A Review of VLSI Structure for the Implementation of Matrix Multiplication

Ruchi Thakkar
Truba College of Science and Technology, Bhopal
House No 48 Jaknore Nivas Ratnagiri Raisai Road
Piplani BHEL Bhopal

Paresh Rawat
Truba College of Science and Technology, Bhopal
House No 48 Jaknore Nivas Ratnagiri Raisai Road
Piplani BHEL Bhopal

ABSTRACT
Matrix multiplication is the kernel operation used in many transform, image processing and digital signal processing application. In this paper, we have studied for parallel-parallel input and single output (PPI-SO), parallel-parallel input and multiple output (PPI-MO) and parallel-parallel fixed input and multiple output (PFI-MO) matrix-matrix multiplication. It is also a well-known fact that the multiplier and adder unit forms an integral part of matrix multiplication. Due to this regard, high speed multiplier and adder become the need of the day. In this paper, we have studied of Vedic mathematics multiplier using compressors.

Keywords
Parallel-Parallel Input and Single Output (PPI-SO), Parallel-Parallel Input and Multiple Output (PPI-MO), Matrix Multiplication (MM)

1. INTRODUCTION
With the development in size of incorporation, more complex sign handling circuits are being executed in VLSI chips. These intricate sign handling circuits request huge computational limit as well as have high vitality and region prerequisites. In spite of the fact that region and rate of operation remain the significant configuration concerns, power utilization is likewise rising as a discriminating variable for present VSLI framework creators [1-4]. The requirement for low power VLSI configuration has two noteworthy inspirations. First and foremost, with increment in working recurrence and handling limit per chip, huge current must be conveyed and the warmth produced because of substantial force utilization must be disseminated by legitimate cooling methods, which represent extra framework cost. Besides, the blasting business sector of convenient electronic apparatuses requests for complex circuits to be controlled by lightweight batteries with long times between re-charges (for occurrence [5].

Another significant ramification of overabundance force utilization is that it points of confinement incorporating more transistors on a solitary chip or on a different chip module. Unless force utilization is significantly lessened, the subsequent warmth will restrain the attainable pressing and execution of VLSI circuits and frameworks. From the ecological perspective, the littler the force scattering of electronic frameworks, the lower warmth pumped into the encompassing, the bring down the power devoured and consequently, brings down the effect on worldwide environment [6].

Lattice augmentation is ordinarily utilized as a part of most flag preparing calculations. It is additionally an as often as possible utilized piece operation as a part of a wide mixture of design, picture handling and automated applications. The framework increase operation includes countless and additionally gathering. Multipliers have vast region, longer dormancy and expend extensive force contrasted with adders. Registers, which are obliged to store the middle of the road item values, are likewise significant force serious segment [7]. These segments represent a noteworthy test for planning VLSI structures for huge request framework multipliers with upgraded speed and chip-zone. In any case, region, speed and force are generally clashing equipment limitations such that enhancing one element debases the other two.

With the attention on low power outline approach, it was found that a great part of the advancement in the field has been on segment examination: better batteries with more power per unit weight and volume; low power CPUs; low power radio handsets; low power shows. In spite of the fact that low-control segments and subsystems are crucial building squares for compact frameworks, we focus on compositional level outlining for accomplishing that objective. A framework wide structural engineering is beneficial on the grounds that there are conditions between subsystems, e.g. enhancement of one subsystem may have results for the vitality utilization of different modules. In this way, vitality decrease strategies must be connected in all outline levels of the framework. Besides, as the best plan choices are gotten from the building and framework level, a mind full outline at these levels can diminish the force utilization extensively [8].

We have proposed design for implementing the matrix multiplication operation in hardware keeping the goal of a power efficient architecture. These designs are verified using various hardware simulating tools.

There are three basic method of implementing the matrix multiplication i.e. parallel to parallel input serial output (PPI-SO), parallel to parallel input multiple output (PPI-MO) and parallel to parallel fixed input multiple output (PFI-MO). The compressors can be used with multiple output (MO) techniques for further reducing the area and delay. In this paper we focus on the matrix multiplication design with PPI-MO method for processing data.

