A Model-based Methodology for Application Specific Energy Efficient Data Path Design using FPGAs

Sumit Mohanty[1], Seonil Choi[1], Ju-wook Jang[2], Viktor K. Prasanna[1]

[1]*Dept. of Electrical Engg.*
Univ. of Southern California
Los Angeles, CA, 90089
{*smohanty, seonilch, prasanna*}*@usc.edu*

[2]*Dept. of Electronic Engg.*
Sogang University
Seoul, Korea
jjang@sogang.ac.kr

Abstract

We present a methodology to design energy-efficient data paths using FPGAs. Our methodology integrates domain specific modeling, coarse-grained performance evaluation, design space exploration, and low level simulation to understand the tradeoffs between energy, latency, and area. The domain specific modeling technique defines a high-level model by identifying various components and parameters specific to a domain that affect the system-wide energy dissipation. A domain is a family of architectures and corresponding algorithms for a given application kernel. The high-level model also consists of functions for estimating energy, latency, and area that facilitate tradeoff analysis. Design space exploration (DSE) analyzes the design space defined by the domain and selects a set of designs. Low-level simulations are used for accurate performance estimation for the designs selected by the DSE and also for final design selection.

We illustrate our methodology using a family of architectures and algorithms for matrix multiplication. The designs identified by our methodology demonstrate tradeoffs among energy, latency, and area. We compare our designs with a vendor specified matrix multiplication kernel to demonstrate the effectiveness of our methodology. To illustrate the effectiveness of our methodology, we used average power density (E/AT), $energy/(area \times latency)$, as the metric for comparison. For various problem sizes, designs obtained using our methodology are on average 25% superior with respect to the E/AT performance metric, compared with the state-of-the-art designs by Xilinx. We also discuss the implementation of our methodology using the MILAN framework.

Keywords: energy optimization, embedded system design, reconfigurable computing

1. Introduction

Field Programmable Gate Arrays (FPGAs) are a flexible and attractive alternative to dedicated signal processing devices such as DSPs and ASICs. The high processing power available in FPGAs makes them an attractive fabric for implementing complex and compute intensive applications, such as the signal processing kernels used in the mobile devices [11]. Mobile devices operate in energy constrained-environments. Thus, in addition to latency and area, energy is a key performance metric.

Traditional methods for energy-efficient data path design involve the use of various low-level design tools to perform optimizations at RTL or gate level. Such techniques are time consuming and are not effective as the return in terms of improvements in energy efficiency is usually much less compared with high-level optimizations. Studies show that optimization of energy dissipation at the algorithmic level has a much higher impact on the total energy dissipation of a system than optimizations at RTL or gate level [17]. It is reported that the ratio of impact on energy optimization is $20 : 2.5 : 1$ for algorithmic, register, and circuit level techniques [16]. Moreover, a design using FPGAs has to achieve a balance among energy, latency, and area performance. To achieve this balance, a designer has to consider various tradeoffs, such as energy versus latency, energy versus area, and energy versus I/O.

There are several challenges that a designer faces while designing energy efficient systems using FPGAs. Flexibility in using FPGAs results in a large design space. It is not feasible to traverse such a large space using time consuming low-level simulations using tools such as Xilinx XPower [18]. Our experience shows that simulators running on a 700 MHz Pentium III Xeon require an average of 2-3 hours to estimate energy dissipation of a simple design for 3×3 matrix multiplication. Also, FPGAs do not exhibit a high-level structure like, for example, a RISC processor. If such a high-level structure is available, then it can be exploited to define a high-level model that facilitates algorithm level design, optimization, and analysis. Finally, it is not possible to define a universal high-level model for FPGAs. The model depends on the design to be mapped on to the device.

In order to address the above issues, we propose a design methodology that exploits domain specific modeling technique to model a family of architectures and corresponding algorithms to implement an application. The resulting high-level model can be perceived as a virtual parameterized data path (Figure 1). Various parameters in such a data path are referred to as the design knobs. For example, operating frequency, memory capacity, bandwidth, and precision are some of the design knobs. Various settings of these knobs provide tradeoffs among energy, latency, and area and allow a system designer to choose an appropriate setting based on the performance requirements. Power functions associated with each component capture the effect of varying the performance knobs on power dissipation of the component. We consider our approach, based on domain specific modeling, top-down, as our design process begins with a high-level abstraction of a domain and applies various algorithm level optimization techniques to optimize the design. Some of the optimization techniques are a) identification of an appropriate setting of the knobs and b) architecture modification based on the algorithmic characteristics. Eventually, once an energy efficient design has been identified the resulting virtual data path is implemented using FPGA.

