Performance-Energy Tradeoffs for Matrix Multiplication on FPGA-Based Mixed-Mode Chip Multiprocessors

X. Wang and S.G. Ziavras
NJIT

Abstract-Chip multiprocessing has demonstrated to be a promising approach in microprocessor design. With ever increasing concerns for energy consumption, performance-energy trade-offs are often necessary, especially in the design of real-time embedded systems. This paper presents our performance and energy study on an in-house developed FPGA-based mixed-mode chip multiprocessor, where the SIMD (Single-Instruction, Multiple-Data), MIMD (Multiple-Instruction, Multiple-Data) and M-SIMD (Multiple-SIMD) computing modes can exist simultaneously in one system. We propose performance-energy trade-off techniques based on the observation that SIMD and MIMD task executions involve substantially different amounts of computation and communication, which result in different time and energy behavior and provide us with opportunities to realize various performance-energy objectives. Generalized matrix-matrix multiplication (MMM) is employed as an example to illustrate our analysis. Experimental results on a Xilinx Virtex II XC2V6000-5 FPGA demonstrate the effectiveness of the proposed approach.

I. INTRODUCTION

Chip multiprocessing has received significant attention in both the academe and industry in the last few years as a result of showing and looming difficulties of traditional approaches to further improve performance [1]. Various architectures have been studied and showed promising results [2-7]. There are two practical computing paradigms in parallel processing, namely SIMD and MIMD. SIMD's superior ability for data parallelism, often enhanced with low inter-PE communication and synchronization overheads, make it superior to MIMD in performing fine-grain tasks. However, SIMD's implicit intra-instruction synchronization makes it difficult to accommodate application dynamics. On the other hand, MIMD machines consisting of independent PEs are good at conditional branching. However, the PE independence property of MIMD makes programming cumbersome. For applications prone to SIMD execution, the need to explicitly synchronize the PEs in MIMD realizations produces substantial overheads. Furthermore, every PE in MIMD requires its own program memory. Mixed-mode heterogeneous computing [8], where the machine's operational mode (i.e., SIMD, MIMD or M-SIMD) changes dynamically as needed by individual subtasks in an application, integrates effectively most of the SIMD and MIMD advantages while alleviating their major drawbacks.

HERA (HEterogeneous Reconfigurable Architecture) [9] is such a mixed-mode chip multiprocessor that we have designed and implemented on Xilinx Virtex II FPGAs. Floating-point (FP) computation-intensive applications are HERA's main target domain. State-of-the-art FPGAs have showed impressive FP performance and provided new opportunities to parallel processing [10-11]. Moreover, FPGAs add to HERA another dimension of flexibility to customize and match the multiprocessor to the computation and communication flavors of a given application. Given the fact that function units for FP operations (addition, subtraction, multiplication, division, square-root, etc) are very resource expensive, application-specific customization brings significant performance benefits. For example, matrix multiplication only needs addition and multiplication. By removing dividers from processors, the number of processors nearly doubles based on our implementation results.

At the same time, energy has become an important design metric for all computing systems, especially when used in wireless and embedded environments, where real-time constraints are combined with requirements for long battery life [12-13]. Compared to their ASIC counterparts, SRAM FPGAs are more energy hungry. Hence, performance-energy trade-offs are often desirable, even necessary, for FPGA-based reconfigurable systems. However, in contrast to extensive low-power or energy-efficient research on chip multiprocessors, very little work has been done on performance-energy trade-offs. Li et al. [17] explores power-performance optimizations targeting chip multiprocessors with Alpha-like processor cores by manipulating multiple interacting factors, including application granularity, voltage/frequency levels and number of processors. Ref. [18] studies energy-performance trade-offs for a shared-memory chip multiprocessor with a shared interconnection bus. A design strategy called spatial voltage scaling is presented in [19] for heterogeneous chip multiprocessors. Ref. [20] presents a domain-specific optimization methodology by resource sharing and pipelining for a reconfigurable coarse-grained architecture supporting integer operations.

In this paper we propose performance-energy trade-off techniques that exploit the flexibility of mixed-mode systems like HERA. Several studies have shown that communication channels consume significant energy in chip multiprocessors, especially for FPGA-based systems [14-15]. For a given task, SIMD and MIMD require different types and amounts of communication among processors. Also, the optimal modes for minimum execution time and energy are different. Hence, by carefully manipulating the presence of SIMD and MIMD execution for given tasks, we can achieve different performance-energy objectives.

