

Energy Efficiency of FPGAs and Programmable Processors for Matrix Multiplication

Ronald Scrofano
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
rscrofan@halcyon.usc.edu

Seonil Choi and Viktor K. Prasanna
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089
{seonil, prasanna}@halcyon.usc.edu

Abstract

Advances in their technologies have positioned FPGAs and embedded processors to compete with digital signal processors (DSPs). In this paper, we evaluate the performance in terms of both latency and energy-efficiency of FPGAs, embedded processors, and DSPs in multiplying two $n \times n$ matrices. As specific examples, we have chosen a representative of each type of device. Our results show that the FPGAs can multiply two $n \times n$ matrices with both lower latency and lower energy consumption than the other two types of devices. This makes FPGAs the ideal choice for matrix multiplication in signal processing applications.

1. Introduction

With improvements in FPGA technology, such as increased logic and embedded multipliers, FPGAs can be considered for computationally demanding applications such as those in signal processing. FPGAs, DSPs, and embedded RISC processors are all candidates for executing signal processing applications.

Traditionally, the performance metrics for signal processing and, indeed, most processing in general, have been latency and throughput. However, the use of mobile devices and sensors makes energy-efficiency another very important metric. It is vital, then, that when designing a sensor or a mobile device the designer pick the most energy-efficient platform on which to base it. The designer can now choose between FPGAs, embedded processors, and DSPs. In this paper, we use high-level estimation to determine which of these platforms provides the most energy-efficient solution for matrix multiplication, which is at the heart of many signal processing applications.

The remainder of this paper is organized as follows. Section 2 explains our method of comparison. Section 3 shows

the results of our comparisons. Section 4 presents a summary of our work.

2. Methodology

To compare FPGAs, DSPs, and embedded processors, we have picked representative devices from each platform and analyzed their performance in executing matrix multiplication. Our device selection is outlined below, followed by a description of our method for finding results for comparison from each device.

2.1. Device selection

Table 1 shows some important features of each of the representative devices which are described below.

The Xilinx Virtex-II Pro serves as the representative FPGA. This is Xilinx's highest density, most advanced FPGA [14]. Of particular interest are the up to 556 18×18 -bit embedded multipliers present on the FPGA. Each of these can run at a clock frequency over 300 MHz.

The Texas Instruments (TI) TMS320C6415 is used as the representative DSP because it is TI's highest performance fixed point DSP [13]. The fixed point nature of this DSP makes it suitable for comparison with FPGAs in which most arithmetic is fixed point. This DSP can be run at 400-, 500-, or 600 MHz. Also, note that it can perform *four* 16×16 -bit or *eight* 8×8 -bit multiplications per clock cycle.

The representative embedded processor is the XScale-based PXA250. This is Intel's newest embedded processor [6]. Key to its use in signal processing applications is the inclusion of a dual-MAC instruction. This instruction can execute two multiply and accumulates in parallel. Further, since this instruction takes longer than the standard instructions, the dual-MAC has a separate pipeline for its execution.

Table 1. Device properties that relate to energy-efficient matrix multiplication

Device	Technology	Clock Rates(s) (MHz)	Multiplies/Clock (precision)	Low-Power Features
Virtex-II Pro	0.13 μ m	300	556 (18 \times 18-bit)	clock gating
TMS320C6415	0.12 μ m	400, 500, 600	8 (8 \times 8-bit)	4 low-power modes
PXA250	0.18 μ m	400	2 (16 \times 16-bit)	2 low-power modes, automatic clock gating

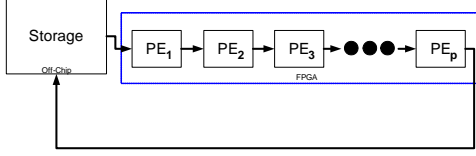


Figure 1. Linear array architecture with off-chip storage

2.2. Latency and energy computations

To calculate energy consumption in $n \times n$ matrix multiplication, the first task is to calculate latency. In each case, latency is first calculated in cycles then converted to seconds by dividing by the clock frequency. Once latency has been calculated, we can use it and the power dissipation of each device to calculate the energy consumption. Each type of device requires a different method for performing the aforementioned two tasks.

