Domain-Specific Modeling for Rapid System-Wide Energy Estimation of Reconfigurable Architectures

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

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

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

Abstract— **Reconfigurable architectures such as FPGAs are flexible alternatives to DSPs or ASICs used in mobile devices for which energy is a key performance metric. Reconfigurable architectures offer several parameters such as operating frequency, precision, amount of memory, number of computation units, etc. These parameters define a large design space that must be explored to find energy efficient solutions. Efficient traversal of such a large design space requires high-level modeling to facilitate rapid estimation of system-wide energy. However, FPGAs do not exhibit a high-level structure like, for example, a RISC processor for which high-level as well as low-level energy models are available.**

To address this scenario, we propose a domain-specific modeling technique that exploits the knowledge of the algorithm and the target architecture family for a given problem to develop a high-level model. This model captures architecture and algorithm features, parameters affecting power performance, and power estimation functions based on these parameters. A system-wide energy function is derived based on the power functions and cycle specific power state of each building block of the architecture. This model can be used to understand the impact of various parameters on system-wide energy and can be a basis for the design of energy efficient algorithms. Our high-level model can be used to quickly obtain fairly accurate estimate of the system-wide energy of data paths configured using FPGAs. We demonstrate our modeling methodology by applying it to two domains.

Keywords— **domain modeling, energy estimation, energy optimization**

I. Introduction

Dramatic increase in the density and speed of FPGAs makes them attractive for complex applications. The state-of-the-art Virtex-II Pro FPGA from Xilinx delivers over 0.3 Tera MACs/sec. at an operating frequency of 300 MHz. Table I [16] shows the peak performance capabilities of the Virtex FPGA compared with the fastest DSP available last year.

With such an available processing power, FPGAs are an attractive fabric for implementing complex and compute intensive applications such as signal processing kernels for mobile devices. Mobile devices operate in power constrained environments. Therefore, in addition to time performance, power performance is a key performance metric [11]. Studies show that optimization at the algorithmic level has a much higher impact on total energy dissipation of a system than RTL or gate level. It is reported that the impact (on energy optimization) ratio is 20 : 2.5 : 1 for algorithmic, register, and circuit level [13]. In this context, there is a need for a high-level energy model which not only enables algorithmic level optimizations but also provides rapid and reasonably accurate energy estimates.

TABLE I
Performance Comparison of FPGAs and DSP (Xilinx Inc.)

Function	Fastest DSP	Virtex-II
8×8 MAC	4.4 billion MACs	600 billion MACs
FIR, 256 tap, 16-bit data/coefficient	17 MSPS (1.1 GHz)	180 MSPS (180 MHz)
1024 point FFT (16 bit data)	7.7 μ *sec.* (800 MHz)	.1 μ *sec.* (140 MHz)

Several issues must be addressed in developing a high-level energy model for FPGAs. There are numerous ways to map an algorithm onto an FPGA as opposed to mapping onto a traditional processor such as a RISC processor or a DSP, for which the architecture and the components such as ALU, data path, memory, etc. are well defined. For FPGAs, the basic element is the lookup table (LUT), which is too low-level an entity to be considered for high-level modeling. Besides, the architecture design depends heavily on the algorithm. Therefore, no single high-level model can capture the energy behavior of all feasible designs implemented on FPGAs. In addition, to elevate the level of abstraction, high-level models do not capture all the details of a system and consider only a small set of key parameters that affect energy. This lowers the accuracy of energy estimation.

In order to address the issues discussed above, we propose a *domain-specific modeling* technique (Figure 1). This technique facilitates high-level energy modeling for a specific domain. A domain corresponds to a family of architectures and algorithms that implements a given kernel. For

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 LG Yonam Foundation.

example, a set of algorithms implementing matrix multiplication on a linear array is a 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. By restricting our modeling to a specific domain, we reduce the number of architecture parameters and their ranges, thereby significantly reducing the design space. A limited number of architecture parameters also facilitate development of power functions that estimate the power dissipated by each component (a building block of a design). For a specific design, the component specific power functions, parameter values associated with the design, and the cycle specific power state of each component are combined to specify a system-wide energy function.

