ASIC Implementation of Bit Matrix Multiplier

K. Swetha Reddy[1], *Surabhi* Seethai[1], *Akanksha*[1],*Meenakshi*[1], *V.* Sagar Reddy[1]
[1] Department of ECE, VNR Vignana Jyothi Institute of Engineering and Technology, Telangana, India

Abstract: In computer science and digital electronics, a bit matrix multiplier (BMM) is a mathematical operation that is used to quickly multiply binary matrices. BMM is a basic component of many computer algorithms and is utilized in fields including machine learning, image processing, and cryptography. BMM creates a new matrix that represents the product of the two input matrices by performing logical AND and XOR operations on each matrix element's binary value. BMM is a crucial method for large-scale matrix operations since it has a lower computational complexity than conventional matrix multiplication. Reduced computational complexity: When compared to conventional matrix multiplication algorithms, BMM has a lower computational complexity since it performs matrix multiplication using bitwise operations like logical AND and XOR. Faster processing speeds are the result, particularly for complex matrix computations. Less memory is needed to store the binary values of the matrices in BMM because these values can be expressed using Boolean logic. As a result, less memory is needed, and the resources can be used more effectively.

1 INTRODUCTION

In order to expedite the management of mixed media, BMM advice can also serve as both sound and visual directions. BMM. N is currently only commercially available on supercomputers as a costly piece of equipment, so smaller, less expensive cycle grid duplicate functionality can be added to software microchips or application-explicit processors. BMM provides digit controls, media sub word activities, and consolidated bit tasks without regard to lattice enhancement and framework rendering (BMMT). Matrix hindering[3] is used to investigate ideal sub-framework sizes for BMM crude guidelines that can all execution of a few enhanced apps and cost trade-offs in the BMM plan.

The BMM, n digit grid duplicate activity has two n x n[2] frameworks, An and B, that are increased. An elective plan, (BMMT.n), which translates the B lattice, has parts from left to right that are numbered ascending up and through in a grid, starting at 1. Although the translation is assured because B serves as the guidance input, it is exceedingly challenging to execute BMM when using B as the enhancement vector. Each column of A and the entire B lattice are prerequisites for each line of the outcome framework C. [11]

*Corresponding Author: swethareddy_k@vnrvjiet.in

Supercomputers have a special 64 x 64 BMM register that is used to store the B lattice. The entire B lattice, including the items in this register, is read in a single cycle.

To transmit the contents of a vector register to the BMM register, it signifies an additional instruction. sub-lattice decay the columns of C back to a standard vector register x after streaming the lines of A from an usual vector register, duplicating each column by the whole B lattice in a single cycle. The A network is held by the two universally useful register operands, and the result C is assembled back into a broadly useful register. In essence, a product or installed processor may not be interested in a large unit like BMM.64[1]. A common method for increasing store use in grid duplication is lattice impeding. It uses the gap and vanquish method, where the initial grid is divided into smaller matrices. This strategy is utilized to perform lattice duplication effectively in memory obliged conditions. We have utilized this strategy to track down better trade-offs, for example the decision of n, m and k, for the plan of a BMM unit that performs $(n \times k) \times (k \times m)$ [14]increase the best boundaries for the deterioration rely upon four variables. The number of terms handled in the augmentation of the submatrix was nkm. When using the more conservative BMM guideline as a crude activity, increasing nkm probably produces the grid duplication that happens the fastest.

The bit matrix multiplier's objective is to obtain the best possible accuracy and performance, as measured by the area and throughput. The BMM, which is now only found in supercomputers but is employed in future applications of staircase encoders, is the most potent and significant bit level operation.[10] It uses a simpler process to generate output for a wider range of input bit sizes. It is also used in error correction codes and gives a codeword to the original data[4].The paper exhibits the encoding part and will encode the input data and the error probability is calculated[5]by analysing the input codeword. Research says that the implementation on FPGA based design have been initiated by reconfigurable hardware[13] and realization of matrix multipliers.

