Digital Signal Processing with General Purpose Microprocessors, DSP and Reconfigurable Logic

Steffen Köhler, Sergej Sawitzki, Achim Gratz, and Rainer G. Spallek

Institute of Computer Engineering
Dresden University of Technology
D-01062 Dresden, Germany
e-Mail: {stk,sawitzki,gratz,rgs}@ite.inf.tu-dresden.de

Abstract. This paper compares selected digital signal processing algorithms on a variety of computing platforms in terms of achievable performance and cost. The experiments were carried out on a standard PC platform, DSP, a RISC microcontroller and on Xilinx XC4013XL FPGA. Our results confirm that general purpose microprocessors are not well suited to these tasks. Both DSP and FPGA achieve higher performance and/or better cost/performance ratios at the expense of lesser generality and a more complicated development cycle. The porting of the algorithms to DSP and FPGA requires about the same amount of work, whereby the cost/performance ratio of the reconfigurable FPGA solution is very attractive.

1 Introduction

Digital signal processing still requires special hardware solutions to achieve an acceptable level of performance. This fact is underlined by the rapid expansion of the DSP market, especially in the embedded domain, and confirmed by the results achieved for general purpose microprocessors in this paper. If the reconfigurability of FPGA is employed, very elegant and cost effective solutions are possible for applications that are otherwise only accessible by high-end DSP. The reconfigurable FPGA is more flexible than the DSP and especially interesting for exploration of newly developed algorithms that are not yet supported by special features of DSP. While dynamic reconfiguration may extend the reach of the FPGA solution even further, the added complexity seems to disfavour that approach. For PC I/O-cards with integrated signal processing, reconfiguration at startup seems to suffice for many conceivable applications.

2 Hardware Equipment

Our experiments were conducted on a standard PC platform with four different system configurations as described below.

General Purpose CPU. Two conventional PC systems were used to determine the performance of general purpose processors (AMD K6 CPU clocked at 200 MHz, Dual Intel Pentium II clocked at 350 MHz. All test programs were ran under the Linux operating system with algorithms coded in C (singlethreaded).

Table 1. DSP Algorithm Run Times (in milliseconds) and Speedups

	FFT (512pt)		LPC Filter		Viterbi Decoder	
	rt/sp		rt/sp		rt/sp	
AMD K6 200	1.2	8.33	0.00100	10.00	6.95	1.00
Pentium II	0.5	20.00	0.00059	16.94	3.20	2.17
DSP 56302	0.6	16.67	0.00030	33.33	1.40	4.96
TMS 320C50	2.2	4.55	0.00100	10.00	3.00	2.32
ARM stand-alone	10.0	1.00	0.01000	1.00	4.80	1.45
ARM with FPGA	1.4	7.14	0.00060	16.67	0.27	25.74

Digital Signal Processors (DSP). We used a PCI extension board with a Motorola DSP 56302 (24 bit fixed point DSP) operating at 85 MHz [1] and an ISA extension board with Texas Instruments TMS 320C50 clocked at 40 MHz (16 bit fixed point DSP) [3]. Both boards were plugged into the K6-based PC which was acting as a host running the Linux operating system and accessing the DSP via native drivers. The algorithms were completely rewritten in assembly language for each DSP.

RISC Microcontroller and FPGA. We used an additional PCI extension card containing an 32 bit ARM microcontroller produced by Atmel and clocked at 8 MHz and two Xilinx XC4013XL FPGA operating at the same clock frequency [2]. In the first experiment the algorithms were compiled with the native ARM compiler and ran on the microcontroller alone. In the second experiment the microcontroller was used as a host to program the FPGA and exchange data with them while the cores of the algorithms were implemented in Handel-C and downloaded to one of the FPGA. To simplify the synchronization, the ARM waited for the FPGA to complete it's respective operation. The development environment was ran under Microsoft Windows 95.

3 Benchmark Suite and Experimental Results

We considered a range of applications which usually serve as benchmarks for DSP vendors and extracted the algorithmic cores. Note that XC4013 devices consist of 576 cells each, and 113 of them (on average) are consumed by the host interface.

Fast Fourier Transform [4]. A one-dimensional FFT was ran for an input set of 512 points. We limited all computations to 16-bit wide operands due to the hardware available. In the case of FPGA the butterfly operation was implemented in hardware consuming 526 CLBs.

LPC Synthesis Filter [6]. Voice synthesis using the LPC10 compression algorithm served as a second test example. We used the FPGA to implement a 10-stage lattice filter (471 CLBs) to accelerate the computations.

Viterbi Decoder [5]. We implemented a $\frac{1}{3}$-rate convolutional decoding algorithm. In the reconfigurable case the Trellis look-up operation was executed in the FPGA while the state matrix and Trellis graph were stored in the SRAM on the FPGA board. 349 CLBs were used to implement the corresponding circuit.

Table 1 summarizes the results of our measurements. For the PC platforms, best timings are shown. It is impossible to eliminate the wide fluctuations in timing val-

Table 2. Cost/Performance Trade-Off

Circuit	Price ($)	Performance/$ (arb. units)	
		achieved	achievable
AMD K6 200	65	99	—
Pentium II	250	52	—
DSP 56302	30	610	—
TMS 320C50	10	560	—
XC4013XL-3	60	280	1120

ues of the general purpose architectures due to interrupt handling and related cache conflicts, which degrade the sustainable performance significantly and make real-time digital signal processing with the host CPU an elusive goal on the PC platform.

4 Conclusions

Although it seems that DSP provide better performance in two of three cases, as Table 1 suggests, the picture changes if the post layout timing data of the designs are considered. Note that Xilinx has announced a new family of XC4000E compatible devices that are claimed to be twice as fast as the currently available parts. Unfortunately, the limitations of the development board do not allow to take advantage of the maximum speed obtainable with even the slower (and cheaper) devices on the FPGA board. The trade-off between cost and performance is another interesting point of view. Setting the circuit prices in relation to the performance data in Table 1 shows that FPGA provide the most performance per dollar on average (at least in best case), see Table 2. This comparison can be continued at the system level with the same result.

Our experience with the Handel-C compiler has shown that typical DSP algorithms can be ported in a reasonable amount of time (2–3 weeks) with good results. To achieve good performance with a DSP, the algorithms often have to be re-coded in assembler, which is a time-consuming process and requires a lot of special knowledge about the DSP used. This suggests that time-to-market for the DSP and the FPGA solution is similar and the better cost/performance ratio of the FPGA solution is very attractive.

References

1. Motorola Inc. DSP56302 24-Bit Digital Signal Processor User's Manual. Austin, TX, 1996
2. Snook, M. ASPIRE Development Board Datasheet. Advanced RISC Machines Ltd., 1998
3. Texas Instruments Inc. TMS320C5x User's Guide. Houston, TX, 1993
4. Mitra, S.K. and Kaiser, J.F. (ed.) Handbook for Digital Signal Processing. John Wiley & Sons, Inc., New York, 1993
5. Benedetto, S., Biglieri, E., and Castellani, V. Digital Transmission Theory. Prentice-Hall, p. 407, 1987
6. Tremain, E.T. The Government Standard Linear Predictive Coding Algorithm: LPC-10. Speech Technology Magazine, pp. 40-49, 1982