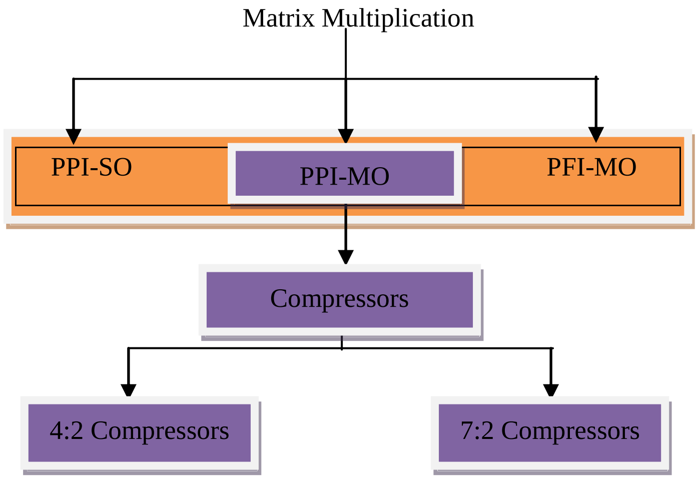


Figure 1: Classification of Matrix Multination Technique

The entire paper has been partitioned into five parts. In II, literature survey for matrix multiplication technique has been discussed. Brief description is Vedic mathematics multiplier using compressors in section III. In IV, proposed work of literature review has been discussed. In V, conclusions and future scope of the paper work has been presented.

2. LITERATURE SURVEY

The objective of our paper work was to design efficient low power architecture for matrix multiplication operation. From the earlier reported works in this field, the major power consuming resource were found to be multipliers and the registers, used to store and move the intermediate data. So, we have studied three designs which reduce as well as optimize the number of multipliers and registers being used in the matrix multiplication operation. For the ease of recognition we have named the designs on the basis of input and output dataflow.

Let us consider the matrix – matrix multiplication for two $n \times n$ matrices A and B given by-

$$C\begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{bmatrix} = A\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \times \mathrm{B}\begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix} \quad \ldots.(1)$$

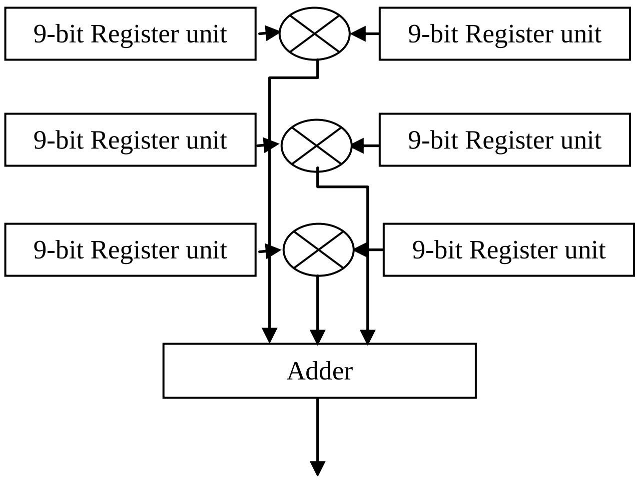
Such that,

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj} \quad \ldots(2)$$

for all i, *j*, a_{ik}, b_{kj}, and c_{ij} represent elements of the n×n matrices A, B and C.

2.1 PPI - SO

In this design we have reduced the resource utilization in terms of number of multipliers and registers in lieu of the completion time.



Output

Figure 2: Proposed PPI – SO Design for n = 3

This design is particularly useful where resources are limited and design can be compromised on basis of increased completion time. The basic working model for a 3 × 3 matrix-matrix multiplication is shown in figure 2 below.

From equation 2, we observe that each element of the output matrix, C, is computed by multiplying and accumulating the elements of the corresponding row and column of the input matrices, A and B respectively. This basic idea is exploited in the design. Considering the matrix – matrix multiplication of two n×n matrices, the calculation is performed using n number of multipliers, n number of registers and n-1 number of adders. n^2 Cycles are required to perform the matrix multiplication operation. Each multiplier factor has 2 input ports: one every from matrix A and B.

