Design Space Exploration for Sparse Matrix-Matrix Multiplication on FPGAs

Colin Yu Lin, Zheng Zhang, Ngai Wong and Hayden Kwok-Hay So
Department of Electrical and Electronic Engineering
University of Hong Kong, Hong Kong
Email: {linyu,zzhang,nwong,hso}@eee.hku.hk

***Abstract*—The design and implementation of a sparse matrix-matrix multiplication architecture on FPGAs is presented. Performance of the design, in terms of computational latency, as well as the associated power-delay and energy-delay tradeoff are studied. Taking advantage of the sparsity of the input matrices, the proposed design allows user-tunable power-delay and energy-delay tradeoffs by employing different number of processing elements (PEs) in the architecture design and different block size in the blocking decomposition. Such ability allows designers to employ different on-chip computational architecture for different system power-delay and energy-delay requirements. It is in contrast to conventional dense matrix-matrix multiplication architectures that always favor the maximum number of PEs and largest block size. In our implementation, the better energy consumption and power-delay product favors less PEs and smaller block size for the 90%-sparsity matrix-matrix multiplications. While in order to achieve better energy-delay product, more PEs and larger block size are preferred.**

I. Introduction

Sparse matrix systems arise in many computationally demanding engineering disciplines. Many sparse matrix computation kernels and algorithms have been accelerated using field-programmable gate arrays (FPGAs) for their higher power and energy efficiencies. In this paper, we study the design issues of sparse matrix-matrix multiplication (SpMxM) on FPGAs. Targeted as a power- and energy-efficient accelerator, we study not only the performance, but also the power and energy tradeoffs of the proposed design. A systolic architecture with variable number of processing elements (PEs) is proposed. By varying the number of PEs in the array, different power-delay and energy-delay tradeoffs can be achieved. A blocking decomposition method is introduced. The block size used in the blocking process also provides different power-delay and energy-delay tradeoffs. Implementation results are used to study the design space exploration (DSE) with different numbers of PEs and different block sizes for SpMxM on a particular FPGA.

To our knowledge, this paper is the first to address sparse matrix-matrix multiplication (SpMxM) on FPGAs. The design for SpMxM proposed in this work is based on the design in both [1] and [2]. Compared to these dense matrix-matrix multiplication implementations and designs, the PE architecture design and I/O data streaming for SpMxM are not so regular and require extra control logic to guarantee best performance.

In Section II, the design for SpMxM is briefly presented. A more detailed explanation is in [3]. In Section III and Section IV, the number of PEs and the block size are presented as factors of system power-delay and energy-delay tradeoffs. Finally, we make the conclusion in Section V.

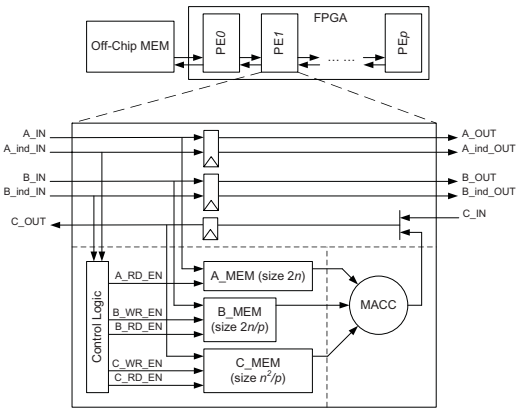

Fig. 1. Architecture Design

II. Proposed Design

A. Architecture Design

As in Figure 1, numbers of processing elements (PEs) are implemented using FPGA on-chip dedicated DSP blocks or reconfigurable logic. These PEs are connected into a linear systolic architecture. Only the first PE can access off-chip memory and read/store data. Input data from off-chip memory is passed from the first to the last PE through the linear systolic. Output data is passed in an opposite direction, and is stored to off-chip memory by the first PE.

B. Computation and Data Partitioning

For a n-by-n sparse matrix-matrix multiplication, the computation is partitioned evenly to all PEs. As the number of PEs p is often much smaller than the size of input matrices n, each PE is assigned to do the computation for one or more than one column data elements of matrix C.

