Challenges and Opportunities in Broadband and Wireless Communication Designs

Jan M. Rabaey[1], Miodrag Potkonjak[2], Farinaz Koushanfar[3], Suet-Fei Li[1], Tim Tuan[1]
[1] EECS Department, University of California, Berkeley, CA 94720
{[2] CS, [3] EE} Departments, University of California, Los Angeles, CA 90095

ABSTRACT

Communication designs form the fastest growing segment of the semiconductor market. Both network processors and wireless chipsets have been attracting a great deal of research attention, financial resources and design efforts. However, further progress is limited by lack of adequate system methodologies and tools. Our goal in this tutorial is to provide impetus for development of communication design techniques and tools.

The first part addresses network processors (NP) that we study from three viewpoints: application, architecture, and system software and compilation tools. In addition to summary of main issues and representative case studies, we identify main system design issues. The second part of the tutorial focuses on wireless design. The main emphasis is on platform-based design methodology that leverages on functional profiling, architecture exploration, and orthogonalization of concerns to facilitate low-power wireless communication systems. The highlight of the paper, an in-depth study of the state-of-the-art wireless design, PicoRadio, is used as explanatory design example.

1 Introduction

Historically, general-purpose (GP) computers have been dominating computation and communication markets. They evolved through several phases, from mainframes to mini computers and workstations, and to personal computers. Currently, we are in the midst of a revolutionary change. Communication devices, both broadband wires and wireless, are the main focus of research attention, financial resources and design efforts.

Wireless communication devices, and in particular mobile phones have shown an exceptional growth rate. Sometime during the next decade, it is expected that the number of mobile phones will surpass the number of wire line phones. This trend has a strong impact on the DSP semiconductor industry: 50% of integrated circuits are used in wireless communication devices.

The DSP market has been growing at rate greater than 40%. It is expected that it will grow from less than $2B in 1996 to over $19B in 2004. Broadband communication devices have been experiencing an even higher growth rate. While it is apparent that optical processing is bound to take over the core Internet traffic, programmable network processors dominate the Internet edge. The edge has many meanings and levels, including one between the core and Internet service providers (ISP), between ISP and enterprise central offices, and between the central offices and branch/small offices. Each of these interfaces requires different levels of throughput and a different type of functionality. Programmable Network Processors (NPUs) are used in a variety of communication systems, such as switches, routers, firewall boxes, set-top boxes, traffic shapers, and aggregation nodes. Progress in both broadband and wireless IC and system development is limited by the lack of adequate system methodologies and tools. The goal of this tutorial is to provide impetus for the development of communication design techniques and tools.

The tutorial has two parts. In order to achieve better coverage, the parts have not only different scopes, but also different emphases and styles of exposition. The first part addresses broadband processors. The emphasis is to provide the field summary, present case studies, and identify main design challenges. We cover applications, architecture, and system software. The second covers wireless design. In this part we only briefly cover the field in order to present an in-depth study of the platform-based methodology.

2 Network Processors

Network processors are specialized, programmable engines that are optimized for performing communications functions. Until late 1990s, an edge device employed a high performance CPU to perform tasks such as processing the routing data, forwarding table lookups, access control and implementing the network stack. Another approach was to use ASIC's which can perform tasks at wire speed. However, there is a lack of flexibility to support customization of an application. Today, with network equipment providers competing to provide sophisticated multimedia communication infrastructures, designers require both flexibility and agility to deliver within very limited time-to-market frames. NPUs have recently emerged as the implementation platform of choice for next generation networking products. With programmable network processors, exceptionally fast packet processing at high-bandwidth is achieved through the optimization of both the instruction set and datapath.

Furthermore, NPUs form the fastest growing segment of the overall market for processors used in networking equipment. While in 2000, ASIC and CPU's revenues were $1,134 and $704 million respectively, and there was $43 million revenues for NPUs, it is projected that by the year 2002, revenues for ASIC and CPU will grow by 38% and 24%, the revenues for NPUs will grow by a factor of higher than seven times.

2.1 Applications

Application specific programmable processors such as DSP processors have significantly different programming and compilation abstractions and technique with respect to GP processors. For example, special attention has to be given to the effective use of multiple memory banks, zero-overhead loops and integrated multiply-accumulator (MAC) units. NP pushes these differences even further. The nature of NP applications

clearly imposes the need for different treatment of "data plane" and "control plane" components.