The data flow to the multipliers are such that, k^{th} multiplier is fed from k^{th} column of matrix A and k^{th} row of matrix B, where 1 < k < n. At the k^{th} multiplier, each element from matrix A is repeated for n consecutive cycles whereas the elements from matrix B are cycled back after n cycles. The partial products are then fed to the adder which computes the final result.

For a better understanding of the process, let us consider the matrix multiplication for n = 3 (as shown in figure 2). In this case, 3 multipliers and 3 registers are used to calculate and store the partial products respectively. These partial products are then fed to the adder block to compute the final result. The first multiplier receives input from the first column of matrix A (a_{k1}) and first row of matrix B (b_{1k}), where. Each element of the matrix A at the first multiplier is repeated for 3 cycles, such that the data flow can be represented as $a_{11}a_{11}a_{11}$ $a_{21}a_{21}a_{21}$ $a_{31}a_{31}a_{31}$.Similarly, at the first multiplier, the elements of B are repeated after 3 cycles, such that the input data-flow will be $b_{11}b_{12}b_{13}$ $b_{11}b_{12}b_{13}$ $b_{11}b_{12}b_{13}$. The other two multipliers receive the component of A and B in the similar order as the first multiplier. After the multiplication, the partial products are fed to the adder which computes the elements of output matrix C in row major order given by

$c_{11}c_{12}c_{13}$ $c_{21}c_{22}c_{23}$ $c_{31}c_{32}c_{33}$. So the entire matrix multiplication operation is performed in n^2 =9 cycles [9].

2.2 PPI-MO

In this design, we opted for faster operating speed by increasing the number of multipliers and registers performing the matrix multiplication operation. From equation 2 we have derived for parallel computation of 3 × 3 matrix-matrix multiplication and the structure is shown in figure 3.

For an n×n matrix – matrix multiplication, the operation is performed using n^2 number of multipliers, n^2 number of registers and $n^2 - n$ number of adders. The registers are used to store the partial product results. Each of the n^2 number of multipliers has one input from matrix B and the other input is obtained from a particular element of matrix A. The dataflow for matrix B is in row major order and is fed simultaneously to the particular row of multipliers such that the i^{th} row of matrix B is simultaneously input to the i^{th} row of multipliers, where 1 < i < n [10] .

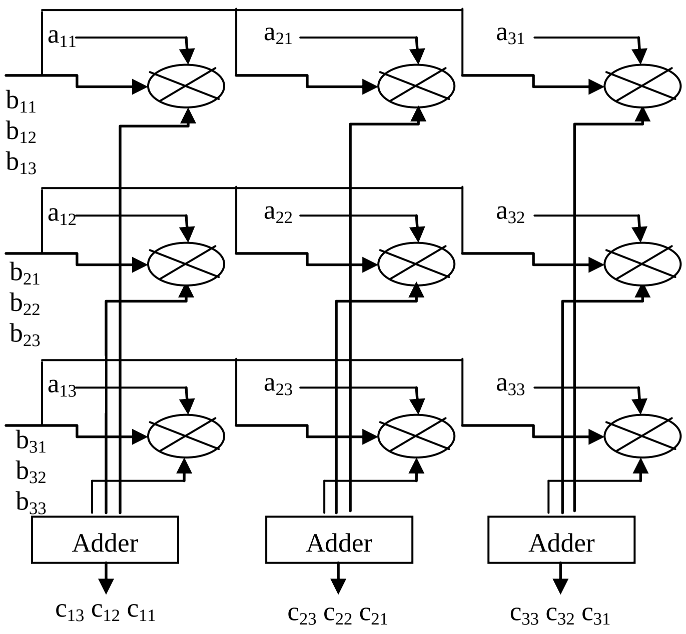


Figure 3: Proposed PPI – MO Design for n = 3

2.3 PFI-MO

Various matrix multiplication applications involving signal processing, filtering and sinusoidal transforms have fixed kernel matrices and only the input matrix is variable. This characteristic is exploited for the proposed PFI – MO design.