In this paper, "application" refers to kernel operations such as window operations, matrix multiplication, FFT, etc. Design space exploration (DSE) refers to traversing the architecture-algorithm space (domain) to evaluate the quality of a design in terms of energy, latency, and area. We use performance to signify some measure of energy, latency, and area. The area metric depends on the target FPGA. For the Xilinx Virtex it is measured in terms of the number of slices used. In our methodology, for a given application, a family of architectures and algorithms (domain) is chosen initially by the designer. Next, a high-level model is defined. This model captures various architecture parameters and provides power functions for rapid (coarse) estimation of energy of the data path for the chosen architecture-algorithm pair. The DSE technique is identified by the designer based on the domain and the high-level model. The model is designed to be at a level of abstraction high enough to expose possible algorithm level optimizations (for example, number of

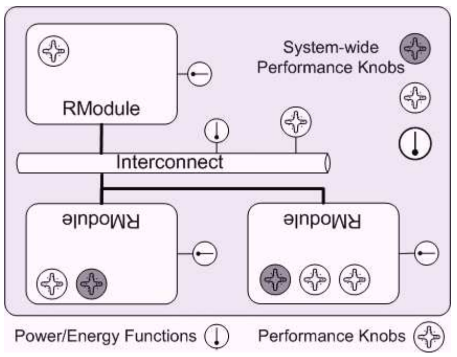


Figure 1. Virtual data path

registers, multipliers, operating frequency, type of multipliers, etc.). Low-level simulations are used to perform off-line simulation of some sample designs in a given domain to accurately estimate the power functions. This paper focuses on an application specific design methodology. More details of the model definition can be found in [3]. We also describe how we have configured the MILAN framework [1] to implement the proposed methodology.

We compare the designs obtained using our methodology with a vendor specified matrix multiplication kernel to demonstrate the effectiveness of our methodology. Our designs demonstrate tradeoffs among energy, latency, and area. For performance comparison, we define *average power density* (E/AT) metric. This metric characterizes a domain in terms of average power dissipated per unit area independent of the problem size. This metric assumes that the designs are area and time optimized. Based on this assumption, the smaller the value for E/AT the better. Based on the E/AT metric our designs are on the average 25% superior than the state-of-the-art designs provided by Xilinx. The E/AT metric is evaluated as $energy/(area \times latency)$.

The rest of the paper is organized as follows. The next section discusses related work. Section 3 discusses our design methodology. Section 4 describes the implementation of this methodology using MILAN. Examples illustrating the methodology and comparison of the resulting designs with the state-of-the-art design is presented in Section 5. We conclude in Section 6.

2. Related work

Several efforts have addressed modeling FPGAs for performance optimization for various signal processing kernels [2, 4, 8]. These efforts focus on latency optimization through efficient data path design. Luk et al. have proposed several techniques to model and optimize time performance of dynamically reconfigurable systems [11]. However, we are not aware of any work that addresses energy optimization of FPGA based implementations through high-level modeling.

Several efforts have addressed rapid but low-level power estimation and optimization. Wolff et al. proposed a technique that uses pre-computed tables to characterize power and latency of the RTL and Intellectual Property (IP) components [17]. This technique achieves energy efficient designs through identification of components with the lowest energy dissipation that meet the given latency requirement. However, it does not exploit the available knobs such as frequency, size of memory, etc. that can be varied to achieve improved energy performance. Nemani et al. proposed a technique that provides rapid power estimates based on the functional description of combinational circuits and their average activity [14]. This effort does not address power optimization. However, it can be integrated into our methodology as a tool for power estimation. Garcia et al. discuss a technique to optimize energy through pipeline architectures [6]. They demonstrate that increase in area does not necessarily result in higher energy dissipation.

Raghunathan et al. have proposed several techniques to estimate power dissipation of different components in an embedded system [16]. Their technique estimates power dissipation based on a bottom-up approach. It modifies available low-level measurement techniques to increase the speed of power measurement. Conversely, we address estimation and optimization through a top-down approach. Our approach begins at the architecture-algorithm (abstraction) level and performs algorithmic optimizations to identify an efficient design before implementing the design using low-level tools.

Xilinx provides a tool set for designing with Virtex-II Pro FPGAs [18]. However, these tools do not provide a high-level abstraction to explore the design space at the algorithm and architecture level. Instead, the Xilinx tools concentrate on gate-level or RT level optimizations. Our methodology is a complementary effort. A design resulting from our methodology can be further optimized using the Xilinx tools.

3. Design methodology

The aim of the design methodology is to design energy-efficient data paths specific to an application. To achieve this goal, our methodology presents a set of designs which provide tradeoffs among energy, latency, and area. The designer explores these designs and identifies an appropriate design based on some selection criteria and performance metrics. Our design methodology is illustrated in Figure 2. The level of automation for each step will be discussed in Section 4.

