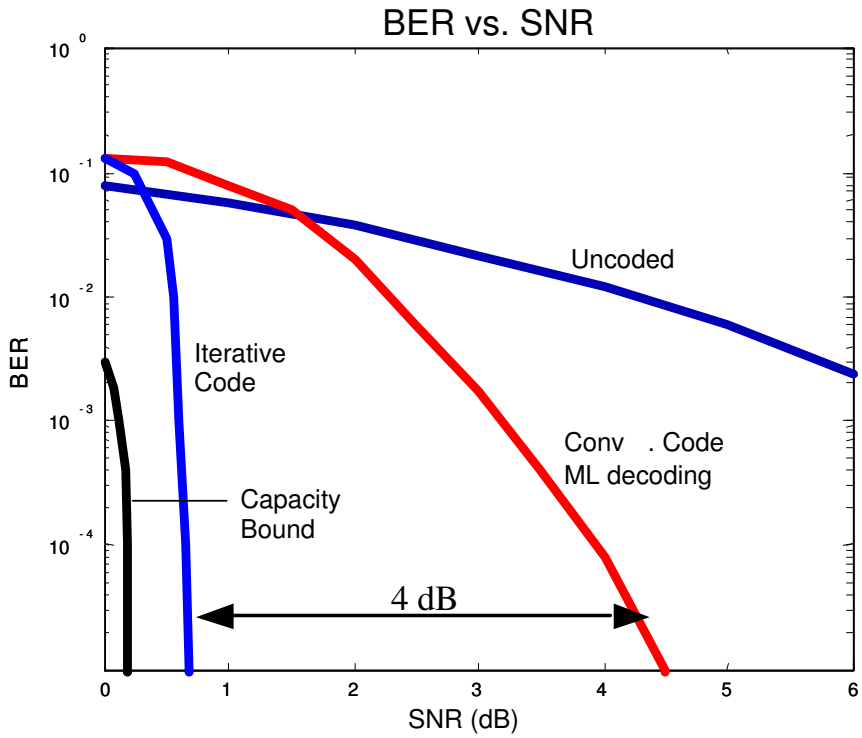

Fig. 4. Decoder Performance with advanced coding schemes. Source: Yeo, 2003 [6]

In order to give an example, we consider decoding of error-control codes at the receiver of a communication processor. Figure 4 shows the benefits of coding in a communication system by reducing the bit error rate (BER) at a given signal-to-noise ratio. We can see that advanced coding schemes such as Low Density Parity Check (LDPC) codes [4] and Turbo codes [5], which are iterative decoders can give 4 dB benefits over conventional convolutional decoders. Such advanced coding schemes are being proposed and implemented in standards such as HSDPA, VDSL, gigabit ethernet, digital video broadcast, Wi-Fi etc.. This is however, at a significant increase in computational complexity. Figure 5 shows the increased complexity of some of the advanced coding schemes [6]. It can be seen that the iterative decoders have $3-5$ orders of magnitude increase in computational complexity over convolutional decoders. Thus, in order to implement these algorithms, reduced complexity versions of these algorithms should be investigated for communication processors that allow simpler hardware designs with significant parallelism without significant loss in performance. An example of such a design is presented in [7].

2.3 Flexibility

As communication systems evolve over time, there is a greater need for communication processors to be increasingly flexible. Communication systems are being designed to support a variable number of parameters such as variable coding rates, variable modulation modes, variable frequency bands etc. This flexibility allows the communication system to adapt itself better

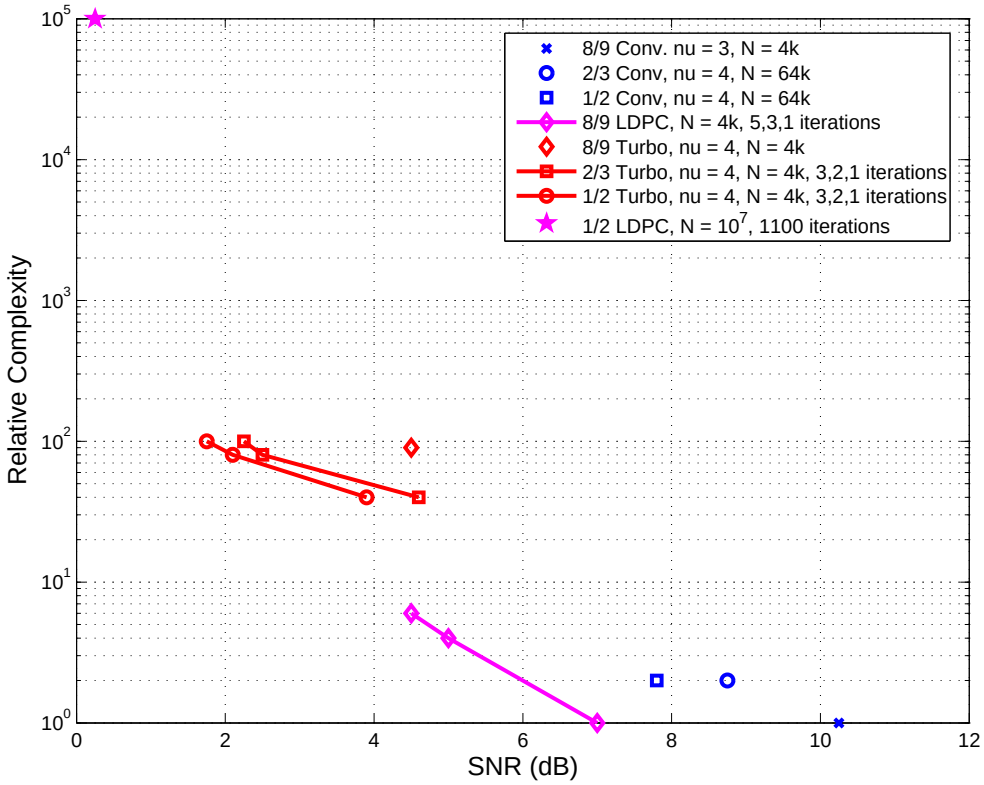

Fig. 5. Decoder Complexity for various types of coding schemes. Source: Yeo, 2003 [6]

to the environment in order to maximize data rates over the channel and/or minimize power. For example, Figure 6 shows base-station computational requirements and the flexibility needed to support variable number of users at variable constraint lengths [8]. The figure also shows an example of say a 2G station at 16 Kbps/user supporting only voice and a 3G base-station at 128 Kbps/user supporting voice, data and multimedia. A 3G base-station processor now has to be backwards-compatible to a 2G base-station processor as well and hence, has to support both the standards as well as adapt its compute resources to save power when the processing requirements are lower. The amount of flexibility provided in communication processors can make the design for test for these systems extremely challenging due to the large number of parameters, algorithms and radio interfaces that need to be tested.

Along with the support for variable standards and protocols, researchers are also investigating the design of a single communication processor that is able to seamlessly switch between different standards, depending on the availability and cost of that standard. The RENE (Rice Everywhere NEtwork) project [9] demonstrates the design of a multi-tier network interface card with a communication processor that supports outdoor cellular (CDMA) and indoor wireless (LAN) and seamlessly changes over the network when the user moves from an office environment with wireless LAN into an outdoor environment using cellular services. Figure 7 shows the design of the wireless multi-tier network interface (mNIC) card concept at Rice University.