C. Data I/O

We use the example of a 4-by-4 multiplication with 2 PEs to explain the data input and output. As seen in Figure 2, the whole computation is divided into $n = 4$ phases. In the i^{th} phase, $1 \leq i \leq n$, the i^{th} column from matrix A and the

978-1-4244-8983-1/10/$26.00 ©2010 IEEE

TABLE I
DESIGN UTILIZATIONS

Parameter	Each PE	Overall Design
Problem size	-	n
Data bit-width	-	w
Number of PEs	-	p, n is divisible by p
Number of I/O ports	-	5
Number of 1-bit registers	$3w + 4\log_2 n$	$(3w + 4\log_2 n)\, p$
Memory size for matrix A	$2n$	$2np$
Memory size for matrix B	$2n/p$	$2n$
Memory size for matrix C	n^2/p	n^2
Total memory size	$2n + \frac{n^2+2n}{p}$	$2np + n^2 + 2n$
Number of MACCs	1	p

TABLE II
DESIGN UTILIZATION COMPARISON

Parameter	Design in [1]	Proposed Design
Problem size	n	n
Data bit-width	w	w
Number of PEs	p	p
Number of I/O ports	3	5
Number of 1-bit registers	$3wp$	$(3w + 4\log_2 n)\, p$
Memory size for matrix A	0	$2np$
Memory size for matrix B	n	$2n$
Memory size for matrix C	n^2	n^2
Total memory size	$n^2 + n$	$2np + n^2 + 2n$
Number of MACCs	p	p

i^{th} row from matrix B are read from off-chip memory. Each PE stores all data elements from matrix A and required data elements from matrix B. Then it starts to do the computation using these data elements.

If the input matrices are dense, the first phase time is from 0 to $n + n^2/p$. n is the data input time, and n^2/p is the computation time.

Before the 1^{st} phase ends, data input for computation in 2^{nd} phase starts. As a result, the data input time n is overlapped with the 1^{st} phase computation time. But that requires larger on-chip memory for data elements from both matrix A and B. The memory size in our design for data elements from matrix A and B are $2n$ and $2n/p$ respectively.

D. Design Summary

Table I summarizes the resource utilizations for the proposed overall design and for each PE. The taget computation is a n-by-n sparse square matrix-matrix multiplication. All input and output data elements of matrix A, B and C have the same bit-width, w. p PEs are implemented using FPGA reconfigurable DSP blocks and/or logic resources.

Five FPGA I/O ports are used to communicate with off-chip memory. Three ports with bit-width w are used to read data elements from matrix A and B and store data elements of matrix C. The other two ports are used to input index information for data elements of matrix A and B. Each of the two ports is $2log_2n$ bits, half of which are row index information and the other half are column index information. The total number of the I/O bits is $3w + 4log_2n$.

For each PE in the proposed design, three w-bit registers are used to pass data elements of matrix A, B and C respectively. Another two registers are used to pass index information for data elements of matrix A and B. The memory block sizes are as mentioned and explained in Subsection II-C. Finally, one MACC is implemented in each PE.

E. Design Comparison

For a n-by-n w-bit matrix-matrix multiplication, Table II gives a comparison on design utilizations between proposed design and the design in [1]. The design in [1] is for dense matrix-matrix multiplication, and the proposed design is for sparse matrix-matrix multiplication. For both designs, the number of PEs is p. The number of I/O ports in [1] is 3, and 2 more I/O ports are required for input matrix index information in the proposed design.

In both the proposed design and the design from [1], there are two limiting factors from the available FPGA on-chip resources. The first limiting factor is the number of MACCs implemented, p. As the MACCs are implemented using dedicated DSP blocks or reconfigurable logic, so p, the number of MACCs, which is also the number of PEs, is limited by the number of DSP blocks or on-chip available logic resources.

The second limiting factor is the FPGA on-chip memory. Since the problem size n is large, and it is larger than the data bit-width and the number of PEs, which is $n < w$ and $n < p$, so the problem size in both designs is limited by FPGA on-chip memory. We use M to represent the total size of FPGA on-chip memory, then the problem size in both designs is limited by $n < \sqrt{M}$.