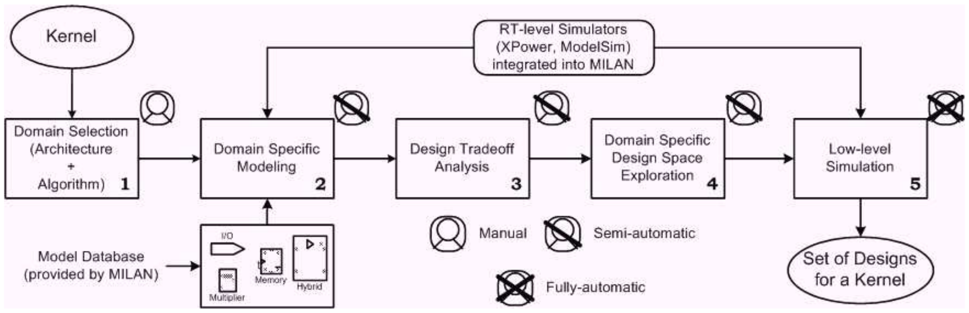

Kernel
RT-level Simulators (XPower, ModelSim) integrated into MILAN
Domain Selection (Architecture + Algorithm) 1
Domain Specific Modeling 2
Design Tradeoff Analysis 3
Domain Specific Design Space Exploration 4
Low-level Simulation 5
Model Database (provided by MILAN)
I/O
Memory
Multiplier
Hybrid
Manual
Semi-automatic
Fully-automatic
Set of Designs for a Kernel

Figure 2. Design methodology

3.1. Domain selection

For each kernel there can be several candidate families of architectures. For example, linear array of processors, two-dimensional array of processors, and cache based uniprocessor architectures are some of the widely used families of architectures. Along with each family of architectures there exists several algorithms that implement each kernel.

Several past efforts have identified various architecture families [5, 10, 15], each having different characteristics in terms of I/O complexity, memory requirements, area, etc. Based on the performance needs and the capabilities and limitations of the target FPGA chip, we identify a suitable architecture-algorithm family. Identification of an appropriate domain ensures that we begin with an efficient design most suitable for the performance requirements and there are various architecture parameters that can be varied to achieve tradeoff among energy, latency, and area. This step is a human-in-the-loop process and exploits the designer's expertise in algorithms and architecture for domain identification.

3.2. Domain specific modeling

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 the analysis of 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. Also for various domains there are several parameters that can not be varied. Therefore, these parameters are excluded during modeling.

Another important aspect of the high-level model is the power functions associated with each component. A power function characterizes the power behavior of a component (RModule or Interconnect). The function captures the effect of variation of model parameters associated with the component on power dissipation. Curve fitting based on sample low-level simulations are used to determine the function. The high-level model also captures functions to evaluate area and latency based on the problem size and possibly other architecture parameters. Finally, the power functions and a set of component power state (CPS) matrices are used to derive the system-wide energy function. The CPS matrices are derived from the algorithm description. These matrices capture the power state for all the components in each cycle. Details of domain specific modeling and techniques to estimate power functions and system-wide energy function can be found in [3].

3.3. Design tradeoff analysis

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 tradeoffs between different performance metrics (energy, latency, and area). These functions also capture the sensitivity of the performance with respect to various architecture parameters. For example, if a component has some variable parameters then the power function associated with the component can be analyzed to identify the power-sensitivity with respect to these parameters.

3.4. Design space exploration (DSE)

During DSE the domain specific design space is traversed to identify designs based on designer specified selection criteria. Our methodology does not propose a DSE technique as the technique depends on the domain. For example, select the design with minimum energy dissipation and select the design with minimum $area \times latency$ are some of the possible selection criteria. As our domain specific modeling technique constraints the design space to allow only valid designs, it typically does not result in a very large design space. Further, as functions are associated with different performance metrics, it is possible to use a brute-force technique that evaluates all possible designs to identify a set of designs that meet the selection criteria. However, it is also possible for a designer to exploit the nature of various performance and power functions to implement a more efficient technique for DSE.

The model parameters and their ranges (as defined by the high-level model) and the functions evaluating various performance metrics are provided as inputs to the DSE phase. In addition, some design selection criteria can also be provided. The output from the DSE is a single design or a set of designs that satisfy the selection criteria.

3.5. Low level simulation

Low-level simulation is applied to the designs selected by the DSE step. The DSE uses various functions to evaluate the designs. While the estimates are reasonably accurate (as shown in Section 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 for any performance metric. The other use is to verify the performance estimates evaluated using the functions provided in the high-level model.

