

# A Methodology for Energy Efficient Application Synthesis Using Platform FPGAs

Jingzhao Ou and Viktor K. Prasanna  
 Department of Electrical Engineering, University of Southern California  
 Los Angeles, California, 90089-2560 USA

**Abstract**—Platform FPGAs incorporate many different components, such as processor core(s), reconfigurable logic, memory, etc., onto a single chip. When an application is synthesized on platform FPGAs, part of it can be executed using hardware implementations on FPGA or software implementations on processor core(s). As the connection between different components on the devices are realized using FPGA routing resources, the designer has many choices for configuring the hardware components to execute the software. We show that these design choices have profound impact on the energy performance of the software programs. We propose a hybrid design approach for energy efficient application synthesis on platform FPGAs. It consists of a bottom-up process which performs simulation based performance modeling, and a top-down process which performs analytical performance optimization. The execution of an FFT software program on a state-of-the-art platform FPGA under various hardware choices is used to illustrate the bottom-up process. For the top-down process, we map an beamforming application onto hardware and software components based on the results from the bottom-up process. Energy reduction up to 46% is observed for the beamforming application using the proposed design approach.

## I. INTRODUCTION

Platform FPGAs, which integrate multi-million gate configurable logic, embedded processors, interconnect, dedicated multipliers, block RAMs, etc. on a single chip, are becoming popular. One major advantage of platform FPGAs is that a single platform (device) can target a wide range of applications. Examples of platform FPGAs include Altera Excalibur [5] and Xilinx Virtex-II Pro [14].

Platform FPGAs are different from general purpose SoCs (System-on-a-Chip) and configurable SoCs (CSoCs). Examples of such devices are OMAP processors [11] and Triscend CSoCs [12]. The various components on platform FPGAs are immersed in multi-million gate configurable logic and are connected through the programmable routing resources. This gives the designer many choices for configuring the embedded processor cores and their peripherals for executing software programs. For example, on Virtex-II Pro, the designer can organize the memory blocks distributed throughout the device into memory systems and attach them to the embedded PowerPC processors through various bus standards. As shown in Section IV-D, these hardware choices have a significant impact on the performance of the software programs. For general purpose SoCs and CSoCs, such communication channels between processors and their peripherals are implemented as ASICs and offer little or no configurability. Therefore, compared with the traditional hardware/software design flow shown in Figure 1(a), there is no clear separation of hardware and software designs in the design flow for platform FPGAs (shown in Figure 1(b)). Programming (configuring) of both hardware and software is required to optimize the performance of the applications synthesized on platform FPGAs.

Energy efficiency has become a key performance metric in the design of various computation and communication systems. It is especially critical in battery operated embedded and wireless systems. Platform FPGAs offer high efficiency with respect to time and energy performance and have been shown to achieve energy reduction and increase in computational performance of at least one order of magnitude compared with traditional processors [1]. Thus, platform FPGAs are being used to implement many of these systems.

This work is supported by United States National Science Foundation (NSF) under award No. CCR-0311823.

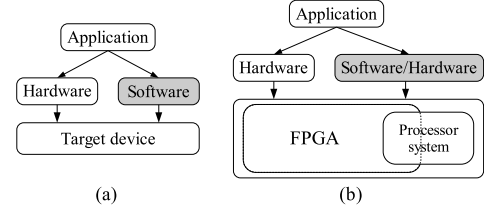


Fig. 1. (1) Traditional hardware/software codesign; (2) design using platform FPGAs

Multi-million gate configurable logic is the major component of platform FPGAs. The basic elements of the configurable logic are look-up tables (LUTs), which are too low-level an entity to be considered for high level modeling and rapid energy estimation. Energy estimation using RTL (Register Transfer Level) simulation (which can be accurate) is too time consuming. The challenges in obtaining energy performance models prevent rapid design space exploration and identification of energy efficient designs.