The implementation of bit matrix multiplier in the staircase encoder[6,7] is to reduce the complexity[9] than other multipliers. The main aim of the bit matrix multiplier is to give high through put rate and be practically possible in implementation. A thorough detailed working specification helps assist the design process with the design less prone to the errors. The design is implemented in Xilinx software and the output is received from the RTL schematic view and simulation is done resulting the multiplication of the matrices. For the bit matrix multiplier, the input data is taken as 12 and 3 parity multipliers are taken which are then taken into modulo 2 multiplication by ANDing and XORing the bits. Similarly for a small number of bits, it is also capable of accommodating large number of bits such as with input data of 478 bits and a ROM of 32 bits is allocated to Pb i.e., the parity generator matrix and is a parallel implementation of the blocks[12] which is also demonstrated clearly in the paper.

2 DESIGN OF BIT MATRIX MULTIPLIER

According to this stair case encoder we are building bit matrix multiplier which is essential block for our stair case encoder. The parallel block staircase encoder's architecture, which aims to provide high throughput and low latency, is shown in the previous figure. It has registers to store the values in, as well as a bit matrix multiplier, bit matrix adder, and parity generator matrix. The bit matrix multiplier needs input data in the initial step of the design process. The Pb submatrix, which provides the multiplier with processing data, is one of two submatrices that make up the parity generation matrix. The bit matrix multiplier's architectural diagram and explanation of how it works are shown below. For example, if we design for a 12-bit input data Ak and let the number of bits from Pb be 3 bits or 3 partial parity multipliers then through multiplication we get a final of 36 bits as the result.

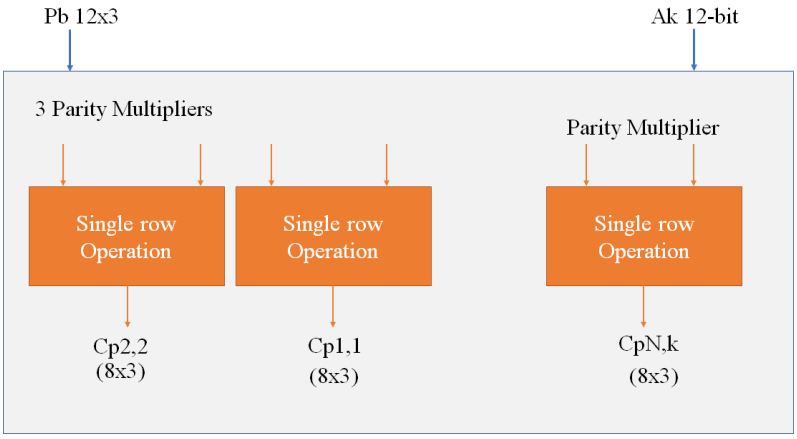

Fig. 1. Block diagram of 36-bit matrix multiplier.

In the figure 1, we see a clear operation of the 12-bit input data being given and 3 parity multipliers which have single row operation in which each block there are 4 multipliers after which all the bits are XORed and gives a single bit as output. The multiplier has a 3 bit modulo two multipliers, which means 4 multipliers have an operation of 4x3 which gives 12 as the input data, and from the parity generator Pb we get 3 parity multipliers which gives 12x3=36 bits for a single row operation.

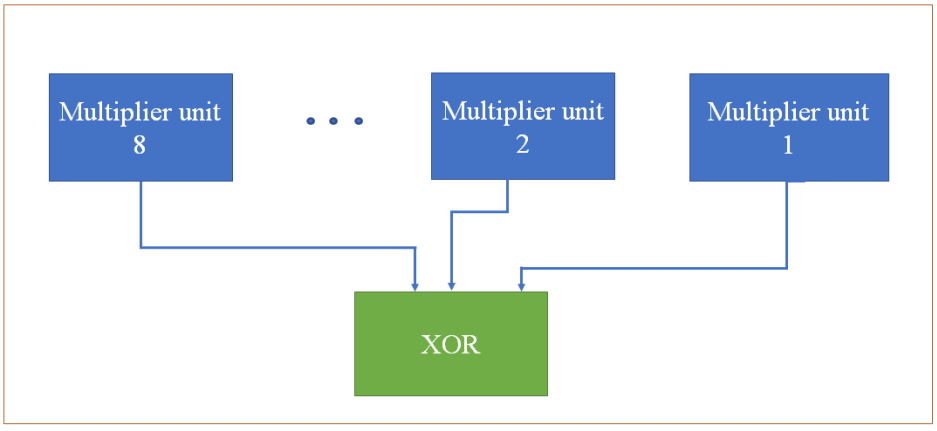

Fig. 2. Block diagram of 4-bit parallel multiplier.