The multiplication of irregular shape and size matrices is used as an example to illustrate our techniques. But this approach can also be applied to other applications with minor modifications. The parallel multiplication of square matrices

has been studied extensively and numerous results targeting various computing platforms exist in the literature. With dramatic increases in the computing capability of current FPGAs, various highly specialized and customized floating-point implementations of MMM on FPGAs are emerging (e. g., [11, 16]). In comparison, HERA is a semi-customized matrix-oriented chip multiprocessor with general-purpose instructions, which is more friendly to software developers without hardware expertise. Moreover, our focus in this study is the efficient processing of matrices of general shape, not only square matrices.

The remainder of the paper is organized as follows. Section II shows our HERA design for the MMM algorithm. Section III describes the MMM algorithm and its mixed-mode mapping onto HERA. Performance and energy characterization and trade-off techniques for the mixed-mode MMM on HERA are presented in Section IV. Experimental results and analysis are shown in Section V. Section VI concludes the paper.

II. MIXED-MODE CHIP MULTIPROCESSOR

The general organization of our multiprocessor for the generalized MMM is shown in Fig. 1. The PEs (Processing Elements) are interconnected via a 2-D q x q torus. We employ fast, direct NEWS (North, East, West and South) connections between nearest neighbors. The computing fabric inside the FPGA is controlled by the *global control unit* (GCU) that communicates with a host processor via the PCI bus for data I/O. Every PE includes a small amount of control logic (CTRL) controlled by the GCU. The GCU fetches instructions from the *global program memory* (GPM) for PEs operating in SIMD. A PE is an in-house designed pipelined RISC processor that consists of an integer function unit (FU), one or more pipelined FP FUs (from an in-house designed hardware library), custom function blocks, dual-port *local data memory* (LDM), and dual-port *local program memory* (LPM). These are the basic PE features. In fact, all the PEs of HERA are semi-customized and generated during the architecture synthesis stage after the target application is given. In this work, only an adder and a multiplier (MAC) are included in each PE. A novel feature of HERA's memory hierarchy is that one port of every PE's LDM is shared with its neighbors to the south and east. This way, a PE can directly write to or read from the LDMs of its west and north neighbors via the shared port. This feature can be used to eliminate omnipresent large block data transfers between nearest neighbors in numerous block-based matrix computations. Although PEs may contain different FP FUs, all of them support the same ISA to facilitate ease of software development. HERA's complete instruction set contains about 30 instructions classified in six major groups: integer arithmetic, FP arithmetic, memory access, jump and branch, NEWS communications, and system control. The operating mode (SIMD or MIMD) of each PE can be configured dynamically by the GCU through its *Operation Mode Register* that can be modified by the *Configure* instruction.

In addition to the NEWS interconnect, HERA also has a hierarchical bus system. Every PE is connected to a column *Cbus* and all the *Cbuses* are connected to the *Column Bus* for broadcasting SIMD instructions and their immediate data. Each column has its own instruction register. SIMD instructions and data are transferred via the Dbus in a pipelined fashion. HERA can be partitioned at run time into several islands, each comprising a group of PEs running in SIMD or MIMD. Partitioning is achieved by global or local PE masking; the mask status is stored in the PE's *Global Mask Register* (GMR) and *Local Mask Register (LMR)*, respectively. A PE in SIMD is active only when both registers are set. LMR can be set by executing locally a comparison instruction. Every PE is assigned a distinct ID that serves in global masking. The last seven bits of an instruction in SIMD form three fields: 3 bits each for the row and column address, and 1 bit for masking. A "1" in this bit sets the GMR of all the PEs in the column and a "0" only sets the GMR of the specific PE whose address is contained in the instruction. Combined with the PE ID and appropriate masks, the system can be configured dynamically into a mixed-mode computing system capable of supporting simultaneously SIMD, MIMD, and M-SIMD.

