# Minimizing Energy Dissipation of Matrix Multiplication Kernel on Virtex-II

Seonil Choi[a], Viktor K. Prasanna[a], Ju-wook Jang[b]

[a]Electrical Engineering-Systems, University of Southern California, Los Angeles, USA
[b]Electronic Engineering, Sogang University, Seoul, Korea

## ABSTRACT

In this paper, we develop energy-efficient designs for matrix multiplication on FPGAs. To analyze the energy dissipation, we develop a high-level model using domain-specific modeling techniques. In this model, we identify architecture parameters that significantly affect the total energy (system-wide energy) dissipation. Then, we explore design trade-offs by varying these parameters to minimize the system-wide energy. For matrix multiplication, we consider a uniprocessor architecture and a linear array architecture to develop energy-efficient designs. For the uniprocessor architecture, the cache size is a parameter that affects the I/O complexity and the system-wide energy. For the linear array architecture, the amount of storage per processing element is a parameter affecting the system-wide energy. By using maximum amount of storage per processing element and minimum number of multipliers, we obtain a design that minimizes the system-wide energy. We develop several energy-efficient designs for matrix multiplication. For example, for $6 \times 6$ matrix multiplication, energy savings of upto 52% for the uniprocessor architecture and 36% for the linear arrary architecture is achieved over an optimized library for Virtex-II FPGA from Xilinx.

**Keywords:** matrix multiplication, domain specific modeling, energy efficient design, FPGAs.

## 1. INTRODUCTION

With the advent of multi-million gate reconfigurable devices, high-level algorithms are being mapped onto FPGAs. Some reconfigurable system-on-chip (SOC) platforms offer implementation of signal processing algorithms directly in hardware. These solutions offer improved performance capability compared with DSP-based solutions. FPGAs have also become attractive platforms to meet the processing requirements of mobile devices. As mobile devices operate in power constrained environments, in addition to time performance, energy efficiency has become a critical performance metric for FPGA-based designs.[12]

Matrix multiplication is a frequently used kernel operation in many mobile applications. Most of the previous work in performing matrix multiplication on FPGAs has focused on latency optimization.[2, 6] We develop designs which minimize the energy dissipation and offer trade-offs for performing matrix multiplication on commercially available FPGAs. Our effort is focused on algorithmic techniques to improve energy performance instead of low-level (implementation level) optimizations.

To develop energy-efficient designs, we use domain-specific modeling techniques that were proposed in Ref. 5. First, we define a domain as a family of architectures and corresponding algorithms to implement matrix multiplication. Then, the architecture parameters that significantly affect the system-wide energy of the domain are identified. The resulting high-level model has various parameters such as operating frequency, memory capacity, I/O bandwidth, and precision that allow us to choose appropriate settings to minimize the energy. The high-level model also provides an energy function to rapidly estimate the system-wide energy of a design.

Further author information: (Send correspondence to Viktor K. Prasanna)
Viktor K. Prasanna: E-mail: prasanna@usc.edu, Telephone: 1 213 740 4483, Fax: 1 213 740 4418, Address: 3740 McClintock Ave. EEB-200C, Los Angeles, CA 90089-2562, USA.
Seonil Choi: E-mail: seonil@halcyon.usc.edu
Ju-wook Jang: E-mail: jjang@sogang.ac.kr

The energy function provides an energy distribution profile that can be used to identify possible energy savings in a design.

Several designs for matrix multiplication were proposed in Ref. 5, 11. The designs in both papers focus on minimizing the energy dissipated in the processing elements and I/O buses. However, if matrix multiplication is part of a large application, the matrix data is usually stored in the memory banks on the FPGA. Storing and accessing data from the memory banks dissipates significant amount of energy. In this paper, we also consider the energy dissipated in the memory banks. The I/O complexity affects the amount of memory access and as a result significantly affects the system-wide energy dissipation. Virtex-II series have Block SRAMs (of size 18K bits each). We utilize this feature for implementing the memory banks. We also propose a compact energy-efficient design which uses approximately the same area as a design in the Xilinx library.

We develop designs for two domains and compare their performance with a vendor specified matrix multiplication kernel to demonstrate the energy performance improvements. Our designs demonstrate superior energy dissipation performance compared with designs in the literature that use approximately the same area or execute in the minimum time. The design in the first domain occupies approximately the same area as the Xilinx design[17] and achieves 52% energy reduction compared with a solution based on the Xilinx design. The designs in the second domain are generated for various problem sizes. The latency of these designs is smaller than those in the first domain. These designs dissipate 26-36% less energy compared with the solutions based on the Xilinx design.