Data plane tasks are comprised of operations of feedforward path through communication pipeline. The broadband optical communication medium places very demanding real-time performance constraints on this component. Tasks are simple, mainly related to receiving, processing and directing packets. Control plane tasks are related to functions such as the control of routing tables, statistics tasks, and high-level management and have significantly more relaxed timing constraints and significantly richer control structure.

Figure 1 summarizes the main control and data plane applications. For the sake of brevity, we restrict our attention only to data plane applications, which include:

- *Media Access Control*. Low-layer protocols, e.g. Ethernet framing and ATM cell processing, that postulate how the data is represented and how the communication channel is accessed. Speeds are currently in the range MB/s to GB/s.
- *Data Parsing*. The main goal is to locate packets of cell headers that carry address and protocol information. Historically, parsing functions were hardwired. However, today flexibility is mandatory in order to support a wide variety of information at multiple layers of the ISO model.
- *Classification*. The objective is to identify a packet against one of the protocols. Actions range from forwarding decisions to quality of service and real-time support.
- *Data Transformations*. Support for data alternation and translation between different protocols. Applications cover a spectrum from simple address translation to complete protocol conversion (e.g. IP to ATM).

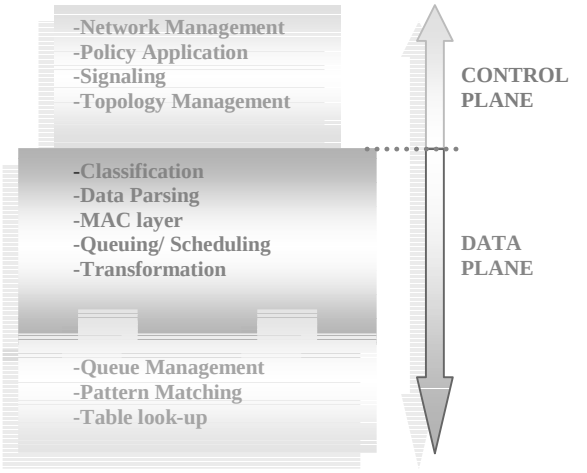

Figure 1. Network Processor Applications

Pattern matching, table lookup, and queue management dominate data plane applications. Pattern matching functions are required for classification and forwarding data blocks to the destination addresses. In particular, destination address forwarding, which includes large bit-field look-ups, requires different techniques depending on the size of the address field and can be optimized for either hardware size or for speed.

Queue management ranges from packet segmentation and assembly to Quality of Service queuing to traffic shaping and data discarding. These functions require buffering of network data into logical queues and a decision engine to determine the rate, priority and type of servicing for each queue.

One of the most effective ways to leverage knowledge about a set of applications is to develop a representative benchmarks suite and to use it evaluate architectural and implementation options. For example, for more than a decade the development and implementation of general-purpose processors have been driven by the SPEC benchmark suite [4]. Recently, the Commbench [3] set of communication programs have been proposed for network processors. It consists of two types of applications. The header processing programs are representative of operations that are conducted on a per packet basis. They typically involve a high amount of "random logic", table lookups and header field interrogation functions. Programs include RTR - Radix tree routing table look-up programming, FRAG - an IP packet fragmentation application, DDR - round robin fair scheduling protocol, and TCP traffic monitoring application. The second group, payload processing applications, process the contents of a packet. Typical reprenstatives are CAST - a DES-like cipher that uses the Bent function in S boxes, ZIP - Lampel-Ziv data compression program, and REED - Reed Solomon error correction code. Two main limitations of this benchmark are its small cardinality and lack of control plane applications. Still, it provides important insights into NPU applications. The Commbench examples are almost an order of magnitude smaller than SPEC programs and almost all computation is related to less than 10% of code. Another emerging benchmark is EDN embedded microprocessor consortium suite, mainly developed by network processor companies such as C-port and Sitera. Key system design challenges related to NPU applications include the development of better benchmark suites including data inputs, development of techniques for the characterizations of examples using profiling, and algorithm and protocol selection and tuning. It is important to emphasize that the profiling of broadband applications is significantly more demanding than profiling of the wireless applications due to an order of magnitude higher throughput.