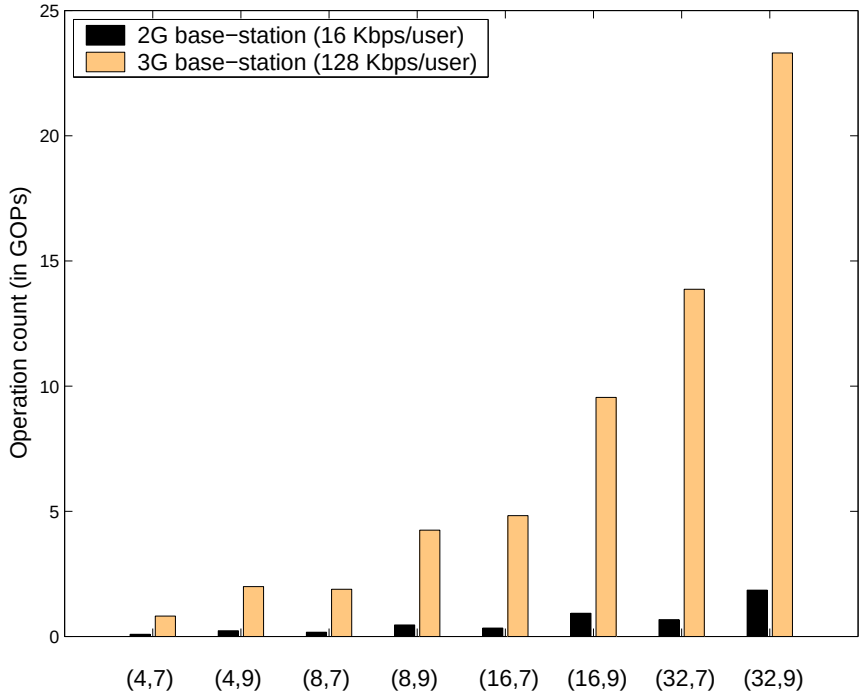

Fig. 6. Flexibility needed to support various users, rates (for example) and backwards-compatibility to standards. Source: Rajagopal [8]

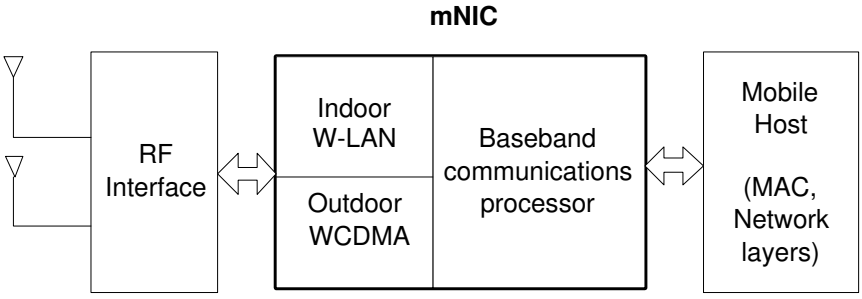

Fig. 7. Multi-tier network interface card concept. Source: Aazhang [9]

The increasing need for greater flexibility to standards and environments. and for power savings with variations in computational requirements is a significant challenge for the design of next-generation communication processors.

2.4 Spectrum Issues

The wireless spectrum is a scarce resource and is regulated by multiple agencies world-wide. As new standards evolve, they have to co-exist with spectrums that are already allocated for existing standards. The regulatory bodies, such as FCC (see www.fcc.gov), demand that new standards meet certain limitations on transmit power and interference avoidance to make sure that the existing services are not degraded by the new standard. Also, due to the a plethora of wireless

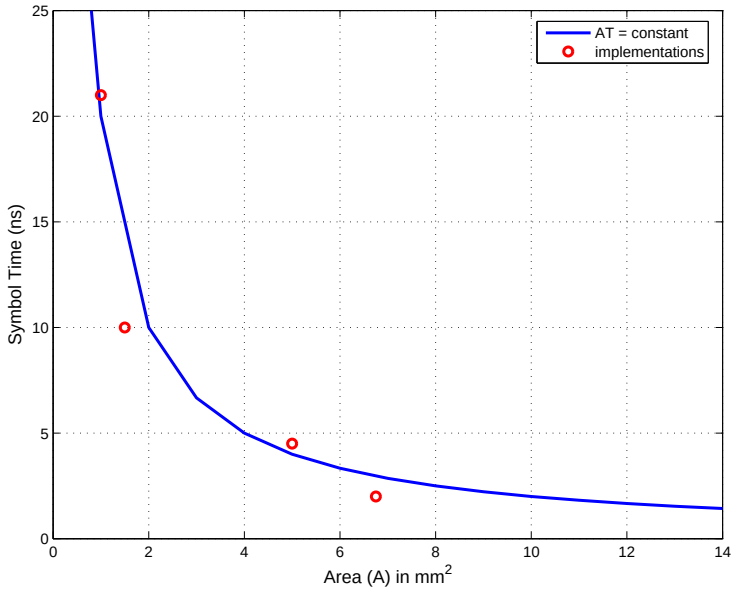

Fig. 8. Normalized Area-Time Efficiency for Viterbi decoding. Source: Gemmeke,JSSC,2002 [11]

standards in the $1-5$ GHz wireless spectrum, new standards are being forced to look at much higher RF frequencies, which make the design of radios more challenging as well as increase the need for transmit power due to larger attenuation at higher frequencies. Newer standards also need to have interference detection and mitigation techniques to help co-existence with existing standards. This involves challenges at the radio level, such as to transmit at different frequencies to avoid interference and gives rise to the need for software defined radios [10]. The lack of a worldwide regulatory standard also implies devices must be programmed differently to meet regulatory specifications in different countries, increasing the complexity of the design.

2.5 Area, Time, Power Tradeoffs

The design of communication processors is further complicated by the nature of optimizations needed for the application and for the market segment. A mobile market segment may place greater emphasis on cost (area) and power while a high-data rate market segment may place a greater focus on performance. Thus, even after new algorithms are designed and computationally efficient versions of the algorithms have been developed, there are tradeoffs between area-time and power consumptions for the implementation of the algorithm on the communication processor. There are also other parameters that need to be traded off such as the technology process nodes (say 0.18 vs 0.13 vs 0.09 μm CMOS process) and voltage and clock frequencies. For example, the area-time tradeoffs for Viterbi decoding are shown in Figure 8. The curve shows that the area needed for the Viterbi decoder can be traded-off at the cost of increasing the execution time for the Viterbi decoder.

