Implementation of Sparse Matrix Vector Multiplication in Verilog HDL

CH. Raviteja
MTech student
Department of ECE G.V.P.College of Engg (A)
Visakhapatnam,A.P.,India

K. R. K.Sastry
Associate Professor
Department of ECE G.V.P.College of Engg(A)
Visakhapatnam,A.P.,India

***Abstract*—This paper presents a high throughput Sparse Matrix Vector multiplication (SpMxV) in FPGA. Besides the throughput the system performance is also obtained. Sparse matrices of university of Florida with less than 0.09 sparsity are used as test pattern for checkingthe design performance. Compressed Row Storage(CRS) minimizes the control logic. Our design implements pipelined architecture which helps in improving system performance over 5x times that of software code running on Pentium4 processor. The design is targeted on XC2VP70-7 Xilinx FPGA and met the operating frequency of 205MHz. The SpMxV is implemented in Verilog HDL.**

***Key Words*—FPGA, SPARSE,Simple Dual Port RAM, MAC, Floating Point,CRS.**

I. INTRODUCTION

Sparse Matrix-Vector multiplication (SpMxV)*y=Ax*, is used in many high-performance scientific computing applications, such as linear system iterative solvers, block LU solvers and eigen value solvers, including information retrieval, medical imaging, and economic modeling. Parallelism in reconfigurable hardware (FPGA) greatly improves the computing performance in integer and floating point operations [1]. Shwetha and Ron used a Variable Dual Compressed Blocks (VDCB) format forsparsematrix-matrix multiplication. The VDCB format works by dividing a matrix into a number of smaller variable sized sub-matrices called BLOCKS [2].Zhuo and Prasanna designed an adder tree based SpMxV implementation for double precision floating point. A reduction circuit is used in their design that needs to be configured according to different matrix parameters [3].

FPGAs have shown great potential in Reconfigurable Computing because of their intrinsic parallelism and flexible architecture. With their rapid increase in gate capacity and frequency, FPGAs can outperform microprocessors on both integer and floating point operations [4].El-Gindy and Shue proposed SpMxV on FPGA for fixed point data [5]. DeLorimier and DeHon arranged the PEs in a bidirectional ring to compute the equation $y=A^i x$, where A is a square matrix and i is an integer. The design they proposed reduces the I/O bandwidth requirement greatly by sharing the results between PEs [6]. El-kurdi et al proposed a stream through architecture for finite element method matrices [7].

In this paper,we implement SpMxV in Compressed Row Storage (CRS) to invariant to number of inputs.

This paper is organized as follows. SpMxV on FPGA and CRS format are explained in section2. Section3 describes basic design of SpMxV and MAC, controller units are explained. Complete design is explained in section4. The implementation results are compared with [1] in section5. Finally we draw conclusion and suggest the future work.

II. SPMXV ON FPGA

The CRS is used in our FPGA and the multiplicand vector *x* (*y=Ax*) is stored in FPGA memory as single column and number of rows as per the input column length. Inorder to minimize the hardware components the multiplicand vector length is limited to 630 elements stored as 63x10.The CRS format is shown in Fig 1.

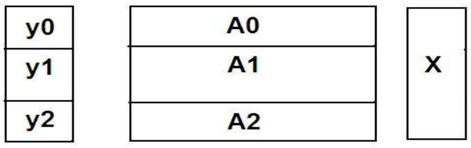

Fig1: Compressed Row Storage(CRS)

As an example, consider 4x7 Sparsematrix as follows:

$$\begin{bmatrix} 10 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 5 \end{bmatrix}_{4x7}$$

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

val	10	6	1	4	3	3	5
col	1	2	1	6	2	4	7

row_cnt	2	2	1	2

In the CRS representation, val vector represents the non-zero elements of the sparse matrix, col vector represents the corresponding column indices of the non-zero elements and row_cnt vector shows number of non-zero elements in each row. First location of row_cnt vector represents first row of sparse matrix, second location represents second row and so on.

