

Energy-Efficient Matrix Multiplication on FPGAs^{*}

Ju-wook Jang¹, Seonil Choi², and Viktor K. Prasanna²

¹ Electronic Engineering,
Sogang University, Seoul, Korea
jjang@sogang.ac.kr

<http://eeca2.sogang.ac.kr/>
² Electrical Engineering-Systems,
University of Southern California, Los Angeles, USA
{seonilch, prasanna}@usc.edu
<http://ceng.usc.edu/~prasanna>

Abstract. We develop new algorithms and architectures for matrix multiplication on configurable devices. These designs significantly reduce the energy dissipation and latency compared with the state-of-the-art FPGA-based designs. We derive functions to represent the impact of algorithmic level design choices on the system-wide energy dissipation, latency, and area by capturing algorithm and architecture details including features of the target FPGA. The functions are used to optimize energy performance under latency and area constraints for a family of candidate algorithms and architectures. As a result, our designs improve the energy performance of the optimized design from the recent Xilinx library by 32% to 88% without any increase in area-latency product. In terms of comprehensive metrics such as EAT (Energy-Area-Time) and E/AT (Energy/Area-Time), our designs offer superior performance compared with the Xilinx design by 50%-79% and 13%-44%, respectively. We also address how to exploit further increases in density of future FPGA devices for asymptotic improvement in latency and energy dissipation for multiplication of larger size matrices.

1 Introduction

Dramatic increases in the density and speed of FPGAs make them attractive as flexible and high-speed alternative to DSPs and ASICs [3][7]. Indeed, FPGAs have become an attractive fabric for implementing computationally intensive applications such as signal and image processing used in mobile devices [8]. Since mobile devices typically operate in multi-mode and energy constrained

^{*} This work is supported by the DARPA Power Aware Computing and Communication Program under contract F33615-C-00-1633 monitored by Wright Patterson Air Force Base and in part by the National Science Foundation under award No. 99000613. Ju-wook Jang's work is supported by Ministry of Information and Communication, Korea.

environments, energy is a key performance metric in addition to latency and throughput.

Matrix multiplication is a frequently used kernel operation in a wide variety of graphic, image, robotics, and signal processing applications. Several signal and image processing operations can be reduced to matrix multiplication. Most previous work for matrix multiplication on FPGAs focuses on latency optimization [1]. In this paper we develop designs which minimize the energy dissipation and offer trade-offs for performing matrix multiplication on commercially available FPGA devices. Our effort is focused on algorithmic techniques to improve energy-performance instead of low level (implementation level) optimizations. Many parameters affect energy dissipation, area, and latency of FPGA-based designs. These parameters depend on the algorithm and the architecture used and the target FPGA features. These parameters give rise to a large design space.

In our approach we narrow down the design space by devising latency-optimal algorithms on “minimal” architectures (in the sense of hardware resources used) which provide trade-offs among energy dissipation, latency, and area. Closed form functions representing system-wide energy dissipation, area, and latency are derived by incorporating architecture and algorithm details, low level power simulation of individual modules, and FPGA vendors’ specifications. A module is a configured block in an FPGA for data processing or storage. The characteristics of power consumption of the modules and the number of configurable logic blocks (CLBs) needed for individual modules are captured in the functions to obtain reasonably accurate estimates.

The functions provide us a high level picture about where we should look for possible savings in energy and area and allow us to make trade-offs to meet the constraints. Using the energy function, algorithm and architecture level optimizations are made. Low level power simulation using XPower from Xilinx is performed to verify accuracy of the energy and latency estimated by the functions. Our experiments show that the estimation of energy dissipation and area using the functions is within 3.8% to 7.8% of the actual values based on low level simulation.

The rest of the paper is organized as follows. An energy model specific to our implementation is described in Section 2. Algorithms and architectures for energy-efficient implementation are presented in Section 3. Section 4 derives functions for energy, latency, and area and illustrates various trade-offs. Section 5 concludes the paper.