The rest of the paper is organized as follows. The next section discusses energy modeling and our design methodology. Section 3 describes algorithms and architectures for energy-efficient designs for matrix multiplication using two domains. We derive system-wide energy functions and illustrate various trade-offs. We conclude in Section 4.

## 2. DOMAIN SPECIFIC MODELING AND DESIGN METHODOLOGY

We follow the design methodology presented in Ref. 11. The methodology generates a set of designs which provide trade-offs among energy, latency, and area. The design methodology is illustrated in Figure 1. In the following, for the sake of completeness, we briefly outline the methodology.

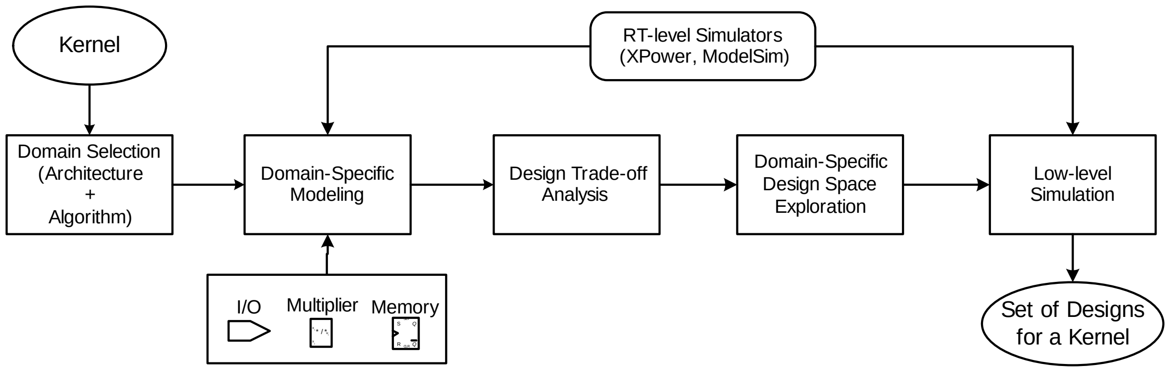
Kernel
RT-level Simulators (XPower, ModelSim)
Domain Selection (Architecture + Algorithm)
Domain-Specific Modeling
Design Trade-off Analysis
Domain-Specific Design Space Exploration
Low-level Simulation
I/O
Multiplier
Memory
Set of Designs for a Kernel

**Figure 1.** Domain-specific modeling and design methodology[11]

First, we select a domain. A domain corresponds to a family of architectures and algorithms that implements a given kernel. For matrix multiplication, several families of architectures are known. We choose two architectures: a cache based uniprocessor and a linear array of processors. For each family of architectures, several algorithms can be used to perform matrix multiplication.

Domain-specific modeling facilitates development of a high-level model for a specific domain. Detailed knowledge of the domain is exploited to identify the architecture parameters for analyzing the energy dissipation of the resulting designs in the domain. The high-level model consists of *Relocatable Modules* (*RModule*) and

*Interconnect* as the basic structural components. In addition, it contains several architecture parameters such as operating frequency ($f$), precision ($w$), size of memory ($s$), power states ($ps$), etc. that are associated with each component and a set of power functions, one for each power state of a component. Domain-specific details identify the range of values for each model parameter, thus reducing the design space. For example, the maximum and minimum problem sizes influence several model parameters if a performance constraint such as maximum tolerable latency has to be met. Additional details of the modeling can be found in Ref. 5.

The high-level model contains several functions such as power functions associated with each component and performance functions for energy, latency, and area. These functions can be analyzed to understand the trade-offs among different performance metrics (energy, latency, and area). These functions also capture the sensitivity of performance metrics with respect to various architecture parameters.

During design space exploration (DSE), the domain-specific design space is traversed to identify designs based on some specified selection criteria. For example, minimum energy dissipation with minimum $area \times latency$ is a possible selection criteria. Our methodology does not propose a general DSE technique as the technique depends on the domain. As our domain-specific modeling technique constraints the design space specific to the domain, it typically does not result in a very large design space.

Low-level simulation is performed on the designs selected by the DSE step. The DSE uses various functions to evaluate the designs. While the estimates are reasonably accurate,[5] we use low-level simulation for two different purposes. Our study shows that the error due to high-level estimation is typically in the range of $\pm 10\%$. Therefore, low-level simulation is necessary to select a design if two candidate designs are within 10% of each other with respect to a performance metric. The other use is to verify the performance estimates obtained using the functions provided in the high-level model.