2.2 Architecture

We now briefly discuss the architectural properties of NPUs. Parallelism and pipelining [1,2] are dominant generic performance enhancement techniques for processor design. They are widely used in general purpose (GP) processor design as well as in network processors (NPU). There are a number of

instantiation of these two generic that can be used in broadband processors:

- *Sequential processing.* In the general purpose market, both RISC and CISC architectures are used due to their unique advantages. However, network processors strongly favor RISC architecture due to lower gate count and lower complexity.
- *Pipelined processing.* Pipelining is probably the most widely used performance enhancement technique for general purpose processors. For example, the most popular general purpose processor, PIII, had 14 pipeline stages. The main advantage of pipelining is that it achieves very high performance/ number of gates ratio. However, latency of the pipeline imposes severe challenges to the programmer. In particular, conditional branches induce a high risk of rendering results of one or more stages in the pipeline irrelevant [6].
- *Application specific function blocks.* Application specific logic units can often perform mathematical calculations or table lookup functions significantly faster than programmable processors. They can also be used within pipeline structures.
- *Parallel processing.* Sequential, pipelined, and application specific function blocks can be used in parallel. Most often application specific function blocks serve as co-processors to the programmable master unit. Parallel processing has potential to achieve very high performance rates, but it also imposes numerous constrains and additional complication such as need for synchronizing parallel threads, need for partitioning of processing task for parallel processing and in particular maintaining contexts and sharing intermediate results between parallel functional units.
- *Input/Output optimization.* I/O operations dominate the network processor's functionality. Addressing the needs of high throughput data streams and high memory bandwidth requirements results in need for unique bus structures. Almost all network processors have the separate busses for I/O, data memory space and program memory space. This implies a need for a high pin count and high power requirements. In addition, a programmer must carefully address synchronization requirements required by I/O optimization.

In order to obtain a more tangible picture of NPU architectures and implementations we now describe the C-5 NPU from Motorola's C-Port Corp [5]. Figure 2 illustrates the internal architecture of the C-5 200 MHz MIMD digital general purpose communication processor. The processor is fabricated using a 0.25um dual-gate CMOS process. It has 6 Al metal layers and 56 millions transistors in a 2.51cm^2 die. At the supply voltage level of 2.4V, it has a maximum of 24W power dissipation.

C-5 processor (Figure 2) has several specialized functional units that make it suitable for the special purposes of routing, switching and data processing in communication tasks. The system, in addition to 16 channel processors (CP) contains a buffer management unit (BFU), a queue management unit (QMU), a table look-up unit (TLU), a fabric port processor (FP), and an executive processor (EP). The internal

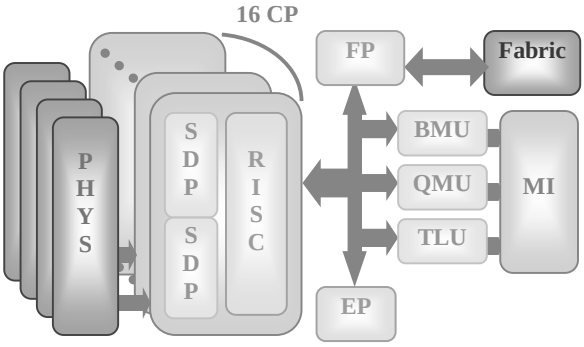

Figure 2 . C-5 Digital Communication Processor: PHYS-Physical Layer; SDP-Serial Data Processor; CP-Channel Processor; FP-Fabric Processor; EP-Executive Proc; MI-Memory Interface; BMU-Buffer Management Unit; QMU-Queue Management Unit; TLU-Table Lookup Unit.

architecture of each of the CPs is also shown in the figure. It contains transmit/receive serial data processors, and a 5 stage RISC processor that has the MIPS-I complete instruction set with 4 program contexts. It also contains a 12kB data memory, a 6kB instruction memory and control logics to interface to global busses. The FP unit processes packets at 2488Mb/s (full-duplex OC-48) rate that is equal to the overall bandwidth of all the CPs. It operates at the core clock frequency although the fabric port operates at 100MHz nominally. The FP unit has an architecture that is similar to one of each of the CPs. The main difference is that in FP all of the data processing and control is done in hardware. FP provides an interface of a digital communication processor to another digital communication processor or to a network fabric. The TLU uses a 64b wide SSRAM to provide 133 million lookups/s. QMU uses a 32b wide SRAM to store descriptor data.