In programmable processors, the number of functional units and the clock frequency can be ad-

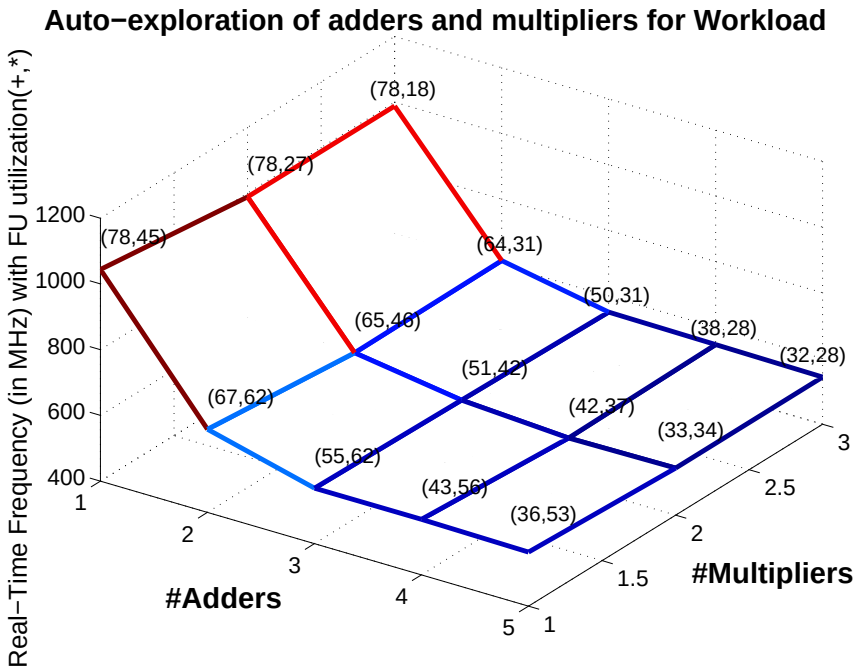

Fig. 9. Number of adders and multipliers to meet real-time requirements in a programmable processor. Source: Rajagopal [12]

justed to meet real-time requirements for an application. An example of this is shown in Figure 9 [12]. The figure shows that as the number of adders and multipliers in a programmable processor are increased, the clock frequency needed to meet real-time for an application decreases until a certain point, at which there are no more operations to be scheduled on the additional adders and multipliers in the processor. The numbers on the graph indicate the functional unit utilization of the adders and multipliers in the processor.

2.6 Interaction between multiple layers

The interaction between the different layers in a communications system also presents challenges to the processor design. As will be shown in the following sections, the characteristics of the physical layer in a communication system are completely different than the characteristics of the MAC or network layer. The physical layer of a communication system consists more of signal processing algorithms that work on estimation of the channel, detection of the received bits and decoding of the data and require computational resources. The MAC and network layers are more data-flow oriented and have more control and data grouping operations. The combination of these two diverse requirements make the task of design of a single communications processor that does both the PHY as well as the MAC difficult and hence, typically these layers are implemented as separate processors.

3 Physical layer or Baseband processors

The physical layer of wireless communication systems present more challenges to the communication processor design than wired communication systems. The nature of the wireless channel implies the need for sophisticated algorithms on the receiver in order to receive and decode the data. While there are challenges both in the analog radio and the digital baseband of the physical layer, we focus on the digital baseband part of the physical layer of wireless communication systems in this article.

3.1 Characteristics of baseband communication algorithms

Algorithms for communication systems in the physical layer process signals for transmission and reception of analog signals over the wireless or even wired link. Hence, most algorithms implemented on communication processors are signal-processing algorithms and show certain characteristics that can be exploited in the design of communication processors.

(1) Signal processing algorithms are typically compute-bound. This implies that the bottleneck in the processing are the computations (as opposed to memory) and the architectures require significant number of adders and multipliers.
(2) Communication processors require very low fixed-point precision in computations. At the transmitter, the inputs are typically sent in as bits. At the receiver, the ADCs reduce the dynamic range of the input signal by quantizing the signal. Quantization in communication processors is acceptable because the quantization errors are typically small compared to the noise added through the wireless channel. This is very useful in designing low power and high speed arithmetic and keeping the size of memory requirements small in communication processors.
(3) Communication algorithms exhibit significant amounts of data parallelism and show regular patterns in computation that can be exploited for hardware design.
(4) Communication algorithms have a streaming dataflow in a producer-consumer fashion between blocks with very little data re-use. This can be used to avoid storage of intermediate values and also eliminate hardware in processors such as caches that try to exploit temporal re-use.

Figure 10 shows a typical transmitter in a communications processor. The transmitter in the physical layer of a communication system is typically much simpler compared to the receiver. The transmitter operations typically consist of taking the data from the MAC layer and then scrambling it to make it look sufficiently random, encoding it for error protection, modulating it on certain frequency tones and then pre-compensating it for any RF impairments or distortions.

Figure 11 shows a typical receiver in a communications processor. The receiver estimates the channel in order to compensate for it, and then demodulates the transmitted data and then decodes the data to correct for any errors during transmission. Although not shown in the figure, many other impairments in the channel and the radio, such as fading, interference, I/Q imbal-

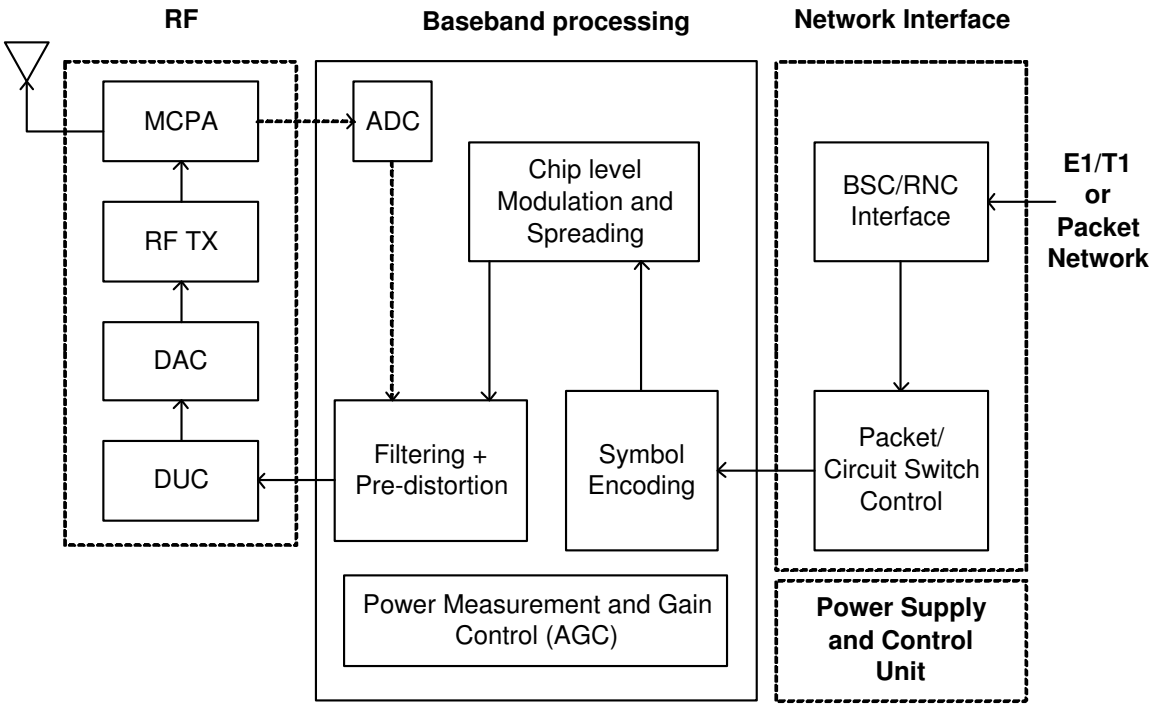

Fig. 10. Typical operations at a transmitter of a baseband processor. Source: Texas Instruments

