A Quantitative Comparison of Reconfigurable, Tiled, and Conventional Architectures on Bit-level Computation

David Wentzlaff and Anant Agarwal
MIT Computer Science and Artificial Intelligence Laboratory
{wentzlaf, agarwal}@cag.csail.mit.edu

Abstract

General purpose computing architectures are being called on to work on a more diverse application mix every day. This has been fueled by the need for reduced time to market and economies of scale that are the hallmarks of software on general purpose microprocessors. As this application mix expands, application domains such as bit-level computation, which has primarily been the domain of ASICs and FPGAs, will need to be effectively handled by general purpose hardware. Examples of bit-level applications include Ethernet framing, forward error correction encoding/decoding, and efficient state machine implementation.

In this work [2] we compare how differing computational structures such as ASICs, FPGAs, tiled architectures, and superscalar microprocessors are able to compete on bit-level communication applications. A quantitative comparison in terms of absolute performance and performance per area will be presented. These results show that although modest gains (2-3x) in absolute performance can be achieved when using FPGAs versus tuned microprocessor implementations, it is the significantly larger gains (2-3 orders of magnitude) that can be achieved in performance per area that will motivate work on supporting bit-level computation in a general purpose fashion in the future.

1 Introduction

Recent trends in computer systems have been to move applications that were previously only implemented in hardware into software on microprocessors. This has been motivated by several factors. Firstly, microprocessor performance has been steadily increasing over time. This has allowed more and more applications that previously could only be done in ASICs and special purpose hardware, due to their large computation requirements, to be done in software on microprocessors. Also, added advantages such as decreased development time, ease of programming, the ability to change the computation in the field, and the economies of scale due to the reuse of the same microprocessor for many applications have influenced this change.

If we believe that this trend will continue, then in the future we will have one computational fabric that will need to do the work that is currently done by all of the chips inside of a modern computer. We have already seen this being done in current computer systems with the advent of all-software modems and software radios. Consequences of implementing all parts of a computer system in one computational fabric are that this fabric will have a significantly different application set than what current microprocessors are tuned for and will have higher computational requirements. In modern computer systems most of the helper chips are there to communicate with different devices and mediums. Examples include sound cards, Ethernet cards, wireless communication cards, memory controllers and I/O protocols such as SCSI and Firewire. For this reason, in this work we focus on Bit-level communication computation.

We investigate how bit-level computation in communication applications maps to differing architectures. Our methodology is to make very carefully optimized implementations on many differing architectures and then study the applications in terms of absolute *performance* and *performance per area*. The architectures examined include an IBM ASIC flow (SA-27E), a Xilinx FPGA (Virtex II), the Pentium 3, the Pentium 4, and the tiled Raw architecture.

Through this work we found some surprising results. To our surprise FGPAs did not have as large of an absolute performance gain versus microprocessors as we had expected, especially considering that the bit-level application mix heavily favored reconfigurable architectures. Rather, FPGAs only had a 2-3x performance improvement versus a microprocessor implementation. This was due largely to the high clock rates of microprocessors and that using lookup tables in a high speed microprocessor does a good job of emulating combinational logic. What we did find is that the real reason to implement bit-level communication processing in FPGAs or ASICs is their large area wins. For our application mix we found that ASICs provide 5-6 orders of magnitude and FPGAs provide 2-3 orders of magnitude better performance per area than software in a microprocessor.

2 Applications and Results

Two applications were examined in this study. Both are widely used in communications processing and have no clear notion of a word that gets computed on in contrast to standard integer applica-

An extended version of this work can be found in MIT-LCS-TR-944 at ftp://ftp.cag.csail.mit.edu/pub/raw/documents/Wentzlaff.2004.MIT_TR.Bit_level.pdf

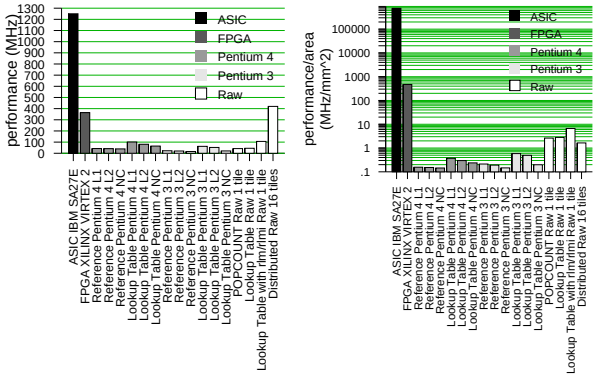

Figure 1. 802.11a Encoding Results.

tions. Verilog was used for the FPGA and ASIC implementations. Optimized 'C' was used for implementation on the Pentium processors, and parallel assembly code with 'C' was used for the Raw implementations. The processor based implementations rely heavily on lookup tables as an efficient emulation layer.