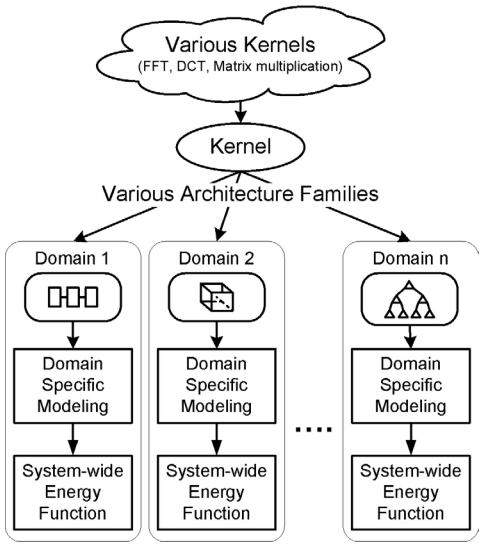

Fig. 1. Domain-Specific Modeling

Our approach is a top-down approach in contrast with other approaches that exploit low-level simulations and estimations for each component and accumulate these results to estimate overall energy dissipation. The advantage of our approach is the ability to rapidly evaluate the system-wide energy using energy function for different designs within a domain. Our high-level energy model also facilitates algorithmic level energy optimization through identification of appropriate values for architecture parameters such as frequency, number of components, precision, etc., early in system design.

The organization of the paper is as follows. The next section describes the domain-specific modeling technique. Section III describes the methodology to estimate the power functions. A detailed description of high-level modeling and energy estimation based on the proposed model for two specific domains is presented in Section IV. We discuss some applications of the high-level model in Section V. Related efforts are discussed in Section VI. Section VII concludes the paper.

II. Domain-Specific Energy Modeling

The goal of our domain-specific modeling (Figure 2) is to represent energy dissipation of the designs specific to a domain in terms of parameters associated with this domain. For a given domain, only those parameters which can significantly affect system-wide energy dissipation and can be varied at algorithmic level are chosen for the high-level energy model. As a result, our model a) facilitates algorithmic level optimization of energy performance, b) provides rapid and fairly accurate estimates of the energy performance, and c) provides energy distribution profile for individual components to identify candidates for further optimization. First, we define the high-level energy model. Then we provide details of energy estimation using this model.

A. High-level Energy Model

Our high-level energy model consists of *RModules, Interconnects, component specific parameters and power functions, component power state matrices,* and a *system-wide energy function.*

Relocatable Module (RModule) is a high-level architecture abstraction of a computation or storage module. It is either a CLB-based logic or a "larger" module composed of multiple RModules and *Interconnects*. For example, a register can be a *RModule* if the number of registers varies in the design depending on algorithmic level choices. One important assumption about RModule is that energy performance of an instance of a RModule is independent of its location on the device. While this assumption can introduce small error in energy estimation, it greatly simplifies the model. *Interconnect* represents the connection resources used for data transfer between the RModules. The power consumed in a given Interconnect depends on its length, width, and switching activity. Interconnect can be of various types. For example, in Virtex-II FPGAs, there are several Interconnects such as *long lines*, *hex lines*, *double lines*, and *single connections* which differ in their lengths [16]. In the rest of the paper, we use *component* to refer to both RModule and Interconnect.

Component specific parameters depend on the characteristics of the component and its relationship to the algorithm. For example, operating frequency or precision of a multiplier RModule can be chosen as parameters if they are varied by the algorithm. Possible candidates parameters include operating frequency (f), input switching activity (sa), word precision (w), power states (ps), number of RModule type i (n_i), and etc. *Component specific power functions* capture the effect of *component specific parameters* on the average power dissipation of the component. The power functions are obtained by implementing sample designs of individual components and simulating them using low-level simulators (See Section III).

Component Power State (CPS) matrices capture the power state for all the components in each cycle. For example, consider a design that contains k different types of components $(C_1, ..., C_k)$ with n_i components of type i. If

the design has the latency of T cycles, then k two dimensional matrices are constructed where the i-th matrix is of size $T \times n_i$ (Figure 3). An entry in a CPS matrix represents the power state of a component during a specific cycle and is determined by the algorithm.

System-wide energy function represents the energy dissipation of the designs belonging to a specific domain as a function of the parameters associated with the domain.

