A Wireless LAN Demodulator in a Pamette: Design and Experience

Tom McDermott[*], Philip Ryan[†], Mark Shand[‡], David Skellern[*], Terry Percival[†] and Neil Weste[*]

[*] Macquarie University Electronics Department, Sydney NSW 2109 Australia
Tel: +61 2 98 50 91 45 Fax: +61 2 98 50 91 28 spon@mpce.mq.edu.au

[†] CSIRO Division of Telecommunications and Industrial Physics, PO Box 76, Epping NSW 2121 Australia
Tel: +61 2 93 72 42 22 Fax: +61 2 93 72 44 94 pryan@rp.csiro.au

[‡] Digital Equipment Corporation, Systems Research Center, 130 Lytton Av., Palo Alto CA 94302-1044 USA
shand@acm.org

1. Abstract

We have implemented the digital section of a wireless local area network (WLAN) demodulator in a reconfigurable interface card called the PCI Pamette. The entire baseband section of the demodulator has been implemented in the Pamette and a simple analog to digital mezzanine board. This is the second implementation of the demodulator, the first being a card-based design using a mixture of discrete and reconfigurable logic. The Pamette implementation took far less time to complete than the card-based design. Moreover, the reconfigurable substrate is much more versatile. This paper describes the Pamette implementation and discusses our experiences with the two different design styles and technologies.

2. Introduction

The Macquarie University Electronics Department and the CSIRO Division of Telecommunications and Industrial Physics are developing a high-speed WLAN [3]. The target of this project is a complete modem in a single chip by 1998. The current stage of the project calls for a versatile hardware prototype which can operate at 30 MHz. This prototype will be used to explore architectural and algorithmic issues, and to characterize the 5 GHz indoor radio environment.

Software simulations based on radio channel measurements at 40 GHz led to the development of a prototype WLAN demodulator architecture. The purpose of the Pamette demodulator implementation is to validate and refine parameter choices for 5 GHz operation. A hardware prototype running in real time is vital to this requirement because software models do not run fast enough to take useful amounts of data. For example, to measure bit error rates of 10^{-9} requires more than 10^{10} transmitted bits. On a lightly loaded HP 9000/735 (125 MHz clock speed) it takes longer than 1 hour to simulate demodulation of 10^6 bits – or more than a year to simulate 10^{10} bits. In the Pamette implementation running

at 30 MHz, 10^{10} bits can be processed in under 10 minutes.

The structure of this paper is as follows. We first describe the WLAN concept and architecture. We next describe the Pamette hardware and associated CAD, cover the Pamette implementation of the demodulator, and close with a comparison of the Pamette and card-based implementations.

2.1 The WLAN System

The basic entity of the WLAN is a radio cell, consisting of a hub station and mobile stations [*5*]. The hub station is responsible for providing connectivity between mobile stations in the cell, and from mobile stations to the wired backbone. The WLAN, shown in Figure 1, consists of one or more radio cells, along with wired terminals, connected to the wider network over a wired backbone. Communication between the hub and mobile stations is via a radio link, using a coded multi-tone modulation

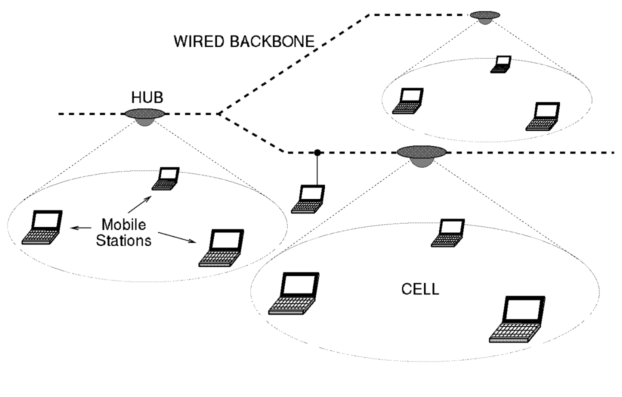

Figure 1: Wireless Local Area Network System

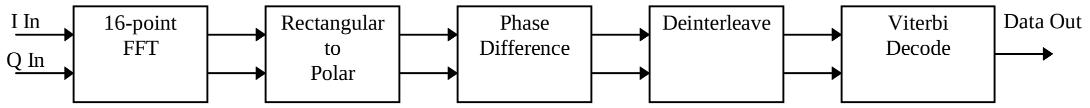

Figure 2: Demodulator Architecture

scheme to overcome multi-path interference.