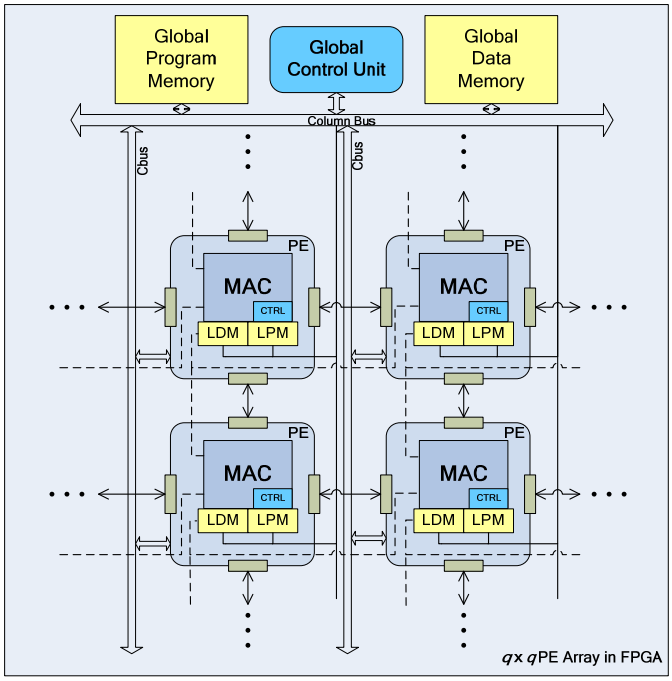

Fig. 1. HERA organization.

III. GENERALIZED MATRIX-MATRIX MULTIPLICATION

Consider A x B = C, where A, B, and C are matrices of size N_1 x N_2, N_2 x N_3, and N_1 x N_3, respectively. N_1, N_2, and N_3 are different for non-square matrices. If A and B are square, Cannon's algorithm [21] works best in the SIMD mode; all the PEs are then busy all the time except during the initial alignment. If A and B are not square or cannot be partitioned in such a way that N_i, i = 1, 2, 3, is a multiple

of q, then the multiplication of border blocks is not efficient in the SIMD mode because the sizes and numbers of blocks are irregular. Some PEs are idle while other PEs are busy at some point because SIMD is implicitly synchronous. We solved this problem by changing the computation mode of PEs. Suppose that the LDM of the PEs can store $3m^2$ floating-point numbers. To be able to store complete blocks from the input and output matrices, the maximum size of a matrix block should be m x m. Let $p1 = \lfloor N1/(q*m) \rfloor$, $p2 = \lfloor N2/(q*m) \rfloor$ and $p3 = \lfloor N3/(q*m) \rfloor$. We first partition A and B into 2 x 2 block-based matrices as shown in the example of Fig. 2, in such a way that the sizes of $A(1,1)$ and $B(1,1)$ are $\{p1*(q*m)\} \text{x} \{p2*(q*m)\}$ and $\{p2*(q*m)\} \text{x} \{p3*(q*m)\}$, respectively. The remaining blocks $A(2,1)$, $A(1,2)$ and $A(2,2)$ of A are decomposed into blocks with maximum dimension m. B is partitioned similarly. $A(1,1)$ and $B(1,1)$ are then partitioned into $p1$ x $p2$ and $p2$ x $p3$ blocks of size $(q*m) \text{x} (q*m)$ again and are distributed into the PEs in a cyclic checkerboard-like fashion.

After we decompose the matrices in this way, the multiplication of A and B involves 8 major tasks, represented by TK_i, i = 1, …, 8, for $A(1,1)$ x $B(1,1)$, $A(1,2)$ x $B(2,1)$, $A(1,1)$ x $B(1,2)$, $A(1,2)$ x $B(2,2)$, $A(2,1)$ x $B(1,1)$, $A(2,2)$ x $B(2,1)$, $A(2,1)$ x $B(1,2)$, and $A(2,2)$ x $B(2,2)$, respectively. Apparently, TK_1 generally involves the most significant work and takes the longest time among the tasks. In SIMD, all PEs execute the same instructions from the GCU on different matrix blocks. In MIMD, PEs work independently and asynchronously on their own instructions and data blocks. In SIMD, both LDM and LPM of a PE are used for data and, hence, reduce the total number of data I/O. However, SIMD execution requires the global broadcasting of instructions and may threaten the energy budget. As discussed in the Introduction, global communication is a major contributing factor to the overall energy consumption. Also, SIMD cause more idle PEs on irregular matrix blocks. MIMD execution may reduce the energy consumption by working on local data. But only the LDM can be used to store matrix blocks in MIMD, which results in an increased number of data transfer. Also, MMM requires significant memory bandwidth for data exchanges between the local memory of the PEs and the on-board memory chips.