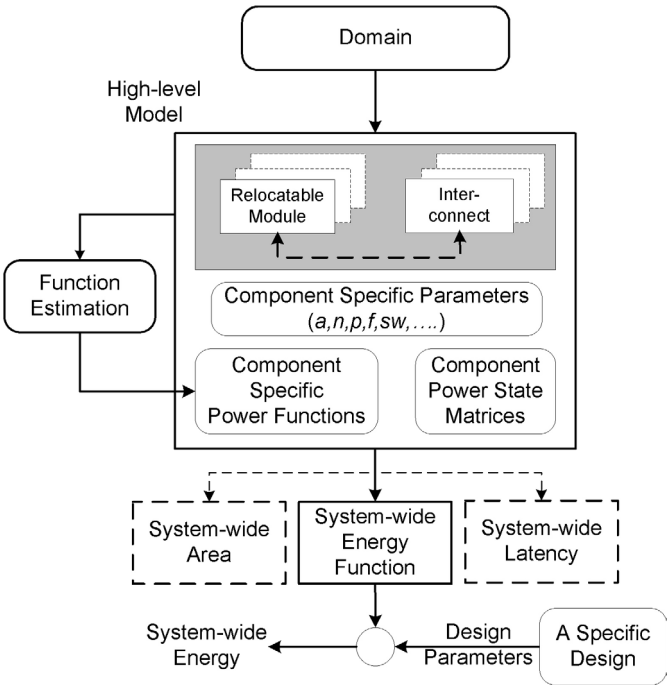

Fig. 2. Domain-Specific Modeling and System-wide Energy Estimation

The domain-specific nature of our energy modeling is exploited when the designer identifies the level of architecture abstraction (RModules and Interconnects) appropriate to the domain and/or chooses the parameters to be used in the *component specific power functions*. This is a human-in-the-loop process and exploits the designer's expertise in the algorithm and the architecture family that constitutes the domain. Well-known power models based on capacitance, voltage, and switching frequency can be more accurate and are generic to be applicable across many domains. However, they do not provide a designer a clear understanding of the impact of his/her algorithmic level design choices on the energy performance. Our modeling enables the designer to rapidly explore a large design space based on the understanding of the effect of the design choices on the overall energy performance.

To handle modeling complexity we follow a hierarchical approach. Each RModule can be recursively divided into RModules and Interconnects. This hierarchical nature allows the designer to capture the details of architecture in the design at various levels of abstraction to define parameters affecting performance.

B. Energy Estimation

Power dissipation by a RModule or Interconnect in a particular state is captured as a power function of a set of parameters. These functions are typically constructed through curve fitting based on some sample low-level simulations (described in Section III). These functions may also be provided by the vendors.

CPS matrices contain cycle specific power state information for each component. The entries in the CPS matrices are determined by the algorithm.

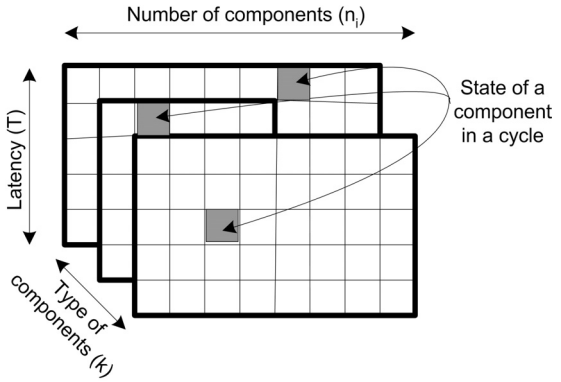

Fig. 3. Component Power State Matrices

Combining the CPS matrices and component specific power functions (See Section III) for individual components, the total energy of the complete system is obtained by summing the energy dissipation of individual components in each cycle. The system-wide energy function SE is obtained as:

$$SE = \sum_{i=1}^{k} \frac{1}{f} \left(\sum_{j=1}^{n_l} \sum_{t=1}^{T} C_i.p.ps \right) \text{ where } ps = CPS(i,t,j) \quad (1)$$

$C_i.p.ps$ is the power dissipated in the j-th component ($j = 1...n_l$) of type i during cycle t ($t = 1...T$) and f is the operating frequency. $CPS(i,t,j)$ is the power state of the j-th component of the i-th type during the t-th cycle.

Since the system-wide energy function is derived using component specific power functions, the energy distribution among various components (the fraction of the total energy consumed by each component) can be obtained. This information is used to identify candidate components to be considered by the designer for energy optimization. Details can be found in Section IV.

Due to the high-level nature of the model, we can rapidly estimate the system-wide energy. In the worst case, the complexity of energy estimation is $O(T \times \sum_{l=1}^{k} n_i)$ (See Equation 1) which corresponds to iterating over the elements of the CPS matrices and adding the energy dissipation by each component in each cycle. However, typically, there is a repeating pattern of state changes for a component (for example, due to loop structures within the algorithms). Also, different components of the same type dissipate the same amount of energy during each cycle.