This is 4-bit parallel multiplier as demonstrated, the block diagram of a single row operation of a single block is shown in the Fig. 2, which consists of 4-bit parallel multipliers. For each multiplier there is a 3-bit input multiplier present which gives 3-bit data and from each multiplier unit we get 12-bit data as 4X3=12 multiplication takes place, and the 12-bit data multiplies with each 3-bit multipliers to give 36-bit data as the output. Similarly for 478 input bits the operation is explained below instead of 36 bits and Pb, the parity generation matrix we take 32 parity multipliers in which for single row operations 159 multipliers are given and 159x3=477 bits are produces and the remaining one bit (478) it is given to an AND operation and all of the bits are XORed to give one single bit.

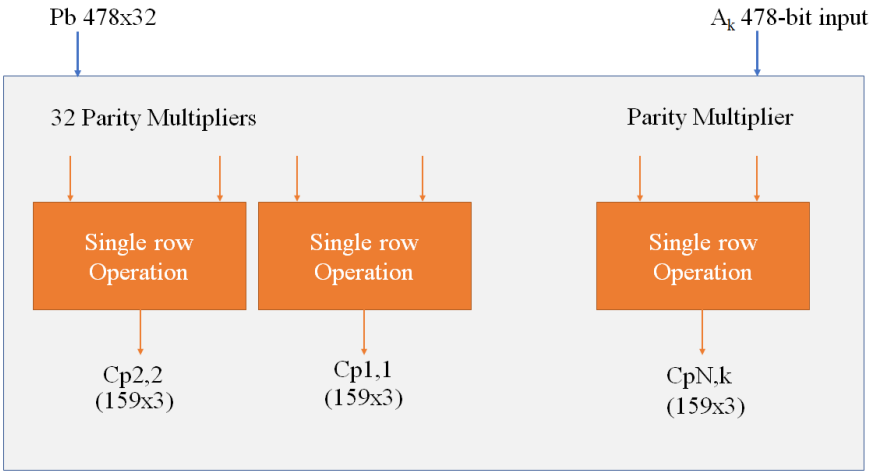

Fig. 3. Block diagram of 478-bit matrix multiplier.

In Figure 4, is the logic diagram of 3-bit input multiplier which consists of AND and XOR gates. The input to the 3 AND gates is the input Ak and Pb memory bit data. The above diagram is an example taken from multiplier unit 2 of single row operation. The input data is given through AND gates and are XORed to give the final 3-bit output from a single multiplier unit and as 4 multipliers are there in parallel, the 3-bit data multiplies with the 4 multiplier units and gives a 12-bit data(4x3=12).

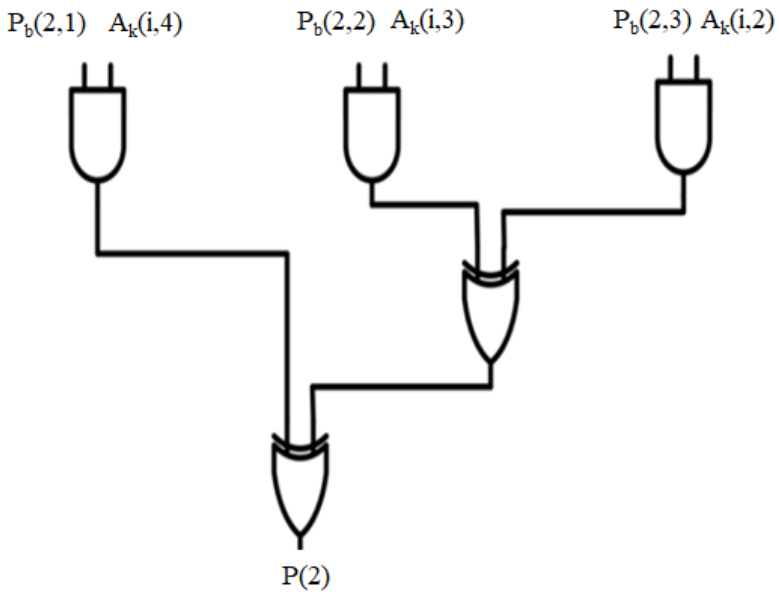

Fig. 4. Block diagram of 3-bit input multiplier.