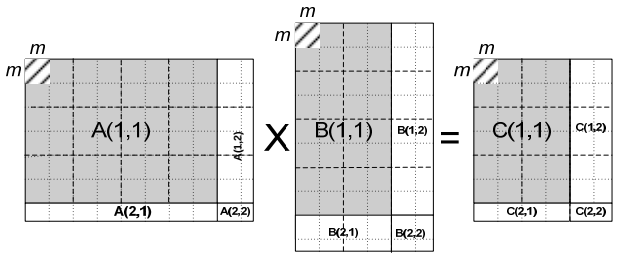

Fig. 2. A partitioning example for matrices A and B. $(q = 2, p1 = 3, p2 = 4, p3 = 2)$

IV. PERFORMANCE-ENERGY TRADE-OFFS

In this section we analyze the performance and energy consumption of the tasks in the SIMD and MIMD modes. This analysis provides a quantitative basis for our performance-energy trade-off techniques in the mixed mode. In contrast to most of the other performance and energy modeling approaches, our equations are based on implementation measurements on the FPGA.

A. Performance and Energy Characterization

In both the SIMD and MIMD modes, the multiplication of a pair of matrix blocks of size n_1 x n_2 and n_2 x n_3 is an indivisible job, which is mapped into one PE only. We denote this type of job as $J_m(n_1, n_2, n_3)$. Each task TK_i involves many such jobs. The clock cycles to finish a $J_m(n_1, n_2, n_3)$ job on HERA is:

$$t_m(n_1, n_2, n_3) = 15*n_1*n_2*n_3 + 6*n_1*n_2 + 4*n_1 + 15 \quad (1)$$

There are several types of communication channels in HERA. Let $J_c(n_1, n_2)$ be the job to transfer a matrix block of size n_1 x n_2 between the GDM and an LDM. Since the GCU can directly access every LDM and LPM, an $J_c(n_1, n_2)$ job takes $t_c(n_1, n_2) = (n_1 * n_2)$ clock cycles. Another type of major communication jobs is the broadcasting of HERA instructions. The MMM code has around 40 assembly language instructions. A few control instructions are added according to different computing modes. These instructions are broadcast via the column buses. The NEWS interconnect is used to transfer matrix blocks or register values between PEs. The clock cycles for the instruction broadcasting and NEWS transfer, denoted by t_b and t_{news}, respectively, are proportional to the data volume.

The energy consumption of the basic jobs is calculated as follows. We distinguish between two power states for the components in HERA: *active* and *idle*. Only the dynamic power is considered since the static power is much less significant in our target device, i.e., the Virtex II FPGAs [22]. The power data of HERA components in the two states are obtained after implementation on a specific FPGA device. The required power for a PE, a NEWS connection, the bus system, or an LDM in the *active* and *idle* states is represented by $P_x^{active}(f)$ and $P_x^{idle}(f)$, respectively, where f is the implemented system frequency of HERA and x represents the respective component. Hence, the energy consumptions of $J_m(n_1, n_2, n_3)$ and $J_c(n_1, n_2)$, respectively, are:

$$e_m(n_1, n_2, n_3) = t_m(n_1, n_2, n_3)/f * P_{PE}^{active}(f) \quad (2)$$

$$e_c(n_1, n_2) = t_c(n_1, n_2)/f * P_{LDM}^{active}(f) \quad (3)$$

Other types of communication include NEWS transfer and instruction broadcasting, with consumptions represented by $e_{NEWS}(n_1, n_2)$ and $e_I(n_1)$, respectively, and calculated by similar equations to (2) and (3).