2 Energy Model and Design Space

Our family of architectures and algorithms for matrix multiplication forms a domain and we limit algorithm-level exploration for energy optimization to the design space spanned by this domain. The family of architectures in Figure 1 and the parameters in Table 1 represent the design space. In the off-chip model, the storage for input matrices is assumed to be outside the FPGA. I/O ports are used for data access. The on-chip model uses on-chip storage such as BSRAM

(Block Select RAM) in the Xilinx FPGA devices. Choosing specific values for the parameters in Table 1 results in a design point in the design space. For example, $n = 24$, $p = 6$, $r = 4$, $m = 1$, $s = 1$, $K_b = 2$, and $K_{io} = 0$ represents a design where 24×24 matrix multiplication is implemented by 6 PEs with 4 registers, one multiplier, and one SRAM (Select RAMs in the Xilinx devices) per PE. The input matrices are stored in two ($\lceil 2 \times 24 \times 24 / 1024 \rceil = 2$) BSRAMs (on-chip memory banks) of the device and no I/O ports are used.

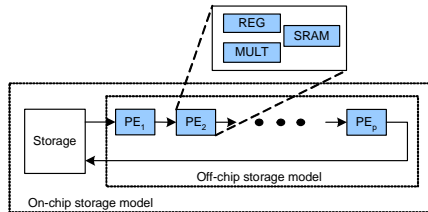


Fig. 1. Family of architectures

An energy model specific to the domain is constructed at the module level by assuming that each module of a given type (register, multiplier, SRAM, BSRAM, or I/O port) dissipates the same power independent of its location on the chip. This model simplifies the derivation of system-wide energy dissipation functions. Energy dissipation for each module can be determined by the number of cycles the module stays in each power state and low level estimation of the power consumed by the module in the power state assuming average switching activity. Additional details of the model can be found in [4].

Table 1. Range of parameters for Xilinx XC2V1500

Parameter	Range	FPGA constraints
Problem size (n)	2, 3, 4, ...	
Number of PEs (p)	n/l , n is divisible by l	
Number of registers per PE (r)	b^{2k+2} , $b = 2, 3, 4, \dots$ ($0 \leq k \leq \log_b n$)	8/16 bit registers
Number of multipliers per PE (m)	b^{2k}	2-stage dedicated pipeline
Number of SRAMs per PE (s)	$\lceil nb^k / 16 \rceil$	16 words minimum
Number of BSRAMs (K_b) (On-chip model)	$\lceil 2n^2 / 1024 \rceil$	1024 16-bit words minimum
Number of I/O ports (K_{io}) (Off-chip model)	$2b^k$	8/16 bits

3 Energy-Efficient Algorithms for Matrix Multiplication

We present our algorithms and architectures in two theorems and one corollary. Pseudo-code for cycle-specific data movement, the detailed architectures, and a snapshot of an example computation are also shown. Theorem 1 improves the best known algorithm for matrix multiplication [10]. This leads to optimal time complexity with leading coefficient of 1 for matrix multiplication on linear array. Theorem 1 is extended for trade-offs among energy dissipation, latency,

and area (Corollary 1). Corollary 1 is used to identify energy-efficient designs under latency and area constraints.

The second algorithm can be used to exploit future increases in the density of FPGA devices to realize improvements in energy dissipation and latency as larger devices become available (Theorem 2). It uses more multipliers and I/O ports. The state-of-the-art design for matrix multiplication from Xilinx reference design uses only 180 slices while a XC2V1500 device can hold upto 10,000 slices in addition to memory banks (BSRAMs) and I/O ports.

Theorem 1. $n \times n$ matrix multiplication can be performed in $n^2 + 2n$ cycles using the architecture in Figures 1 and 2 (b) with the number of processing elements, $p = n$.