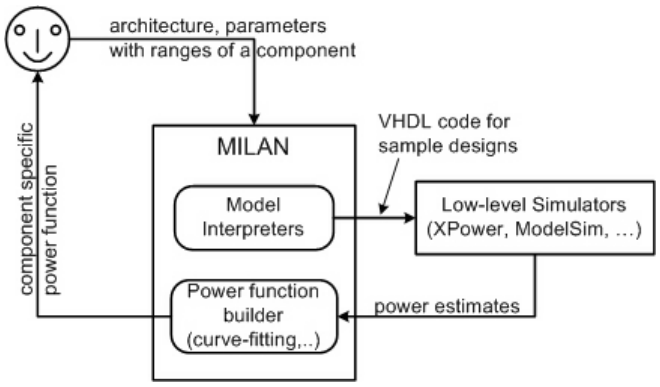

Fig. 4. Power Function Estimation using MILAN

Therefore, based on these observations the time to compute the energy is better than the worst case complexity of energy estimation stated above. Further, even if we compute the system-wide energy based on each cycle we do not analyze the activities at the level of individual gates. Typically, there are only a few distinct components within a domain that affect energy dissipation of the designs in that domain. Indeed, for the illustrative examples considered in this paper, the time for energy estimation does not depend on the problem size.

III. Estimating Component Specific Power Functions

We use the MILAN framework [1] to derive the component specific power functions associated with the high-level energy model. MILAN is a **M**odel based **I**ntegrated simu**LA**tio**N** framework for embedded system design and optimization by integrating various simulators and tools into a unified environment. In order to use the framework, the designer first models the target system using the modeling paradigms provided in MILAN. The designer provides the architecture and the parameters (with their possible ranges) that significantly affect the power dissipation of the component. Model interpreters(MI) in the MILAN are used to drive the integrated tools and simulators. Model interpreters (MI) translate the information captured in the models into the format required by the low level simulators and tools.

Let $F(p_1, \ldots, p_n)$ be the component specific power function and $p_1, \ldots, p_n$ be the parameters associated with the component. Figure 4 illustrates the process of deriving component specific power functions. This process involves estimation of power dissipation through low-level simulation of the component at different design points (a design point is a unique combination of parameter values). For low-level simulations, we have integrated simulator such as XPower [16] and ModelSim [8] into the MILAN framework. The switching activity for the input to the component can be provided by the designer or specified as some default values, depending on the desired accuracy. One can choose appropriate values based on prior experience for the ease of analysis. However, if higher accuracy is needed, behavioral simulation of the complete application over expected input vectors to the whole system can be performed to obtain exact values for switching activity at the input of each component.

Low-level simulation is performed at each of the chosen design points to estimate the power dissipation. These power estimates are fed to the power function builder. A typical low-level simulation for power estimation of a sample design point proceeds as follows. The chosen sample VHDL design is synthesized using Synopsys FPGA Express on Xilinx ISE 4.1i. 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 power function builder is driven by an MI from the MILAN framework. For components with a single parameter, the power function can be obtained from curve-fitting on sample simulation results. In case of larger number of the parameters, surface fitting can be used. Currently, we only focus on building component specific power functions with at most two parameters. The resulting power functions are provided back to the designer.

The component specific power function of an interconnect depends on its length, operating frequency, and the switching activity. We use Equation 2 to estimate power dissipation in an interconnect. $\Phi.p$ denotes the power dissipation of a cluster of k RModules connected through the candidate interconnect and $M.p_i$ represents power dissipation of the i-th RModule. The power dissipated by the cluster is obtained by low-level simulation.

$$IC.p = \Phi.p - \sum_{i=1}^{k} M.p_i \tag{2}$$

While the initial effort to build the component specific power function might be costly compared with ad hoc approaches, the benefits are noticeable when the same components are re-used in different designs within and across domains.

IV. Illustrative Examples of Domain-Specific Energy Modeling

To illustrate our domain-specific energy modeling, we apply the techniques discussed in the previous sections to define a high-level model for two domains implementing matrix multiplication, a frequently used kernel operation in wide variety of signal processing algorithms. For each domain we identify the components and the component specific parameters, identify the power functions for each component, and finally derive a system-wide energy function. Two architecture families, a uniprocessor architecture and a linear array architecture, are chosen to demonstrate our approach.