At the transmitter, raw data is encoded by a rate-½ convolutional encoder, and then interleaved. The interleaving process is performed to decorrelate errors in the radio channel, making the channel appear Gaussian. The interleaved data is then packed into frames of block-differentially-modulated quadrature phase-shift-keyed (DQPSK) symbols suitable for generating a multi-tone signal. These frames are passed to a 16-point Inverse Fast Fourier Transform (IFFT) processor, where the DQPSK symbols in the frame set the phase of 16 individual tones. The output of the IFFT is a complex time domain signal consisting of the sum of these 16 tones – a multi-tone signal. This signal is passed to a radio-frequency (RF) transmitter, where it is mixed up to RF frequencies (5 GHz in our system) and transmitted.

At the receiver the RF radio signal is down-converted to a pair of signal components (in-phase and quadrature-phase) at baseband (30 MHz). These recovered signal components are used by the demodulator to reconstruct the transmitted data sequence.

2.1.1 Demodulator Architecture

The digital section of the demodulator is shown in Figure 2. The in-phase (I) and quadrature-phase (Q) components of the received signal are sampled at the baseband frequency by a pair of analog-to-digital converters, yielding two data streams.

The data passes into a Fast Fourier Transform (FFT) processor which computes the I and Q frequency components of the received signal. These frequency components are estimates of the transmitted DQPSK symbols. A rectangular to polar conversion transforms the I and Q signals into phase and magnitude form. A block phase difference is then performed, and the symbols deinterleaved to produce a stream of QPSK symbol estimates, from which a Viterbi decoder recovers the final bit stream.

2.1.2 Computational Complexity

The demodulator has a computational complexity of about 200 DSP operations per sample, or 10^9 DSP operations per second for 30MHz operation. A DSP operation is here taken to be an 8-bit addition, multiply-accumulate or load-store. Because much of the arithmetic in the demodulator requires only 8-bit words, it is not apparent that significant advantage could be gained by using a DSP chip with much wider word-widths. Even assuming a DSP clock that is five times the sample speed (i.e. 150 MHz), a DSP-based implementation would require some 40 DSP chips, with corresponding difficulties resulting from the partitioning of the design across a large number of devices.

3. The PCI Pamette

The PCI Pamette is a reconfigurable interface card developed by Digital Equipment Corporation [1]. The Pamette consists of a user area of 4 Xilinx FPGAs [9], a PCI interface (implemented in a fifth FPGA), 256 kB of scratchpad SRAM, and slots for up to 256 MB of DRAM SIMMs. The Pamette provides mezzanine card support through standard IEEE CMC connectors [8]. This allows

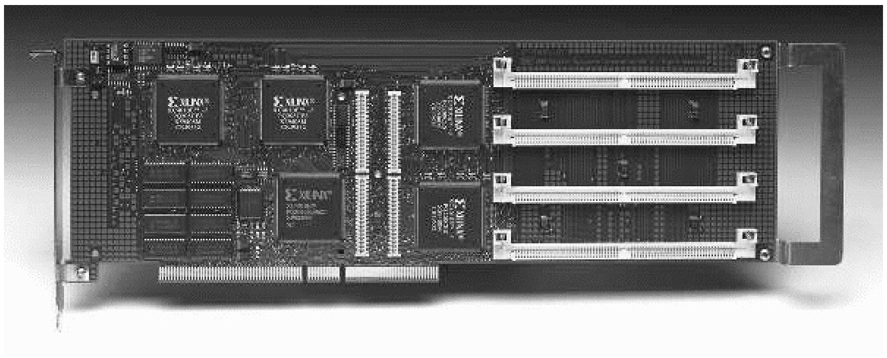

Figure 3: PCI Pamette

external I/O through either a standard or a custom mezzanine card.

