Adv. Radio Sci., 8, 289–294, 2010
www.adv-radio-sci.net/8/289/2010/
doi:10.5194/ars-8-289-2010

Advances in Radio Science

Sparse matrix-vector multiplication on network-on-chip

C.-C. Sun[1], J. Götze[1], H.-Y. Jheng[2], and S.-J. Ruan[2]

[1]Dortmund University of Technology, Information Processing Lab, Otto-Hahn-Str. 4, 44227 Dortmund, Germany
[2]National Taiwan University of Science and Technology, Low-Power System Lab, Taipei 106, Taiwan

Abstract. In this paper, we present an idea for performing matrix-vector multiplication by using Network-on-Chip (NoC) architecture. In traditional IC design on-chip communications have been designed with dedicated point-to-point interconnections. Therefore, regular local data transfer is the major concept of many parallel implementations. However, when dealing with the parallel implementation of sparse matrix-vector multiplication (SMVM), which is the main step of all iterative algorithms for solving systems of linear equation, the required data transfers depend on the sparsity structure of the matrix and can be extremely irregular. Using the NoC architecture makes it possible to deal with arbitrary structure of the data transfers; i.e. with the irregular structure of the sparse matrices. So far, we have already implemented the proposed SMVM-NoC architecture with the size 4×4 and 5×5 in IEEE 754 single float point precision using FPGA.

1 Introduction

Over the past 30 years, scientists have tried to mitigate the poor performance of sparse matrix computations through various approaches, such as reordering the data to reduce wasted memory bandwidth, modifying the algorithms to reuse the data, and even building specialized memory controllers. Despite these efforts, sparse matrix performance on GPPs (General Purpose Processors) still depends on the sparsity structure of the matrices (Morris and Prasanna, 2007).

Sparse matrix computations occur in various applications. For example, the Finite Element Method (FEM) is a widely used engineering analysis tool based on obtaining a numerically approximate solution for a given mathematical model of a structure. The resulting linear system is characterized by the system matrix A which is usually large and sparse (Elkurdi et al., 2008). Iterative solvers, mainly the Conjugate Gradient (CG) method, are almost dominated by SMVM operations. The CG method is the best-known iterative method for numerically solving linear equation, $A \cdot x = b$, whenever A is a Symmetric Positive-Definite (SPD) sparse matrix. In the past few years, many researchers have presented the hardware solutions utilizing the feature of the pipeline ability and the parallelism inherent from the SMVM computation (Sun et al., 2007; Gregg et al., 2007; Gotze and Schwiegelshohn, 1988; Williams et al., 2007). On the other hand, Google's PageRank (PR) Eigenvalue problem is the world's largest sparse matrix calculation. This algorithm is almost dominated by SMVM operations where the target matrix is extremely sparse, unsymmetrical and unstructured. This problem has also been investigated for acceleration with a FPGA solution in (McGettrick et al., 2008; Zhuo and Prasanna, 2005).

Traditional architectures of SMVM implementations usually focused on a dedicated internal chip interconnection to forward the vector components and nonzero matrix elements between multiple processors. For example, the fat-tree style designs, which required presorting and preordering before input the data (Kapre and DeHon, 2007), will become extremely difficult when the matrix is very large and sparse. This challenge brings current applications and technology trends to motivate a paradigm shift in on-chip interconnect architectures from bus-based point-to-point network to packet-based switch network. This packet-based architecture is called Network-on-Chip (NoC) (Bertozzi and Benini, 2004; Wolf, 2004). The basic idea of the NoC is that we regard a System-on-Chip (SoC) device as a micro network of components. In this paper, in order to solve the problems arising from large sparse matrices with their extremely irregular structures, we select a chip-internal NoC architecture as the main transmission network bone for the data transfers required for the SMVM (SMVM-NoC). Utilizing the packets forwarding functionality is beneficial concerning

Correspondence to: C.-C. Sun
(chichia.sun@tu-dortmund.de)

Published by Copernicus Publications on behalf of the URSI Landesausschuss in der Bundesrepublik Deutschland e.V.

reconfiguration possibilities, flexibility and high resource utilization.