Key architectural challenges include, high speed and scheduled interconnect bus structures, the development of an instruction set for the NPU, design space exploration, in particular with respect to the flexibility vs. performance trade-off. Also, there is a strong need for the development of testing, debugging, and performance evaluation hardware support. For all these tasks CAD tools are needed. An early example of this type of research is related to communication processors pipeline sizing.

2.3 Software and Development Tools

It is well known that in both GP and application specific processors, the key performance and time to market bottleneck is software. For example, Hennessy and Paterson demonstrated that a utilization factor of the GP datapath is rarely higher than 3% even when the most aggressive optimization compilers are used. In order to harness the power of the NPU, software toolsets often provide comprehensive software suites, which include one or more version simulators, debuggers, and performance analyzers.

The centerpiece of software tools is the compiler. Three main NPU programming choices are available: microcode, 4GL, and standard language programming. Microcode engines mainly implement data-plane functionality high performance

through multithreading. The hardware and speed efficiency is achieved at the expense of programming convenience and the need for complete comprehension of both targeted architecture and applications. Fourth Generation Languages (4GL) for NPUs provide search and pattern matching algorithms for parsing and classification. They drastically reduce programming efforts at the expense of lower performance and restrictions of proprietary environments. Finally, standard language programming leverage most often on C/C++. While for some tasks they provide higher programming efficiency, if high performances are needed, they often imply a need for strong knowledge of architecture and applications.

A majority of NPU standard language compilers are GNU-based and of rather low efficiency. Communication Programming Interface (CPI) abstracts the most common networking functions such as physical interface management, data forwarding, table lookups, queue manipulation operations, and fabric and kernel services. Furthermore, several companies provide reference designs and libraries for the most popular applications, such as 10/100 Mb and 1Gb Ethernet Switch-Routers, OC-3 and OC-12 Packet over SONET router, ATM Switch, and AAL5 SAR.

The single greatest challenge for NPUs is to minimize down time [9]. In order to accomplish this goal, designers need traffic generators and protocol analyzers. Protocol analyzers provide complete and accurate information about fault tolerance of the network elements. Unfortunately, they can provide only post-deployment information. Their major functions include large capture buffers, for storing data for off-line data analysis the coding protocols for all seven layers of the protocol stack, real-time analysis and interpretation of data and flexible filtering capabilities.

On the other side of the coin are traffic generators. They can be used during the verification phase in the lab set-up before deployment. The state-of-the-art traffic generators have capabilities which include generation and transmission of data at the maximum rate, creation and simulation of data streams with user given statistical characteristics, ability to time-stamp packets, support for variety of protocols, filtering capabilities and the ability to capture and interpret data for analysis. High throughput and low latency requirements place very demanding requirements on both traffic generators and protocol analyzers.

In addition to the simulation environment, the hardware evaluation kit is a mandatory component of the NPU development environment in order to provide an extensive and robust validation process. Intel's Level One IXP12DE Workbench tool kit is an example of the state-of-the-art NPU software tool. It facilitates development and analysis of software for Level One's IXP1200 processor. The processor has a StrongArm core and six 32 bit RISC independent programmable coprocessors. The processors are linked using a 4.2 Gbps IX bus interface to I/O devices. The interface contains a hash unit, FIFO buffers, system-level control-status registers and a 1K word of scratchpad memory.

Each RISC processor has 1K instruction storage and 128 general purpose registers and provides support for four threads. Each thread determines when it should go to sleep and allow another thread to start execution; arbiter decides which thread to run next. The Workbench includes a microcode assembler/optimizer, debugger, and the Transactor - cycle-accurate simulator. The transactor provides access to all threads in the IXP NPU. Graphical color interface allows simultaneous observation of the execution state in all six coprocessors. Colors indicate pipeline-stage states and thread history.

The workbench also provides comprehensive statistics for each micro-engine and all threads. The reference design is 16 ports 100-Mbit Fast Ethernet design. The key challenges related to NPU software and development tools include the development of high quality compilers, high speed simulation and emulation techniques, and better debugging tools. There is a wide consensus that software tools are the main bottleneck in the development of communication systems.