Based on the above analysis, let us look at the execution time and energy consumption of the MMM tasks. If TK_1 is to be scheduled in SIMD, we have two choices: Cannon's algorithm and a naive algorithm. The naïve algorithm treats all jobs like $J_m(m, m, m)$ and each round can finish $2*3*q^2$

$J_m(m, m, m)$ jobs since each PE has direct access to three LDMs for itself and the two neighbors. Results are then collected and new matrix blocks are loaded into the LDMs and LPMs followed by a new round of multiplications. Each round requires (3+4)*q^2 $J_c(m, m)$ jobs. The pattern continues until all the jobs are done. Totally we have $p1*p2*p3*q^3$ $J_m(m, m, m)$ jobs. We do not have J_{news} jobs in this case. The total number of clock cycles for computation and communication, and the energy consumption for this naïve scheme are

$$t^n_{1,comp} = t_m(m,m,m)*p1*p2*p3*q \quad (4)$$

$$t^n_{1,comm} = t_c(m,m)*p1*p2*p3*q^3*\tfrac{1}{6} \quad (5)$$

$$e^n_1 = \frac{1}{f}*\{(P^{active}_{PE} + P^{active}_{bus} + P^{active}_{LDM+LPM})*t^n_{1,comp} + (P^{idle}_{PE} + P^{idle}_{bus} + P^{idle}_{NEWS})*t^n_{1,comm} + P^{active}_{LDM+LPM}*t^n_{1,comm} + P^{idle}_{NEWS}*t^n_{1,comp}\} \quad (6)$$

If Cannon's algorithm is used for TK_1, then the needed clock cycles and energy consumption are:

$$t^c_{1,comp} = t_m(m,m,m)*p1*p2*p3*q \quad (7)$$

$$t^c_{1,NEWS} = 2*t_{news}(2m,2m)*p1*p2*p3*q^3/8 \quad (8)$$

$$t^c_{1,c} = 2*t_c(2m,2m)*p1*p2*p3 \quad (9)$$

$$t^c_{1,comm} = t^c_{1,NEWS} + t^c_{1,c} \quad (10)$$

$$e^c_1 = \frac{1}{f}*\{(P^{active}_{PE} + P^{active}_{bus} + P^{active}_{LDM+LPM})*t^c_{1,comp} + (P^{idle}_{PE} + P^{idle}_{bus} + P^{idle}_{NEWS})*t^c_{1,c} + (P^{active}_{NEWS} + P^{active}_{LDM+LPM})*t^c_{1,NEWS} + P^{idle}_{NEWS}*(t^c_{1,comp} + t^c_{1,c})\} \quad (11)$$

If MIMD is the computing mode, then we need to update the LDM only ($J_c(m, m)$ jobs) in each round but finish only q^2 $J_m(m, m, m)$ jobs. Also, we have a larger number of data transfers. The data transfers can overlap with the multiplication jobs since PEs work independently in MIMD. The worst-case clock cycles and energy consumption can be found by:

$$t^{MIMD}_{1,comp} = t_m(m,m,m)*p1*p2*p3*q \quad (12)$$

$$t^{MIMD}_{1,comm} = t_c(m,m)*p1*p2*p3*q^3*2/3 \quad (13)$$

$$e^{MIMD}_1 = \frac{1}{f}*\{(P^{active}_{PE} + P^{active}_{LDM+LPM})*t^n_{1,comp} + (P^{idle}_{PE} + P^{idle}_{NEWS} + P^{idle}_{bus})*t^n_{1,comm} + P^{active}_{LDM+LPM}*t^n_{1,comm} + (P^{idle}_{bus} + P^{idle}_{NEWS})*t^n_{1,comp}\} \quad (14)$$

For the sake of simplicity, we do not take data locality into account in these equations. It will be considered during task scheduling. Also, the accumulation time of the partial products is not included.

Similarly, other tasks (TK_i i = 2, …, 8) can be treated in SIMD, M-SIMD, MIMD, or the mixed mode. In the mixed-mode, PEs are divided into multiple SIMD and MIMD groups. Synchronization may be needed at some point. We can derive similar equations for clock cycles and energy consumption. One particular thing with these tasks is that they involve non-square matrix blocks. The irregularity will cause more idle PEs in SIMD than MIMD. Hence, it is beneficial to execute these tasks in the mixed mode. For example, consider a task involving one J_m(2, 5, 17), six J_m(10, 15, 11), and eight J_m(14, 25, 7) jobs. We can construct one SIMD group consisting of six PEs working on the six J_m(10, 15, 11) jobs and another SIMD group with eight PEs for the eight J_m(14, 25, 7) jobs. An independent PE will work on the J_m(2, 5, 17) job. This way we can avoid the idleness of PEs and potentially save on energy and time.