Low-level simulation for energy estimation of a design proceeds as follows. The design specified in VHDL is synthesized using Synopsys FPGA Express and Xilinx XST on the Xilinx ISE 4.1i design environment. The place-and-route file (.ncd file) is obtained for the target FPGA device, Virtex-II XC2V1500. Mentor ModelSim 5.5e is used to simulate the module and generate simulation results (.vcd file). These two files are then provided to the Xilinx XPower tool to estimate the energy dissipation. The switching activity for the input of the design can be provided by the designer or specified as some default values.

## 3. ENERGY-EFFICIENT DESIGNS FOR MATRIX MULTIPLICATION

Based on our design methodology, we develop energy-efficient designs for matrix multiplication using two domains. Two architecture families, a uniprocessor architecture and a linear array architecture, are chosen. We refer to "system-wide energy" as the total energy dissipated to perform matrix multiplication on FPGAs.

### 3.1. Uniprocessor Architecture

We define a uniprocessor (PE) implementing the "usual" block matrix multiplication as the first domain. This domain uses a single multiplier and results in compact energy-efficient designs. There are two possible scenarios: on-chip design and off-chip design. If the matrix multiplication kernel is a stand-alone application, all matrix data are stored in an external memory outside the FPGA. We refer to such a design as off-chip design. If the matrix multiplication kernel is one of many kernels in an application, it is desirable to have the matrix data reside (in a Block SRAM) on the device. We refer to such a design as on-chip design. The Block SRAM is a dedicated on-chip memory in Virtex-II and is usually used for storing intermediate data between (kernel) computations. Figure 2 shows our target architectures.

In the off-chip design, the PE has one MAC (multiplier and accumulator), a cache (local buffer) of size $c$, and I/O ports (See Figure 2 (a)). Each word of cache is three 8-bit registers. The data matrices are stored in an external memory. For $n \times n$ matrix multiplication, the computational complexity of the algorithm is $O(n^3)$. Block matrix multiplication (BMM) is performed with block size $\sqrt{c} \times \sqrt{c}$. The I/O complexity (amount of traffic between the PE and external memory) is $O(n^3/\sqrt{c})$. It can be observed that a large cache decreases the I/O traffic and as a result improves the energy dissipation in performing I/O.

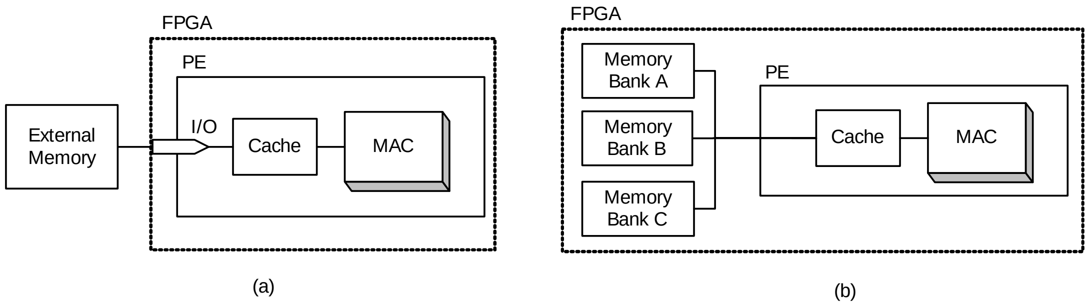

**Figure 2.** Uniprocessor architecture: (a) off-chip design and (b) on-chip design.

In the on-chip design, the PE has one MAC, a cache of size $c$, and three memory banks for storing three matrices (See Figure 2 (b)). BMM is performed with block size $\sqrt{c} \times \sqrt{c}$. The energy for I/O (outside the device) is not included, but the energy dissipated in the three memory banks is considered. The read/write access frequency of the memory banks depends on the traffic between the memory banks and the PE. It can be observed that as the cache size increases, the number of memory bank accesses decreases and as a result the energy dissipated in the memory banks reduces.

### 3.1.1. Identifying Components and Parameters

We identified four components: MAC, cache, and the memory banks as RModules, the I/O as an Interconnect. The RModules have $w$ bit precision. We set $w = 8$, the same precision used in the Xilinx library.[17] The cache size ($c$) is the only parameter that can be varied at design time.