As shown in Table II, the proposed design uses more registers for index information, and requires larger memory to store data elements of input matrices. But with the same reason mentioned above, which is that n is large and $n < p$, the extra register and memory requirement will not have a large effect on n. The problem size in the proposed design is also limited by $n < \sqrt{M}$.

III. NUMBER OF PES

In this section, implementation results are shown and discussed for the proposed design presented in Section II.

The sparsity of a matrix is used to represent the zero-element percentage in the whole matrix. A zero-sparsity matrix is a dense matrix. All data elements in input matrices are 16-bit and fixed point, and they are randomly generated using the Verilog random number generation function `$random`. The non-zero elements of the input matrices are input to FPGA with their row and column index information. This input format can be easily translated from different sparse matrix formats, such as Dictionary of Keys and Compressed Sparse Row/Column.

The FPGA on-chip architecture is implemented using the Xilinx FPGA XC5VLX110T. Different numbers of processing elements (PEs) are implemented using the FPGA on-chip dedicated DSP blocks. The PE individual storage memory blocks for matrix A and B are implemented using Xilinx

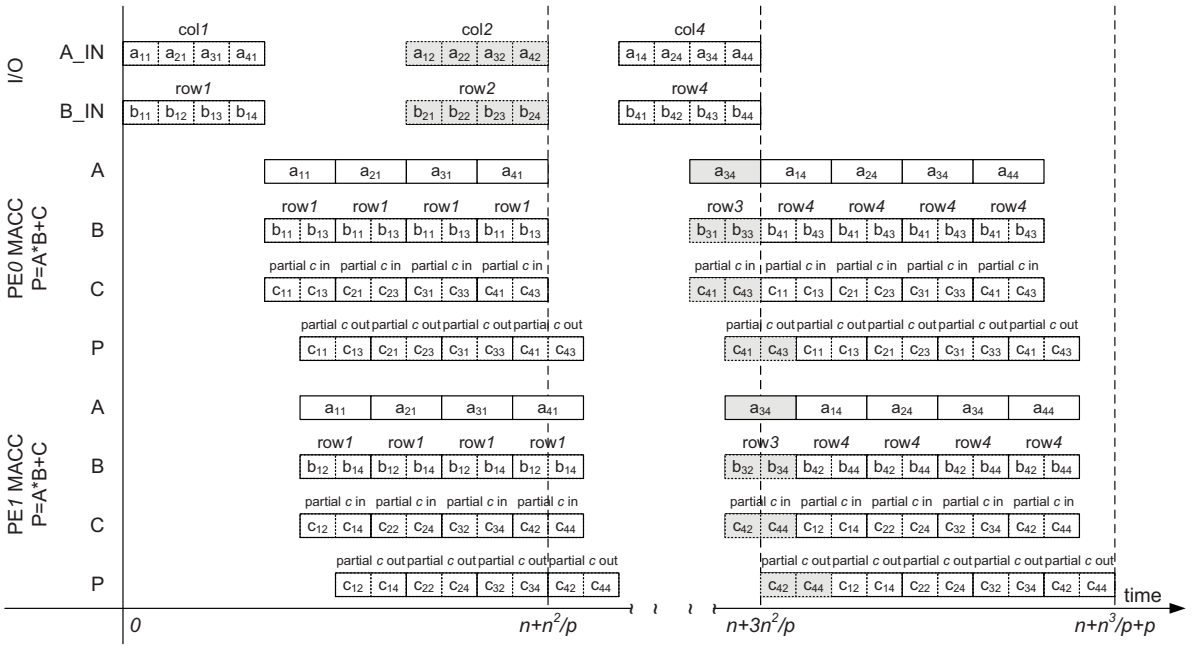

Fig. 2. Data Input

distributed memory IP cores. The sizes of these two memory blocks are $2n$ and $2n/p$. As discussed in Subsection II-C, using larger memory to store more data elements from both matrix A and B gives better delay performance. And the memory for output data and partial output data of matrix C is implemented using Xilinx Block RAM IP cores.