Another application of low-level simulation is to estimate the power functions associated with the high-level model. Detailed description of the use of low-level simulation for power function estimation can be found in [3].

4 Implementing the methodology using MILAN

Model-based **I**ntegrated Simu**LA**tio**N** (MILAN) is a model based extensible framework that facilitates rapid, multi-granular performance evaluation of a large class of embedded systems, by seamless integration of different widely used simulators and design tools into a unified environment [12]. MILAN provides a formal paradigm for specification of the structural and behavioral aspects of embedded systems and a unified software environment for system design and simulation. MILAN can be configured for a specific domain to provide a modeling language suitable for that domain. MILAN is suitable for our design methodology as it can be configured for a specific domain and it can seamlessly integrate various simulators and tools. However, the use of MILAN is not limited only to the proposed methodology. MILAN provides an easy plug-n-play environment. Simulators and tools (both henceforth referred to as tools) are integrated into MILAN through software components known as Model Interpreters (MI). Each tool is associated with its own set of MIs. MIs translate the information stored in the models into the format required by the tool. For example, integration of SimpleScalar (a popular cycle accurate simulator targeting MIPS processors) involves a set of MIs to generate a "config" file for SimpleScalar, to generate "C" code for the application, and to provide feed-back to the model by sending the performance results obtained through simulations to the models. More details of the MILAN framework can be found in [1].

We describe how MILAN is configured to implement the design methodology described in Section 3. The first step does not involve MILAN. This step exploits the designer's expertise in algorithm and architecture to identify a suitable domain for the target application. Once a domain has been identified, MILAN is configured to provide a modeling language suitable for domain specific modeling. Once configured, MILAN provides a user interface (UI) through GME 2000, a graphical modeling tool. The UI consists of representative graphical blocks for basic modules such as registers, multipliers, adders, and SRAMs, that are used to model the candidate domain in the second step. Each of the architecture parameters are associated with the appropriate range of values obtained through domain analysis. MILAN uses low-level simulation and information provided in the high-level model to automatically estimate the power functions [3].

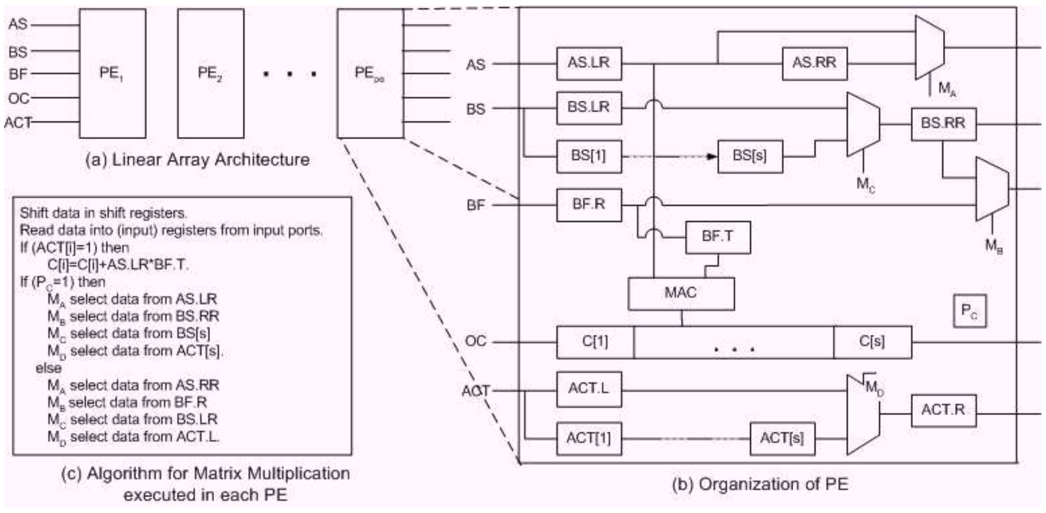
For the third step, MILAN is used to plot various functions for visual inspection by the designer. The details of the high-level model captured in MILAN are available for design space exploration (DSE), the fourth step. DSE tools can be integrated into MILAN and the MIs associated with the tool feed information from the models to the tool. An illustrative DSE using MILAN can be found in [13]. The second, third, and fourth step are semi-automatic. These steps involve the designer to provide required inputs such as the details of the high-level model, choice of DSE tool, selection criteria for the DSE, etc. to the MILAN framework.

The DSE step (Step 5) identifies a set of candidate designs. These designs are stored in the MILAN design database. The designer then invokes appropriate model interpreters that configure the low-level simulator based on the candidate designs to perform low-level simulation. We have integrated XPower and ModelSim in to MILAN to perform low-level simulation for power and latency respectively.