Proof. The algorithm in Figure 2 (a) and the architecture in Figure 1 and Figure 2 (b) are devised to compute $c_{ij} = \sum_{k=1}^n a_{ik} \times b_{kj}$ for all i, j . a_{ik}, b_{kj} , and c_{ij} represent an element of $n \times n$ matrices A, B , and C . PE_j denotes the j -th PE from the left in Figure 1, $j = 1, 2, \dots, n$. It computes the $c_{1j}, c_{2j}, \dots, c_{nj}$. Matrix B is fed to the lower I/O port of PE_1 in row major order $(b_{11}, b_{12}, b_{13}, \dots, b_{1n}, b_{21}, b_{22}, \dots)$. Matrix A is fed to the upper I/O port of PE_1 in column major order $(a_{11}, a_{21}, a_{31}, \dots, a_{n1}, a_{12}, a_{22}, \dots)$, n cycles behind the matrix B. For example, a_{11} is fed to the upper I/O port of PE_1 at the same cycle as the b_{21} is fed to the lower I/O port of PE_1 . At the end of the processing, c_{ij} is in the SRAM or registers in PE_j , for $1 \leq i \leq n$. The a_{ik} traverses the $PE_1, PE_2, PE_3, \dots, PE_n$ in order and allows PE_j to update $c'_{ij} = c'_{ij} + a_{ik} \times b_{kj}$ where c'_{ij} represents the intermediate value of c_{ij} . Since a_{ik} stays at each PE_j for just one cycle, it is essential to ensure that b_{kj} arrives at PE_j no later than a_{ik} . We show how the algorithm in Figure 2 (a) satisfies this requirement. The number of cycles needed for b_{kj} to arrive at PE_j is $(k-1)n + 2j - 1$. a_{ik} needs $n + (k-1)n + i + j - 1$ cycles to arrive at PE_j . The requirement is satisfied since $2j - 1 \leq n + i + j - 1$ for all i, j . Next we show why two registers (denoted BM and BL in Figure 2 (b)) are enough to hold b_{kj} ($k = 1, 2, \dots, n$) until they are no longer needed. b_{kj} is needed until a_{nk} arrives at PE_j in the $\{n + (k-1)n + n + j - 1\}$ -th cycle. $b_{(k+2)j}$ arrives at PE_j in the $\{(k+1)n + 2j - 1\}$ -th cycle. Since $(k+1)n + 2j - 1 > n + (k-1)n + n + j - 1$ for all j, k , and n , b_{kj} can be replaced when $b_{(k+2)j}$ arrives at PE_j . This proves that PE_j needs only two temporary registers (denoted as BM, BL in Figure 2 (b)) to hold b_{kj} ($k = 1, 2, \dots, n$). Now we show $n^2 + 2n$ cycles are needed to complete the matrix multiplication. The computation finishes one cycle after a_{nn} arrives at PE_n , which is the $\{n + (n-1)n + n + n - 1\}$ -th or $\{n^2 + 2n - 1\}$ -th cycle. \square

Corollary 1. $n \times n$ matrix multiplication can be performed in $(n/p)^3(p^2 + 2p)$ cycles using the architecture in Figure 2 (b), where p is the number of processing elements and $p \leq n$.

Proof Sketch. $n \times n$ matrix multiplication can be decomposed into n^3/p^3 $p \times p$ matrix multiplications. Using Theorem 1 with n replaced by p , the result follows. \square

Corollary 1 provides trade-offs between area and latency. Smaller values for p reduces the number of PEs, resulting in less area. But it increases the number