In this section, the effects of the number of PEs upon power-delay product and energy-delay product of both dense and sparse matrix-matrix multiplications are studied. The 256-by-256 matrix-matrix multiplications are used as the target computation. Four sparsity percentages of the input matrices, 0%, 70%, 80% and 90% are considered. The 0%-sparsity is used to represent the dense matrix-matrix multiplication. The other three are used to represent the sparse matrix-matrix multiplication, since high sparsity matrices are more often seen in real applications.

A. Power-Delay Product

In the proposed design, more PEs can be used to shorten the computation delay. But more on-chip resources will be used, as a result, the power consumption increases when more PEs are used. The power-delay product can be used to indicate the power-delay tradeoff. The delay per operation is used to give a comparison between different sparsity percentages. And the results of power-delay product for different number of PEs are shown in Figure 3.

For dense matrix-matrix multiplications, using more PEs can give a better power-delay product. The implementations consume less energy when using more PEs. But for sparse matrix-matrix multiplications, using more PEs is not always reducing energy consumption. As seen in Figure 3, for a 256-by-256 sparse matrix-matrix multiplication, if 70% of the data

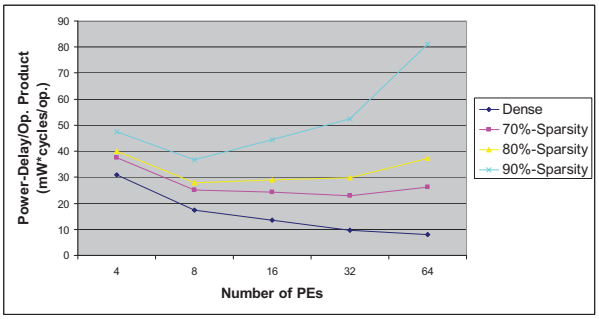

Fig. 3. Power-Delay Product vs. Number of PEs

elements in input matrices A and B are zero, the proposed architecture design with 32 PEs gives the best power-delay product and consumes least energy. If the sparsity of input matrices is 80-90%, using only 8 PEs gives the best power-delay product and consumes least energy.

B. Energy-Delay Product

The energy-delay product is another important parameter for system design. It indicates the tradeoff between system performance and energy consumption.

The energy-delay products of implementations with different number of PEs are shown in Figure 4. For the dense 256-by-256 matrix-matrix multiplications, the architecture design with 64 PEs has the best energy-delay product. The same architecture design with 64 PEs also gives the best energy-delay product for sparse matrix-matrix multiplications with a sparsity no more than 70%. When the sparsity of input matrices is 80%, using 32 PEs can provide better energy-delay