5. Case study: Data path design for matrix multiplication

We illustrate our methodology using matrix multiplication (MM), a frequently used kernel operation in signal and image processing. Also, MM is a fundamental operation that has been widely studied by the architecture and algorithm communities and a rich family of architectures and algorithms implementing MM is available [9].

5.1. Defining a Domain



```
Shift data in shift registers.
Read data into (input) registers from input ports.
If (ACT[i]=1) then
    C[i]=C[i]+AS.LR*BF.T.
If ($P_C$=1) then
    $M_A$ select data from AS.LR
    $M_B$ select data from BS.RR
    $M_C$ select data from BS[s]
    $M_D$ select data from ACT[s].
  else
    $M_A$ select data from AS.RR
    $M_B$ select data from BF.R
    $M_C$ select data from BS.LR
    $M_D$ select data from ACT.L.
```

Figure 3. Architecture for matrix multiplication, PE organization, and corresponding algorithm

For the sake of illustration, we consider the family of linear array of processing elements (PEs) as the candidate architecture family. This family of architectures offers several advantages compared to other architecture families. These architectures have a low I/O-bandwidth requirement and scale as the problem size grows. An optimal family of algorithms for these architectures is known [15]. Each processor in the linear array has a fixed storage of size s, $1 \leq s \leq n$. The architecture can perform $n \times n$ matrix multiplication in $O(n^2)$ time using $n\lceil n/s \rceil$ PEs. The number of PEs (pe) varies from n to n^2. However, as our target device is an FPGA with limited amount of logic and memory that can be synthesized on it, we consider two related families of architecture-algorithm:

- linear array architecture with total storage n^2 for a $n \times n$ matrix multiplication when n is small, and
- a block matrix multiply algorithm for $N \times N$ matrices using an linear array implementation for a sub-matrix of size $n \times n$, where N is a multiple of n.

5.2. A High-level Model for Matrix Multiplication on Linear Array of Processors

The structure of the linear array is shown in Figure 3.a. It consists of two components: processing elements (PEs) and busses connecting the adjacent PEs. For the purpose of high-level modeling, we identified the PE as an RModule and the bus between two adjacent PEs as an Interconnect The PE (see Figure 3.b) has a MAC of precision w and a memory of size s. The PE has two power states on and off. During the on state the multiplier is on and thus the PE dissipates more energy than 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 precision (w), number of PEs (pe), amount of memory within a PE (s), and power states (ps). We refer to this design as *Design 1.*

A matrix multiplication algorithm for linear array architectures is proposed in [15]. There are several constraints imposed by the algorithm which are exploited to identify component specific parameters and their ranges. Also, to achieve the minimum latency, the minimum number of PEs needed for a $n \times n$ matrix multiplication is n [15]. Therefore, the range of s is given by $1 \leq s \leq n$. To achieve the minimal I/O complexity $O(n^2)$, the total amount of memory across all PEs should be n^2. Therefore, the total number of PEs (pe) is $n \lceil n/s \rceil$. The latency (T) of this design using $n \lceil n/s \rceil$ PEs and s memory per PE is [15]:

$$T = \frac{1}{f}(n^2 + 2n \lceil n/s \rceil - \lceil n/s \rceil + 1). \tag{1}$$

We consider problems in the range $1 \leq n \leq 16$. For the sake of illustration, we fixed w at 8. The parameters and their ranges are shown in Table 1.

Table 1. Model parameters

Parameters	Values or ranges
s	$1 \leq s \leq n$
pe	$1 \leq pe \leq n \lceil n/s \rceil$
w	8
ps	on, off

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

$$PE.p.ps = \begin{cases} 7.01s + 31.04\,mW, & (ps = on) \\ 7.01s + 14.04\,mW, & (ps = off) \end{cases} \tag{2}$$

The bus has a constant amount of power dissipation of 39.74 mW. The area of the design can be expressed as: $A = A_{mult} \times pe + A_{reg8} \times (3s + 6) \times pe + 50 \times n$ where, A_{mult} is the area for a multiplier and A_{reg8} is the area for a 8 bit register. This equation is derived based on the design of the PE (see Figure 3) and the factor of 50 is added to account for the other components such as multiplexers present in each PE.

5.3. Tradeoff analysis

Tradeoff analysis involves plotting various functions associated with the high-level model. For example, Figure 4 shows the effect of variation of number of memory modules (s) on the power dissipation of a PE and effect of variation of number of PEs (pe) on the system-wide energy for a specific problem size (16 $\times$ 16). Based on these curves, the design that consumes the minimum energy is the design with $pe = s = n$, where n is the problem size.