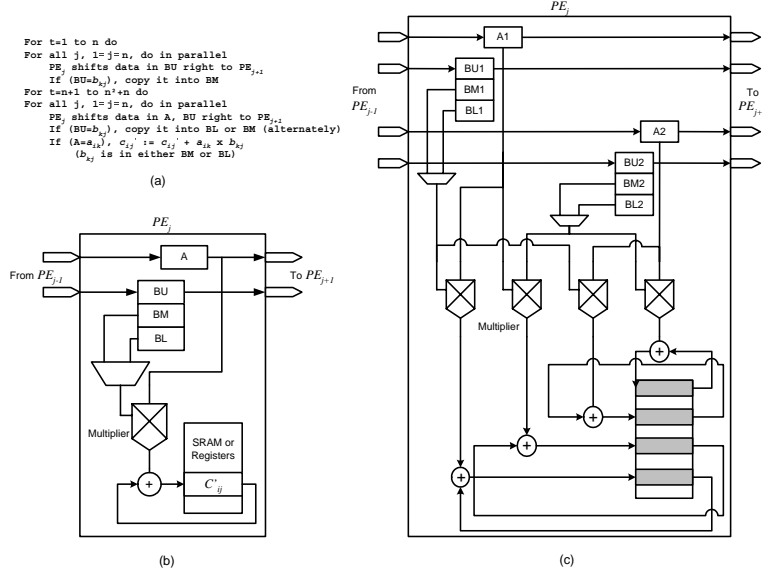


Fig. 2. (a) Algorithm used in the proof of Theorem 1, (b) Architecture of PE_j used in the proof of Theorem 1, and (c) Architecture of PE_j used in the proof of Theorem 2

of cycles to complete the matrix multiplication. Combined together with power estimation of modules, Corollary 1 provides trade-offs among energy dissipation, latency, and area.

Theorem 2. $n \times n$ matrix multiplication can be performed in $(n^2/b^k + 2n/b^k)$ cycles using b^k PEs in Figure 2 (c), where b is the block size for block multiplication, n divisible by b^k , and $k \leq \log_b n$.

Proof Sketch. The proof follows from recursive application of the idea in Theorem 1. In general, an $n \times n$ matrix multiplication requires b^{2k} multipliers per PE, and $2b^k$ I/O ports. Local storage of b^{2k} words is used to store intermediate results of the output matrix. Figure 2 (c) shows our architecture for the case $k = 1$ and $b = 2$. An $n \times n$ matrix multiplication is decomposed into 8 $n/2 \times n/2$ matrix multiplications. Let A_{ij}, B_{ij} , and $C_{ij}, 1 \leq i, j \leq 2$ denote a submatrix of size $n/2 \times n/2$, respectively. In the first phase, $C'_{ij} = A_{i1} \times B_{1j}, 1 \leq i, j \leq 2$ are computed simultaneously using the idea in Theorem 1 by feeding A_{11}, A_{21}, B_{11} and B_{12} into the 4 input ports. In the second phase, $C'_{ij} = C'_{ij} + A_{i2} \times B_{2j}, 1 \leq i, j \leq 2$ are performed. Each phase takes $n^2/4 + n$ cycles from Theorem 1. Since overlapping is possible between the end of the first phase and the start of the second phase, the total number of cycles is $n^2/2 + n$. For other values of k and b , the proof follows from successive recursion using the same idea. Detailed proof is omitted due to space limitations. \square

4 Performance Analysis

4.1 Functions to Estimate Energy, Area, and Latency

Functions to represent energy dissipation, area, and latency are derived for Theorem 1, Corollary 1, and Theorem 2. Energy function of a design is obtained

by $\sum_i t_i P_i$, where t_i and P_i represent the number of active cycles and average power per cycle for module i . For example, P_{MULT} denotes the average power dissipation of the multiplier module. The average power is obtained from low level power simulation of the module. The area function is given by $\sum_i A_i$ where A_i represents the area used for module i . In general, these simplified energy and area functions may not be able to capture all the implementation details needed for accurate estimation. However, we are concerned with algorithmic level comparison, rather than accurate estimation. Since our architectures are simple and have regular interconnection, the error between these functions and the actual values based on low level simulation is expected to be small. In Section 4.4 we confirm the accuracy of the energy and area functions.