The component specific power functions for MAC ($M.p$), cache ($R.p$), I/O ($IO.p$), and memory bank ($MEM.p$) were obtained through low-level simulation. Sample VHDL code for each component was synthesized and simulated in the Xilinx ISE4.1i environment. In the simulation, 1000 sets of 8 bit input data (waveforms) were randomly generated. The data sets were fed to the ModelSim to obtain the simulation output (.vcd files) based on the place-and-routed file of a component (.ncd file). The switching activity of the input sets to each component was found to be on an average 25%. However, the switching activity in a circuit depends on the behavior of a component (.ncd file). Also, by simulating design using the above data sets, we obtained the average power dissipation. The power dissipation was measured using XPower.

For implementing the MAC in Virtex-II, there are two design choices: a CLB-based multiplier and a dedicated multiplier. A dedicated multiplier is a stand-alone ASIC-based multiplier. A CLB-based multiplier is built using CLBs and it was observed that it consumes more power than a dedicated multiplier. There are two design choices for implementing the cache using CLBs. If the cache size is small, the cache can be realized using CLBs configured as register modules. Larger cache can be realized using CLBs configured as SRAM.[17] However, a SRAM-based cache can only be configured to be a multiple of 16 bytes. We noticed that for $c > 6$, the SRAM-based cache consumes less power than the register-based cache of the same size. $M.p$ and $IO.p$ are constants. The power function for the register-based cache is: $R.p(c) = 22.88 + 2.34c$ (mW). The power function for the SRAM-based cache is: $R.p(c) = 33.56 + 8.36 \lceil c/16 \rceil$ (mW). The memory bank is implemented using Block SRAM in Virtex-II. The power state of Block SRAM can be controlled by gated clocking. However, there is not much difference ($< 2\%$) in the power dissipation between the on and off states. Instead, the power dissipated in Block SRAM depends mainly on the access frequency ($fm$) of the memory bank. A Block SRAM can be configured to be a multiple of 2 K bytes with 8 bit precision. The power function for 2 K byte Block SRAM is: $MEM.p(fm) = 0.28 + 102.92fm$ (mW).

### 3.1.2. System-wide Energy Function

We now consider the system-wide energy dissipated by the design. In both the on-chip and off-chip designs, the amount of computation performed by the MACs is the same and the MACs dissipate the same amount of energy. For the off-chip design, we do not consider the energy dissipated in the external memory. The system-wide energy function ($SE$) for performing $n \times n$ matrix multiplication is:

$$SE(n, c) = \frac{1}{f}(n^3 \times M.p + 3n^3 \times R.p(c) + 3(n^3/\sqrt{c}) \times IO.p)$$

Note that as $c$ varies, we obtain a family of architectures each implementing matrix multiplication using BMM with different block sizes. To compare with the Xilinx design, the operating frequency of our design was set to 166 MHz. Figure 3 shows how different values of $c$ affect the system-wide energy dissipation and the energy distribution among the components of the design for $6 \times 6$ matrix multiplication. As $c$ increases, the energy for performing I/O decreases but the energy dissipated in the cache increases. Initially, the system-wide energy decreases as $c$ increases but for large values of $c$, the system-wide energy increases.

For the on-chip design, the energy dissipated in the memory banks is considered instead of the energy dissipated in the I/O. The system-wide energy function is:

$$SE(n, c) = \frac{1}{f}(n^3 \times M.p + 3n^3 \times R.p(c) + 3(n^3/\sqrt{c}) \times MEM.p)$$

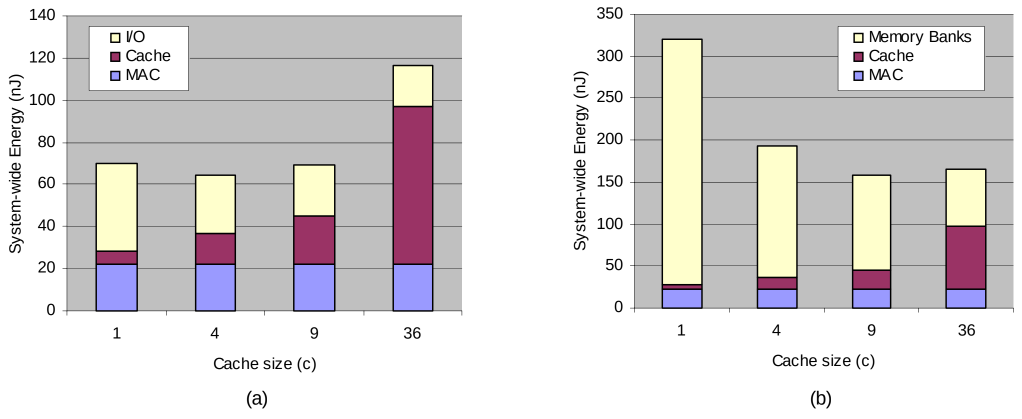

**Figure 3.** System-wide energy dissipation and energy distribution for $6 \times 6$ matrix multiplication as a function of cache size: (a) off-chip design and (b) on-chip design.