Let γ_i be the percentage of computation in TK_i, i = 1, …, 8, working in SIMD and M-SIMD. The remaining work in TK_i is assigned to PEs in MIMD. The clock cycles and system energy consumption for all the tasks can be found by:

$$T_\Sigma = \sum_8 \{T^{SIMD}_i * \gamma_i + T^{MIMD}_i (1-\gamma_i)\} \quad (15)$$

$$E_{\Sigma,sys} = \sum_x \{C^{active}_x * P^{active}_x(f) + C^{idle}_x * P^{idle}_x(f)\} * \frac{1}{f} \quad (16)$$

where C^{active}_x and C^{idle}_x are the clock cycles of the system components, i.e., PEs, NEWS, bus, or memory, in the *active* and *idle* states, respectively, for all the tasks. They are collected by hardware counters in the respective components at runtime.

B. Performance-Energy Tradeoffs

From the above analysis, we can see that the SIMD and MIMD executions of a task involve different amounts of execution time and energy consumption. By varying the appearance weight of different modes, we can achieve different performance-energy objectives. In particular, we explore three performance-energy scenarios:

(1) *Optimize the performance with no energy constraints*

The focus is to reduce the communication time and also consider data locality when distributing matrix blocks to available PEs. This case also helps us to learn the best performance and the corresponding energy consumption of the application on the specific architecture. The objective is to find a set of γ_i which results in minimum T_Σ (Eq. 15). The possible choices for the γ_i's are determined by partitioning. After a matrix and a HERA configuration are given, the total number of matrix blocks is fixed. This resulting problem can be explored by a linear programming solver. The energy cost of each task E_{Σ,TK_i} can be found by summarizing all the energy cost of its jobs using the equations in the last subsection.

(2) *Optimize the performance with energy constraints*

Let E_B be the upper bound on the energy. We first analyze the energy consumption for TK_i, i = 1, …, 8, in the first case, and then estimate the difference between the actual consumption and its upper bound. We take advantage of the fact that different γ_i values for the same task has different impact on the execution time and energy consumption. For an application-system pair, there is an optimal γ_i for minimum execution time. Since the PEs consume different power in different states, this optimal γ_i does not necessarily correspond to minimum energy consumption. Optimality involving both energy and performance depends on the task

characteristics as well as the architecture. We aim to optimize across two dimensions for each task: energy and/or performance vs. γ_i. Moreover, a hardware technique, clock gating, is employed to save energy at runtime. The clock signal of idle PEs will be disabled until they are assigned new jobs. Algorithm-1 is applied.

(3) *Reduce the energy cost for a given performance loss*

When energy consumption is also of paramount importance, performance can be sacrificed to an allowable extent in order to reduce the required energy. Let β be the allowed loss percentage. We decrease the performance of each task TK_i by the ratio β. We find a set of γ_i, i = 1, …, 8, to meet the time budget of TK_i

$$t_i^b = (1+\beta) * t_\Sigma^i \qquad (17)$$

An algorithm similar to Algorithm-1 but for the time budget instead of the energy budget is applied.

Algorthm-1: Meeting the energy budget

1. Find the energy gap: $E_g = E_B - \sum_{i=1}^{8} E_{\Sigma,TK_i}$;
2. IF E_g < 0, Stop;
3. Find the total energy cost of TK_i i = 2, …, 8, $\sum_{i=2}^{8} E_{\Sigma,TK_i}$;
4. Calculate the energy budget for $E_B^{2-8} = E_B - \sum_{i=2}^{8} E_{\Sigma,TK_i}$;
5. Assign an energy budget, $e_B^i = E_B^{2-8} * \frac{O_i}{\sum_i O_i}$, to each TK_i i = 2, …, 8, according to its computation weight;
6. Find the optimal γ_i, i = 2, …, 8, with the minimum t_Σ^i and $e_\Sigma^i < e_B^i$, where t_Σ^i and e_Σ^i are the total execution time and energy consumption of TK_i.
7. If the above procedure fails, include TK_1 and repeat Steps 2-6.

V. EXPERIMENTAL RESULTS

The FPGA device used in our experiments is the Xilinx VIrtex II XC2V6000-5 FPGA [22], which contains 33,792 slices and 144 x 512 x 36-bit BlockRAM blocks. The performance of the single-precision floating-point adder and multiplier used to construct the HERA PEs is shown in Table I. The HERA system runs at 125MHz. 36 PEs with 512 x 36-bit LDM and LPM were implemented for the experiments.