Table 2. Number of modules used and the latency of various designs (note: 1 cycle delay added for flushing the 2-cycle multiplier pipeline)

Design	Number of multipliers	Number of registers	Size of SRAMs	Size of BSRAMs	I/O	Latency(cycles)
Xilinx	1	14 (8 bits), 5 (16 bits)	0	$\lceil 3n^2/1024 \rceil$	3	$(n/3)^3 \times 45$
[10]	n^2/m , $1 \leq m \leq n$	n^2 (16 bits), $n^2 + 6n^2/m$ (8 bits)	0	$\lceil 2n^2/1024 \rceil$	4	$n^2 + 2n^2/m + n/m$
Theorem 1	n	$4n$ (8 bits)	$n \lceil n/16 \rceil$	$\lceil 2n^2/1024 \rceil$	2	$n^2 + 2n + 1$
Corollary 1	p	$4p$ (8 bits)	$p \lceil p/16 \rceil$	$\lceil 2n^2/1024 \rceil$	2	$(n/p)^3 (p^2 + 2p + 1)$
Theorem 2	$b^k n$	$4n$ (8 bits)	$n/b^k \lceil b^k n/16 \rceil$	$\lceil 2n^2/1024 \rceil$	$2b^k$	$(n^2/b^k + 2n/b^k + 1)$

Table 2 shows the number of modules to be used by the designs for $n \times n$ matrix multiplication with 8-bit elements. For the off-chip model, I/O ports are used to fetch elements from outside the FPGA. In the on-chip model, BSRAMs of 16-bit 1024 words are used for on-chip storage of input matrices. The SRAMs are CLB-based memory blocks used for storing intermediate results. Using Table 2, functions to represent energy, area, and latency for Corollary 1 are shown in Table 3. Functions for other designs can be obtained in the same way. An average switching activity of 25% at running frequency of 166MHz is assumed. Dedicated multipliers available in the device are used. A_{offset} denotes the difference between the area of a PE and $\sum_i A_i$ for a PE and accounts for glue logic.

4.2 Trade-offs among Energy, Latency, and Area

The functions in Table 3 are used to identify trade-offs among energy, latency, and area. For example, Figure 3 illustrates the trade-offs among energy, latency, and area for 24×24 matrix multiplication for the off-chip model. It can be used to choose energy-efficient designs to meet given area and latency constraints. For example, if 800 slices are available and latency should be less than 6,000 cycles ($36\mu s$), an energy-efficient design is obtained using $p = 4$. Energy dissipation, area, and latency evaluated using the functions in Table 3 are $6.85\mu J$, 524 slices, and 5400 cycles ($32.4\mu s$), respectively. Actual values for energy and area were found to be $7.37\mu J$ and 559 slices based on low level simulation. The simulation

Table 3. Functions to represent energy, area, and latency for $n \times n$ 8-bit matrix multiplication using p PEs. They are based on Corollary 1 with 166MHz

Metric	Functions
Energy (J)	on-chip $(n^3/p^3) \cdot (p^2 + 2p + 1) \{pP_{MULT} + p\lceil p/16 \rceil P_{SRAM} + 4pP_{R8} + \lceil 2n^2/1024 \rceil P_{BSRAM}\} / (166 \times 10^6)$
	off-chip $(n^3/p^3) \cdot (p^2 + 2p + 1) \{pP_{MULT} + p\lceil p/16 \rceil P_{SRAM} + 4pP_{R8} + 2P_{IO}\} / (166 \times 10^6)$
Area (slices)	on-chip $pA_{MULT} + p\lceil p/16 \rceil A_{SRAM} + 4pA_{R8} + pA_{offset}$ and $\lceil 2n^2/1024 \rceil$ BSRAMs
	off-chip $pA_{MULT} + p\lceil p/16 \rceil A_{SRAM} + 4pA_{R8} + pA_{offset}$ and 2 8-bit I/O ports
Latency (cycles)	$(n^3/p^3)(p^2 + 2p + 1)$