Similarly, as shown in the figure 3, for input data such as 478-bit input data in form of Ak, where 32 parity multipliers are present and Pb which is the memory bit input data(32x478), 159 multiplier units are there in every single row operation and 32 such

multipliers are placed parallelly. Each multiplier has a 3-bit multiplier, therefore for 159 such units we get the result as 477(159x3) and for 32 such parity multipliers 32x478 we get the output. Hence, we prove here that, as for 36 bits or as large as 478 bits the operation performed is same and we get the accurate results. In the bit-matrix multiplier the modulo-2 augmentation is applied to Ak (I) and Pb in lined up by ANDing and XORing of pieces in Ak (i) with columns in Pb to deliver 32 halfway equality bits Cp2, k(i). The piece grid multiplier comprises of 32 equality multipliers that process the 32 equality pieces of Cp2, k (I) simultaneously. Every equality multiplier registers 1 bit of Cp2, k(i). The equal calculation likewise occurs inside the equality multiplier by utilizing 159 multiplier units. Every multiplier unit process 3 pieces from a column of PT b and Ak (I), separately. The quantity of information pieces of multiplier units is chosen to accomplish high parallelization level. The pieces produced from the multiplier units are XORed with the pieces created by ANDing the last piece of Ak (i) with the last piece of line of PT b. The last processed esteem structures one of the pieces in Cp2, k(i).

3 RESULTS

Through the figure 4, simulation result for a 36-bit output data it is shown that Ak is 12 bits, Pb which is the outcome is 36 bits and y is the 3-bit parity multiplier. Through the bit matrix multiplication of Ak and y the total outcome is observed to be 36 bits.

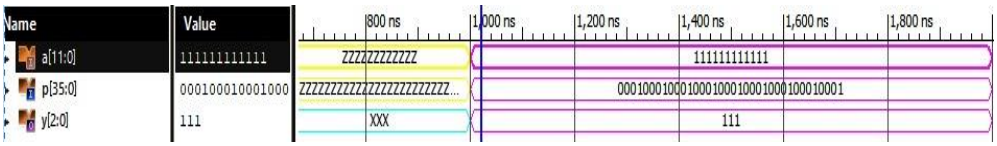

Fig. 5. 36-bit matrix multiplier simulation result.

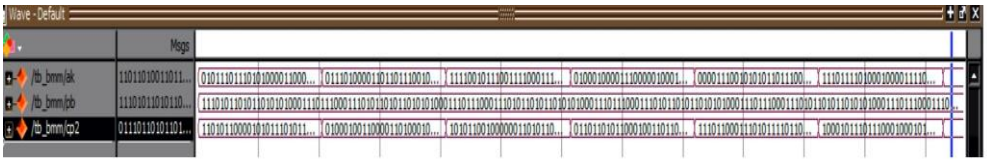

Fig. 6. 478-bit matrix multiplier simulation result.

Similar to that of the 36 bits bit matrix multiplier, for generation of larger number bits can be done by taking the input data bits as 478(Ak), with the help of 32 parity multipliers and 159 multiplier units, where each unit gives a 32-bit output data. Through multiplication of Ak and Pb which is 478x32 we obtain 15296 bits of data through bit matrix multiplication. Here in figure, it is observed that the schematic view of the design of 478 bits which proves that similarly generation of smaller or larger bits can be executed which can be used in the real time applications such as error free coding. The synthesis results of the Bit matrix Multiplier obtained from Synopsys tool. The figure 7 specifies the mapped gate level netlist-based schematic for bit matrix multiplier using 90nm technology.