We first evaluated the accuracy of our performance and energy equations in the SIMD and MIMD modes shown in Section IV. A variety of non-square matrices of different shapes were used. Cannon's algorithm was applied to TK_1 in all the matrix pairs for the SIMD execution. The measured execution time and energy consumption of the tasks are listed in Table II. These results were compared with those calculated with our time and energy equations. The energy results were measured with the Xilinx XPower tool. The average activity rates were extracted from ModelSim files. The average difference between the actual and the measured time and energy is 2.1% and 4.5%, respectively. The difference in time mainly comes from the overheads of system administration and bus conflicts. Data locality during scheduling also adds to dynamic effects on performance and energy. This energy error rate is acceptable for system-level estimation models. HERA components consume a continuous range of power with different activity rates while we assume only one state to represent any active behavior. The key is to obtain the accurate activity rate by extensive simulations with benchmark matrices. Another reason is that the energy measurements for the bus system tend to be less accurate than for PEs and memory blocks. However, our objective is to develop fast, yet useful models for exploring performance-energy optimizations without involving tedious and time-consuming low-level simulations. Table II also shows that different execution modes require different execution times and energy consumptions, which provides room for performance-energy trade-offs. The exploration space increases with the increases in the matrix size.

Table I
Implementation results of the floating-point function units

Function Unit	Number of Pipelines	Area (slices)	Freq. (MHz)	Power (mW) at 125MHz	
				Active	Idle
Adder	3	390	163	227.2	87.4
Multiplier	3	134	174	85.9	59.2

Finally we evaluated our optimization techniques. Table III shows results for matrices of size 565 x 767 and 767 x 999. Scenario-II evaluates the impact of clock gating on the energy consumption. A reduction of 7.3% in energy consumption was observed by putting the idle PEs into sleep without major switching penalty on the execution time. A performance penalty of 5.7% was observed when reducing the energy consumption by 13%, as shown in Scenario-III. In Scenario-IV and -V, we relaxed the performance by 10.6% and 15% to reduce the energy consumption by 14.5% and 18.9%, respectively. The benefits of the approach should be better with more closely coupled algorithms that have more data dependences among tasks, which expose more flexibility for performance-energy trade-offs.

VI. CONCLUSIONS

Continuous advances in silicon technology and increasing difficulties in realizing superscalar processors have brought a significant shift in microprocessor design. Chip multiprocessing has recently emerged in general-purpose computing and will continue to develop further in many application scenarios, including embedded and wireless systems. While performance is always desirable, trade-offs between performance and energy are necessary in many such systems. We have presented our performance-energy trade-off study for an in-house designed and implemented mixed-mode reconfigurable chip multiprocessor. The flexibility of mixed-mode execution provides us with a tremendous exploration space to achieve

various performance-energy objectives. The experimental results prove the effectiveness of our approach.

REFERENCES

[1] R. Ronen, A. Mendelson, K. Lai, S-L. Lu, F. Pollack, and J. Shen, "Coming challenges in microarchitecture and architecture," *Proc. IEEE*, vol. 89, no. 3, March 2001.

[2] R. Krashinsky, C. Batten, M. Hampton, S. Gerding, B. Pharris, J. Casper, and K. Asanovic, "The vector-thread architecture," *IEEE Intern. Symp. Computer Archi.,* Munich, Germany, June 2004.

[3] L. Hammond, B. Hubbert, M. Siu, M. Prabhu, M. Chen, and K. Olukotun, "The Stanford Hydra CMP," *IEEE MICRO*, March-April 2000.

[4] L. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. V. Piranha, "A scalable architecture based on single-chip multiprocessing," *IEEE Int.. Symp. Comp. Arch.,* June 2000.

[5] D. Burger, S. W. Keckler, K. S. McKinley, et al., "Scaling to the End of Silicon with EDGE Architectures," *IEEE Computer*, vol. 37, no. 7, pp. 44-55, July 2004.

[6] H. P. Hofstee, "Power efficient processor architecture and the cell processor," *11th Intern. Symp. High-Perfor. Computer Archi.*, pp. 258-262, Feb. 2005.