Note that as $c$ increases, the traffic between the memory banks and the PE decreases and as a result the energy dissipated in the memory banks decreases.

### 3.1.3. Performance Analysis

As the system-wide energy function is a well-behaved function with easily determinable minima, we were able to identify the most energy-efficient designs from the trade-off graphs (See Figure 3). We compared the performance of our design with a design for $3 \times 3$ matrix multiplication provided by Xilinx.[17] Since Xilinx library does not provide on-chip design, we added Block SRAMs to the Xilinx design. All the designs execute at 166 MHz. For $n > 3$, we used block matrix multiplication using the $3 \times 3$ design.

To measure the system-wide energy for the design based on the Xilinx library, we used the same 1000 input data sets (waveforms) which were used to obtain the component specific power functions in Section 3.1.1. Thus, the switching activity of the input set are the same for both designs. Notice, however, the switching activity in the circuit can be different from the activity in our designs. We compared our design with the design based on the Xilinx library. Table 1 shows the energy, latency, and area of the designs for $n = 6$. For the off-chip design, the cache size $c = 4$ gives the minimum system-wide energy while for the on-chip design, the optimal cache size was found to be 9. All designs use a single dedicated multiplier. The on-chip designs use three memory banks. The multiplier and memory banks are not included in the total area. The improvement in energy dissipation and latency of our design compared with the Xilinx design are also shown.

**Table 1.** Comparison of on-chip designs

| Size | Design based on Xilinx Library | | | Design based on Uniprocessor | | | | Performance Improvement | |
|---|---|---|---|---|---|---|---|---|---|
| | T cycles | A slices | E nJ | T cycles | A slices | Cache size | E nJ | E % | T % |
| 6x6 | 360 | 179 | 330.7 | 320 | 158 | 9 | 157.9 | 52 | 13 |

## 3.2. Linear Array Architecture

For the second domain, we consider a linear array of processing elements (PEs) as the candidate architecture (See Figure 4 (a)). Each PE has one multiplier and storage. We start with an algorithm for optimal latency on linear array.[13] $PE_j$ in Figure 4 (a) computes $c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj}$ for all $i$, $1 \leq i \leq n$ where $a_{ik}$, $b_{kj}$, and $c_{ij}$ represent an element of $n \times n$ matrices $A$, $B$, and $C$. In iteration $(i, k)$, $1 \leq i, k \leq n$, $c_{ij} = c_{ij} + a_{ik} \times b_{kj}$ is computed in $PE_j$. Elements of matrix $A$ and $B$ are fed to the array via two input ports of $PE_1$ in column major and row major order, respectively. It is critical to ensure that $a_{ik}$ "meets" $b_{kj}$ in a cycle in $PE_j$. For this, $a_{ik}$ and $b_{kj}$ pass through two and one delay(s), respectively in each PE. The resulting architecture for each PE is shown in Figure 4 (b). $a_{ik}$ enters input port $AS$ and goes through two delays ($AS.LR$ and $AS.RR$), while $b_{kj}$ enters $BS$ and goes through one delay. Details of the algorithm, its analysis, and proof of correctness can be found in Ref. 13.

Compared with the uniprocessor design in Section 3.1, we use more multipliers to reduce the latency. The above family of architectures offers several advantages compared to other architecture families. These architectures have a low I/O-bandwidth requirement and they scale as the problem size grows. An optimal family of algorithms for these architectures is known.[13] Each PE in the linear array has storage of size $s$, $1 \leq s \leq n$. The number of PEs ($pe$) is chosen such that $n \leq pe \leq n^2$. To achieve the minimal I/O complexity ($O(n^2)$), the total amount of storage across all the PEs should be $n^2$. As shown in Ref. 13, this architecture can perform $n \times n$ matrix multiplication in $O(n^2)$ time using $n\lceil n/s \rceil$ PEs. For the sake of illustration, we consider the on-chip design for this domain.