3 Wireless Communication Designs

On a different front of communication system design, there is an increasing need for mobile, wireless processors [7,8]. Recently, a number of wireless standards have emerged, including BlueTooth [10], IEEE 802.11 [11], and HomeRF [12]. These standards will appear in embedded systems for numerous applications, such as cellular phones, home automation, digital audio players, and many more. In the second part of this paper, we discuss the key aspects of processor design for next-generation wireless embedded systems.

To deal with the unique constraints and challenges in wireless system design, we must rethink the traditional approach to system design. In the following sections, we first introduce the concepts of a novel design methodology for wireless systems. Later, we demonstrate our methodology by describing its application in the early stage design of a low-energy reconfigurable radio.

3.1 Design Methodology

Today, wireless system designers are faced with a new set of challenges. Market pressure due to fierce competition is rapidly shrinking the design time, while inventive wireless applications are imposing stricter requirements in energy consumption, cost, size, and flexibility. Although technology scaling provides the designer with more hardware resources, epitomized by the emergence of System-on-a-Chip solutions, to meet the ambitious constraints in a short period of time, the design process must make efficient utilization of highly reusable architectures.

3.1.1 Platform-based Design

Our solution for these challenges is a design methodology called platform-based design [13]. Platform-based design facilitates design reuse by abstracting hardware to a higher level (platforms) that is visible to the application software. The hardware platform should comprise a family of flexible (parameterizable) architectures that adequately support the functions in the application space. Also, a software platform is needed to abstract the hardware platform into a programmer's model to allow effective mapping. We call the union of hardware and software platforms the *system platform.* Once a system platform has been identified for the application space and the architecture space, the final chip design involves design exploration within the system platform to determine the best mapping of application to architecture.

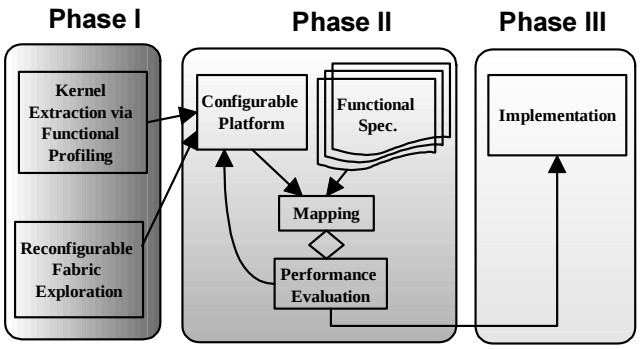

Figure 3. Design flow for wireless protocol processor

Figure 3 graphically captures the three phases of platform-based design flow. In phase I, the system platform is conceived through consideration of the application domain and the available architectural modules. Phase II performs the design exploration to find a suitable platform instance for a given set of target applications and constraints. Lastly, phase III completes the final implementation (hardware and software synthesis) of a specific application onto the platform instance. The following sections briefly describe each phase.

3.1.2 Platform Conception (Phase I)

The first step in platform-based design is to conceive a system platform that identifies a set of architectural modules to support the class of functions in our applications domain.

A typical platform for wireless systems consists of programmable processors, reconfigurable logic, dedicated logic, memories, and peripherals. To construct a hardware platform that supports the key functions of the application domain is a two-fold problem: We need to (1) identify the key functions and their constraints, and (2) explore the available architecture modules and their performance behavior. The former is achieved through *functional profiling* of a suite of candidate applications, and extracting a set of key operations (kernels) common to these applications. The latter requires *architecture exploration* of existing implementation fabrics to obtain first-order performance, energy and area estimations for these basic modules. The first part of Figure 3 depicts this duality. In sections 3.2.1 and 3.2.2, we present these two steps in greater detail in the context of a low-energy radio design case study.

The second part of platform conception involves the construction of a software layer that interfaces the hardware platform to application programmer. Having an application program interface (API) simplifies the subsequent efforts of mapping applications to the hardware.

3.1.3 Platform Instantiation (Phase II)

Once a system platform has been defined, we need to explore within the system platform to find a platform instance that is suitable for a given set of applications. To do so, we employ the Y-chart approach [14], which involves an iterative process of mapping functions to parameterized architectural modules, and evaluating the performance of the resulting platform under the given set of functional constraints. This process is illustrated in second part of Figure 3.