The proposed PFI – MO design can be considered as an improvement over the proposed PPI – MO design. The hardware constraints are similar to the previous design with n^2 number of multipliers, n^2 number of registers and $n^2 - n$ number of adders required for computing an n×n matrix – multiplication. The presence of known fixed values is exploited by initially storing the values of matrix A, thus reducing the requirement of input ports for the matrix. The dataflow for matrix B is in row major order and is fed simultaneously to the particular row of multipliers such that the i^{th} row of matrix B is simultaneously input to the i^{th} row of multipliers, where 1 < i < n. Other input to the $(i, j)^{th}$ multiplier is accessed from the register storing $(j, i)^{th}$ element of matrix A, where 1 < i,j < n. The resultant products from each column of multipliers are then added to give the elements of output matrix C. per cycle. So, the entire matrix multiplication computation is performed in n cycles [10].

Table 1: Comparisons table for 3×3 matrix multiplication

	Multiplier	I/P Port	Adder	Register	Cycle
PPI-SO	3	6	2	6	9
PPI-MO	9	12	6	3	3
PFI-MO	9	3	6	3	3

2.4 Summary

Due to large size of the inner-product or matrix - vector or matrix – matrix multiplication, huge amount of computation is involved. Inner-product and matrix multiplication are being implemented in VLSI system for low-cost and real-time applications. Since, filtering and transforms are most common in multi-media applications, a power efficient VLSI design is required to meet the requirement of the such applications.

From the literature review we found that multiplier consumed large area compare to adder. So we are going to Vedic mathematics lies in the fact that it reduces the otherwise cumbersome-looking calculations in conventional mathematics to a very simple one. This is so because the Vedic formulae are claimed to be based on the natural principles on which the human mind works. This is a very interesting field and presents some effective algorithms which can be applied to various branches of engineering such as computing and digital signal processing.

The multiplier architecture can be generally classified into three categories. First is the serial multiplier which emphasizes on hardware and minimum amount of chip area. Second is parallel multiplier (array and tree) which carries out high speed mathematical operations. But the drawback is the relatively larger chip area consumption. Third is serial-parallel multiplier (based on compressor) which serves as a good trade-off between the times consuming serial multiplier and the area consuming parallel multipliers.

3. VEDIC MATHEMATICS MULTIPLIER USING COMPRESSORS

Vedic science is an antiquated quick figuring math system which is taken from recorded old book of intelligence. Vedic science is an antiquated Vedic math which gives the exceptional procedure of mental estimation with the assistance of straightforward tenets and standards. Swami Bharati Krishna Tirtha (1884-1960), previous Jagadguru Sankaracharya of Puri selected arrangement of 16 Sutras (axioms) and 13 Sub - Sutras (culminations) from the Atharva Veda. He created strategies and methods for opening up the standards contained in the equations and their sub-recipes, and called it Vedic Mathematics. As indicated by him, there has been extensive writing on Mathematics in the Veda-sakhas [11].

Vedic science is a piece of four Vedas (books of shrewdness). It is a piece of Sthapatya- Veda (book on structural building and construction modeling), which is an upa-veda (supplement) of Atharva Veda. It covers clarification of a few present day numerical terms including number-crunching, geometry (plane, co-ordinate) and trigonometry.

• 4:2 COMPRESSOR

To add binary numbers with minimal carry propagation we use compressor adder instead of other adder. Compressor is a digital modern circuit which is used for high speed with minimum gates requires designing technique. This compressor becomes the essential tool for fast multiplication adding technique by keeping an eye on fast processor and lesser area.