Also provided with the Pamette is a set of CAD tools called PamDC for implementing designs [2]. PamDC is derived from Perle1DC, the CAD system of DECPeRLe-1 [1]. PamDC is a C++ class library which allows netlist descriptions to be embedded in user-written C++ code. For example, the code fragment shown in Figure 4 implements an 8-bit adder. The Pamette CAD provides support for attaching placement directives to nets at the C++ level. C++ classes are used to represent the hierarchy of a design, equivalent to blocks and sub-blocks in a schematic.

```
WireVector<Bool, 8> A,B,Carry,Sum;

/* Bottom bit has no carry in */
Sum[0] = A[0] ^ B[0];
Carry[0] = A[0] & B[0];

for(int i = 1; i < 8; i++)
{ /* Specify the top 7 bits */
 Sum[i] = A[i] ^ B[i] ^ Carry[i-1];
 Carry[i] = (A[i] & B[i]) |
            (A[i] & Carry[i-1]) |
            (B[i] & Carry[i-1]);
}
```

Figure 4: An 8-bit Adder in the Pamette C++ Hardware Description Language

The Pamette design flow consists of writing a C++ program which, when compiled and run, produces a netlist. This netlist is then passed to the Xilinx backend tools to produce a Xilinx bitstream. Use of the CAD tools is not mandatory; any technique which generates a Xilinx bitstream can be used to configure the Pamette FPGAs.

The version of the Pamette used in the demodulator is populated with four user-programmable XC4010Es, giving the user access to about 40k gates. Another variant of the Pamette replaces these chips with the XC4020E, yielding up to 80k gates.

We will call a particular configuration imposed on the reconfigurable hardware on the board, and a corresponding program executing on the host, a Pamette *design*. The host can transfer data to and from the Pamette over the PCI bus, reprogram the Pamette, and (in a debug mode) single-step the Pamette's clock. Part of the CAD set is a library of functions for programming and accessing the Pamette from the host.

4. Pamette Demodulator

The Pamette implementation of the demodulator was largely a straightforward translation of the existing card-based design [5]. Our first step was to calculate the size of the blocks to ensure that the demodulator could be implemented in the Pamette, and could be partitioned across the four user-area FPGAs. Each block in Figure 2 was then implemented by transcribing from existing schematics to the Pamette CAD. The digital section of the design runs at 30 MHz. The FFT and Viterbi decoder blocks are fully placed – the location of every CLB is specified in the C++ description of the design. This was necessary to reach the target design speed and ensure that the blocks would fit into their allocated FPGAs. The phase difference and deinterleaver blocks are only partially placed – some obvious placement constraints have been applied, but much of the control logic was left to the Xilinx backend tools because they gave sufficient performance.

The FFT block is implemented in a single XC4010E, with the design based heavily on an earlier semi-custom implementation [4]. The 16-point FFT is decomposed into two 4-point FFTs separated by a complex multiplier. The 4-point FFTs are further decomposed into two (trivial) 2-point FFTs and two degenerate multipliers. The resulting 16-point FFT requires only a single complex multiplier embedded in a pipeline of simple 2-point FFT processors and degenerate multipliers. The FFT is symbol-serial/bit-parallel – the 16 symbols are processed sequentially, but each symbol is dealt with in a bit-parallel fashion.

The implementation of the rectangular-to-polar conversion demonstrates one of the powerful abilities of the Pamette hardware and CAD. The conversion is done using one of the Pamette SRAM banks as a lookup table. At startup time, the Pamette is configured into a simple design that can be used to read and write the SRAM. This is then used to load the table lookup values. The Pamette is then reconfigured to the demodulator design. By using a separate design to do the once-off initialization the demodulator design is simplified, and the design effort is decreased because we can reuse an existing SRAM read-write design.

Another powerful feature of the Pamette is apparent in the deinterleaver block. The deinterleaving is done by writing data to an *n* by *m* memory array row by row, then reading it out column by column. To generate different deinterleaver sizes, we simply pass the two parameters *n* and *m* to the C++ description of the design. This allows us to select the deinterleaver size simply by generating a number of different Pamette configurations and downloading the appropriate one. The implementation of any *particular* interleaver - say the 3-by-7 case - is quite simple. The complexity is moved from the logic design to the CAD, where complexity is easily handled because we have all the power of a full-blown programming language.

4.1 Analog interface

Input to the demodulator is two analog signals – the I and Q components of the recovered radio signal, and two digital signals – an external clock and a start-of-packet flag. In order to bring these to the Pamette digital section an analog interface was implemented on a simple plug-in mezzanine card. This card consists of analog signal conditioning hardware and a dual analog-to-digital converter. This card, shown in Figure 5, was fabricated at Macquarie University using an in-house 2-layer printed circuit board (PCB) technology.