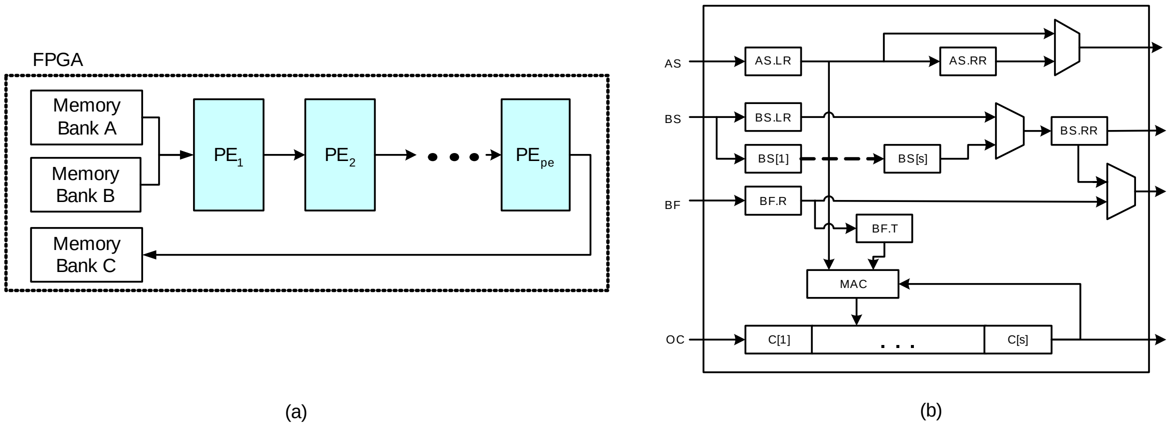

**Figure 4.** (a) Linear array architecture and (b) details of the PE

### 3.2.1. Identifying Components and Parameters

The structure of the linear array is shown in Figure 4 (a). It consists of three components: processing elements (PEs), buses connecting adjacent PEs, and memory banks. For the purpose of high-level modeling, we identified the PE and the memory bank as RModules, and the bus between two adjacent PEs as an Interconnect. The PE has a MAC of precision $w$ and storage of size $s$ (See Figure 4 (b)). The MAC is implemented using a dedicated multiplier. The PE has two power states $on$ and $off$. In the $on$ state, the multiplier is on and thus the PE dissipates more power than in the $off$ state when the multiplier is off. The power state of the multiplier is controlled by clock gating. The PE also includes 6 registers and 3 multiplexers of $w$ bits. The key parameters affecting energy are the number of PEs ($pe$), the amount of storage within a PE ($s$), and power states ($ps$).

We implemented the PE using a Virtex-II FPGA operating at $f = 166$ MHz and performed simulations to obtain the power functions for the PE and the bus. The power function for the PE is:

$$PE.p.ps = \begin{cases} 7.01s + 31.04\ mW & (ps = on) \\ 7.01s + 14.04\ mW & (ps = off) \end{cases} \tag{1}$$

The bus has constant amount of power dissipation of 39.74 mW. The power function for the memory bank is the same as in the uniprocessor architecture (See Section 3.1.1).

### 3.2.2. System-wide Energy Function

There are several constraints imposed by the algorithm which is exploited to identify component specific parameters and their ranges. The value of $s$ determines the total number of PEs ($pe$). The latency ($T$) of this design using $n\lceil n/s \rceil$ PEs and $s$ storage per PE is[13]: $T = (n^2 + 2n\lceil n/s \rceil - \lceil n/s \rceil + 1)$.

We consider problems in the range $1 \leq n \leq 16$. Precision ($w$) is set to 8. In each PE, the multiplier is on for $T\ /\ (\lceil n/s \rceil)$ cycles and is off for $T \times (1 - 1/\lceil n/s \rceil)$ cycles.[13] $PE.p.ps$ refers to the power dissipation of PE when its multiplier is in state $ps$ (See Equation 1). Note that the I/O traffic between the PEs and the memory banks is $O(n^2)$. The system-wide energy function is:

$$SE(n,s) = \frac{1}{f}(n \times T \times PE._{p.ps=on} + T \times (n\lceil n/s \rceil - n) \times PE._{p.ps=off} + 3n^2 \times MEM.p)$$

### 3.2.3. Design Trade-offs and Performance Analysis

Figure 5 (a) shows the effect of varying the amount of storage ($s$) on the power dissipation of a PE. Figure 5 (b) shows the effect of varying the amount of storage ($s$) on the system-wide energy for three problem sizes ($n = 4, 8, 16$). Based on these plots, to obtain energy-efficient designs we choose $pe = s = n$, where $n$ is the problem size.