III. BASIC DESIGN

A. Top level Block diagram

In our design Sparse Engine(SE) is the main computational block. Top level architecture of SE consisting of two Block RAMs (Coeff.RAM size 64X32, Sparse RAM size 1024x32), two FIFOs (size 1024x6), Floating point MAC (32bit) and a Controller block is shown in Fig2.

MAC consists of multiplier and adder with 8 clock cycle latency eachfor performing 32 bit floating point operations. The design is developed as parallel MAC. A stall is generated when FIFO of either of Col_id, or row_id is full.

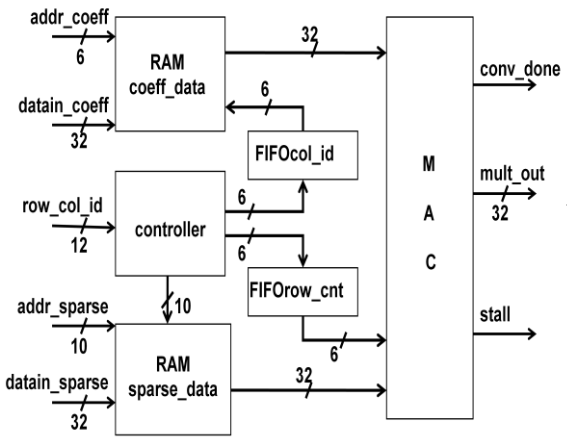

Fig2: Sparse Engine (SE)

B. Controller Block

Controller block is responsible for manipulating the row vector sparse multiplication process. It continuously monitors row_col_id. In the row_col_id, upper nibble 6 bits represents row index and lower nibble 6 bits represents column index of non zero elements.

Controller compares present row_col_id with pre row_col_id, if the row index of two row_col_ids are equal the row_cnt increment by one else row_cnt is written into FIFO row_cnt and repeats the same process for remaining. The lower nibble of each row_col_id is stored in FIFO col_id. Whenever a valid row_cnt is available at FIFO row_cnt, then that particular row elements of sparse matrix and corresponding multiplicand vector elements are processed to MAC. For reading multiplicand vector elements col_id indices are used as reading addresses.

C. MAC UNIT

Detailed Block diagram of MAC is shown in Fig3.Pipelining is used in MAC for improving the speed. The row_count indicates the number of multiplications and additions to be carried on data stored in col and row vectors. Simple Dual port RAM is used in write first read next configuration.

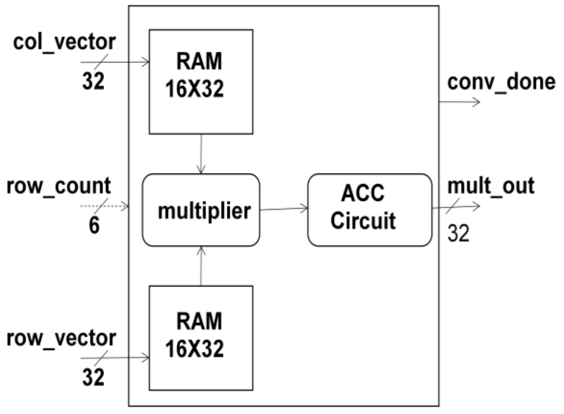

Fig 3: Floating point MAC unit

In order to perform dot product of one row, multiplier executes row_cnt number of multiplications and Accumulation circuit accumulates each multiplication result.

conv_done signal is generated to indicate the valid row multiplication subsequent to the valid signal another row could be fetched to the MACwhich indicates valid mult_out result. The result mult_out represents dot product between row vector of sparse matrix and column vector of coefficient matrix.

IV. COMPLETE DESIGN

Parallel structure of SEs shown in Fig4 will help in developing the multiple row operations.The results of completed rows are stored in result BRAM. In general based on FPGA we can use n number of SEs. In our design 8 SEs are used for SpMxV operation.