2.2.1. FPGA. We choose a linear array of processing elements (PEs) as the architecture and choose as the algorithm that which is presented in [8] as that paper's Corollary 1. With the linear array architecture, data can be stored either on- or off-chip. In the latency and energy estimations, we use off-chip storage of the input matrices. A diagram of the linear array architecture with off-chip data storage is given in Figure 1. The PEs in the linear array architecture each have multipliers, registers for storing input to the multipliers, and SRAMs for storing intermediate results.

The algorithm computes the matrix product efficiently, both in terms of latency and energy, by cleverly moving the entries of the input matrix through the linear array. As shown in [8], the equation for calculating the latency required to multiply two $n \times n$ matrices is

$$L = \left(\frac{n}{p}\right)^3 (p^2 + 2p + 1) \quad (1)$$

cycles, where p is the number of PEs in the linear array.

It is shown in [8] that to execute the algorithm a total of p multipliers, $4p$ registers, and 2 I/O ports must be used. Additionally, SRAM of size $p \lceil \frac{p}{16} \rceil$ must be available. The total power is the sum of the power consumed by the interconnection network, the I/O, and the components in the PEs. It is assumed that location on the FPGA does not affect the power consumed by any of these modules. Energy is equal

to the product of latency and average power consumption. Thus, the energy for performing the multiplication is given by

$$E = L \left(pP_M + p \left\lceil \frac{p}{16} \right\rceil P_S + 4pP_R + 2P_{IO} + P_{IC} \right) \quad (2)$$

where L is the latency when p PEs are used; P_M , P_S , P_R , P_{IO} , and P_{IC} are the powers used by a multiplier, an on-chip memory cell, a register, the data transfer, and the interconnection network, respectively. The values of the power variables are given in Table 2 [8]. These values were obtained by performing low-level simulations of the Virtex-II FPGA.

Each PE component was modeled in VHDL and then synthesized using Xilinx XST. After synthesis, the design was placed and routed using Xilinx PAR. The output from the place and route tool was converted back to VHDL and simulated using ModelSim 5.5e with input stimuli that had an average switching activity of 50%. The output from the simulation and the output from the place and route tool were used as input to Xilinx XPower. XPower was used to get the power data for each component of a PE. This data is shown in Table 2.

Table 2. Values for power parameters in the energy consumption equation

Variable	Power (mW)
P_M	17.00
P_S	8.39
P_R	2.34
P_{IO}	11.31
P_{IC}	10.00

The Virtex-II FPGA has a clock frequency of 150 MHz, while the Virtex-II Pro has a clock frequency of 300 MHz. Consequently, we scale our energy results for the Virtex-II by a factor of $\frac{300}{150}$ to account for the higher power used by a higher clock frequency. However, we have ignored any gain in energy efficiency from the Virtex-II Pro's use of 0.13 μ m technology, as opposed to the Virtex-II's use of 0.15 μ m technology.

2.2.2. DSP. TI provides an equation for finding the latency, in cycles, of using its optimized algorithm for matrix multiplication on the TMS320C6415. For $n \times n$ matrix multiplication, this equation is

$$L = \left[(2n + 18) \frac{n}{4} + 7 \right] \frac{n}{2} + 5 \quad (3)$$

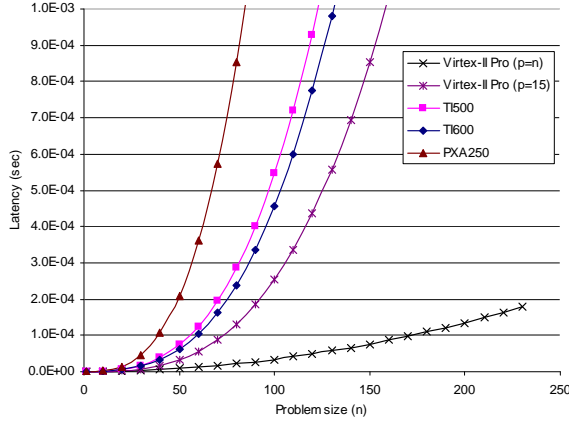


Figure 2. Latency vs. problem size

cycles. The method for calculating the energy consumption of the TMS320C6415 is to find its average power dissipation and multiply it by the latency. TI classifies two activity models for the DSP: the high activity model and the low activity model [13]. High activity represents the time the DSP spends performing compute-intensive, optimized algorithms, such as a FIR filter. Low activity represents the time spent performing less compute-intensive tasks, such as setting up registers or executing less optimized algorithms. TI then presents data for two power activity categories: 75% high/25% low and 50% high/50% low. We have determined that matrix multiplication more closely falls into the 75% high/25% low category because it is an optimized algorithm, spending more of its time in high activity. Table 3 shows the average power consumption for each category when the DSP is run at clock frequencies of 500 and 600 MHz.