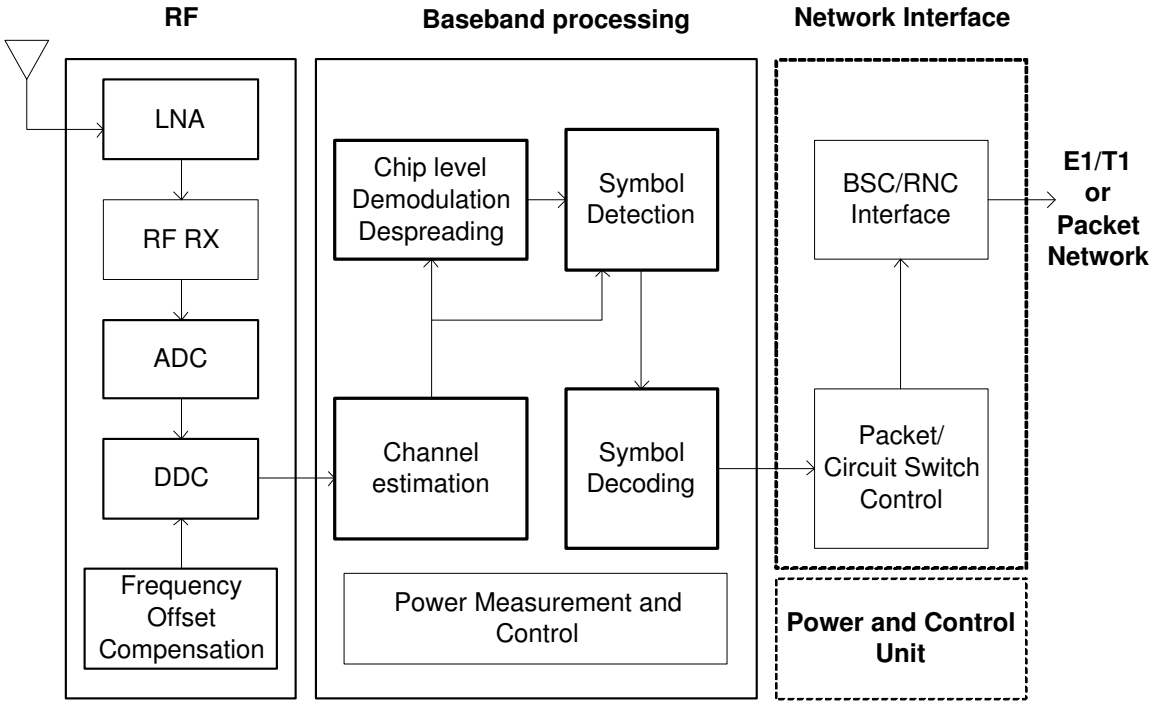

Fig. 11. Typical operations at the receiver of a baseband processor. Source: Texas Instruments

ance, frequency offsets and phase offsets are also corrected at the receiver. The algorithms used at the receiver involve sophisticated signal processing and in general, have increased in complexity over time, while providing more reliable and stable communication systems.

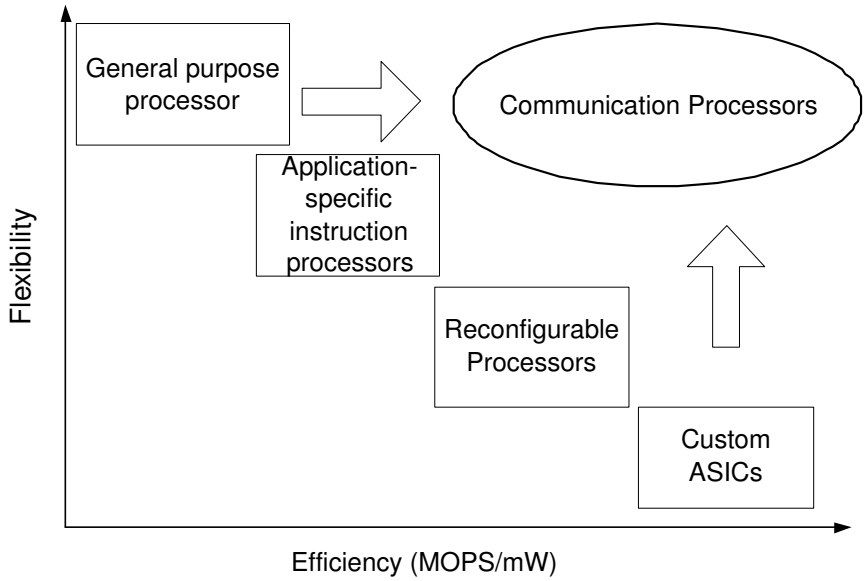

Fig. 12. Flexibility of different architectures

3.2 Architecture designs

A wide range of architectures can be used to design a communications processor. As shown in Figure 12, these architectures trade-off flexibility with performance/power and are dependent on the application. A custom ASIC has the best efficiency in terms of data rate at unit power consumption – but at the same time it has the least amount of flexibility [13]. A fully programmable processor on the other hand, is extremely flexible but is not area/power/throughput efficient. We discuss the trade-offs among the different types of architectures in order to use them as communication processors.

3.2.1 Custom ASICs

Custom ASICs are the solution for communication processors that provide the highest efficiency and the lowest cost in terms of chip area and price. This, however, comes at the expense of a fairly large design and test time and lack of flexibility and scalability with changes in standards and protocols. A partial amount of flexibility is provided as register settings which are then controlled by the MAC or higher layers in firmware (software). For example, the data rate to be used for transmission can be programmed into a register in the custom ASIC from the MAC and that can be used to set the appropriate controls in the processor.

3.2.2 Reconfigurable processors

Reconfigurable processors are a relatively new addition to the area of communication processors. Reconfigurable processors typically consist of a CISC type instruction set processor with a reconfigurable fabric attached to the processor core. This reconfigurable fabric is typically