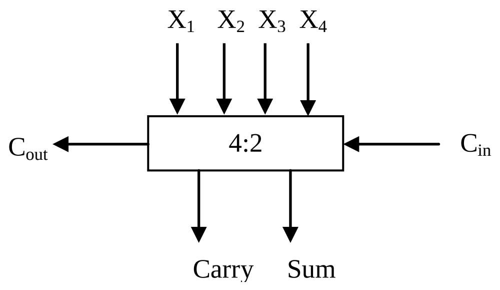


Figure 4: Block Diagram of 4:2 Compressors

4:2 compressors are capable of adding 4 bits and one carry, in turn producing a 3 bit output. The 4-2 compressor has four inputs X_1, X_2, X_3 and X_4 and Sum and Carry output along with a Carry-in (Cin) and a Carry-out (Cout) as shown in Figure 4. The input Cin is the output from the previous lower significant compressor.
The Cout is the output to the compressor in the next significant stage. The critical path is smaller in comparison with an equivalent circuit to add 5 bits using full adders and half adders.

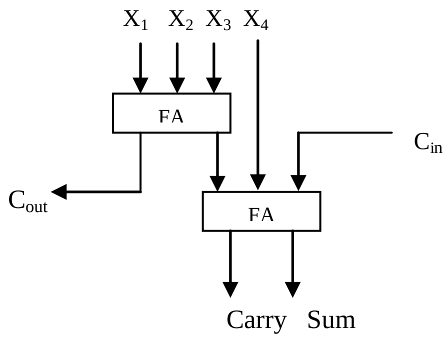


Figure 5: Logical Diagram of 4:2 Compressors

$$C_{out} = X_1X_2 + X_2X_3 + X_1X_3 \quad (3)$$

$$S_1 = X_1 \oplus X_2 \oplus X_3 \quad (4)$$

$$C_{arry} = S_1X_4 + X_4C_{in} + S_1C_{in} \quad (5)$$

$$Sum = S_1 \oplus C_{in} \oplus X_4 \quad (6)$$

• 7:2 COMPRESSOR

Similar to its 4:2 compressor counterpart, the 7:2 compressors as shown in Figure 6, is capable of adding 7 bits of input and 2 carry's from the previous 4:2 compressors, at a time. We have designed a novel 7:2 compressor utilizing two 4:2 compressors, two full adders and one half adders. The architecture for the same has been shown in Figure 7.

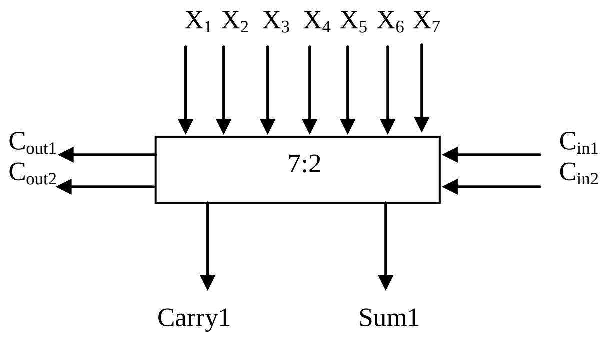


Figure 6: Block Diagram of 7:2 Compressors

$$Sum1 = S_1 \oplus S_2 \quad (7)$$

$$Carry1 = S_3 \oplus C_1 \oplus C_{21} \quad (8)$$

$$C_{out1} = C_3 \oplus C_2 \oplus C_{22} \quad (9)$$

$$C_{out2} = C_3C_2 + C_{22}C_2 + C_3C_{22} \quad (10)$$

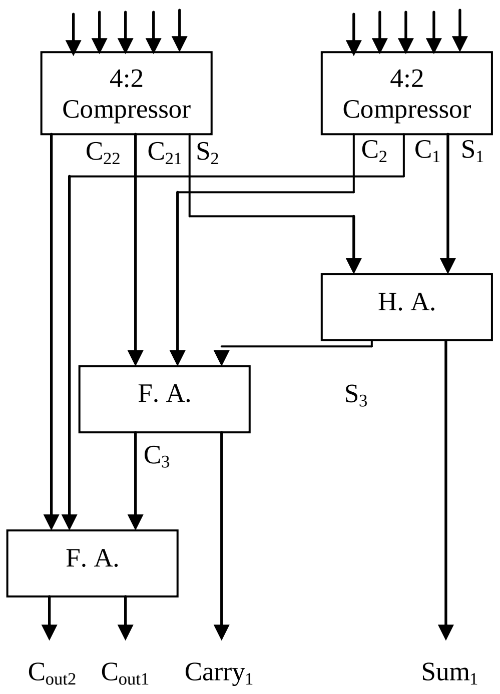


Figure 7: 7:2 Compressor using 4:2 Compressor

4. PROPOSED ARCHITECTURE

Therefore it is required to design a matrix multiplication with low maximum combinational path delay (MCPD). So to achieve this we proposed to reduce and optimize the number of multipliers and adders required for the matrix multiplication operation.