A. Example 1: Uniprocessor Architecture

We define a uniprocessor (PE) implementing the usual block matrix multiplication as the first domain. The PE has one MAC (multiplier and accumulator), a cache of size c, and I/O ports (see Figure 5). For the sake of illustration, we assume that all operations have a single cycle latency and the data matrices are stored in an external memory. For $n \times n$ matrix multiplication, computational complexity of the algorithm is $O(n^3)$. Block matrix multiplication (BMM) is performed with block size of $\sqrt{c} \times \sqrt{c}$. The I/O complexity (amount of traffic between the PE and external memory) is $O(n^3/\sqrt{c})$. This complexity corresponds to the minimum achievable I/O complexity for matrix multiplication [5]. It can be observed that a large cache decreases the I/O complexity and as a result improves the energy performance for I/O.

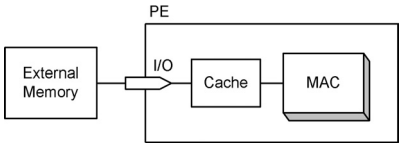

Fig. 5. Uniprocessor Architecture

A.1 Identifying Components and Parameters

We identified the MAC and the cache as RModules and the I/O as an Interconnect. The size of the cache can be controlled by the algorithm. The RModules have w bit precision and the cache has one more parameter, c, the cache size. For the sake of illustration, the cache size is the only variable parameter and $w = 8$ is used. The component specific power functions for MAC ($M.p$), cache ($R.p$), and I/O ($IO.p$) are obtained through low-level simulation. The MAC is implemented using CLB-based multiplier and the cache is realized using register modules provided in the Virtex-II library. $M.p$ and $IO.p$ are constants. The power function for the cache is: $R.p(c) = 22.88 + 2.34c$ (mW).

A.2 System-wide Energy Function

We consider the energy dissipated by the PE on an FPGA. We do not consider the energy dissipated by the external memory. The system-wide energy function (SE) is:

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

Note that as c varies, we obtain a family of the architectures each implementing matrix multiplication using BMM with different block sizes. The operating frequency f is 166 MHz. Figure 6 shows how different values of c affect the system-wide energy and the energy distribution between the components of the complete system for a 6×6 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 later for large values of c the system-wide energy goes up.

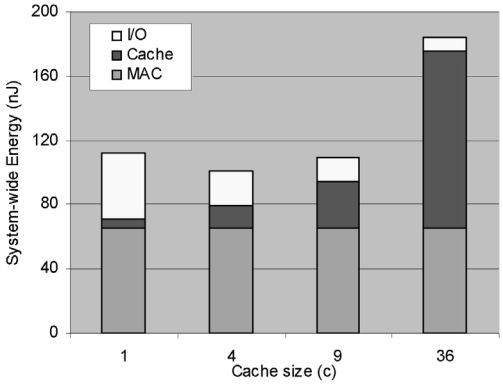

Fig. 6. System-wide Energy for a PE and Energy Distribution as a Function of Cache Size

B. Example 2: Linear Array Architecture

For the second domain, we considered a linear array of processing elements (PEs) with constant I/O bandwidth (independent of the problem size) as the architecture family to perform matrix multiplication. The following discussion is based on optimal algorithms for matrix multiplication on a linear array family of architectures [12].

B.1 Defining Components and Parameters

The structure of the linear array is shown in Figure 7. It consists of two components: processing elements (PEs) and interconnects connecting adjacent PEs. For the purpose of high-level modeling, we identified the PE as an RModule and the bus between two PEs as an Interconnect.

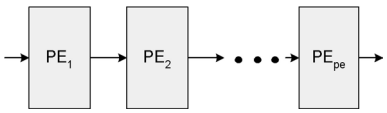

Fig. 7. Linear Array of PEs

In order to identify the component specific parameters, we analyze the structure of each PE. The PE (See Figure 8) has a MAC of precision w and a memory of size s. The memory is realized by using register modules provided in the FPGA library. 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 gated clocking. The PE also includes 6 registers and 3 multiplexers of w bits. The key parameters affecting energy are precision (w), amount of memory within a PE (s), and power states (ps).