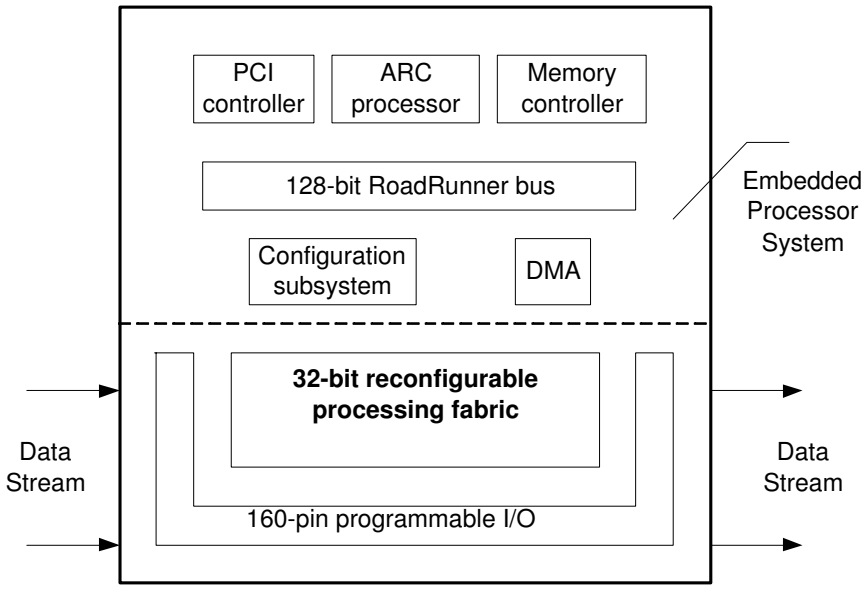

Fig. 13. Reconfigurable communications processors. Source: Chameleon Systems [14]

used to run complex signal processing algorithms that have sufficient parallelism and need a large number of adders and multipliers. The benefits of the reconfigurable fabric compared to FPGAs is that the reconfiguration can be done dynamically during run-time. Figure 13 shows an example of the Chameleon reconfigurable communications processor [14].

The reconfigurable fabric and the instruction set computing seek to provide the flexibility needed for communications processor while providing the dedicated logic in the reconfiguration fabric for efficient computing which can be reprogrammed dynamically. One of the major disadvantages of reconfigurable processors is that the software tools and compilers have not been progressed to a state where performance/power benefits are easily visible along with the ease of programming the processor. The Chameleon reconfigurable processor is no longer an active product. However, several researchers in academia, such as GARP at Berkeley [15], RAW at MIT [16], Stallion at Virginia Tech [17] and industry such as PACT [18] are still pursuing this promising architecture for communication processors.

3.2.3 Application-specific instruction processors

Application-specific instruction processors (ASIPs) are processors with an instruction set for programmability and with customized hardware tailored for a given application [19]. The programmability of these processors followed by the customization for a particular application to meet data rate and power requirements make ASIPs a viable candidate for communication processors [20, 21].

A DSP is an example of such an application-specific instruction processor with specific optimizations to support signal processing operations. Since standards are typically driven by what is possible from an ASIC implementation feasibility for cost, performance and power, it is difficult for a programmable architecture to compete with a fully custom based ASIC design for

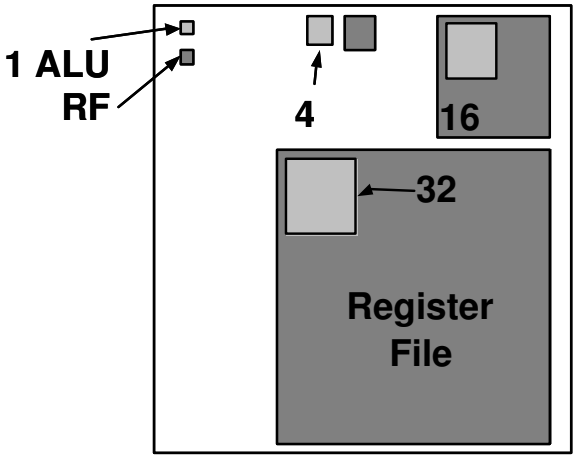

Fig. 14. Register File expansion with increasing number of functional units in a processor. Source: Rixner [26]

wireless communications. DSPs fail to meet real-time requirements for implementing sophisticated algorithms due to the lack of sufficient functional units. However, it is not simple to just increase the number of adders and multipliers in a DSP. Traditional single processor DSP architectures such as the C64x DSP by Texas Instruments [22] employ VLIW architectures and exploit instruction level parallelism (ILP) and subword parallelism. Such single processors DSPs can only have limited arithmetic units (less than 10) and cannot directly extend their architectures to 100's of arithmetic units. This is because, as the the number of arithmetic units increases in an architecture, the size of the register files and the port interconnections start dominating the architecture [23, 24]. This growth is shown as a cartoon in Figure 14. While the use of distributed register files may alleviate the register file explosion at the cost of increased penalty in register allocation [23], there is an associated cost in exploiting ILP due to limited size of register files, dependencies in the computations and the register and functional unit allocation and utilization efficiency of the compiler. It has been shown that even with extremely good techniques, it is very difficult to exploit ILP beyond 5 [25]. The large number of ALUs also make the task of compiling and scheduling algorithms on the ALUs and keeping all the ALUs busy difficult.

Another popular approach to designing communication processors is to use a DSP with co-processors [27–29]. The co-processors are still needed to perform more sophisticated operations that cannot be done real-time on the DSP due to the lack of sufficient adders and multipliers. Co-processor support in a DSP can be both tightly-coupled and loosely-coupled [29]. In a tightly-coupled coprocessor (TCC) approach, the co-processor interfaces directly to the DSP core and has access for specific registers in the DSP core. The TCC approach is used for algorithms that work with small datasets and require only a few instruction cycles to complete. The DSP processor freezes when the co-processor is being utilized since the DSP will have to interrupt the co-processor immediately in the next few cycles. In time, the TCC is integrated into the DSP core with a specific instruction or is replaced with code in a faster or lower-power DSP. An example of such a TCC approach would be the implementation of a Galois field bit manipulation that may not be part of the DSP instruction set [29]. The loosely coupled approach (LCC) is used for algorithms that work with large datasets and require significant amount of cycles to complete