Table 3. Average power for the TMS320C6415

Clock Frequency(MHz)	Operating Voltage (V)	Power (W)	
		50%/50%	75%/25%
500	1.2	1.04	1.19
600	1.4	1.47	1.61

2.2.3. Embedded Processor. In our calculation of latency, we have assumed that, because of the pipelined dual-MAC instruction, the PXA250 can perform 1 dual-MAC per clock cycle (i.e. 2 MACs/cycle). We further assume that these dual-MAC instructions will dominate the computation time. In the multiplication of two $n \times n$ matrices using the standard matrix multiplication algorithm, there are n^3 MAC operations. These operations can be parallelized and the hardware of the PXA250 supports two MAC operations simultaneously. Therefore, the latency estimate for multiplying together two $n \times n$ matrices is given by

$$L = \frac{n^3}{2} \quad (4)$$

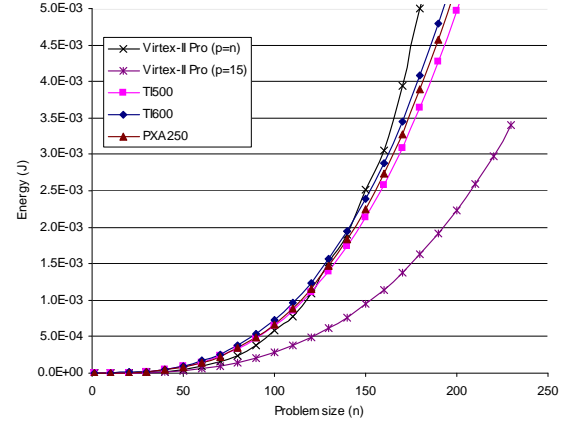


Figure 3. Energy vs. problem size

cycles. Intel has provided power dissipation data for the PXA250 running at a clock frequency of 300 MHz. So, we use this power data and a clock frequency of 300 MHz in our comparison, even though the device is capable of running at up to 400 MHz. According to [6], at 300 MHz, the PXA250 dissipates 400 mW of power.

3. Results

Figure 2 shows the latencies, in seconds, of each device when multiplying two $n \times n$ matrices, $1 \leq n \leq 224$.

Equation 4, used to calculate the latency for the PXA250, provides only a lower bound on latency because it ignores all instructions other than the MAC instructions. Therefore, the actual latency of the PXA250 is most likely much higher than that shown here. Yet, the PXA250 is still the slowest in performing the multiplication. The PXA250, it seems, is not a good choice for multiplying together two $n \times n$ matrices if low latency is the highest priority.

Two sets of results are presented for the TMS320C6415: one set each for when it is run at 500 MHz (labeled TI500) and when it is run at 600 MHz (labeled TI600). Two sets of results are also presented for the Virtex-II Pro. In one set the number of multipliers equals the number of rows in the matrices being multiplied ($p = n$). This configuration leads to the lowest latency. In the other set of results, $p = 15$. We have chosen 15 PEs because this configuration is one of the most optimal in terms of energy consumption [8]. Despite this low energy consumption, the configuration with $p = 15$ still has lower latency than the DSP and the embedded processor.

Figure 3 shows the energy consumed by each device when multiplying two $n \times n$ matrices, $1 \leq n \leq 224$.

The Virtex-II Pro with $p = n$, the TMS320C6415 running at both 500 and 600 MHz, and the PXA250 all consume about the same amount of energy. There is too much

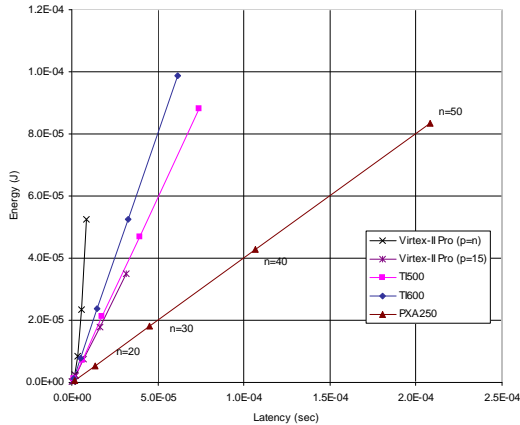


Figure 4. Energy vs. latency

experimental error to say conclusively which consumes the most energy. However, it is clear that the Virtex-II Pro with $p = 15$ consumes much less energy than any of the other devices or the other Virtex-II Pro configuration. Thus, it is clear that the FPGA can perform both the most rapid and the most energy-efficient multiplication of two $n \times n$ matrices.