To fully explore the design space, we need to adhere to the principle of *orthogonalization of concerns* [15][16], which advocates the need for separation of functional and implementation concerns. Doing so provides the designer with the greatest degree of freedom in choosing the best solution. An important consideration is the need for a purely functional design specification. It has been shown that such a functional specification should have an underlying formal mathematical model, as is the case with concurrent finite state machines (CFSMs) [17].

Given that a well-defined system platform is in place, platform analysis becomes a relatively simple process. In the system platform, we have available a library of architectural modules with corresponding performance, energy and area prediction models. From these modules, we can construct different platforms with varying performance for our target applications. By examining multiple platforms, we can quickly converge to an optimal platform instance that satisfies the set of design constraints of our applications.

3.1.4 Implementation (Phase III)

Once we have a platform that meets all of the design constraints, mapping any specific application onto hardware becomes a software issue. Through the software API, the hardware platform can be programmed or configured to perform the desired functionality. Remaining issues include generation and compilation of application code, real-time operating system (RTOS), and any necessary design synthesis.

By thinking at a platform level, we are able to produce a solution that allows fast design time through extensive software and hardware reuse. To effectively use platform-based design methodology, it is imperative to have a good system platform in place before proceeding further in the design flow. The next section describes in detail the platform conception phase of the design of a wireless, reconfigurable radio.

3.2 Case Study: PicoRadio

The emergence of ubiquitous computing has created a demand for *ad hoc*, sensor-based networks that comprise hundreds of programmable and self-organizing radios, which we call PicoNodes [18]. Through self-configuration and adaptation to environmental changes, these networks are perfect for data-acquisition and environment-control applications. To achieve large, dense networks, PicoNodes must be inexpensive, lightweight. To avoid periodic battery replacement for hundreds of nodes, they must exhibit extremely low-energy consumption to enable energy scavenging [19].

In designing the PicoRadio network processor, we employ the platform-based design approach to devise an architecture that is optimized for size, cost, and most importantly, energy. In this section, we present the design of the PicoRadio network processing architecture as a case study to facilitate further understanding of the design methodology for wireless systems. The project is currently in the platform conception phase (phase I) of the design methodology, which consists of functional profiling and architecture exploration.

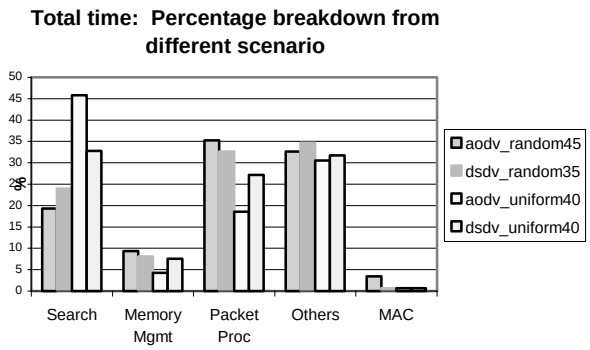

Figure 4. Profiling results of an ad hoc wireless network

3.2.1 Functional Profiling

Before starting the implementation process, we need to gain an in-depth understanding of our application space. A highly efficient implementation could only be realized if the performance critical operations in the applications are classified and specially targeted. Functional profiling explores regularity and extracts common operations (kernel extractions) in the application. The important issues of functional profiling are profiling granularity and classification and interpretation of the collected data. If the granularity is either too coarse or too fine, regularity and commonality may not be fully exposed. To reach an optimal granularity, some reorganization of the application code (e.g. insertion of some wrapper functions) is often needed. To classify and interpret profiling data in a meaningful fashion requires some insight into the class of application algorithms.

Referring back to the first section of this paper, the list of critical operations identified by the high performance wired network processor (WNP) community consists of: *parsing, searching (table lookup), packet modifying and re-assembly*. We will use this list as initial guideline for profiling wireless applications.

To facilitate our understanding of the PicoRadio application space, we have collected a benchmark suite with a wide range of mobile wireless applications. Preliminary results have been obtained from our first benchmark application, a mobile *ad hoc* network that supports different protocols. The application program is written in OPNET Radio Modeler from Millennium 3 Technologies [20].