The major contributions of this paper are:

- An idea of SMVM operations based on NoC architecture.
- The SMVM-NoC provides a superior vision to handle the large sparse matrix; especially it is able to deal with the arbitrary structure from sparse matrix.
- Primitive implementation results of 4×4 and 5×5 SMVM-NoC in FPGA.

This paper is organized as follows: In Sect. 2 we clarify the definition of the SMVM operations and the NoC concepts. Furthermore, the basic idea of SMVM operations on NoC is presented. In Sect. 3 the design issues of the NoC will be introduced, which lead to the SMVM-NoC architecture in FPGA. Section 4 shows the primitive implementation results. Section 5 concludes this paper.

2 SMVM on network-on-chip

2.1 Sparse matrix-vector multiplication

A typical SMVM operation is computed as follows:

$$A \cdot x = b, \tag{1}$$

where x and b are vectors of length n, A is a $n \times n$ sparse matrix. Since the matrix A can be very large and sparse, iterative solvers are typically used to solve the system of linear equations due to their low storage requirements and good convergence properties (Golub and Van Loan, 1996). Many researchers have already utilized pipelining and parallelism to improve the performance. However, the computational complexity is usually determined by the sparsity of the matrix A. As mentioned before, the CG method is one of the most popular iterative methods used for solving large and sparse systems. It is almost dominated by SMVM operations (deLorimier and DeHon, 2005).

In this paper, the sparse matrix is stored in a Compressed Sparse Row (CRS) format, in which only the nonzero matrix elements will be stored in contiguous memory locations. In CRS format, there are three vectors: val for nonzero matrix elements; col for the column index of the nonzero matrix elements; and ptr stores the locations in the val vector that start a new row (Zhuo and Prasanna, 2005). As an example, consider a simple SMVM operation with 5×5 sparse matrix A as follows:

$$\begin{bmatrix} 4&0&8&0&0 \\ 6&0&3&0&0 \\ 0&1&0&0&0 \\ 0&0&0&5&0 \\ 7&0&0&0&4 \end{bmatrix} \odot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}. \tag{2}$$

The CRS format of this matrix can be described by three vectors given below:

val:	4	8	6	3	1	5	7	4
col:	1	3	1	3	2	4	1	5
ptr:	1	3	5	6	7	8		

By subtracting consecutive entries in the ptr vector, a new vector len can be generated, where len vector stores the size of each row (i.e., the number of nonzero in each row). Note that the last entry in len vector is calculated by using $n_z - \text{ptr}(n_{\text{ptr}} - 1)$, where n_z represents the total number of nonzero elements and $\text{ptr}(n_{\text{ptr}} - 1)$ is the next to last in ptr vector. For our example, the len vector is:

len:	2	2	1	1	2

2.2 Network-on-chip

Recently, the growing complexity of embedded multiprocessor architectures for digital media processing will soon require highly scalable communication infrastructure. Today most of the current communication architectures in System-on-Chip (SoC) are still based on dedicated wiring. However, the dedicated wiring architecture has its limitation. For instance, when the VLSI technology keeps shrinking down into the nanoscale level, the synchronization issue of nanoscale ASIC implementation with a single clock source will be extremely tough (Hemani et al., 2000; Benini and Micheli, 2002). Therefore, NoC architecture is proposed to replace the traditional bus-based on-chip interconnections to packet-based switch network architecture.

The NoC was proposed to structure the top-level wires on a chip and facilitate the implementation into a modular design. As shown in Fig. 1, this multiprocessor platform consists of a regular 4×4 array of Processing Elements (PEs) where each PE could be a general-purpose processor, a DSP, a memory or a subsystem, etc. A switch (router) is embedded within each PE for connecting itself with its neighboring PEs; the communication can be achieved by routing packets as a switching network. This network is the abstraction of the communication among components and must satisfy quality-of-service requirements, such as reliability, performance, and energy bounds.

2.3 Basic idea of the SMVM-NoC

Based on the Fig. 1, we build up a 4×4 SMVM-NoC with a simple NoC network, which is connected by a two dimensional mesh style chip-internal network and uses the packet forwarding function to transmit the data.