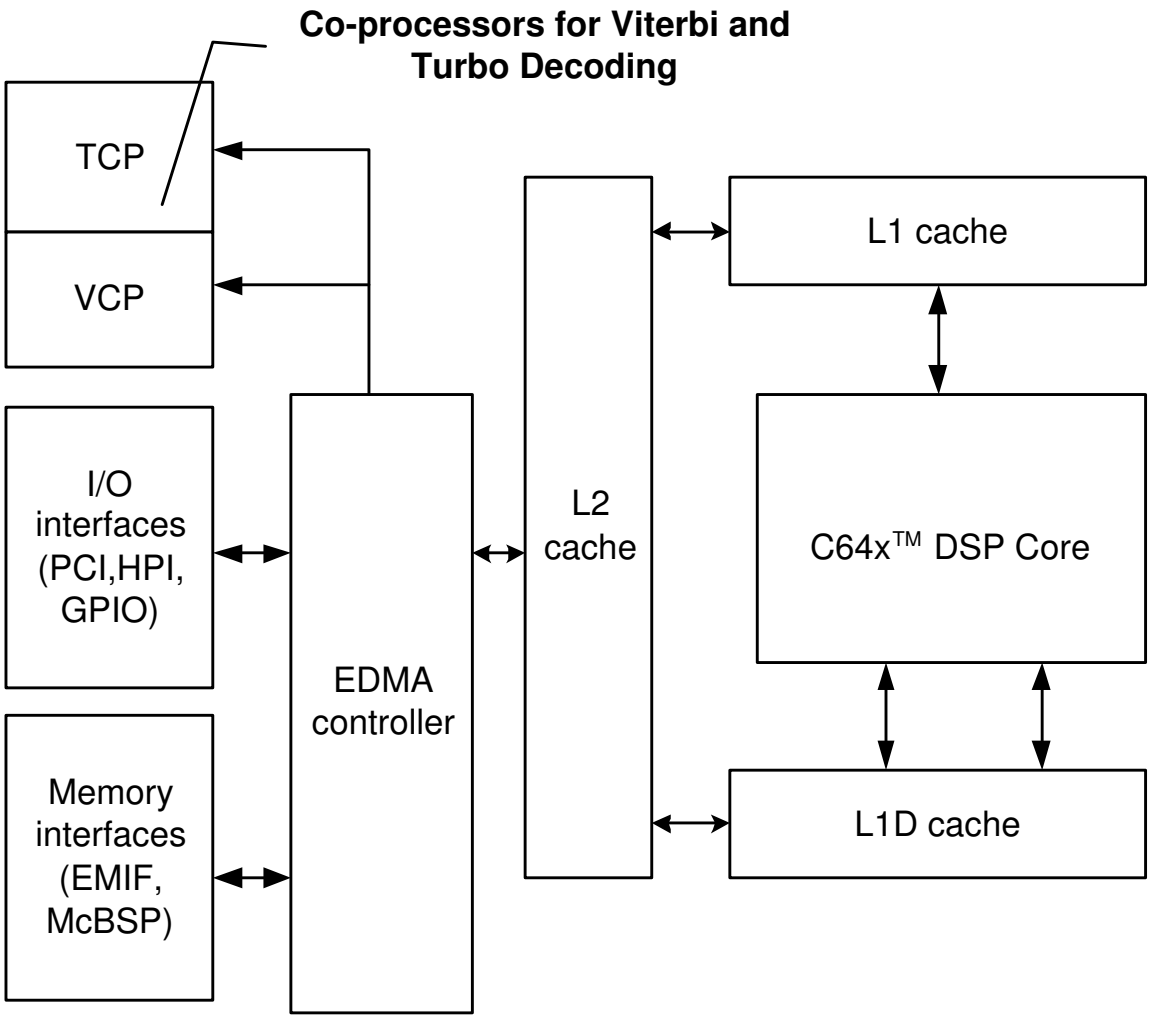

Fig. 15. DSP with co-processors for decoding. Source: Agarwala, JSSC 2002 (Texas Instruments) [22]

without interruption from the DSP. The LCC approach allows the DSP and co-processor to execute in parallel. The co-processors are loaded with the parameters and data and are initiated through application-specific instructions. The coprocessors sit on an external bus and do not interface directly to the DSP core, allowing the DSP core to execute in parallel. Figure 15 shows an example of the TMS320C6416 processor from Texas Instruments which has Viterbi and Turbo co-processors for decoding [22] using the loosely coupled coprocessor approach. The DSP provides the flexibility needed for applications and the co-processors provide the compute resources for more sophisticated computations that are unable to be met on the DSP.

3.2.4 Programmable processors

In order to be precise with definitions, in this subsection, we consider programmable processors as processors that do not have an application-specific optimization or instruction set. For example, DSPs without co-processors are considered in this subsection as programmable processors.

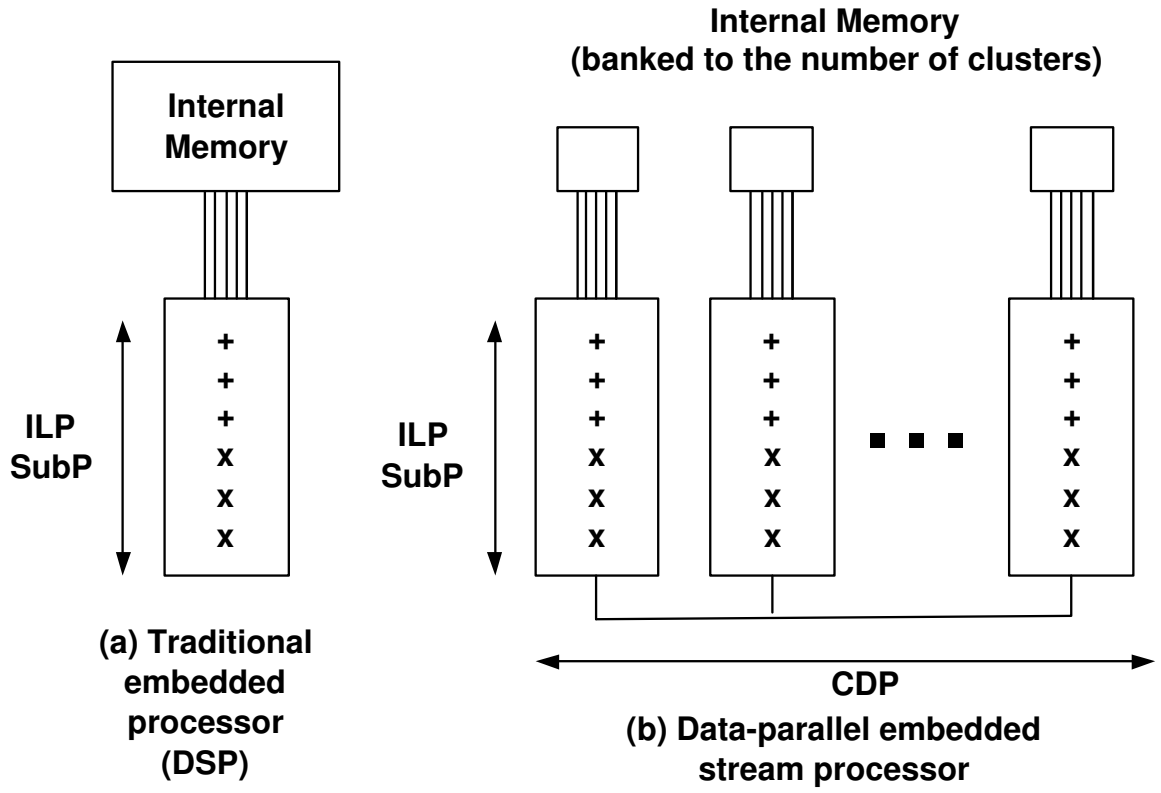

Fig. 16. DSP and stream processors. Source: Rajagopal [12]

Stream processors are programmable processors that have optimizations for media and signal processing. They are able to provide 100's of ALUs in a processor by arranging the ALUs into groups of clusters and exploiting data parallelism across clusters. Stream processors are thus able to support GOPs of computation in the processor. Figure 16 shows the distinction between DSPs and stream processors. While typical DSPs exploit just instruction level parallelism (ILP) and sub-word parallelism (SubP), stream processors also exploit data-parallelism across clusters (CDP) to provide the needed computational horsepower.