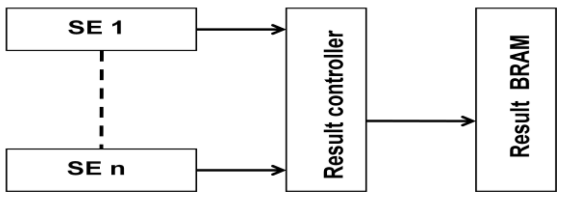

Fig4: Complete Design

V. IMPLEMENTATIONRESULTS

The design is targeted on Xilinx XC2VP70-7 FPGA. The implementation results are summarized in Table1. We checked our results for single floating point values and our design can adoptable to integer and double floating point values also with certain optimizations.

Table 1: Characteristics of SpMxV

	[1]	Proposed
Target Device	XC2VP70-7	
Design	Single FP	
Achievable Frequency	200 MHz	210 MHz
Slices	10528 (31%)	6877 (20%)
MULT18x18	32 (9%)	32 (9%)

We use our design of 8 SEs to compare with software on microprocessor. The machine is dual 2.8GHz intel Pentium 4 with 16KB L1, 512KB L2 Cashe and 1GB memory. The speedup of our design over pentium4 is shown in Fig 5. The two test matrices are belong to Bai group taken from Florida Sparse matrix collection[8]. As the number of non zero elements increases the performance is increased.

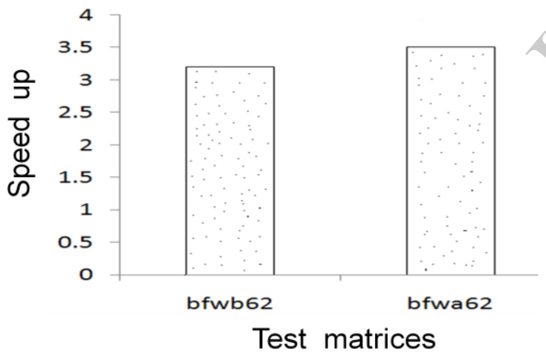

Fig 5: Speedup over 2.8GHz Pentium4

VI. CONCLUSIONS

We used CRS format for storing sparse matrices in SpMxV design. Compared to [1], our design has significant area, speed improvement and depends less on matrix sparsity. Our future work includes implementing our design on the Cray XD1 supercomputer for scientific applications and performance analysis.

REFERENCES

[1] Junqing Sun, Gregory, Olaf Storaasli. "Sparse Matrix-Vector Multiplication Design on FPGAs", IEEE, Feb 2007.

[2] Shwetha Jain-Mendon and Ron Sass, "Performance Evaluation of Sparse Matrix-Matrix Multiplication", IEEE, June 2013.

[3] L. Zhuo and V. K. Prasanna, "Sparse matrix-vector multiplication on FPGAs", FPGA, Feb 2005.

[4] K. D. Underwood. "FPGAs vs. CPUs: Trends in peak floating-point performance", FPGA, Feb 2004.

[5] H. A. ElGindy and Y. L. Shue., "On Sparse Matrix-Vector Multiplication with FPGA- based System", FCCM, Apr 2002.

[6] M. deLorimier and A. DeHon. "Floating-Point Sparse Matrix-Vector Multiply for FPGAs".International Symposium on Field Programmable Gate Arrays, Feb 2005.

[7] Y. El-kurdi, W. J. Gross and D. Giannacopoulos. "Sparse Matrix-Vector Multiplication for Finite Element Method Matrices on FPGAs".2006 IEEE Symposium on Field Programmable Custom Computing Machines, April 2006.

[8] T. Davis, University of Florida Sparse Matrix Collection, http://www.cise.ufl.edu/research/sparse/matrices, NA Digest, 92(42), October 16, 1994, NA Digest, 96(28), July 23, 1996, and NA Digest, 97(23), June 7, 1997.