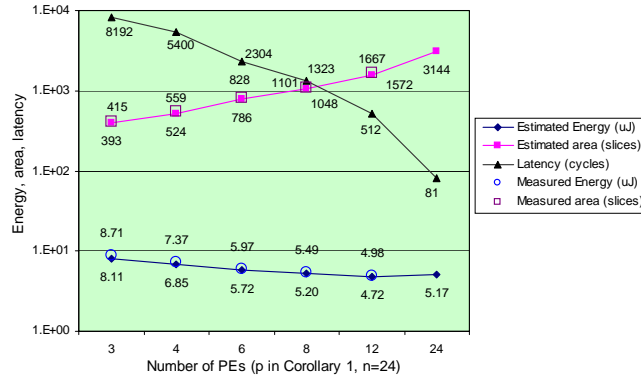


Fig. 3. Trade-offs among energy, latency, and area for 24×24 matrix multiplication. Estimated values from functions are plotted along with measured values in empty circles and boxes

was based on a VHDL code of the design and XPower tool. Compared with the Xilinx design, energy dissipation and latency are reduced by 44% and 77%, while the area increases 3.4x. The resulting design occupies 5% of the slices available in a XC2V1500 device. The largest reduction in energy dissipation is 70% and can be obtained using $p = 12$. Trade-off analysis for the on-chip model also shows similar behavior and is omitted due to space limitations.

4.3 Performance Comparison

Using the functions in Table 3, energy-efficient designs are identified for various sizes of matrices under (an arbitrary) area constraint of 1,800 slices. Thus the area does not necessarily increase with the size of input matrices. Actual values for energy and area based on low level simulation involving VHDL coding and Xilinx XPower tool are obtained for each size of input matrices. Energy is obtained by multiplying the average power (averaged over 1,000 cycles) by the latency. The average power is obtained by XPower. Table 4 compares the performance of our designs against the Xilinx design and a best known latency-optimal design [10] on a linear array for various sizes of matrices. The area and energy are obtained by using synthesized designs and XPower. Xilinx provides an optimized design for 3×3 matrix multiplication only; block matrix multiplication is

employed for larger sizes. These comparisons are based on individual metrics as well as more comprehensive metrics of energy \times area \times latency (EAT) product [2] and energy/(area \times latency) (E/AT). Note that the E/AT metric is the average power density of the design.

Our designs improve the performance of the Xilinx reference design by 32%-66% with respect to energy and 2.8-17x with respect to latency while increasing the area by 2.8x to 9.3x. The designs based on [10] with the same latency as ours fail to reduce the energy dissipation compared with the Xilinx reference design. Analysis of energy and area functions reveal that our designs improve the Xilinx design due to reduction in latency.

In terms of EAT (Energy-Area-Time) and E/AT (Energy/Area-Time), our designs based on Corollary 1 offer superior performance compared with the Xilinx design by 50%-79% and 13%-44%, respectively.

Table 4. Performance comparison of various designs against the Xilinx design. Reduction in energy is represented in %. The reduction in latency and the increase in area are represented as times