Streams are stored in a stream register file, which can efficiently transfer data to and from a set of local register files between major computations. Local register files (LRFs), co-located with the arithmetic units inside the clusters, directly feed those units with their operands. Truly global data, data that is persistent throughout the application, is stored off-chip only when necessary. These three explicit levels of storage form an efficient communication structure to keep hundreds of arithmetic units efficiently fed with data. The Imagine stream processor developed at Stanford is the first implementation of such a stream processor [30]. Figure 17 shows the architecture of a stream processor, with C arithmetic clusters. Operations in a stream processor all consume and/or produce streams which are stored in the centrally located stream register file (SRF). The two major stream instructions are memory transfers and kernel operations. A stream memory transfer either loads an entire stream into the SRF from external memory or stores an entire stream from the SRF to external memory. Multiple stream memory transfers can occur simultaneously, as hardware resources allow. A kernel operation performs a computation on a set of input streams to produce a set of output streams. Kernel operations are performed within a

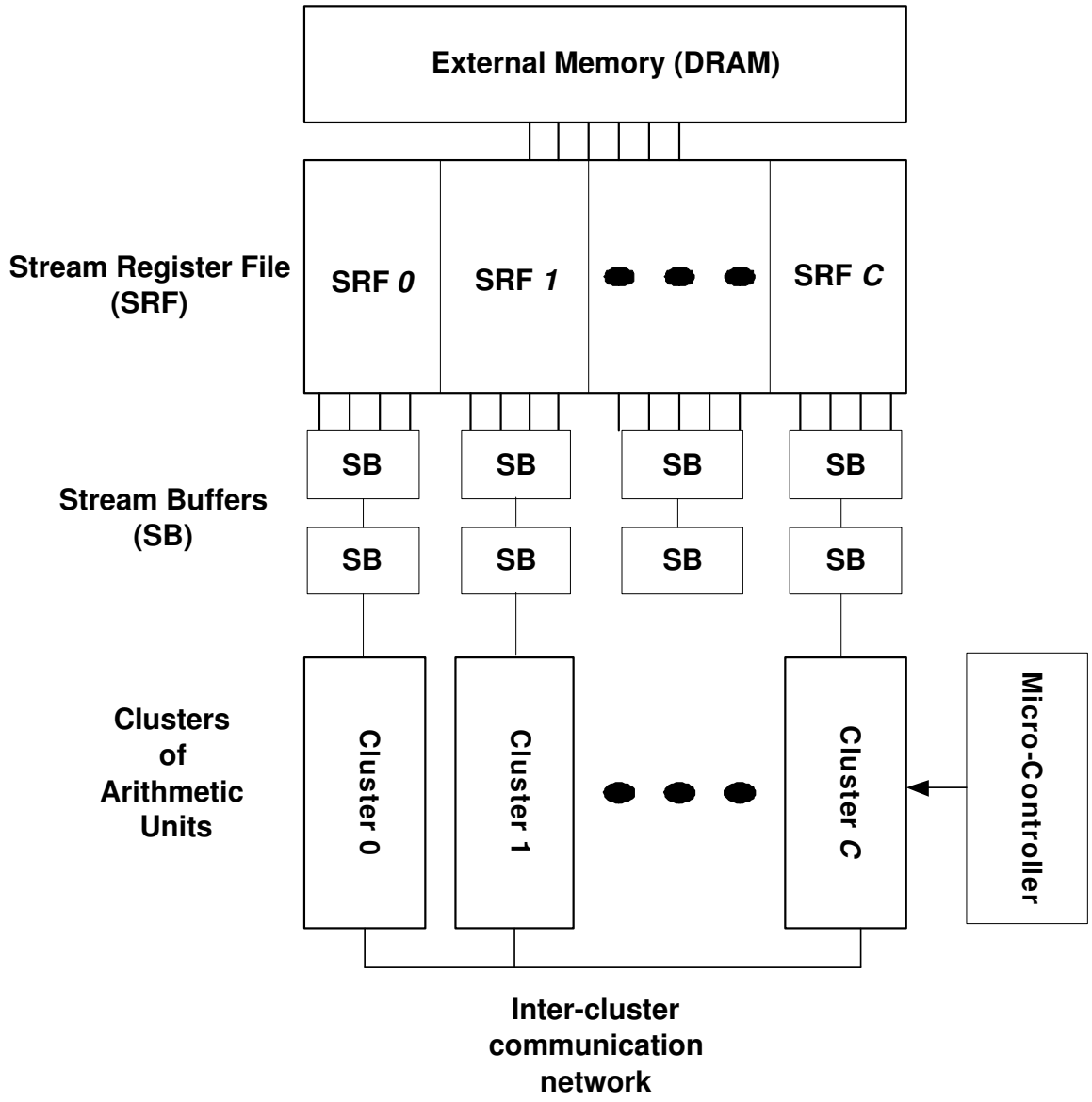

Fig. 17. Stream Processor Architecture. Source: Rixner [26]

data parallel array of arithmetic clusters. Each cluster performs the same sequence of operations on independent stream elements. The stream buffers (SBs) allow the single port into the SRF array (limited for area/power/delay reasons) to be time-multiplexed among all the interfaces to the SRF, making it appear that there are many logical ports into the array. The stream buffers (SBs) also act as prefetch buffers and prefetch the data for kernel operations. Both the SRF and stream buffers are banked to match the number of clusters. Hence, kernels that need to access data in other SRF banks need to use the inter-cluster communication network for communicating data between the clusters.

The similarity between stream computations and communication processing in the physical layer makes stream-based processors an attractive architecture candidate for communication processors [8, 31].

4 Medium Access Control (MAC) and Network Processors

While the focus of this chapter is on the physical layer of the communication processor, the MAC and network layers have a strong interaction with the physical layer, especially in wireless networks. In this section, we briefly discuss the challenges and functionality needed in processors for MAC and network layers [32].

MACs for wireless networks involve greater challenges than MACs for wired networks. The wireless channel necessitates the need for re-transmissions when the received data is not decoded correctly in the physical layer. Wireless MACs also need to send out beacons to notify the access point (AP) that there is an active device present on the network.

Typical functions of a wireless MAC include

(1) Transmissions of beacons in regular intervals to indicate the presence of the device on the network
(2) Buffering frames of data that are received from the physical layer and sending requests for re-transmissions for lost frames
(3) Monitoring radio channels for signals, noise and interference
(4) Monitoring presence of other devices on the network
(5) Encryption of data using AES/DES in order to provide security over the wireless channel
(6) Rate control of the physical layer to decide what data rates should be used for transmission of the data

From the above, it can be seen that the MAC layer typically involves significant data management and processing. MACs are typically implemented as a combination of a RISC core which provides the control to different parts of the processor and dedicated logic for parts such as encryption for security and host interfaces.

Some of the functions of the network layer can be implemented on the MAC layer and vice-versa, depending on the actual protocol and application used. Typical functions at the network layer include:

(1) Pattern matching and lookup. This involves matching the IP address and TCP port
(2) Computation of checksum to see if the frame is valid and any additional encryption and decryption
(3) Data manipulation involving extracting and insertion of fields in the IP header and also, fragmentation and re-assembly of packets
(4) Queue management for low priority and high priority traffic for QoS
(5) Control processing for updating routing tables and timers to check for re-transmissions and backoff etc.