To address the above issues, we propose a *hybrid design approach*. It consists of a bottom-up process which performs simulation based performance modeling, and a top-down process which performs analytical performance optimization. The execution of an FFT software program on Virtex-II Pro, a state-of-the-art platform FPGA, under various hardware choices is used to illustrate the bottom-up process. Then, to illustrate the top-down process, we map an adaptive beamforming application onto the corresponding hardware and software components of Virtex-II Pro, based on the results from the bottom-up process. Energy reduction up to 46% is observed for the beamforming application using the proposed design approach.

The paper is organized as follows. Section II discusses the related work. Section III describes the hybrid design flow. Section IV exemplifies the bottom-up process through the execution of an FFT software program on a platform FPGA. Hardware choices that affect the performance of the software program are also analyzed. Section V presents the top-down design process. The energy performance of different mappings of an adaptive beamforming application onto platform FPGAs is given in this section to show the effectiveness of the proposed design approach. We conclude in Section VI.

## II. RELATED WORK

Experiments for re-mapping of critical software loops from a microprocessor to hardware implementations using configurable logic are carried out by Villarreal *et al.* [13]. Significant energy savings is achieved for a class of applications. As their target devices are CSoCs, the impact of different hardware configurations on the execution of software programs is not addressed in their research.

A hardware-software bipartitioning algorithm based on network flow techniques for dynamically reconfigurable systems is proposed by Rakhmatov *et al.* [7]. While their algorithm can be used to minimize the energy dissipation, design using platform FPGAs is more complicated than a hardware-software bipartitioning problem considering the various hardware choices for configuring the device. Therefore, their technique cannot be applied to energy efficient application synthesis for platform FPGAs.

Integrated design environments (IDEs) such as Embedded Development Kit (EDK) [14] are becoming available. They include both hardware and software design tools for application development on platform FPGAs. These IDEs take a bottom-up design approach. The energy performance of a design is available only after the design is completed and the time consuming low-level simulation is performed on the final design.

### III. DESIGN FLOW

We propose a *hybrid design approach* for energy efficient application synthesis on platform FPGA, which is shown in Figure 2. The input to our design flow is a task graph. That is, the target application is decomposed into a set of tasks with communication or precedence requirements between them. Based on the task graph, the application synthesis consists of a bottom-up process and a top-down process.

In the *bottom-up* process, the designer analyzes the performance of the various implementations of the tasks. For hardware implementations on FPGAs, techniques such as the one proposed by Choi *et al.* [2] can be applied to derive energy functions for various hardware implementations of the tasks. Then, energy performance of these hardware implementations can be rapidly obtained from vendor sheets or calculated using these energy functions. For software implementations on processor cores, the hardware choices that have significant impact on energy performance are identified. Low-level simulation is performed to obtain the energy performance for executing the software programs under various settings of these hardware choices. The performance values obtained in the bottom-up process is made available to the top-down process for optimizing the energy performance of the entire application.

In the *top-down* process, based on the task graph, the performance models of the tasks, as well as constraints from the application requirements and the target device, we formulate the energy efficient application synthesis problems as optimization problems with minimization of energy dissipation as the objective. Various optimization algorithms can be used to solve these optimization problems and realize the most energy efficient designs. The design process can be iterative. Low-level simulation can be performed based on the output of the top-down process and generate more accurate information on energy and time performance of the tasks. These information can be applied to the bottom-up process to refine the values of the performance models. Our previous work in [10] shows that the accuracy of the energy estimation of task  $T_4$  discussed in Section V-A is improved from 17.4% to 9.4% using such feedback information.

There are two major advantages offered by the hybrid design approach. First, performance analysis during the bottom-up process is confined to be within tasks. This enables techniques such as [2] for rapid and fairly accurate estimation of the energy performance of hardware implementations on FPGAs. For software implementations using processors, the different hardware configurations of the processor systems and their impact on energy performance are analyzed in this paper. Second, the performance models derived in the bottom-up process provide an abstraction of the platform FPGA device. Based on this abstraction, various optimization algorithms can be used to optimize the energy performance of the applications running on it. While hybrid design approach has been applied in designing applications on SoCs, to the best of our knowledge, we are the first to introduce it for energy efficient application synthesis using platform FPGAs.