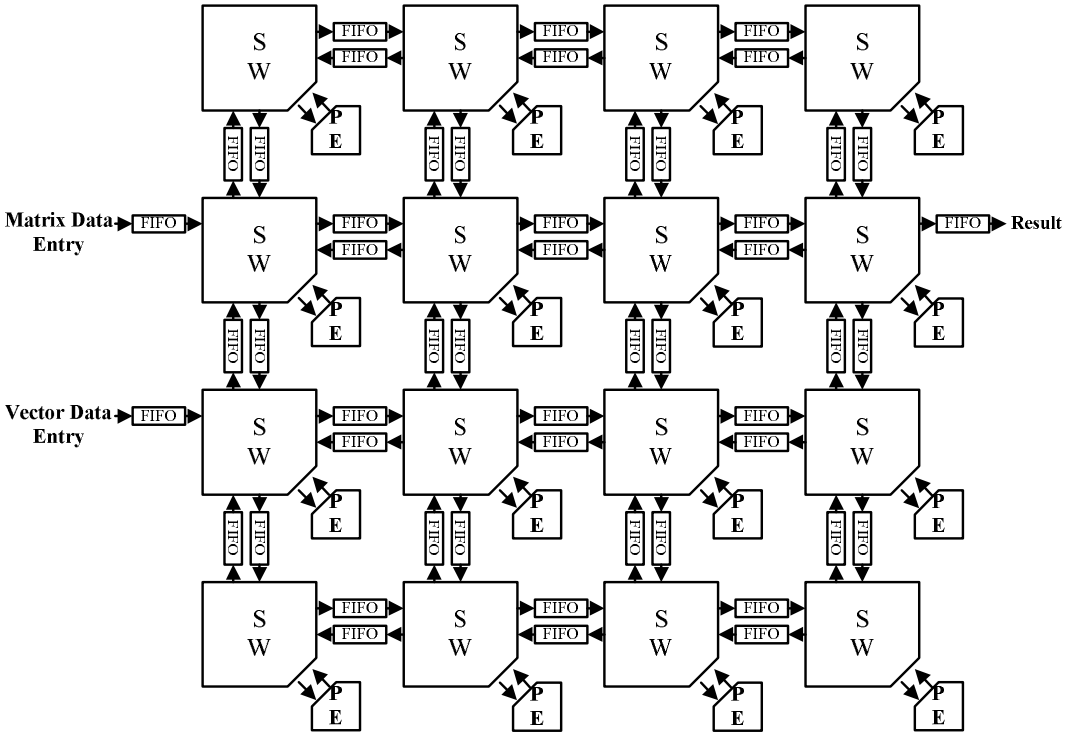

Fig. 1. A 4×4 PE array with a mesh style network, includes three entries for matrix/vector elements and multiplication results.

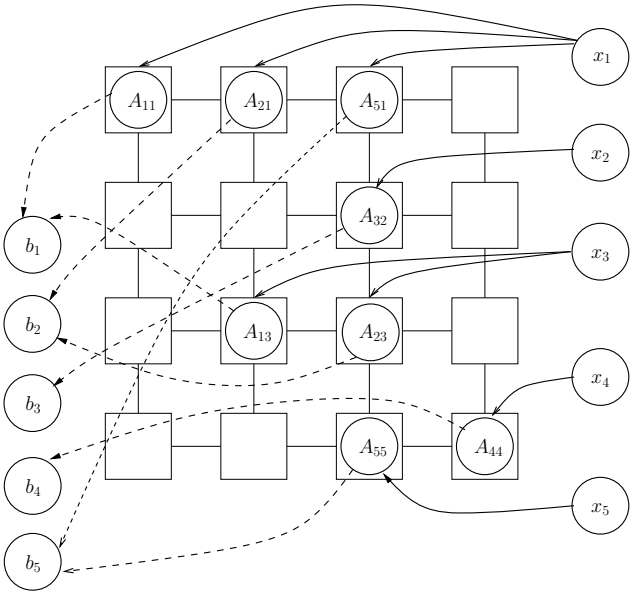

Fig. 2. A simple mapping of a parallel SMVM operations in NoC architecture.