A matrix multiplication algorithm for linear array architectures is proposed in [12]. 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

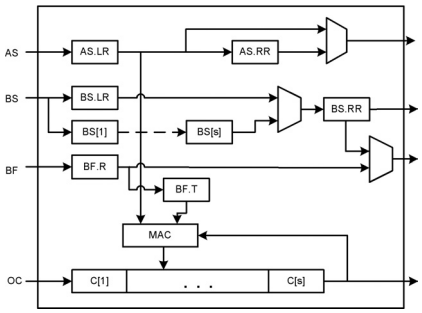

Fig. 8. The details of the PE

needed for a $n \times n$ matrix multiplication is n [12]. 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 [12]:

$$T = \frac{1}{f}(n^2 + 2n\lceil n/s \rceil - \lceil n/s \rceil + 1). \quad (3)$$

We consider the problems of size $1 \leq n \leq 16$. For the sake of illustration, we fixed w at 8. The parameters and their ranges are shown in Table II. Note that the parameters of interest are pe, ps, and s. The system-wide energy function is specified using these three parameters.

TABLE II
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

B.2 Estimating Power Functions

To estimate the power function of the PEs, we applied the technique described in Section III. The amount of memory per PE (s) was varied. 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 chose the operating frequency as 166 MHz since we compare our designs with the matrix multiplication provided in the Xilinx library which operates at the same frequency. The comparison can be found in Section V. The power function for the PE is given by:

$$PE.p.ps = \begin{cases} 7.01s + 31.04 \text{ (mW)}, & ps = on \\ 7.01s + 14.04 \text{ (mW)}, & ps = off. \end{cases} \quad (4)$$

The interconnect power function is constant. It is estimated using Equation 2 since the interconnect between the PEs is localized in the design and is regular. We implemented two PEs and the interconnect, and measured the power dissipation while both PEs are in ON state. The power dissipated in the interconnect is $IC.p = 39.74$ mW.

Thus, our high-level model for matrix multiplication on linear array architecture consists of PEs, Interconnects, component specific parameters and their ranges as shown in Table II, the power function for the PE (Equation 4), and the power function for the interconnect.

B.3 Specifying System-wide Energy Function

To derive the system-wide energy function (SE) (shown in Equation 5), we use Equation 1. In this domain, we can separate the equation into three parts: the total energy dissipated in the PE when the multiplier is on ($E_{PE}|_{on}$), the total energy dissipated in the PE when the multiplier is off ($E_{PE}|_{off}$), and the total energy dissipated in interconnect (E_{IC}). Therefore,

$$SE(n, s) = E_{PE}|_{on} + E_{PE}|_{off} + E_{IC} \quad (5)$$

Note that the power dissipation of all the PEs (at a given state of multipliers) is identical. In each PE, the multiplier is on for a duration of T / ($\lceil n/s \rceil$) [12]. $E_{PE}|_{on}$ is shown in Equation 6. In each PE, the multiplier is off for a duration of $T \cdot (1 - 1/\lceil n/s \rceil)$. $E_{PE}|_{off}$ is shown in Equation 7. Equation 8 shows the total energy of interconnect. $PE.p.ps$ refers to the power dissipation of PE when its multiplier is in state ps (See Equation 4).

$$\begin{aligned} E_{PE}|_{on}(n, s) &= \frac{T}{\lceil \frac{n}{s} \rceil} \cdot n \left\lceil \frac{n}{s} \right\rceil \cdot PE_{.p.ps=on} \\ &= n \cdot T \cdot PE_{.p.ps=on} \end{aligned} \quad (6)$$

$$\begin{aligned} E_{PE}|_{off}(n, s) &= T(1 - \frac{1}{\lceil \frac{n}{s} \rceil}) \cdot n \left\lceil \frac{n}{s} \right\rceil \cdot PE_{.p.ps=off} \\ &= T(n \left\lceil \frac{n}{s} \right\rceil - n) \cdot PE_{.p.ps=off} \end{aligned} \quad (7)$$

$$E_{IC}(n, s) = T(n \left\lceil \frac{n}{s} \right\rceil - 1) \cdot IC_{.p} \quad (8)$$

In order to verify the accuracy of our high-level energy modeling, we performed the following experiments. We considered several sample designs with various problem sizes ($n = 3, 6, 8, 9, 12, 16$). We set $s = n$. We used the system-wide energy function to estimate the system-wide energy dissipation of these designs. We compared this result with a complete VHDL simulation using the Xilinx tools. From Equation 3 the latency is $n^2 + 2n$ clock cycles. Further, the PE, when $s = n$, is always in the ON state. Therefore, the system-wide energy function simplifies to $SE(n, n) = E_{PE}|_{ON} + E_{IC}$. The system-wide energy for $n \times n$ matrix multiplication with $w = 8$ is given in column labeled "Estimated" in Table III.