5. CONCLUSION

Most of the digital signal processing (DSP) algorithms is formulated as matrix-matrix multiplication, matrix-vector multiplication and vector-vector (Inner-product and outer-product) form. On the other hand, most of these algorithms are currently implemented in hardware to meet the temporal requirement of real-time application [9]. This future work has proposed matrix multiplication using compressors based multiplier. Both parallel and pipelining techniques have also been used in the proposed designs.

A compressor adder is a logical circuit which is used to improve the computational speed of the addition of 4 or more bits at a time. Compressors can efficiently replace the combination of several half adders and full adders, thereby enabling high speed performance of the processor which incorporates the same. Compressors are generally designed by XOR-XNOR gates and multiplexers. A compressor is a device which is used to reduce the operands while adding terms of partial products in multipliers.

6. REFERENCES

[1] T. Arslan, D.H. Horrocks, and A.T. Erdogan,“Overview and Design Directions for Low-Power Circuits and Architectures for Digital Signal Processing,” IEE Colloquium on Low Power Analog and Digital VLSI: ASICS, Techniques and Applications, pp. 6/1 – 6/5, 1995.

[2] Massoud Pedram, “Design Technologies for Low Power VLSI,” Encyclopedia of Computer Science and Technology, pp. 1 – 32,1995.

[3] O. Mencer, M. Morf, and M. J. Flynn, “ PAM-Blox: High performance FPGA design for adaptive computing,” in Field Programmable Custom Computing Machines (FCCM), pp. 167 – 174, 1998.

[4] K. K. Parhi, VLSI Digital Signal Processing Systems. John Wiley & Sons, Inc. 1999.

[5] A. Amira, A. Bouridane, and P. Milligan, “Accelerating matrix product on reconfigurable hardware for signal processing,” in Proceedings 11th International Conference on Field-Programmable Logic and Its Applications (FPL), pp. 101 – 111, 2001.

[6] S. Tugsinavisut, S. Jirayucharoensak and P. A. Beerelt, “An Asynchronous Pipeline Comparisonswith Applications to DCT Matrix-vector Multiplication,” in Proceedings of the 2003 International Symposium on Circuits and Systems(ISCAS), vol. 5, pp. V-361 - V-364, 2003.

[7] Ju-Wook Jang, Seonil B. Choi, and Viktor K. Prasanna,” Energy- and Time-Efficient Matrix Multiplication on FPGAs”, IEEE Transaction on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 11, pp. 1305 – 1319, 2005.

[8] Song Sun, Madhu Monga, Phillip H. Jones, “An I/ O Bandwidth-Sensitive Sparse Matrix-Vector Multiplic Engine on FPGAs”, IEEE Transactions On Circuits And Systems—I: Regular Papers, Vol. 59, No. 1, January 2012.

[9] Shivangi Tiwari and Nitin Meena, “Efficient Hardware Design for Implementation of Matrix Multiplication by using PPI-SO”, International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 4, July 2013 ISSN (Online): 2320 – 9801.

[10] Shivangi Tiwari, Shweta Singh and Nitin Meena, “FPGA Design and Implementation of Matrix Multiplication Architecture by PPI-MO Techniques”, International Journal of Computer Applications (0975 – 8887)Volume 80 – No1, October 2013.

[11] Sushma R. Huddar and Sudhir Rao, “Novel High Speed Vedic Mathematics Multiplier using Compressors”, 978-1-4673-5090-7/13/$31.00 ©2013 IEEE.