2.1 802.11a Convolutional Encoder

IEEE 802.11a [1] is the wireless Ethernet standard used in the 5GHz ISM band and provides for transmission of information up to 54Mbps. In this study we examine the convolutional encoder from 802.11a. A convolutional encoder is a method of forward error correction which consists of a shift register that the input data is passed sequentially though. The output encoded data is computed by tapping off the shift register at various locations and computing some combinational function of the tap locations. The studied convolutional encoder generates two bits for each input bit and is characteristic of convolutional encoders used in many digital wireless communication systems. Figure 1 presents the results for 802.11a encoding.

2.2 8b/10b Block Encoder

8b/10b encoding is a byte oriented binary transmission code which translates 8 bits at a time into 10-bit codewords. This particular block encoder was designed by Widmer and Franszek [3] and has nice features such as being DC balanced, detection of single bit errors, clock recovery, addition of control words, and ease of implementation in hardware. This encoder is a line encoder used for both fiber-optic and wired applications. Most notably it is used to encode data right before it is transmitted in fiber-optic Gigabit Ethernet. This application is not inherently parallelizable because a running parity of transmitted bits is computed and then used to alter subsequent transmitted bits. Thus a feedback loop is introduced. Figure 2 presents the results for 8b/10b encoding.

3 Conclusion

From our experimental results we have found some interesting trends that hold for these two applications:

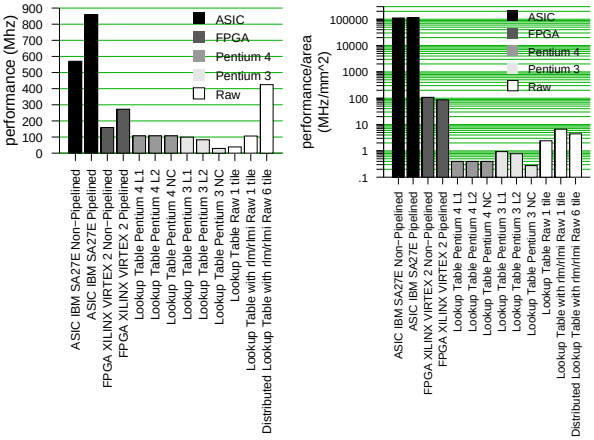

Figure 2. 8b/10b Encoding Results.

1. ASICs provide a 2-3x absolute performance improvement over an FPGA implementation.
2. FPGAs provide a 2-3x absolute performance improvement over a microprocessor implementation.
3. ASICs provide 5-6 orders of magnitude better performance per area than software implementation on a microprocessor.
4. FPGAs provide 2-3 orders of magnitude better performance per area than software implementation on a microprocessor.
5. Parallel implementations on Tiled architectures yield competitive absolute performance to that of FPGAs but use at least an order of magnitude more area to do so.

In this study we found that it is usually possible to use one grain size to run an application with a smaller grain size without a significant loss in absolute performance through some emulation mechanism. One effective emulation mechanism is the ability to use a microprocessor's cache as a lookup table to substitute for custom logic. But, unfortunately these forms of grain size mismatch cause large, multiple orders of magnitude, inefficiencies when it comes to area utilization and possibly power. This motivates the study of integration of fine grain computational structures such as FPGAs with architectures that have larger grain size such as word-based microprocessors to be able to support both bit-level and general purpose computation in an area efficient manner.

References

[1] IEEE. *IEEE Standard 802.11a-1999 Supplement to IEEE standard for Information Technology*, 1999.

[2] D. Wentzlaff and A. Agarwal. A quantitative comparison of reconfigurable, tiled, and conventional architectures on bit-level computation. Technical Report MIT-LCS-TR-944, Massachusetts Institute of Technology, Apr. 2004.

[3] A. X. Widmer and P. A. Franaszek. A DC-balanced, partitioned-block, 8b/10b transmission code. *IBM Journal of Research and Development*, 27(5):440–451, Sept. 1983.