Design	Metric	Matrix size						
		3 \times 3	6 \times 6	9 \times 9	12 \times 12	15 \times 15	24 \times 24	48 \times 48
Xilinx	energy (nJ)	25	201	680	1,612	3,150	12,902	103,219
	latency (cycles)	45	360	1,215	2,880	5,625	23,040	184,320
	area (slices)	180	180	180	180	180	180	180
	$EAT/10^6$	0.2	13	148	836	3200	53×10^3	3.4×10^6
	$(E/AT) \times 10^3$	3	3	3	3	3	3	3
Proposed (Corollary 1)	energy (nJ)	17	93	228	622	1,598	5,952	39,808
	(reduction, %)	(32%)	(53%)	(66%)	(61%)	(49%)	(53%)	(61%)
	latency (cycles)	16	49	103	172	972	3,136	11,008
	(speedup, times)	(2.8)	(7.3)	(12)	(17)	(5.8)	(7.3)	(17)
	area (slices)	415	828	1,279	1,667	708	828	1,667
	(increase, times)	(2.3)	(4.6)	(7.1)	(9.3)	(3.9)	(4.6)	(7.1)
	$EAT/10^6$	0.1	3.7	30	178	1100	15	700×10^3
	$(E/AT) \times 10^3$	2.5	2.3	1.7	2.2	2.3	2.3	2.2
Design based on [10]	energy (nJ)	25.6	185	721	1,943	2,970	11,904	124,532
	(reduction, %)	(-2%)	(8%)	(-6%)	(-20%)	(6%)	(8%)	(-20%)
	latency (cycles)	16	49	103	172	972	3,136	11,008
	(speedup, times)	(2.8)	(7.3)	(11.8)	(17)	(5.8)	(7.3)	(17)
	area (slices)	487	1,260	2,359	3,683	975	1,260	3,683
	(increase, times)	(2.7)	(7)	(13)	(20)	(5.4)	(7)	(20)
	$EAT/10^6$	0.2	11	175	1230	2800	47×10^3	5×10^6
	$(E/AT) \times 10^3$	3	3	3	3	3	3	3

Theorem 2 provides asymptotic improvement in energy and latency performance for the on-chip model in Figure 1. From Table 2, for large problems, energy dissipated in BSRAMs and the latency of the Xilinx reference design increase as $O(n^5)$ and $O(n^3)$, respectively, assuming a unit of energy is consumed per cycle for retaining a word in the BSRAM. Energy dissipation and latency for designs based on Theorem 1 and [10] increase as $O(n^4)$ and $O(n^2)$, respectively

under similar assumptions. Theorem 2 improves these complexities to $O(n^4/b^k)$ and $O(n^2/b^k)$, respectively. b is the basic block size for block multiplication and n divisible by b^k with $k \leq \log_b n$. Future increase in the density of FPGAs can be used to increase the number of multipliers and hence b^k , leading to asymptotic reduction in energy dissipation and latency. Details of performance improvements and trade-off analysis are omitted due to space limitations.

4.4 Accuracy of Energy and Area Functions

To test the accuracy of the energy and area functions in Table 4, we compared estimates from the functions against actual values based on synthesized designs and low level simulation. The low level simulation involved VHDL coding of designs in Section 4.3. These designs were synthesized using Synopsys FPGA Express and XST (Xilinx Synthesis Technology) in Xilinx ISE 4.1i. The place-and-route file (.ncd file) was obtained for Virtex-II XC2V1500 FPGA. Mentor Graphics ModelSim 5.5e was used to simulate the module and generate simulation results (.vcd file). These two files are then provided to the Xilinx XPower tool to evaluate the average power dissipation. Energy dissipation is obtained by multiplying the average power by the latency. We observed that the estimation error using our functions (see Table 3) is 3.8% to 7% for energy dissipation and 4.1% to 7.8% for area, respectively.

To address the dependency of energy dissipation on input matrices, matrices were randomly generated and fed as input to our design and the Xilinx design for 3×3 matrix multiplication. Equation 1 is employed to estimate the confidence interval for our simulation. The confidence interval is the interval into which the real average (over all possible input matrices in this example) falls with certain probability(confidence) [6]. \bar{x} , $z_{\alpha/2}$, s , and M represent the average energy over (randomly generated) sample input matrices, a given constant, standard deviation, and number of sample matrices, respectively [6]. The probability that the real average energy dissipation belongs to the interval in Equation 1 is $1 - \alpha$.

$$\bar{x} \pm z_{\alpha/2} \frac{s}{\sqrt{M}} \tag{1}$$

Figure 4 (a) compares the energy dissipation over 50 randomly generated 3×3 input matrices for our design and the Xilinx design. The 95% confidence intervals are compared in Figure 4 (b). Based on 95% confidence intervals, the average energy dissipation of our design for 3×3 input matrices is 7.81 nJ (32%) less than that of the Xilinx design.