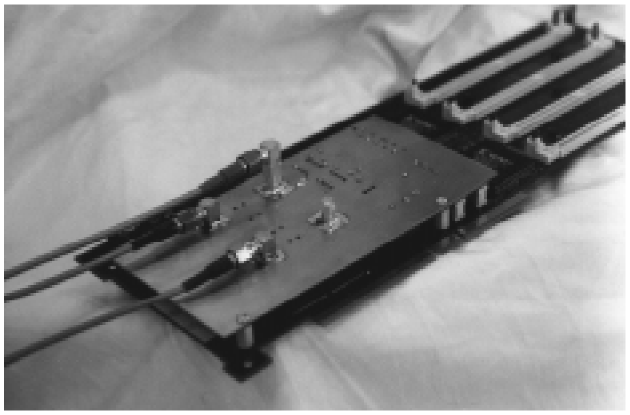

Figure 5: Analog Interface Card

Because the input data is synchronous with an external clock, an asynchronous boundary between the Pamette and the analog interface was required. This was implemented in the first user FPGA, using the XC4010E's edge-triggered RAM to implement a simple 16-deep asynchronous FIFO [7]. The Pamette provides support for such asynchronous boundaries by routing a clock signal from the mezzanine connectors to a pin near a Xilinx clock buffer.

5. Design Experience

It is commonly believed that the use of reconfigurable substrates can dramatically reduce the time required to develop prototype systems (for example, see [10]). Our experience with the Pamette implementation of the demodulator supports this. In this section, we describe our experiences in building the card-based and Pamette demodulator prototypes, and discuss the reasons for the differences in design effort required.

During construction of the 40 GHz prototype the US FCC released its draft proposal for spectrum assignment in the 5 GHz band for the US National Information Infrastructure [6]. This prompted an additional focus on 5 GHz WLANs with a target speed for the demodulator prototype of 30 MHz.

The first, card-based, implementation has each of the blocks shown in Figure 2 implemented as a separate (PCB). The implementation of this prototype took four people working full time a year to complete. It should be noted that this coincided with the introduction of a new set of PCB CAD tools.

The time it took to design a single working PCB has been estimated from the log books of the designer:

Learning CAD tools:	9 weeks
Designing circuit:	4 weeks
Fabrication:	2 weeks
Board population:	1 week
Debugging the board:	8 weeks
Total	**6 months**

This contrasts very strongly with the experience of a first-time Pamette user. One of the authors, an experienced hardware engineer, took only a single day to come 'up to speed' with the Pamette hardware and CAD tools. The C++ examples provided with the CAD were quickly and easily modified to implement real designs. The most troublesome concept proved to be the extra layer of abstraction – writing C++ code which generates structures which implement the design, rather than drawing schematics which describe the design directly. This rapid learning curve was echoed by the rapid development time – the entire digital section of the demodulator was completed in 8 person-weeks.

5.1 Conclusions – why the Pamette implementation was easier.

Our first impression when we finished the Pamette implementation of the prototype demodulator was the (to us) astonishing speed at which it was completed. Even taking into account that doing any design a second time will inevitably be considerably faster (for example, in the Pamette demodulator we did not have to design logic – it was taken from the existing card-based demodulator) the question remained: *why was the Pamette implementation so much faster and easier?*

The most obvious answer to this question is – quite simply – that we didn't build any hardware. The Pamette implementation is much closer to the idealized digital domain than the card-based version – ones are ones, zeroes are zeroes. There is a much smaller requirement for the designer to understand the fundamentally analog nature of the circuit – no ground bounce, ringing, or crosstalk. Because these problems have been solved by the designers of the Pamette, the design process is substantially easier and less error-prone for Pamette users.

The reconfigurable nature of the Pamette allowed us to develop the demodulator block by block. Blocks were implemented and tested in isolation by different designers. This contrasts strongly with the PCB design flow, where our inability to simulate complete cards prevented a block-by-block development path, and mechanical constraints allowed only limited opportunity for 'reconfiguration' once the cards had been fabricated. Because the Pamette provides enough reconfigurable logic to implement the demodulator, we were able to partition and build the demodulator without putting a lot of effort into minimizing the area of the blocks. This advantage remains as long as the target design fits within the speed and area constraints imposed by the Pamette.