We also implemented each of the sample designs in VHDL and simulated them using Xilinx ISE 4.1i and Modelsim 5.5e. The energy estimation is performed using Xilinx XPower. The input switching activity to these designs is the same (25%) as used during evaluation of component

TABLE III
Illustration of Accuracy of Our Modeling in Example 2

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

specific power functions. The measured energy values are shown in column labeled "Measured" in Table III.

Table III also shows the error percentage of our high-level estimation method when compared with energy estimation values obtained through low-level simulation. The error on the average is within 6.4% and is 7.4% in the worst case. The time needed to perform high-level estimation (assuming the power functions are pre-computed) is on the order of minutes on a Pentium III Xeon running at 700 MHz, whereas the time needed for low-level simulation and power estimation was 3 hours per design on the same machine.

V. Design Methodology Using the Model and Energy Optimization

Our domain-specific modeling provides an energy estimation methodology to facilitate design decisions in the early phases of the design cycle. The system-wide energy function captures the impact of the architecture parameters on the system-wide energy at the algorithmic level. Using this, the designer identifies trade-offs among area, latency, and energy. The designer explores a domain and identifies an appropriate design based on a selection criteria. The detailed design methodology can be found in [10].

To demonstrate the energy efficiency of our data path designs, we compare them with the designs provided by the Xilinx library [16]. Xilinx provides a module for 3×3 matrix multiplication. To perform $\mathbf{C} = \mathbf{A} \times \mathbf{B}$, the module uses one set of 3 registers for $\mathbf{A}$ matrix to store one row and the another set of 9 registers to store the complete $\mathbf{B}$ matrix. A single multiplier is used. A row of data from $\mathbf{A}$ and the complete $\mathbf{B}$ are brought in to the module to compute the first row of $\mathbf{C}$. This process is repeated for the other two rows of $\mathbf{A}$ to generate the complete $\mathbf{C}$. For larger problem size, block matrix multiplication with block size 3×3 is used. The operating frequency of the module is 166 MHz. The energy dissipation was measured using XPower. Table IV shows the area, latency, and the system-wide energy of the designs.

For our design, we estimate the energy dissipation using the system-wide energy function (Equation 5) with the same operating frequency as the Xilinx module, $f = 166$ MHz. The other parameters are $w = 8$, $n = 6$, and $s = 6$. Our design uses more area compared with the Xilinx design since we use 6 multipliers. However, the larger I/O requirement of the Xilinx library results in higher energy dissipation. Our design dissipates about 30% less energy.

TABLE IV
Comparison for 6×6 matrix multiplication

Metric	Xilinx design [16]	Design based on our methodology
Area (# of slices)	179	1,074
Latency (μ sec.)	2.17	0.295
Energy (nJ)	260.5	169.1

Also, using our model and design methodology, the designer can further optimize a chosen design by improving the performance of the components that consume the significant energy. Initially, the system-wide energy function is analyzed to identify the distribution of energy dissipation among various types of components (See Figure 6). Components with higher percentage of energy dissipation are chosen as possible candidates for design modification. The detailed optimization techniques can be found in [10].

VI. Related Work

Several research efforts have focused on rapid energy estimation of a design on FPGAs. Shang and Jha [14] proposed a black-box approach to estimate energy based on input and output signal statistics. This approach is suitable for estimation of average power dissipation of a RT-level component to be embedded into a system. However, it is not applicable for algorithm level power analysis. On the other hand, our model captures various architecture parameters that can be manipulated at algorithmic level for energy optimization.

XPower, the power estimation tool provided by Xilinx [16], estimates the energy dissipation of FPGAs based on low-level simulation. The input to the tool is LUT-level place-and-route information along with details of switching activity for LUT-level components. While its accuracy is comparable with the actual execution of the design, it does not support energy estimation early in the design phase when the complete system description in some HDL is not available. Stammermann et al. presented ORINOCO, a software tool for power dissipation analysis and optimization at the algorithmic level from C/C++ and VHDL description [15]. However, C/C++ or VHDL descriptions do not capture parameters affecting system-wide energy and also a designer requires a complete knowledge of the final system before the code can be generated in these languages. Both ORINOCO and XPower are essentially estimation tools and can be used in our methodology to perform low-level sample simulations necessary for specifying our component specific power functions. We have compared our estimation accuracy against XPower.

Chou et al. proposed a hardware/software co-synthesis CAD tool (IPChinook) [4] primarily targeted towards ASIC design from a single high-level specification (for both hardware and software) and hardware/software co-simulation. This tool does not capture the effect of parameter variation on energy dissipation for individual compo-

nents which is essential for algorithmic level analysis.