5 Conclusions

New algorithms and architectures were developed for matrix multiplication to significantly reduce the energy dissipation and latency compared with the state-of-the-art FPGA-based designs. These improve the best known design and provide trade-offs among energy dissipation, latency, and area. In our methodology, "energy hot spots," which consume most of the energy, are identified through energy estimation functions. Algorithm level optimizations are performed to reduce

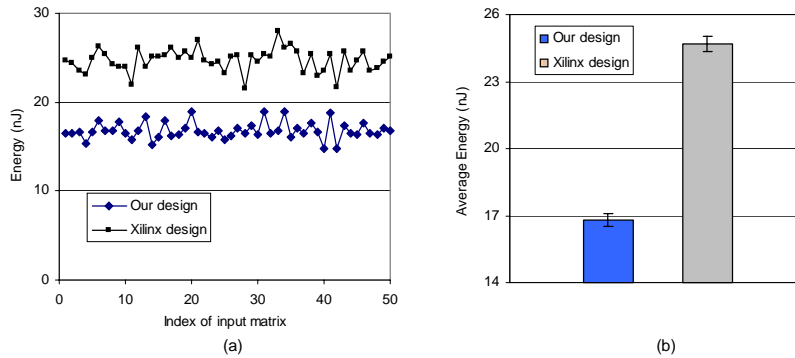


Fig. 4. Comparison between our design (based on Theorem 1) and Xilinx design for 3×3 matrix multiplication: (a) Energy dissipation for randomly generated matrices and (b) Average energy dissipation with confidence intervals

”energy hot spots” without increase in latency. Low level simulation using Xilinx ISE 4.1i and Mentor Graphics ModelSim 5.5e and recent Xilinx XC2V1500 as a target FPGA were performed to evaluate the chosen designs. XPower, a low level power estimation tool for FPGAs from Xilinx, was employed for accurate energy estimation. Additional details can be found in [5].

References

1. Amira A., Bouridane A., Milligan P.: Accelerating Matrix Product on Reconfigurable Hardware for Signal Processing. *Field-Programmable Logic and Applications (FPL)* (2001) 101-111
2. Bass B.: A Low-Power, High-Performance, 1024-Point FFT Processor. *IEEE Journal of Solid-State Circuits*, Vol. 34, No. 3 (1999) 380-387
3. Brebner G., Bergman N.: Reconfigurable Computing in Remote and Harsh Environments. *Field-Programmable Logic and Applications (FPL)* (1999)
4. Choi S., Jang J., Mohanty S., Prasanna V. K.: Domain-Specific Modeling for Rapid System-Level Energy Estimation of Reconfigurable Architectures. To appear in *ERSA 2002 conference* (2002)
5. Choi S., Jang J., Prasanna V. K.: Domain-Specific Modeling and Energy-Efficient Designs for Matrix Multiplication. Technical Report in preparation, Department of Electrical Engineering-Systems, University of Southern California (2002)
6. Hogg R., Tanis E.: *Probability and Statistical Inference*. 6th Eds. Prentice Hall (2001) 656-657
7. Luk W., Andreou P., Derbyshire A., Dupont-De-Dinechin F., Rice J., Shirazi N., Siganos D.: A Reconfigurable Engine for Real-time Video Processing. *Field-Programmable Logic and Applications (FPL)* (1998) 169-178
8. Master P., Athanas P. M.: Reconfigurable Computing Offers Options For 3G. *Wireless Systems Design*. (1999) 20-23
9. Model-based Integrated Simulation. <http://milan.usc.edu>
10. Prasanna Kumar V. K., Tsai Y.: On Synthesizing Optimal Family of Linear Systolic Arrays for Matrix Multiplication. *IEEE Transactions on Computers*. Vol. 40, No. 6 (1991)
11. Xilinx Application Note: Virtex-II Series and Xilinx ISE 4.1i Design Environment. <http://www.xilinx.com> (2001)