### IV. THE BOTTOM-UP PROCESS

#### A. FPGA Designs

As discussed in Section III, domain-specific modeling can be applied to obtain the performance of hardware implementations of the tasks on FPGA. While more details about this technique can be

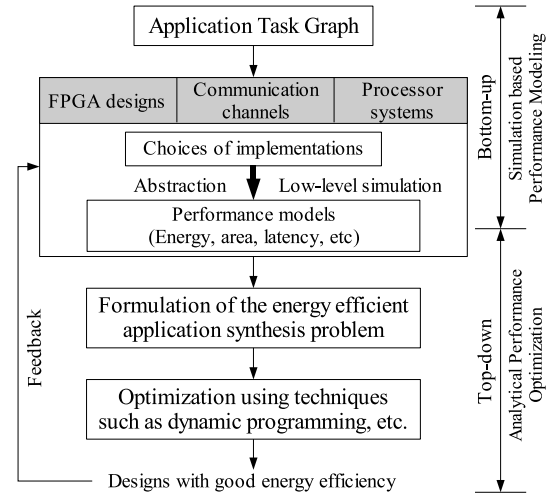


Fig. 2. The hybrid design approach

found in [2], we summarize it here for the sake of completeness. We divide different implementations of a task into groups of similar architecture. By doing so, we impose a high-level architecture onto the FPGA implementations within each domain. Then, energy functions are derived for each domain through low level simulation. The performance of the implementations is calculated rapidly using these energy functions.

#### B. Processor Systems

Software programs are executed on the processor systems on platform FPGAs. *Processor systems* are defined as systems that use processors as the main processing elements and use the programmable on-chip routing resources to connect to the processors and their other peripherals such as memory, instruction and data buses, etc. One example of processor systems is shown in Figure 3. There are many configurations of the processor systems. For example, on Virtex-II Pro, the processors can be the embedded PowerPC processor, or the MicroBlaze soft processor [14]; the buses can be one of three types of IBM CoreConnect buses, or the Xilinx Local Memory Bus (LMB); the memory is divided into blocks and can be organized to form various memory system and attached to the processors through various bus protocols.

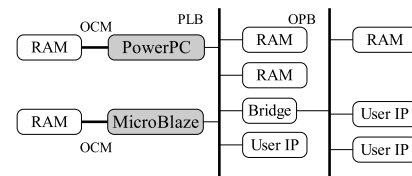


Fig. 3. An example processor system

#### C. Communication Channels

In the section, we will explore various ways for creating communication channels between processors and their peripherals on platform FPGAs and show their impact on the performance of application synthesis. Our analysis focuses on Virtex-II Pro due to wide availability of tools and devices. Also, the bus protocols and bus controllers used in Xilinx FPGAs are widely used in many other processor related SoC designs.

As the communication channels are implemented using FPGA routing resources, various buses can be implemented on platform FPGAs. Our analysis focuses on the following two types of buses.

TABLE I

ENERGY DISSIPATION OF AN 8-POINT FFT SOFTWARE PROGRAM ON THE POWERPC BASED PROCESSOR SYSTEM UNDER VARIOUS CONFIGURATIONS

Precision	Float	Float	Float	Float	Float	Float	Float	Float
Sine/Cosine	Library	Library	Library	Library	LUT	LUT	LUT	LUT
Bus (MHz)	OCM (50)	OCM (100)	PLB (100)	PLB (100)	OCM (50)	OCM (100)	PLB (100)	PLB (100)
Cache	Enable	Enable	Disable	Enable	Enable	Enable	Disable	Enable
Time ( $\mu$ s)	4239.76	3378.58	6806.28	1377.71	787.88	624.4	1327.88	241.61
Energy ( $\mu$ J)	400.66	334.48	673.82	136.39	74.45	61.82	131.46	23.92