For large size matrix multiplication we use the well known block matrix multiply technique. The performance in terms of energy, latency, and area for 16 $\times$ 16 matrix multiplication is shown

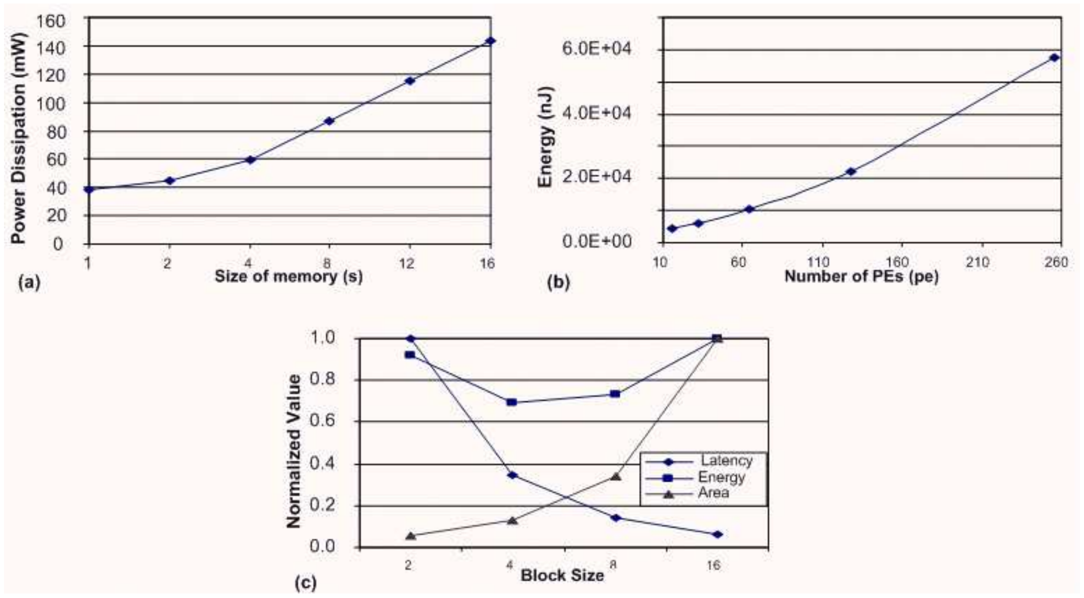


Figure 4. Various tradeoffs associated with the two families of architectures

in Figure 4.c. For analysis, we have normalized the values for energy, latency, and area in this graph with respect to the maximum value in each category. Thus, depending on the performance requirement, the best block size can be chosen. For example, block size of 4 is the most efficient with respect to energy. However, if we optimize for latency or area, block size 16 and 2 respectively are the optimal designs within the design space defined by the domain.

5.4. Design space exploration

Table 2. Performance comparison

Size	Design based on Xilinx library				Design based on our methodology				Performance Improvement	
$n \times n$	T cycles	A slices	E nJ	E/AT	T cycles	A slices	E nJ	E/AT	E %	T times
6x6	480	207	414.8	0.007	49	1074	158.9	0.0050	62	10
9x9	1620	207	1400.0	0.007	100	1935	590.6	0.0050	58	16
15x15	7500	207	6481.5	0.007	256	4305	3459.9	0.0052	47	29
33x33	79860	207	69015.0	0.007	21296	429	28485.1	0.0052	59	4
48x48	245760	207	212385.8	0.007	25088	1074	81331.5	0.0050	62	10

For the sake of illustration, we consider "minimum energy dissipation" as our selection criteria for both the domains. Table 2 shows the energy efficient designs identified for different problem sizes based on the total energy consumed to perform matrix multiplication. We compare these designs with the state-of-the-art design for matrix multiplication provided by Xilinx [18].

As our power functions are well-behaved functions with easily determinable minima, we were able to identify the most energy efficient design through visual inspection of the tradeoff curves. We compared the performance of our design with a design for a 3×3 matrix multiplication provided by Xilinx [18]. All the designs were executed at the same clock frequency of 166 MHz. We used block matrix multiplication to implement larger sized matrix multiplication using the 3×3 base design.

For each problem size, we compared the design provided by Xilinx to the most efficient designs based on our methodology. Table 2 shows the energy, latency, and area values for these 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 performed 57% better with respect to system-wide energy dissipation than the Xilinx design. The latency improvement varies from $4\times$ to $30\times$. Also, on the average our designs are 25% superior in terms of average power density (E/AT). Note that same or similar values for E/AT indicate that a similar design is used for different problem sizes.

The energy dissipation for various designs discussed in this section are based on high-level estimations using the system-wide energy function for the domain. In order to validate our energy estimation technique, we performed the following experiments. For a particular design, we used the system-wide energy function to estimate the total energy dissipation. We compared this result with a complete VHDL simulation of our designs using the Xilinx tools. In the sample simulations, the input data to the components in estimating the power dissipation was randomly generated and its switching activity (sa) was found to be 25%. We performed this experiment for various designs for different problem sizes. Table 3 shows the comparison. Our energy estimations were (on the average) within 6.4% of the estimations using low-level simulation tools. In the worst case, the error was 7.4%.