For example, a simple direct mapping of Eq. (2) on this NoC platform is shown in Fig. 2. First of all, the vector components x_j and nonzero matrix elements A_{ij} will be distributed according to the column index j through the mesh network; i.e. vector x_1 arrives in the PEs with A_{11}, A_{21} and A_{51} elements (4, 6, 7), vector x_2 arrives in the PE with A_{32} element (1), vector x_3 arrives in the PEs with A_{13}, A_{23} elements (8, 3) and vector x_4 arrives in the PE with A_{44} element (5), vector x_5 arrives in the PE with A_{55} element (5) in the network, respectively. Second, each local PE will perform the multiply operations $A_{ij} \cdot x_j$ in parallel. After the multiplications, the results of the products $b_i^{(j)} = A_{ij} \cdot x_j$ are routed to other PEs (as indicated by the dashed lines). Finally, the $b_i^{(j)}$ are added up according to the row index i of matrix A_{ij} to obtain b_i.

Although the basic idea of the SMVM-NoC is very simple, there are a lot of research topics, which must be addressed in order to obtain an efficient realization of the general concept. First, the investigation of the suitability of different topologies, routing schemes and congestion avoidance methods in the SMVM-NoC architecture is important. Second, how can different iterative methods (Jacobi, Gauss Seidel and Conjugate Gradient) be realized on the SMVM-NoC architecture efficiently? Meanwhile, how do we use the fact that each iteration step of the iterative methods is actually identical and what does that mean for the SMVM-NoC implementation. Third, further design issues/strategies such as integration of preconditioners, monitoring convergence and matrix partitioning for large matrices should be taken into consideration in the future.

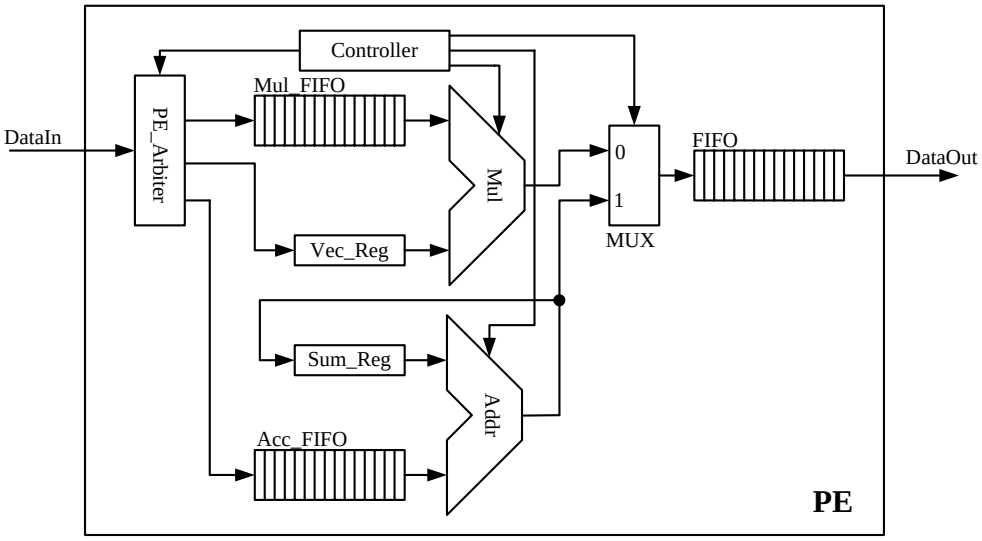

Fig. 3. Schematic view of the PE.

3 Implementation

Our basic idea described in Sect. 2.3 is now implemented on the architecture of Fig. 1. In Fig. 1, the vector components and nonzero matrix elements will distribute from the data entry points at the left side of the platform to each PE through the mesh network. The transmissions are done by packet-switched networks overlaid on top of commodity FPGAs. Finally, the PEs will add up all row index products and send it to the result port in packet style at the right side of the platform. Since the routing scheme for sending data packets between PEs negotiates dynamically at run time, this promises no additional memory for storing schedules and no further off-line setup.