Our protocol model has simple MAC and physical layers, but a rather sophisticated network layer that enables us to explore different types of routing protocols. Node distribution and mobility can also be specified. Routing protocols has a significant impact on implementation parameters such as routing and forwarding table sizes. The distribution of nodes in the network affects the network activity and hence the protocol performance. We studied four different scenarios with two different routing protocols and two types of node distributions. The first protocol is the Ad-hoc On-Demand Distance Vector Routing (AODV) [21], a reactive protocol. The second is the Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) [22], a proactive protocol. The first type of network distribution is a uniform grid of nodes in which the neighbors can hear each other. The second is a randomly generated distribution. The nodes have fewer neighbors on average in the random distribution case and generate less network traffic.

OPNET only produces profiling information at the level of the leaf funtions. The generated fine grained profiling data are grouped and classified using the WNP guidelines. The results, presented in Figure 4, look very similar to the NP cases. Searching consumes 20%-45% of the total time, and packet manipulation (parsing, modification, re-assembly, etc.) consumes 18%-28% of the total time. This clearly suggests that the implementation of dedicated packet processing and table searching engines may greatly improve the overall system performance. Furthermore, since searching is mostly routing and forwarding table lookups, a protocol that does not involve the maintenance of large tables should lead to a cheaper implementation.

3.2.2 Architecture Exploration

To ensure the most usable system platform for the remainder of the design process, the components of the hardware platform should provide sufficient coverage of the functions of the application domain. Functional profiling identifies the most crucial functions that we need to address. Architectural exploration generates a set of architectural modules that best supports these functions.

To enable software reuse and allow application exploration, a wireless platform must provide some degree of programmability. Some of the most popular programmable hardware fabrics include customizable microprocessor [23], field-programmable gate arrays (FPGA), and programmable logic device (PLD) [24]. While the microprocessor provides the most flexibility, they fail to meet the tight energy requirements of wireless applications. In contrast, FPGA and PLD are more promising options; therefore they shall be our focus.

In fabric exploration, we are looking for creative ways of implementation. Both FPGA and PLD exhibits promising characteristic for certain types of functions. To make the proper tradeoffs, one must understand their characteristics and fundamental differences.

The traditional FPGA architecture and PLD architecture primarily differ in granularity. The FPGA comprises an array of thousands of configurable blocks. Using lookup tables (LUTs), these blocks can each implement any arbitrary N-input logic (typically N<5). On the other hand, the PLD comprises several (<32) logic array blocks that can each implement two-level (AND-OR) logic of wide inputs (up to 64), but limited outputs (<20). To achieve reasonable speed performance, the wide-input AND-gate is usually implemented with a single pseudo-NMOS pull-up transistor [24].

The structural differences result in several important performance implications. Protocols designs are often specified using extended FSM models, which are FSMs with datapath elements. Datapath elements such as adders and multipliers usually exhibit highly irregular logic structure, which is better suited for the structure-blind LUTs. On the other hand, FSMs directly map to standard two-level logic. Also, datapath elements typically require a non-trivial fixed I/O width (8b-32b), and FSMs tend to have very wide input with just a few outputs. Consequently, the FPGA tends to be more suitable for datapath elements, and the PLD structure lends itself to efficient implementation of FSMs. Empirical studies (Figure 5) using commercial tools for performance estimation precisely

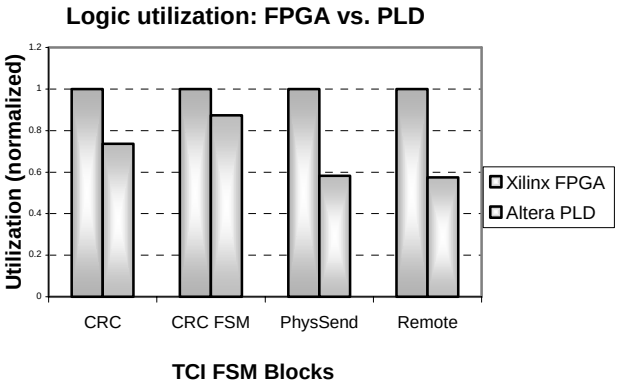

Figure 5. Normalized utilization comparison of FPGA vs. PLD, implementing functional blocks from a wireless protocol.

reflect these trends. The exact partition of the design should be determined through careful consideration of these tradeoffs.

In PicoRadio design, fabric exploration has helped us identify the functional capability and tradeoffs of several architectures. To meet the key constraints of our applications domain, most importantly, low energy-consumption, we have done research specifically in low-energy FPGA and PLD architectures [25].