Table 3. Accuracy of our high-level estimation technique

Problem size	n	3	6	8	9	12	16
Energy (nJ)	Estimated	21.4	159.9	399.6	590.6	1,576.9	4,357.8
	Measured	23.1	169.1	430.2	633.0	1,671.5	4,646.4
	Error	7.4%	5.4%	7.1%	6.7%	5.7%	6.2%

5.5. Applying our design methodology for energy optimization

Our design methodology facilitates algorithmic optimizations based on the high-level model and tradeoff analysis. These optimizations are performed as a "human-in-the-loop" process. In the following, we illustrate a scenario where a designer uses the energy distribution among various components to improve energy efficiency of the complete design. We consider Design 1 described in the previous example as the candidate design. As shown earlier, for Design 1, $s = n$ results in an energy efficient design within the domain. We illustrate how energy efficiency can be improved further based on the details captured by the high-level model. Figure 5 shows the distribution of energy dissipation in Design 1 for $s = n$ and $n = 3, 12$. For Design 1 (Figure 3), 47% and 76% of the energy is dissipated in registers for problems of size 3×3 and 12×12, respectively. Note that the bulk of the energy is dissipated in the registers. Also, it is difficult to reduce energy dissipation in multipliers and I/O ports without increasing latency (hence energy dissipation). We have developed some techniques at algorithm and architecture level to reduce the number of registers used in matrix multiplication.

Our techniques reduce the number of registers from $2n^2 + 6n$ to $n^2 + 4n$ [7]. For example, two registers ($AS.LR$ and $AS.RR$) in Design 1 (Figure 3.b) are replaced by one register by feeding elements of matrix A n cycles after feeding those of matrix B.

Careful analysis of data movement reveals that only two registers ($B.T1$ and $B.T2$) are enough to store the elements of B. We refer to this new design as Design 2. As shown in Figure 5, in Design 2, overall energy dissipation is reduced by 16% and 25% for problems of size 3×3 and 12×12, respectively. Amount of energy dissipation in registers is reduced from 47% and 76% to

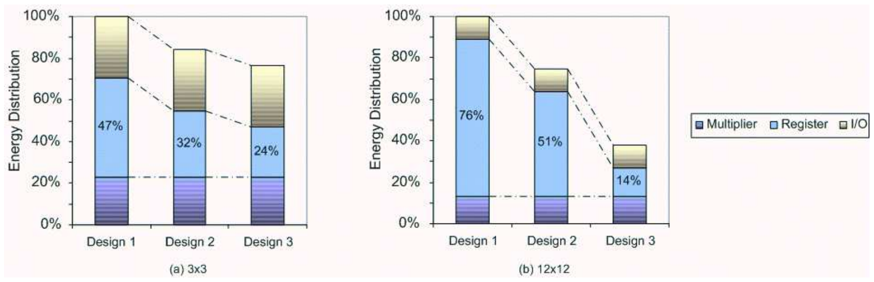


Figure 5. Change in distribution of energy dissipation among modules during optimization

32% and 51%, respectively.

Besides reduction of energy via optimizations at architecture and algorithm level, further reduction is possible at the implementation level. For example, the registers to store intermediate results ($C[j]$ in Figure 3) can be replaced by CLB-based SRAMs to reduce the power per unit storage. However, the minimum number of words for the SRAM in the target FPGA [7] must be 16. We denote this design as Design 3.

Compared with the original design (Design 1), in Design 3, energy dissipation was reduced by 23% and 62% for problems of size 3×3 and 12×12, respectively. For these designs, only 24% and 14% of the total energy is dissipated in the registers.

To compare the energy efficiency with the Xilinx design, Design 3 with problem size 3×3 is chosen for low level synthesis. Low level synthesis was performed using Synopsys FPGA Express and Xilinx XST (Xilinx Synthesis Technology) in Xilinx ISE 4.1i design environment. The place-and-route file (.ncd file) was obtained for a target FPGA device, Virtex-II XC2V1500. Mentor ModelSim 5.5e was used to simulate the design and generate simulation results (.vcd file). These two files were then provided to the Xilinx XPower tool to estimate energy dissipation. The result is compared against the highly optimized Xilinx reference design [18] in Table 4. Design 3 is superior to the Xilinx design by 32%, 64%, and 35% in terms of energy, latency, and average power density. However, in terms of area, our design is 2.3× of the Xilinx design. Details of the designs, energy functions, and additional results can be found in [7].