3.1 Processor element

Fig. 3 shows the schematic view of a PE, including one floating point adder, one floating point multiplier, controllers and FIFOs. The input packets will be first decomposed by the "PE Arbiter" then forward to the ACC/MUL FIFOs or the Vector Register. After that, PE will perform a multiplications and the accumulation in IEEE 754 single float point precision. In the current configuration, each PE can hold at most 32 nonzero matrix elements in the Mul FIFO, one vector component, 32 accumulation matrix elements in the ACC FIFO and 32 packets in the output FIFO.

3.2 Switch structure

A basic switch component for the local PE to communicate its neighboring PEs consists of a set of input/output ports, a crossbar network, four input FIFO buffers and a controller circuit as shown in Fig. 4. The switch we used has five data input ports and five data output ports. These ports are named North, East, West, South and Local respectively. In order to be able to prevent the deadlock hazard, Round-Robin strategy is used when several packets arrive at a switch simultaneously.

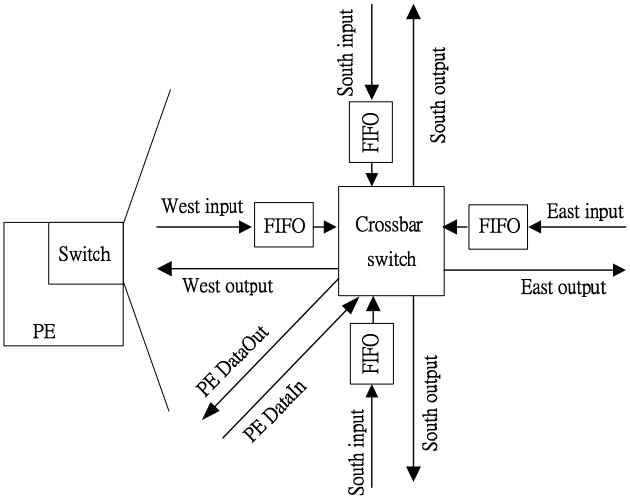

Fig. 4. Detailed switch interconnections of a PE for its neighboring communication.

3.3 Routing algorithm

For simplicity and flexibility, we select direct routing (a.k.a, XY deterministic routing) algorithm for our NoC architecture design. In direct routing, each switch in the network is indexed by their XY coordinates. When a switch receives a packet from the other switches, it will first extract the header

Table 1. Synthesis Results for SMVM Calculator based on NoC Architecture.

Family	Device	Model	Logic utilization	Used	Available	Utilization	Max frequency
Virtex 5	xc5vlx110t	4×4	Slice Registers	24 734	69 120	35%	265 MHz
			Slice LUTs	23 058	69 120	33%	
			DSP48E	32	64	50%	
		5×5	Slice Registers	39 121	69 120	56%	259 MHz
			Slice LUTs	36 716	69 120	53%	
			DSP48E	50	64	78%	

information of this packet and arbitrate the direction, then transmit to next switch in the network. Each packet is first routed along X coordinate and then along Y coordinate until this packet reaches the destination.

4 Experimental result

We have first modeled our proposed architecture in Verilog HDL and synthesized it with Xilinx ISE 10.1 SP3. Later, we verified our implementation on the Xilinx XUPV505 (XC5VLX110T). The synthesized resource utilization reports are shown in Table 1. When the network size is 4×4, it can achieve a peak performance 8.29 GFLOPS in theory. Each PE can hold at most $(32\times3+2)=98$ packets and in total $(4\times4\times98)+(24\times2\times32)=3104$ packets of the SMVM-NoC architecture. A simple testing with a 16×16 dense matrix vector multiplication in a 4×4 network has been verified. Note that so far we have only tested the dense matrix multiplication. Although we have only experimented on dense matrix multiplication, the only limiting is the number of the nonzero elements in the sparse matrix. Therefore, no matter these nonzero elements are from a sparse matrix or a dense matrix the NoC network does not treat them differently. On the other hand, since we used the Xilinx MicroBlaze in FPGA to inject the packets, it is effortless to extend the current architecture to calculate the sparse matrix by modifying the testing program later.