[7] A. Jerraya and W. Wolf (eds.), *Multiprocessor Systems-on-Chips.* Morgan Kaufman, 2004.

[8] H. J. Siegel, M. Maheswaran, D. W. Watson, J. K. Antonio, and M. J. Atallah, "Mixed-mode system heterogeneous computing," in *Heterogeneous Computing*, Eshaghian, M. M. (Ed.), Artech House, Norwood, MA, pp. 19-65, 1996.

[9] _____, "Exploiting mixed-mode parallelism for matrix operations on the HERA architecture through reconfiguration," *IEE Proceedings, Computers and Digital Techniques,* in press.

[10] K. Underwood, "FPGAs vs. CPUs: trends in peak floating-point performance," *12th ACM/SIGDA Intern. Symp. Field-Program. Gate Arrays*, pp.171-180, 2004.

[11] L. Zhuo and V. K. Prasanna, "High performance linear algebra operations on reconfigurable systems," *ACM/IEEE Conf. Supercomputing*, Washington, 2005.

[12] C. Im and S. Ha, "An energy optimization technique for latency and quality constrained video applications," *IEEE Design & Test of Computers*, vol. 21 pp. 358-366, Sept. 2004.

[13] P. H. Chou and C. Park, "Energy efficient platform designs for real-world wireless sensing applications," *IEEE International Conference on Computer Aided Design (ICCAD 2005),* Nov. 2005.

[14] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex™-II FPGA family," *10th ACM/SIGDA Intern. Symp. Field-program. Gate Arrays*, pp. 157-164, 2002.

[15] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, "Power modeling and characteristics of Field Programmable Gate Arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 11, pp. 1712-1724, Nov. 2005.

[16] M. deLorimier and A. DeHon, "Floating-point sparse matrix-vector multiply for FPGAs," *13th ACM/SIGDA Intern. Symp. Field-Program. Gate Arrays*, pp. 75-85, 2005.

[17] J. Li and J. F. Martínez, "Power-performance considerations of parallel computing on chip multiprocessors," *ACM Trans. on Architecture and Code Optimization*, vol. 2 no.4, pp. 397-422, Dec. 2005.

[18] I. Kadayif, M. Kandemiret, and U. Sezer, "An integer linear programming based approach for parallelizing applications in on-chip multiprocessors," *IEEE Design Automation Conference,* New Orleans, LA, June 2002.

[19] B. H. Meyer, J. J. Pieper, J. M. Paul, J. E. Nelson, S. M. Pieper, and A. G. Rowe, "Power-performance simulation and design strategies for single-chip heterogeneous multiprocessors," *IEEE Transactions on Computers*, June 2005.

[20] Y. Kim, M. Kiemb, C. Park, J. Jung, and K. Choi, "Resource sharing and pipelining in coarse-grained reconfigurable architecture for domain-specific optimization," *IEEE Design, Automation and Test in Europe (DATE 2005)*, vol. 1, pp. 12-17, 2005.

[21] L. E Cannon, "A cellular computer to implement the kalman filter algorithm," Ph.D. Thesis, Montana State University, 1969.

[22] Virtex II FPGA datasheet, http://direct.xilinx.com/bvdocs/publications/ds031.pdf.

Table II
Execution time and energy consumption for various parallel execution modes

Matrix Dimensions			SIMD		MIMD	
N1	N2	N3	Time (sec)	Energy (J)	Time (sec)	Energy (J)
150	150	172	0.028	0.43	0.033	0.45
245	261	375	0.11	1.61	0.13	1.55
312	595	303	0.25	3.55	0.30	3.27
205	611	613	0.41	5.94	0.48	5.69
508	311	528	0.43	6.19	0.49	6.17
687	202	676	0.55	7.20	0.63	6.64
711	713	403	1.51	21.30	1.68	18.64
999	997	996	6.40	96.76	7.16	82.43

Table III
Performance-energy trade-offs in mixed-mode computing

Scenario	Objective	Constraints	Energy (J)	Execution Time (sec)
I	Minimize T	None	32.8	2.45
II	Minimize E	T < 2.45	30.4	2.44

III	Minimize T	E < 28.6	28.55	2.59
IV	Minimize E	T < 2.70	28.03	2.70
V	Minimize E	T < 2.82	26.6	2.80