In [2] regression tree [3] is used to improve the power estimation of a RT-level component. Starting with candidate variables (I/O bits), the variable v_i, which has the maximum impact on the power dissipation is identified. Then the sample power dissipation results of power measurement is split in two subsets based on this variable. The splitting is recursively performed to build a regression tree which ranks variables in their significance with respect to the power. It is a bottom-up approach starting from low-level implementation and ends in identifying significant variables affecting the power.

In contrast, our model starts with candidate parameters chosen from a high-level view of the architecture and algorithm. The effect of the parameters on the system-wide energy is captured in the component specific power functions. The component specific power functions are used to obtain parameter values for optimal power performance by traversing the design space at an algorithmic level.

VII. Conclusion

This paper introduced domain-specific energy modeling for rapid system-level energy estimation and algorithmic level optimization for reconfigurable architectures. The modeling captures the details of the architecture and the algorithm to identify parameters affecting the power performance, hence facilitating derivation of a system-wide energy function. Matrix multiplication on a uniprocessor and a linear array architecture were chosen as two domains to illustrate construction of a high-level model, derivation of power functions for individual components, and combine them to obtain an system-wide energy function.

For one specific domain, the system-wide energy function was tested on several sample designs for its accuracy against time-consuming low-level energy estimation. The reference low-level energy evaluations were obtained on a Virtex-II chip through synthesis using Xilinx ISE 4.1i, simulation by ModelSim 5.5e, and power estimation by Xilinx XPower. The error in the system-wide energy estimation using the high-level model was within 6.4% and was 7.4% in the worst case. Using our modeling, the time needed to evaluate the system-wide energy function is in the order of minutes on a Pentium III Xeon running at 700 MHz while the low-level energy estimation takes more than 3 hours (on the average) for each sample design.

Our modeling methodology provides a virtual malleable data path. It is virtual because at the time of performance estimation we do not implement the design on target FPGAs. It is malleable because there are several parameters that can be varied to understand the trade-offs between different performance metrics such as energy, area, and latency. Such a characteristic makes our proposed model suitable to be considered during large application synthesis where several different kernels are integrated to implement a system such as MPEG encoding or Software Defined Radio. In such scenarios, once we have models for various kernel implementations, we can exploit the multi-level design space exploration (DSE) technique provided in the MILAN framework [1]. It evaluates the system-level design space against user specified constraints [9] to identify an appropriate design.

Currently, analytical techniques are being developed for various options for interconnection resources available in the state-of-the-art FPGA devices to predict their energy behavior. Family of architectures that contains interconnection networks such as hypercubes and binary trees are being considered for future analysis.

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. Bogliolo, L. Benini and G. Micheli, "Regression-based RTL Power Modeling," ACM Transactions on Design Automation of Electronic Systems, Vol. 5, No. 3, 2000.

[3] B. L. Bowerman and R. T. O'Connell, "Linear Statistical Models-An Applied Approach," 2nd Edition, Brooks/Cole Pub Co.

[4] P. Chou, R. Ortega, K. Hines, K. Partridge, and G. Borriello, "IPCHINOOK: An Integrated IP-based Design Framework for Distributed Embedded System," Design Automation Conference, 1999.

[5] J.-W. Hong and H. T. Kung, "I/O Complexity: The Red-Blue Pebbling Game," ACM Symposium on Theory of Computing (STOC), 1981.

[6] J. Jang, S. Choi, and V. K. Prasanna, "Energy-Efficient Matrix Multiplication on FPGAs," submitted to International Conference on Field Programmable Logic and Applications, 2002.

[7] V. Mathur and V. K. Prasanna, "A Hierarchical Simulation Framework for Application Development on System-on-Chip Architectures," IEEE Intl. ASIC/SOC Conference, 2001.

[8] ModelSim, Model Technologies, http://www.model.com/.

[9] 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," to appear in Language Compilers and Tools for Embedded Systems, 2002.

[10] 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 Intl. Conference on Application-specific Systems, Architectures and Processors, 2002.

[11] T. Mudge, "Power: A First-Class Architectural Design Constraint," IEEE Computer , Volume. 34, April 2001.

[12] 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.

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

[14] L. Shang and N. K. Jha, "High-Level Power Modeling of CPLDs and FPGAs," International Conference on Computer Design, 2001.

[15] A. Stammermann, L. Kruse, W. Nebel, and A. Prastsch, "System Level Optimization and Design Space Exploration for Low Power," Proc. of ISSS, 2001.

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