Table 4. Performance comparison between Design 3 and the Xilinx design

Metric	Xilinx	Design 3	Ratio
Energy (*nJ*)	25	17	68%
Latency (cycles)	45	16	35%
Area (slices)	180	415	2.3x
E/AT	0.67	0.43	64%

6. Conclusion

In this paper, we proposed a model-based design methodology for designing energy efficient data paths using FPGAs for a specific application. This methodology integrated domain specific high-level modeling, design space exploration, high-level energy estimation, and low-level simulations. Matrix multiplication was chosen as a candidate application to illustrate the design of an energy efficient data path. We considered two domains that implement matrix multiplication. We also demonstrated algorithm-level optimization techniques to improve the energy efficiency of one of the designs implementing matrix multiplication.

For both the domains, we demonstrated the tradeoffs among energy, latency, and area for various sizes of matrix multiplication. The designs based on our methodology demonstrated (on the average) an improvement of 57% in energy dissipation and 14x in latency when compared with the state-of-the-art designs for matrix multiplication provided by Xilinx. High-level modeling based on a family of architectures and corresponding algorithms (domain) results in fairly accurate estimate of the system-wide energy dissipation. Our energy estimates were within 7.4% of the estimates using low-level simulations.

Acknowledgment
This work is supported by the US DARPA Power Aware Computing/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. In addition, Ju-wook Jang's work is supported by the Ministry of Information and Communication, Korea. We thank Amol Bakshi for his inputs regarding design space exploration.

References

[1] A. Agrawal, A. Bakshi, J. Davis, B. Eames, A. Ledeczi, S. Mohanty, V. Mathur, S. Neema, G. Nordstrom, V. Prasanna, C. Raghavendra, M. Singh, "MILAN: A Model Based Integrated Simulation for Design of Embedded Systems," Language Compilers and Tools for Embedded Systems, 2001.

[2] K. Bondalapati and V. K. Prasanna, "Loop Pipelining and Optimization for Reconfigurable Architectures," Reconfigurable Architectures Workshop (RAW), May 2000.

[3] S. Choi, S. Mohanty, J. Jang, and V. K. Prasanna, "Domain-Specific Modeling for Rapid System-Level Energy Estimation of Reconfigurable Architectures," Intl. Conference on Engineering of Reconfigurable Systems and Algorithms, 2002.

[4] A. Dandalis and V. K. Prasanna, "Signal Processing using Reconfigurable System-on-Chip Platforms," International Conference on Engineering of Reconfigurable Systems and Algorithms, June 2001.

[5] J. A. B. Fortes, K. S. Fu, and B. Wah, "Systematic Approaches to the Design of Algorithmically Specified Systolic Arrays," International Conference on Acoustics, Signal and Speech Processing, 1985.

[6] A. Garcia, W. Burleson, and J.L. Danger, "Power Modeling in FPGAs," 9th International Conference on Field Programmable Logic and Applications, 1999.

[7] J. Jang, S. Choi, and V. K. Prasanna, "Energy-Efficient Matrix Multiplication on FPGAs," Manuscript, Dept. of Electrical Engg., University of Southern California, April, 2002.

[8] D. Kumar and K. Parhi, "Performance Trade-off of DCT Architectures in Xilinx FPGAs," The 33rd Asilomar Conference on Signals, Systems, and Computers, 1999.

[9] V. Kumar, A. Grama, A. Gupta, and G. Karypis, "Introduction to Parallel Computing: Design and Analysis of Algorithms," Benjamin Cummings, November, 1993.

[10] S. Lei and K. Yao, "Efficient Systolic Array Implementations of Digital Filtering," IEEE International Symposium on Circuits and Systems, 1989.

[11] W. Luk, N. Shirazi, and P.Y.K. Cheung, "Modeling and Optimizing Run-time Reconfigurable Systems," IEEE Symposium on FPGAs for Custom Computing Machines, 1996.

[12] Model-based Integrated Simulation, http://milan.usc.edu/.

[13] 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.

[14] M. Nemani and F. N. Najm, "High-level Area and Power Estimation for VLSI Circuits," IEEE/ACM International Conference on Computer-Aided Design, 1997.

[15] 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.

[16] A. Raghunathan, N. K. Jha, and S. Dey, "High-level Power Analysis and Optimization," Kluwer Academic Publishers, 1998

[17] F. G. Wolff, M. J. Knieser, D. J. Weyer, C. A. Papachristou, "High-level Low Power FPGA Design Methodology," National Aerospace and Electronics Conference, 2000.

[18] Xilinx Application Note: Virtex-II/Virtex-II Pro Series and Xilinx ISE 4.1 Design Environment, http://www.xilinx.com.