5 Conclusions

In this paper, we presented an idea of the SMVM-NoC architecture. In this architecture, we have tested a 16×16 dense matrix vector multiplication in IEEE 754 single float point precision in FPGA. Note that a sparse matrix with 16·16=256 nonzero elements requires the same resources. The advantages of introducing the NoC structure into SMVM computation are given by high resource utilization, flexibility and the ability to communicate among heterogeneous systems. Since the NoC structure can receive data from and forwards results to different entries simultaneously, this makes it able to deal with very large sparse matrix and disrespect the structure of the sparse matrix. The synthesis results showed that the advanced FPGA with the chip-internal NoC network provides a solution for sparse matrix computation to further accelerate many numerically problems in hardware, such as solving linear equation (CG method), FEM problem and so on.

Future work will extend this architecture to be able to process sparse matrices of arbitrary structure. A detailed performance analysis of different routing algorithms, network topologies and switch architecture will be done. At the end, we will compare it with other sparse matrix computation architectures in different design criteria (area, timing and power).

References

Benini, L. and Micheli, G. D.: Networks on Chips: A New SoC Paradigm, Computer, 35, 70–78, 2002.

Bertozzi, D. and Benini, L.: Xpipes: a network-on-chip architecture for gigascale systems-on-chip, Circuits and Systems Magazine, IEEE, 4, 18–31, 2004.

deLorimier, M. and DeHon, A.: Floating-point sparse matrix-vector multiply for FPGAs, in: FPGA '05: Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays, 75–85, ACM, New York, NY, USA, doi:http://doi.acm.org/10.1145/1046192.1046203, 2005.

Elkurdi, Y., Fernández, D., Souleimanov, E., Giannacopoulos, D., and Gross, W. J.: FPGA architecture and implementation of sparse matrix-vector multiplication for the finite element method, Computer Physics Communications, 178, 558–570, 2008.

Gregg, D., Mc Sweeney, C., McElroy, C., Connor, F., McGettrick, S., Moloney, D. and Geraghty, D.: FPGA Based Sparse Matrix Vector Multiplication using Commodity DRAM Memory, in: International Conference on Field Programmable Logic and Applications, 786–791, 2007.

Golub, G. H. and Van Loan, C. F.: Matrix computations (3rd ed.), Johns Hopkins University Press, Baltimore, MD, USA, http://portal.acm.org/citation.cfm?id=248979, 1996.

Gotze, J. and Schwiegelshohn, U.: Sparse matrix-vector multiplication on a systolic array, 2061–2064 4, doi:10.1109/ICASSP.

1988.197034, 1988.

Hemani, A., Jantsch, A., Kumar, S., Postula, A., Oeberg, J., Millberg, M., and Lindquist, D.: Network on a Chip: An Architecture for Billion Transistor Era, in: Proceedings of IEEE NorChip, 24–31, New York, 2000.

Kapre, N. and DeHon, A.: Optimistic Parallelization of Floating-Point Accumulation, in: ARITH '07: Proceedings of the 18th IEEE Symposium on Computer Arithmetic, pp. 205–216, IEEE Computer Society, Washington, DC, USA, 2007.

McGettrick, S., Geraghty, D., and McElroy, C.: An FPGA architecture for the Pagerank eigenvector problem, Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on, 523–526, 2008.

Morris, G. R. and Prasanna, V. K.: Sparse Matrix Computations on Reconfigurable Hardware, Computer, 40, 58–64, doi:http://doi.ieeecomputersociety.org/10.1109/MC.2007.103, 2007.

Sun, J., Peterson, G., and Storaasli, O.: Sparse Matrix-Vector Multiplication Design on FPGAs, in: FCCM '07: Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 349–352, 2007.

Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K., and Demmel, J.: Optimization of sparse matrix-vector multiplication on emerging multicore platforms, in: SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, 1–12, ACM, New York, NY, USA, doi:http://doi.acm.org/10.1145/1362622.1362674, 2007.

Wolf, W.: The future of multiprocessor systems-on-chips, Annual ACM IEEE Design Automation Conference, 681–685, 2004.

Zhuo, L. and Prasanna, V. K.: Sparse Matrix-Vector multiplication on FPGAs, in: FPGA '05: Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays, 63–74, ACM, New York, NY, USA, doi:http://doi.acm.org/10.1145/1046192.1046202, 2005.