5 Conclusions

Communication processor designs need to be re-evaluated significantly for future systems as the rate of technology increase is unable to catch up with the increase in data rates and the increase in algorithm complexity. The need for greater flexibility in order to support multiple protocols and be backwards-compatible only exacerbates the design problem due to the need to design programmable solutions that can provide high throughput and meet real-time requirements while being area and power efficient. The stringent regulatory requirements on spectrum, transmit power and interference mitigation makes the design of the radio difficult while the complexity, diverse processing characteristics and interaction between the physical layers and the higher layers complicates the design of the digital part of the communication processor. Various tradeoffs can be made in communication processors to optimize throughput vs. area vs. power vs. cost and the decisions are dependent on the actual application under consideration. We present a detailed look at the challenges involved in designing these processors and present sample communication processor architectures that are being considered for communication processors in the future.

Acknowledgements

Sridhar Rajagopal and Joseph Cavallaro were supported in part by Nokia Corporation, Texas Instruments, Inc., and by NSF under grants EIA-0224458, and EIA-0321266.

References

[1] T. Ojanpera and R. Prasad, editors. *Wideband CDMA for Third Generation Mobile Communications*. Artech House Publishers, 1998.

[2] H. Honkasalo, K. Pehkonen, M. T. Niemi, and A. T. Leino. WCDMA and WLAN for 3G and beyond. *IEEE Wireless Communications*, 9(2):14–18, April 2002.

[3] J. M. Rabaey. Low-power silicon architectures for wireless communications. In *Design Automation Conference ASP-DAC 2000, Asia and South Pacific Meeting*, pages 377–380, Yokohama, Japan, January 2000.

[4] T. Richardson and R. Urbanke. The renaissance of Gallager's Low-Density Parity-Check codes. *IEEE Communications Magazine*, pages 126–131, August 2003.

[5] B. Vucetic and J. Yuan. *Turbo Codes: Principles and Applications*. Kluwer Academic Publishers, 1 edition, 2000.

[6] E. Yeo. Shannons Bound: At What Costs? Architectures and Implementations of High Throughput Iterative Decoders. Berkeley Wireless Research Center Winter Retreat, January 2003.

[7] S. Rajagopal, S. Bhashyam, J. R. Cavallaro, and B. Aazhang. Real-time algorithms and architectures for multiuser channel estimation and detection in wireless base-station receivers. *IEEE Transactions on Wireless Communications*, 1(3):468–479, July 2002.

[8] S. Rajagopal, S. Rixner, and J. R. Cavallaro. Improving power efficiency in stream processors through dynamic cluster reconfiguration. In *Workshop on Media and Streaming Processors*, Portland, OR, December 2004.

[9] B. Aazhang and J. R. Cavallaro. Multi-tier wireless communications. *Wireless Personal Communications, Special Issue on Future Strategy for the New Millennium Wireless World, Kluwer*, 17:323–330, June 2001.

[10] J. H. Reed, editor. *Software Radio: A modern approach to radio engineering*. Prentice Hall, 2002.

[11] T. Gemmeke, M. Gansen, and T. G. Noll. Implementation of Scalable and Area Efficient High-Throughput Viterbi decoders. *IEEE Journal on Solid-State Circuits*, 37(7):941–948, July 2002.

[12] S. Rajagopal, S. Rixner, and J. R. Cavallaro. Design-space exploration for real-time embedded stream processors. *IEEE Micro (special issue on embedded systems)*, 24(4):54–66, July-August 2004.

[13] N. Zhang, A. Poon, D. Tse, R. Brodersen, and S. Verdú. Trade-offs of performance and single chip implementation of indoor wireless multi-access receivers. In *IEEE Wireless Communications and Networking Conference (WCNC)*, volume 1, pages 226–230, New Orleans, LA, September 1999.

[14] B. Salefski and L. Caglar. Re-configurable computing in wireless. In *Design Automation Conference*, pages 178–183, Las Vegas, NV, June 2001.

[15] T. C. Callahan, J. R. Hauser, and J. Wawrzynek. The GARP Architecture and C Compiler. *IEEE Computer*, pages 62–69, April 2000.

[16] A. Agarwal. RAW computation. *Scientific American*, 281(2):60–63, August 1999.

[17] S. Srikanteswara, R. C. Palat, J. H. Reed, and P. Athanas. An overview of configurable computing machines for software radio handsets. *IEEE Communications Magazine*, pages 134–141, July 2003.

[18] PACT: eXtreme Processing Platform (XPP) white paper. http://www.pactcorp.com.

[19] K. Keutzer, S. Malik, and A. R. Newton. From ASIC to ASIP: The Next Design Discontinuity. In *IEEE International Conference on Computer Design*, pages 84–90, September 2002.

[20] J. R. Cavallaro and P. Radosavljevic. ASIP Architecture for Future Wireless Systems: Flexibility and Customization. In *Wireless World Research Forum (WWRF)*, June 2004.

[21] P. Radosavljevic, J. R. Cavallaro, and A. de Baynast. ASIP Architecture Implementation of Channel Equalization Algorithms for MIMO Systems in WCDMA Downlink. In *IEEE Vehicular Technology Conference (VTC)*, volume 3, pages 1735–1739, September 2004.

[22] S. Agarwala et al. A 600 MHz VLIW DSP. *IEEE Journal on Solid-State Circuits*, 37(11):1532–1544, November 2002.

[23] S. Rixner, W. J. Dally, U. J. Kapasi, B. Khailany, A. Lopez-Lagunas, P. R. Mattson, and J.D. Owens. A bandwidth-efficient architecture for media processing. In *31st Annual ACM/IEEE International Symposium on Microarchitecture (Micro-31)*, pages 3–13, Dallas, TX, December 1998.

[24] H. Corporaal. *Microprocessor Architectures - from VLIW to TTA*. Wiley International, 1 edition, 1998.

[25] D. W. Wall. Limits of Instruction-Level Parallelism. In *4th international conference on Architectural support for programming languages and operating systems(ASPLOS)*, pages 176–188, Santa Clara, CA, April 1991.

[26] S. Rixner. *Stream Processor Architecture*. Kluwer Academic Publishers, 2002.

[27] C-K. Chen, P-C. Tseng, Y-C. Chang, and L-G. Chen. A digital signal processor with programmable correlator architecture for third generation wireless communication system. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, 48(12):1110–1120, December 2001.

[28] A. Gatherer and E. Auslander, editors. *The Application of programmable DSPs in mobile communications*. John Wiley and Sons, 2002.

[29] A. Gatherer, T. Stetzler, M. McMahan, and E. Auslander. DSP-based architectures for mobile communications: past, present and future. *IEEE Communications Magazine*, 38(1):84–90, January 2000.

[30] U. J. Kapasi, S. Rixner, W. J. Dally, B. Khailany, J. H. Ahn, P. Mattson, and J. D. Owens. Programmable stream processors. *IEEE Computer*, 36(8):54–62, August 2003.

[31] S. Rajagopal, S. Rixner, and J. R. Cavallaro. A programmable baseband processor design for software defined radios. In *IEEE International Midwest Symposium on Circuits and Systems*, volume 3, pages 413–416, Tulsa, OK, August 2002.

[32] P. Crowley, M. A. Franklin, H. Hadimioglu, and P.Z. Onufryk. *Network Processor Design : Issues and Practices*, volume 1. Morgan Kaufmann Publishers, 2002.