Precision	Integer	Integer	Integer	Integer
Sin/Cosine	LUT	LUT	LUT	LUT
Bus (MHz)	OCM (50)	OCM (100)	PLB (100)	PLB (100)
Cache	Enable	Enable	Disable	Enable
Time ( $\mu$ s)	232.88	193.29	375.69	76.02
Energy ( $\mu$ J)	22.01	19.14	37.19	7.53

• PLB (Processor Local Bus) is a shared bus directly attached to the PowerPC processor. Several advanced techniques are employed in order to achieve high data throughput. For example, the read and write transfers are overlapped on PLB to allow two data transfers per clock cycle. In principle, PLB has the largest bandwidth over all the supported bus protocols and can be used to attach high-performance and high-speed IP cores to the processor. However, there are several factors that deteriorate its energy and time performance. First, if the data access pattern of a program fails to make good use of the advanced techniques, communication through PLB would turn out to be very inefficient. Second, as more peripherals are attached to the PLB bus, the energy efficiency and the maximum operating speed of the bus will go down. Third, as more workloads and more bus transactions occur on the bus, the bus utilization will also decrease. Load balancing by moving some data communication to the other buses is required to maintain the bus utilization.

• OCM (On-Chip Memory) bus is a 32-bit dedicated bus that can be directly attached to the PowerPC processor. The greatest advantage for using OCM buses is that it guarantees a fixed data access time. Since transmission over OCM buses does not require bus arbitration, such transmission is expected to have high energy efficiency. One major disadvantage of using OCM buses is that they are unable to use the cache available on the PowerPC processor and thus are unable to make use of the benefits provided by the cache for saving energy dissipation. For example, storing instructions in the OCM BRAM may turn out to be very inefficient when the process is executing a loop. In this case, the instructions with the loop have to be fetched through OCM buses each time they are executed.

#### D. Energy Performance

In this section, using the execution of an FFT software program on a PowerPC based processor system on Virtex-II Pro, we show the trade-offs in energy performance of the PowerPC based processor system on Virtex-II Pro platform FPGAs.

All the design discussed in this paper are described using EDK 3.2 [14], implemented using ISE 5.2, and simulated using [14] ModelSim 5.7 [8]. The power consumption is measured using XPower [14].

The PowerPC processor operates at 100 MHz. The FFT program is executed with the following choices.

- Precision: both *float* and *int* data types are considered.
- Sine/cosine functions: for *float* data, we consider the implementation through the *math.h* software emulation program or by looking up a table which contains the precomputed values. For integer data, only table look-up is considered.
- Bus: both OCM and PLB bus protocols are considered. The OCM bus operates at either 50 MHz or 100 MHz.
- Cache: cache is disabled for transactions over OCM bus in order to guarantee a fixed data access time. For PLB bus, both cases (cache is enabled and disabled) are considered.

The experimental results are shown in Table I. For the execution of the same software program, different configurations of the PowerPC based processor systems on Virtex-II Pro cause more than 5x difference in the time and energy costs. Such difference would affect the overall performance of the hardware and software mapping during application synthesis.

Implementations using the PLB bus and with caching *enabled* dissipate the least amount of energy while those using PLB bus and with caching *disabled* dissipate the largest amount of energy. Communication through PLB bus introduces more overhead than that using OCM bus. This increases the energy dissipation on both the processor and the bus as the PowerPC spends more time waiting for instructions and data to be fetched through the PLB bus. However, when caching is enabled, as the FFT code mainly consists of loops, this can effectively lower the amount of instruction and data transmission on the PLB bus. Thus, using the PLB bus achieves the best performance in this case. Besides, different operating frequencies of the buses and the processor result in more time and energy costs due to the introduction of more overhead for the communication between them.

## V. THE TOP-DOWN PROCESS

To illustrate the top-down process, we implement an MVDR (Minimum Variance Distortionless Response) beamforming application which uses the FFT kernel discussed above. This application is widely deployed in many embedded signal processing systems, e.g. software defined radio [4], where energy efficiency is an important performance metric. Our objective is to minimize the energy dissipation for executing one instance of the beamforming application.