Figure 4 shows the energy consumption compared to the latency for problem sizes ranging from $n = 0$ to $n = 50$. The points on the graph represent problem sizes that are multiples of 10.

In Figure 4, it is desirable for a device's curve to remain as close to the origin as possible. That is, the device that has the lowest latency and the lowest energy consumption for a given problem size will be best for that problem size. The line for the Virtex-II Pro with $p = n$ has very low latency, but when n becomes large, the energy consumption increases significantly. The TMS320C6415 run at either 500 or 600 MHz shows a large increase in both energy and latency as the problem size increases. The PXA250 has the longest latency of any of the devices being compared. Its energy consumption is slightly lower than that of the TMS320C6415 running at 500 or 600 MHz. So, by choosing the PXA250 over the TMS320C6415, the designer could make a slight gain in energy efficiency while sacrificing quite a bit of speed. The lowest latency and energy values are found in the curve for the Virtex-II Pro with $p = 15$. Clearly, the Virtex-II Pro with the linear array architecture using 15 PEs is the best in this category.

4. Conclusion

In this paper, we have shown that the multiplication of two $n \times n$ matrices can be done most efficiently by an FPGA. We have shown this result by choosing a representative FPGA, DSP, and embedded processor. For each device used in comparison, we have given our rationale in choos-

ing that device as representative. Finally, we have presented our method for estimating the energy and latency each device requires to multiply two $n \times n$ matrices.

5 Acknowledgements

This work is part of the USC PACMAN project (<http://pacman.usc.edu>). This project is supported by the US DARPA Power Aware Computing and Communications program.

References

- [1] Altera Corporation, <http://www.altera.com>.
- [2] C. Chen and M. Sarrafzadeh. An effective algorithm for gate-level power-delay tradeoff using two voltages. In *Proceedings of the International Conference on Computer Design*, 1999.
- [3] A. DeHon. The density advantage of configurable computing. *IEEE Computer*, 33(4):41–49, April 2000.
- [4] J. A. B. Fortes, K. S. Fu, and B. Wah. Systematic approaches to the design of algorithmically specified systolic arrays. In *Proceedings of the 1985 International Conference on Acoustics, Speech and Signal Processing*, 1985.
- [5] A. D. Garcia, W. P. Burlinson, and J. L. Danger. Reducing the power consumption in FPGAs with keeping a high performance level. In *Proceedings. IEEE Computer Society Workshop on VLSI, 2000*, pages 47–52, April 2000.
- [6] Intel Corporation, <http://www.intel.com>.
- [7] J. Jang, S. Choi, and V. K. Prasanna. Area and time efficient implementation of matrix multiplication on FPGAs. In *Proceedings of the IEEE International Conference on Field-Programmable Technology*, 2002. (to appear).
- [8] J. Jang, S. Choi, and V. K. Prasanna. Energy efficient matrix multiplication on FPGAs. In *Field-Programmable Logic and Applications*, volume 2483 of *Lecture Notes in Computer Science (LNCS)*. Springer Verlag, 2002.
- [9] O. Mencer, M. Morf, and M. J. Flynn. Hardware software tri-design of encryption for mobile communication units. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 3045–3048, 1998.
- [10] A. Raghunathan, N. K. Jha, and S. Dey. *High-level Power Analysis and Optimization*. Kluwer Academic Publishers, 1998.
- [11] L. Shang, A. Kaviani, and K. Bathala. Dynamic power consumption in Virtex-II FPGA family. In *International Symposium on Field Programmable Gate Arrays*, 2002.
- [12] R. Tessier and W. Burlinson. Reconfigurable computing and digital signal processing: A survey. *Journal of VLSI Signal Processing*, May/June 2001.
- [13] Texas Instruments, Inc., <http://www.ti.com>.
- [14] Xilinx, Inc., <http://www.xilinx.com>.
- [15] G. Yeap. *Practical Low Power Digital VLSI Design*. Kluwer Academic Publishers, 1998.