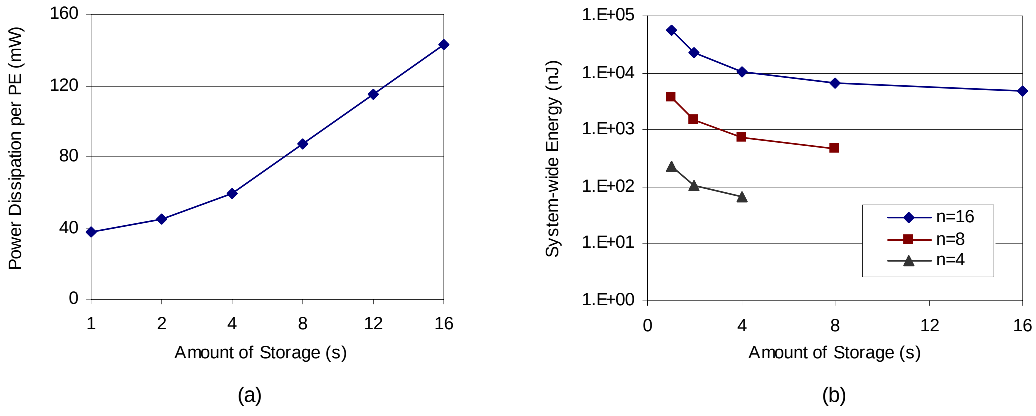

**Figure 5.** (a) Power dissipation for a single PE and (b) the system-wide energy as a function of the amount of storage ($s$) for $n = 4, 8, 16$.

Table 2 shows the energy, latency, and area of the designs for various problem sizes. The improvement in energy dissipation and latency in our designs compared with the Xilinx designs are also shown. On the average our designs consume 32% less energy compared with the Xilinx design. The latency improvement varies from 7× to 22×. However, our designs occupy more area.

**Table 2.** Comparison of our designs on linear array architecture with Xilinx design

| Size | Design based on Xilinx Library | | | Design based on Linear Array Architecture | | | Performance Improvement | |
|---|---|---|---|---|---|---|---|---|
| | T | A | E | T | A | E | E | T |
| $n \times n$ | cycles | slices | nJ | cycles | slices | nJ | % | times |
| 6x6 | 360 | 179 | 330.7 | 49 | 1074 | 213.2 | 36 | 7.3 |
| 9x9 | 1215 | 179 | 1116.0 | 100 | 1935 | 715.5 | 36 | 12.2 |
| 15x15 | 5625 | 179 | 5166.8 | 256 | 4305 | 3812.5 | 26 | 22.0 |

The energy dissipation of the designs discussed in this section is based on high-level estimation using the system-wide energy function for the domain. In order to validate these energy estimations, we performed the following experiment. For a particular design, we used the corresponding system-wide energy function to estimate the total energy dissipation. We compared this result with a complete VHDL simulation of the design using Xilinx tools. In the simulations, the same input data used to obtain the component specific power functions were used. As noted earlier, the average switching activity was observed to be 25%. We performed this experiment for various problem sizes using designs in Section 3.2.2. Table 3 shows the accuracy of our estimates. Our energy estimations were (on the average) within 7.0% of the estimations using low-level simulation tools.

**Table 3.** Accuracy of the high-level energy estimation of our designs

| Problem size | $n$ | 3 | 6 | 8 | 9 | 12 | 16 |
|---|---|---|---|---|---|---|---|
| Energy (nJ) | Estimated | 34.0 | 213.2 | 497.8 | 715.5 | 1801.2 | 4759.7 |
| | Measured | 37.4 | 228.6 | 536.9 | 768.4 | 1913.6 | 5078.6 |
| | Error | 9.0% | 6.7% | 7.3% | 6.9% | 5.9% | 6.3% |

## 4. CONCLUSIONS

This paper developed several energy-efficient designs for matrix multiplication. The energy models specific to the domains were defined by our algorithms, architectures, and specification of target FPGA devices. In the models, key architecture parameters such as caches size and amount of storage per PE that affect the system-wide energy were identified. Then the system-wide energy functions were obtained using these parameters. Trade-off analysis using the system-wide energy functions facilitated development of energy-efficient designs. To realize compact designs, we developed the uniprocessor design in which the cache size affects the system-wide energy. To reduce the latency, we developed designs based on the linear array architecture in which the amount of storage per PE affects the system-wide energy. Our results demonstrate superior energy performance.