### A. MVDR Beamforming

The MVDR beamforming application processes the data coming from  $M$  antenna elements. Its task graph consists of a linear pipeline of five tasks, from  $T_0$  to  $T_4$ . In  $T_0$ ,  $T_1$  and  $T_2$ , we implement a fast algorithm described in [6] for MVDR spectrum calculation. It consists of: Levinson Durbin recursion to calculate the coefficients of a prediction-error filter ( $T_0$ ), correlation of the predictor coefficients ( $T_1$ ), and the MVDR spectrum computation using FFT ( $T_2$ ). This fast algorithm eliminates a lot of computation that is otherwise required by direct calculation of the spectrum. We employ an LMS (Least Mean Square) algorithm ( $T_3$ ) to update the weight coefficients of the filter due to its simplicity and numerical stability. A spatial filter ( $T_4$ ) is used to filter the input data. The coefficients of the filter are determined by the previous tasks.

Various implementations of the tasks are developed. Different degrees of parallelism are employed in the hardware implementations of tasks  $T_0$  and  $T_1$ . Task  $T_2$  uses FFT to calculate the MVDR spectrum. We employ the radix-4 based hardware implementations

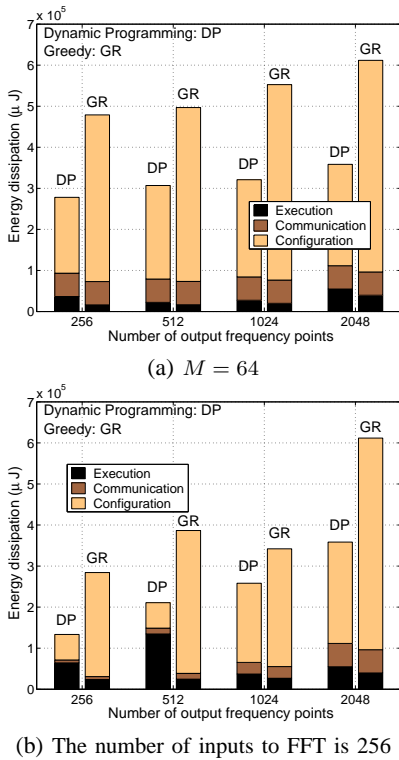


Fig. 4. Energy dissipation of the beamforming application

for FFT developed by Choi *et al.* [3]. Clock gating, various degrees of parallelism, and various memory bindings are used in these implementations to improve energy efficiency.  $V_p$  and  $H_p$  are the vertical and horizontal parallelism employed by the designs. Two different memory bindings, one using slice based RAMs and one using BRAMs, are used to store the intermediate values. In addition, we also consider an implementation using an IP core from Xilinx [14]. For task  $T_3$  and  $T_4$ , we employ different numbers of dedicated multipliers in their hardware implementations. Hardware implementations of  $T_0$ ,  $T_1$ , and  $T_2$  are developed as parameterized VHDL code while those of  $T_3$  and  $T_4$  are developed using a MATLAB/Simulink based design tool developed by Ou *et al.* [10]. All these hardware and software implementations are mapped on the corresponding components for execution. Performance values of these implementations are obtained through low-level simulation. Hardware implementations run at a clock rate of 50 MHz and the data precision is 10 bits. See [3] and [10] for additional details about these implementations and their performance.

Note that as mentioned in the previous sections, the energy performance of the hardware implementations of the tasks can be obtained rapidly and fairly accurately using the domain-specific modeling techniques proposed in [2].

### B. Optimization of Energy Performance

We create a trellis using the performance values obtained as discussed in the previous section. We let the nodes on the trellis represent the costs of the various implementations of the tasks and the edges between the nodes represent the communication costs between the tasks. Energy efficient application synthesis is then formulated as an optimization problem to find a path on the trellis with minimum cost. A dynamic programming algorithm is proposed to identify such a path in an iterative manner. For the sake of comparison, we also consider a greedy algorithm. See [9] for more details on the problem formulation and the optimization algorithms.