4 Conclusion

Broadband and wireless communication designs are the fastest growing segment of the integrated circuits market. Their further exponential development is impeded by lack of proper system development techniques and synthesis and compilation tools. We surveyed challenges and opportunities in communication designs. In addition to several smaller case studies, we presented in-depth discussion of the platform-based design methodology for wireless systems.

5 References

[1] G. Qu and M. Potkonjak, Techniques for Energy Minimization of Communication Pipelines, ICCAD. pp.597-600, Nov 1998.

[2] R.Y Wang; A. Krishnamurthy; R.P. Martin; T.E. Anderson and others, Modeling Communication Pipeline Latency. SIGMETRICS'98. pp.22-32, 1998.

[3] T. Wolf and M. Franklin, CommBench – A Telecommunication Benchmark for Network Processors. Proceedings of IEEE International Symposium on Performance Analysis of Systems and Softwares (ISPASS-2000), Austin, Texas, pp. 154-162, April 2000.

[4] J.L. Henning, SPEC CPU 2000: Measuring CPU Performance in the New Millennium. Computer, vol33, (no.7), pp. 28-35, July 2000.

[5] D. Husak and R. Gohn, Communication Processor Programming Models: Keys to The Promise. Proceedings of Gigabit Ethernet Conference(GEC), pp. 298-309, March 2000.

[6] M. Hathaway, Building Next Generation Network Processors. Proceedings of Gigabit Ethernet Conference(GEC), pp. 310-319, March 2000.

[7] F. Koushanfar, V. Prabhu, M. Potkonjak and J.M. Rabaey, Processors for Mobile Applications. To appear in Proceedings of International Conference on Computer Designs (ICCD), Sept. 2000.

[8] I. Verbauwhede and C. Nicol, Low Power DSP's for Wireless Communications. Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), pp. 303-310, July 2000.

[9] H. Soor, Performance Analysis on Gigabit Ethernet Switches, Proceedings of Gigabit Ethernet Conference(GEC), pp. 77-81, March 2000.

[10] The Bluetooth Special Interest Group http://www.bluetooth.com/.

[11] IEEE 802.11 Working Group for WLAN http://www.manta.ieee.org/groups/802/11/.

[12] The HomeRF Working Group http://www.homerf.org/.

[13] A. Ferrari and A. Sangiovanni-Vincentelli, System Design: Traditional Concepts and New Paradigms, Proceedings of the 1999 Int. Conf. On Comp. Des., Austin, Oct. 1999.

[14] B. Kienhuis et al, *An Approach for Quantitative Analysis of Application-specific Dataflow Architectures*, Proceedings of International Conf. of Application-specific Systems, Architectures and Processors, pp. 338-349, Zurich, Switzerland 1997.

[15] J. Rowson, A. Sangiovanni-Vincentelli, *System Level Design*, EE Times, 1996.

[16] J. Rowson and A. Sangiovanni-Vincentelli, Interface-based Design, Proceedings of the 34th Design Automation Conference (DAC-97). pp. 178-183, Las Vegas, June 1997.

[17] E. Lee and A. Sangiovanni-Vincentelli, A Unified Framework for Comparing Models of Computation, IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems, Vol. 17, N. 12:1217-1229, December 1998.

[18] J. Rabaey et al. PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking. IEEE Computer, Vol. 33, No. 7, pp. 42-48, July 2000.

[19] R. Amirtharajah, A. P. Chandrakasan, Self-Powered Signal Processing Using Vibration-based Power Generation. IEEE Journal of Solid State Circuits, vol. 33, no. 5, pp. 687-95, May 1998.

[20] OPNET Radio Modeler, OPNET Technologies, Inc., http://www.mil3.com

[21] C. Perkins and E. Royer, Ad-hoc On-Demand Distance Vector Routing, Proceeding of the 2nd IEEE Workshop. Mobile Comp. Sys. and Apps. pp. 90-100, Feb. 1999.

[22] C. Perkins and P. Bhagwat, Highly Dynamic Destination Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, Computer Communications Review, pp. 234-44, Oct. 1994.

[23] Tensilica, Inc. http://www.tensilica.com/.

[24] M. Smith, Application Specific Integrated Circuits, Addison-Wesley, 1997.

[25] BWRC homepage, http://bwrc.eecs.berkeley.edu/