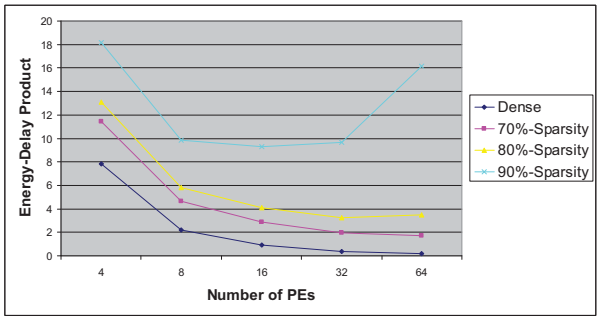

Fig. 4. Energy-Delay Product vs. Number of PEs

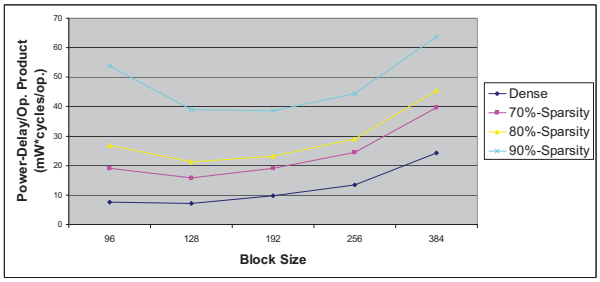

Fig. 5. Power-Delay Product vs. Block Size

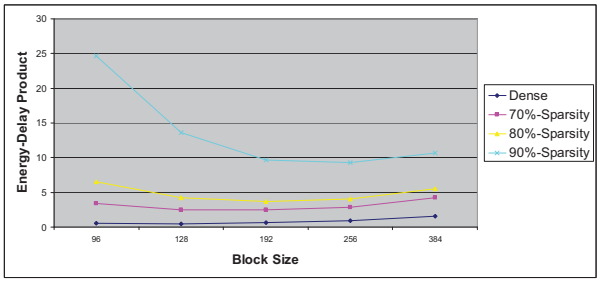

Fig. 6. Energy-Delay Product vs. Block Size

product than 64 PEs. And only 16 PEs are necessary to get the best energy-delay product with the sparsity of 90%.

IV. Blocking Decomposition

In Section II and Section III, the proposed design and implementation results are presented without considering blocking decomposition. In these sections, we assume that the FPGA on-chip resources are able to support computing the matrix-matrix multiplication without blocking. As discussed in Subsection II-E, the problem size for both the proposed design and the design in [1] is limited by FPGA on-chip memory. $n < \sqrt{M}$, where n is the problem size and M is the total size of FPGA on-chip memory. The largest n which can be computed by both designs without blocking is given by $n < \sqrt{M}$.

Since the problem size of dense and sparse matrix-matrix multiplication is often very large, a n-by-n multiplication is blocked into $\left(\frac{n}{b}\right)^3$ b-by-b multiplications, where b is the block size and $b < \sqrt{M}$. In this section, we discuss the effects of the block size b upon performance, power and energy consumptions of the designs for both sparse and dense matrix-matrix multiplications.

A. Power-Delay Product

In the proposed design, larger block size can shorten the computation delay. But more on-chip resources will be used, as a result, the power consumption increases when the block size increases. The power-delay product can be used to indicate the power-delay tradeoff. The delay per operation is used to give a comparison between different sparsity percentages. And the results of power-delay product for different block sizes are shown in Figure 5.

As seen in Figure 5, for dense, 70%- and 80%-sparse matrix-matrix multiplications, block size 128 gives the best power-delay product and consumes least energy. If the sparsity of input matrices is 90%, using block size 192 gives the best power-delay product and consumes least energy.

B. Energy-Delay Product

The energy-delay products of implementations with different block sizes are shown in Figure 6. For the dense matrix-matrix multiplications, the architecture design with block size 128 has the best energy-delay product. The same architecture design with block size 128 also gives the best energy-delay product for sparse matrix-matrix multiplications with a sparsity no more than 70%. When the sparsity of input matrices is 80%, using block size 192 can provide slightly better energy-delay product than block size 128. And larger block size 256 is preferred to get the best energy-delay product with the sparsity of 90%.

V. Conclusions

A design methodology for sparse matrix-matrix multiplication on FPGAs is presented. In the proposed design, the number of PEs and the block size are two important parameters in architecture design and blocking decomposition process respectively. Using different design parameters gives different tradeoffs of power-delay and energy-delay. In our implementation for the 90%-sparsity matrix-matrix multiplications, the better energy consumption and power-delay product favors less PEs and smaller block size. While in order to achieve better energy-delay product, more PEs and larger block size are preferred.

References

[1] J. Jang, S. Choi, and V. Prasanna, "Energy- and time-efficient matrix multiplication on FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 11, pp. 1305–1319, 2005.

[2] L. Zhuo and V. Prasanna, "Scalable and modular algorithms for floating-point matrix multiplication on reconfigurable computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 433–448, 2007.

[3] C. Y. Lin, Z. Zhang, N. Wong, and H. K.-H. So, "Power-Delay and Energy-Delay Tradeoffs for Sparse Matrix-Matrix Multiplication on FPGAs," in *International Workshop on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART 2010) in conjunction with 24th International Conference on Supercomputing (ICS'10)*, 2010.