The performance when the adaptive beamforming application is synthesized on Virtex-II Pro when  $M = 64$  and when the number of inputs to FFT is 256 are shown in Figure 4. For all the considered cases considered, energy reduction from 41% to 46% are achieved by the dynamic programming algorithm over the greedy algorithm.

For both the dynamic programming and the greedy algorithms, tasks  $T_0$  and  $T_1$  are always mapped to the configurable logic. However, the dynamic programming algorithm maps it to the design using 2-input complex MAC (Multiplier-and-Accumulator) while the greedy algorithm maps them to the designs based on 4-input complex MAC in cases such as when  $M = 16$ . Designs based on the 2-input complex MAC is not the ones that always dissipate the least amount of execution energy. However, the designs based on the MAC with 2 inputs occupy less amount of area than those based on the MAC with 4 and 8 inputs. The energy reduction for partially reconfiguring the device between the execution of the task ranges from 34% to 66%. For task  $T_2$ , the dynamic programming algorithm always maps it to the software implementation using PowerPC while the greedy algorithm always maps it to the configurable logic. While the parameterized FFT designs in [3] minimize the execution energy dissipation of  $T_4$  through the employment of parallelism, radix and choices of storage types, such energy minimization is achieved by using more FPGA resources. This increases the reconfiguration energy costs and is not viable in realizing energy efficient design.

## VI. CONCLUSION

We proposed a hybrid design approach for energy efficient application synthesis using platform FPGAs. The execution of an FFT software program and the synthesis of an adaptive beamforming application were used to illustrate the proposed design process and demonstrate its effectiveness.

Our work is still in a preliminary stage. A simulator that can rapidly and fairly accurately obtain energy performance depending on hardware configurations of the processor system is under development. Besides, we are extending the optimization algorithms to address applications with directed acyclic task graphs.

## REFERENCES

- [1] J. Becker, "Configurable Systems-on-Chip: Challenges and Perspectives for Industry and Universities," *ERSA*, 2002.
- [2] S. Choi, J.-W. Jang, S. Mohanty, V. K. Prasanna, "Domain-Specific Modeling for Rapid System-Wide Energy Estimation of Reconfigurable Architectures," *ERSA*, 2002.
- [3] S. Choi, G. Govindu, V. K. Prasanna, "Energy-Efficient and Parameterized Designs for Fast Fourier Transform on FPGAs," *Inter. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.
- [4] C. Dick, "The Platform FPGA: Enabling the Software Radio," *Software Defined Radio Technical Conf. (SDR)*, 2002.
- [5] Altera Inc., <http://www.altera.com/>.
- [6] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Third Edition, 1991.
- [7] D. Rakhmatov and S. Vrudhula, "Hardware-software Bipartitioning for Dynamically Reconfigurable Systems," International Conference on Hardware Software Codesign (CODES), 2002.
- [8] Mentor Graphics, Inc., <http://www.mentor.com>.
- [9] J. Ou and V. K. Prasanna, "Energy-Efficient Hardware/Software Co-Synthesis for a Class of Applications on Reconfigurable SoCs," (to appear) *International Journal of Embedded Systems*, June 2004.
- [10] J. Ou and V. K. Prasanna, "Parameterized and Energy Efficient Adaptive Beamforming Using System Generator," *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004.
- [11] OMAP processors, Texas Instrument, Inc., <http://www.ti.com>.
- [12] Triscend, Inc., <http://www.triscend.com>.
- [13] J. Villarreal, D. Suresh, G. Stitt, F. Vahid, and W. Najjar, "Improving Software Performance with Configurable Logic," *Kluwer Journal on Design Automation of Embedded Systems*, 2002.
- [14] Xilinx Corporation, Inc., <http://www.xilinx.com>.