Currently, we are automating several design steps by using the MILAN framework.[9] MILAN is a model based hierarchical simulation framework for domain-specific system design. MILAN can be configured for a specific domain to provide a modeling language suitable for the domain. It also facilitates integration of several tools and simulators into the framework. MILAN is used to automate several steps in our design methodology to realize a semi-automatic design environment for energy-efficient designs using FPGAs. Additional details of the MILAN framework can be found in Ref. 1, 10.

## ACKNOWLEDGMENTS

The work described in this paper is part of the Model based Integrated SimuLAtioN (MILAN) project.[9] MILAN is a joint project between the University of Southern California and Vanderbilt University. It is supported by the US DARPA Power Aware Computing and Communication (PAC/C) program under contract F33615-C-00-1633 monitored by Wright Patterson Air Force Base. Ju-wook Jang's work is supported by the Ministry of Information and Communication, Korea. We would like to thank the MILAN team, particularly, Sumit Mohanty for many helpful discussions.

## REFERENCES

1. A. Agrawal, A. Bakshi, J. Davis, B. Eames, A. Ledeczi, S. Mohanty, V. Mathur, S. Neema, G. Nordstrom, V. Prasanna, C. Raghavendra, and M. Singh, "MILAN: A Model Based Integrated Simulation for Design of Embedded Systems," *Language Compilers and Tools for Embedded Systems*, 2001.
2. A. Amira, A. Bouridane, and P. Milligan, "Accelerating Matrix Product on Reconfigurable Hardware for Signal Processing," *Field-Programmable Logic and Applications (FPL)*, Springer Lecture Notes in Computer Science 2147, pp. 101-111, 2001.
3. A. Bogliolo, L. Benini, and G. Micheli, "Regression-based RTL Power Modeling," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 5, No. 3, 2000.
4. B. L. Bowerman and R. T. O'Connell, *Linear Statistical Models - An Applied Approach*, 2nd Edition, Brooks/Cole Pub Co.
5. S. Choi, J. Jang, S. Mohanty, and V. K. Prasanna, "Domain-Specific Modeling for Rapid System-Wide Energy Estimation of Reconfigurable Architectures," to appear in *International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, June 2002.
6. H. ElGindy and Y.-L. Shue, "On Sparse Matrix-Vector Multiplication with FPGA-based Systems," *IEEE Symposium on Field Programmable Custom Computing Machines (FCCM)*, April 2002.
7. J.-W. Hong and H. T. Kung, "I/O Complexity: The Red-Blue Pebbling Game," *ACM Symposium on Theory of Computing (STOC)*, 1981.
8. J. Jang, S. Choi, and V. K. Prasanna, "Energy-Efficient Matrix Multiplication on FPGAs," submitted to *International Conference on Field Programmable Logic and Applications (FPL)*, 2002.
9. Model-based Integrated Simulation, http://milan.usc.edu.
10. S. Mohanty, V. K. Prasanna, S. Neema, and J. Davis, "Rapid Design Space Exploration of Heterogeneous Embedded Systems using Symbolic Search and Multi-Granular Simulation," *Language Compilers and Tools for Embedded Systems*, 2002.
11. S. Mohanty, S. Choi, J. Jang, and V. K. Prasanna, "A Model-based Methodology for Application Specific Energy Efficient Data path Design using FPGAs," to appear in *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2002.
12. T. Mudge, "Power: A First-Class Architectural Design Constraint," *IEEE Computer*, Volume. 34, April 2001.
13. V. K. Prasanna Kumar and Y. Tsai, "On Synthesizing Optimal Family of Linear Systolic Arrays for Matrix Multiplication," *IEEE Transactions on Computers*, Vol. 40, No. 6, 1991.
14. A. Ragunathan, N. K. Jha, and S. Dey, *High-level Power Analysis and Optimization*, Kluwer Academic Publishers, 1998.
15. L. Shang and N. K. Jha, "High-Level Power Modeling of CPLDs and FPGAs," *International Conference on Computer Design*, 2001.
16. A. Stammermann, L. Kruse, W. Nebel, and A. Prastsch, "System Level Optimization and Design Space Exploration for Low Power," *International Symposium on System Synthesis (ISSS)*, 2001.
17. Xilinx Application Note: *Virtex-II Series and Xilinx ISE 4.1i Design Environment*, http://www.xilinx.com.