An important advantage of the Pamette over the card-based demodulator prototype is the tight integration of all the resources required to develop the demodulator: the design, implementation and test were all done from a single workstation. The card-based prototype was debugged using a logic analyzer interfaced to a workstation. The interface required custom software, which itself required debugging. The cards had special test structures built in, a source of implementation overhead. In the Pamette implementation, test structures can be removed once a block has been tested.

The Pamette implementation of the demodulator is substantially smaller than the card-based version. As one purpose of the prototype is to take radio channel measurements in a variety of environments, this additional degree of portability is an advantage. Further, the Pamette's PCI interface means that, unlike the card-based implementation, custom demodulator/host interface hardware does not need to be developed.

The Pamette demodulator can be modified rapidly and over a wide range of parameters- a significant advantage in a prototype. Experiments may prove that it is necessary to change arithmetic word widths or to make more significant modifications such as changing from a convolutional to a block code. Such modifications to the card-based prototype would generally require a new PCB to be designed, manufactured and debugged. Further, the prototype can be duplicated and disseminated simply by purchasing a commercially available device rather than going through the manufacture-populate-test loop again.

6. Acknowledgments

This research was funded in part by the CSIRO Division of Telecommunications and Industrial Physics.

The WLAN architecture and card-based implementation are the result of collaborative work between Macquarie University Electronics and the CSIRO Division of Telecommunications and Industrial Physics. The authors thank the other members of the WLAN project: John Deane and Rob Shaw of CSIRO; John Dalton, Jeff Graham, and Charles Lee of Macquarie University.

The authors thank John Haddy of Macquarie University Electronics Technical Services, for his assistance with the design of the analog mezzanine card, and his informative discussions on PCB design. Macquarie University Electronics Technical Services fabricated the card.

7. References

[1] J.E. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. V. Touati, P. Boucard, "Programmable Active Memories: Reconfigurable Systems Come of Age," *IEEE Trans. On VLSI*, Vol. 4, No. 1, March 1996.
<URL:http://www.research.digital.com/SRC/pamette/>

[2] H. Touati, M. Shand, "PamDC: a C++ Library for the Simulation and Generation of Xilinx FPGA Designs," 23 September 1996.
<URL:http://www.research.digital.com/SRC/pamette/>

[3] D.J. Skellern and T.M.P. Percival, "High Speed Wireless LANs: Technologies for the Missing Link," *IEEE 1994 Microwave and Millimetre-Wave Monolithic Circuits Symposium.*
<URL:http://www.mpce.mq.edu.au/elec/networks/ wireless_res.html>

[4] P.J. Ryan, T.M.P. Percival, and D.J. Skellern, "A 16-Point FFT IC for Wireless Communication Systems,"

Workshop on Applications of Radio Science (WARS 95), 25-27 June 1995
<URL:http://www.rp.csiro.au/~pryan/papers/wars>

[5] D.J. Skellern, T.M.P. Percival, C. Lee, P.J. Ryan, T. McDermott, N.H.E. Weste, J.W. Dalton, J.A. Graham, T.F. Wong, A.F. Myles, "A mm-wave High Speed Wireless LAN for Mobile Computing – Architecture and Prototype Modem/Codec Implementation," *Hot Interconnects*, 1996.

[6] FCC, Notice of Proposed Rule Making, "Amendment of the Commission's Rules to Provide for Unlicensed NII/SUPERNet Operations in the 5 GHz Frequency Range," *ET Docket Number No. 94-102*, RM-8648 and RM-8653, Document FCC96-193, 25 April 1996.

[7] P. Alfke, "Synchronous and Asynchronous FIFO Designs," *Xilinx Application Note XAPP 051*, 17 September 1996.

[8] Bus Architecture Standards Committee of the IEEE Computer Society, "Draft Standard for a Common Mezzanine Card Family: CMC," P1386/Draft 2.0, 4 April 1995.

[9] Xilinx Corporation, "XC4000 Series Field Programmable Gate Arrays," 1996.
<URL:http://www.xilinx.com/ >

[10] S. Hauck, G. Borriello, C. Ebeling, "Springbok: A Rapid-Prototyping System for Board-Level Designs," *ACM/SIGDA 2nd International Workshop on Field-Programmable Gate Arrays*, February 1994.