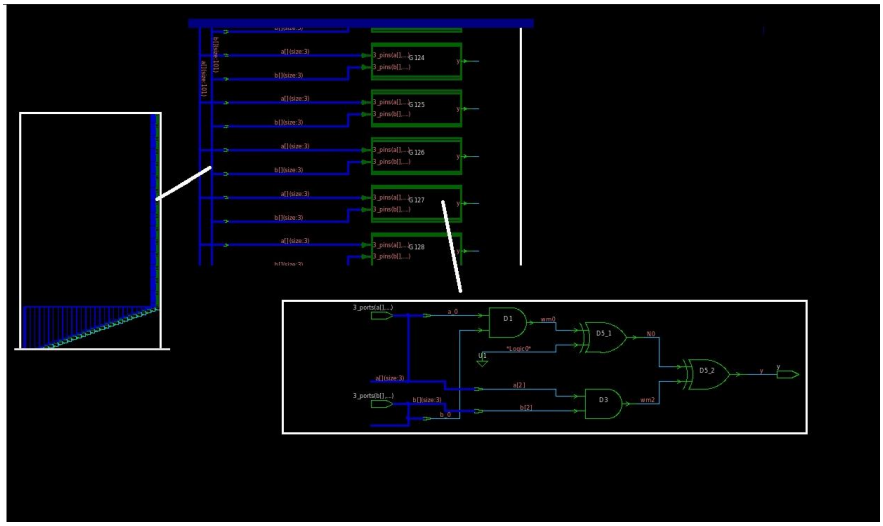

Fig. 7. Internal schematic view of 478-bit matrix multiplier.

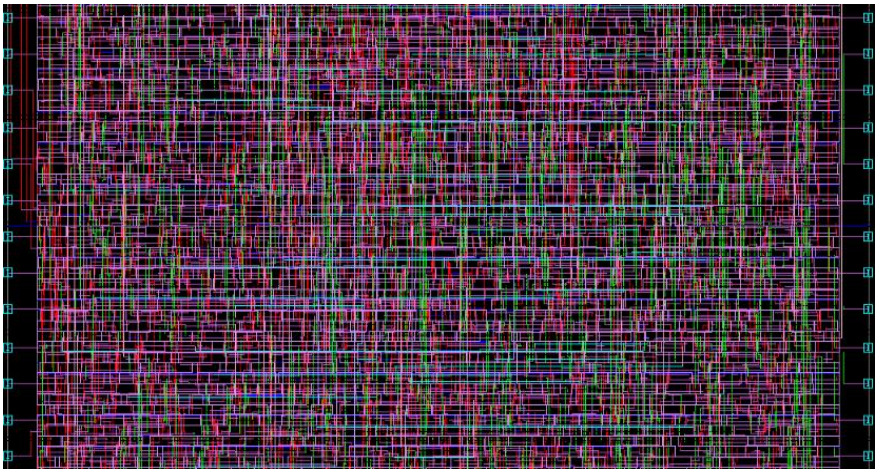

Fig. 8. ASIC design layout of the 478-bit matrix multiplier.

4 CONCLUSION

The Bit Matrix Multiplier operates using combinational logic and does not receive clock input, which results in a reduction in power consumption. We will support an equal flight of stairs encoder strategy to achieve high throughput and low inertness. A Piece framework Multiplier, Spot Network Viper, Equality Age Units, Equality Touch Registers, ROMs, Multiplexer, Regulator, Information Buffer, and Data Formatter are all included in the engineering of a flight of stairs encoder. As a result, we used the Bit-Matrix Multiplier in our project, which is one of the key building elements in the design of staircase encoders and aids in lowering power consumption. As the Bit-Matrix Multiplier is used, power consumption is lowered. It helped with the design of a forward error correction (FEC) encoder with high throughput, low dormancy, and power. The multistage pipelined design of this encoder enables excellent effectiveness in terms of throughput and region. The area consumed by the bit matrix

multiplier given by generating the area report is 195618.33mm^2 with 82174 ports and 34 sequential cells. The power generated through the design compiler obtained is a total power of 38.68mW with a operating voltage of 1.2V and leakage power of 620.43uW. FEC increases the throughput of the functional framework while reducing the number of transmission errors and the power requirements for correspondence systems. To use less electricity, the flight of stairs encoder's control theory can be further developed.

References

[1] Dou Yong & Vassiliadis, G.N. Gaydadjiev, G.K. Kuzmanov, "*64-bit floating-point FPGA matrix multiplication"*, Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays, 86-95 (2005).

[2] Al-Ghuribi, Sumaia & Thabit Khalid, *Matrix Multiplication Algorithms*", International Journal of Computer Network and Information Security. 12, 74-79 (2012)

[3] Sandhya Rai1, Prof. Suresh. S. Gawande2, "*Survey of Matrix Multiplication using IEEE 754 Floating Point for Digital Image Compression*", *International Journal Of Innovative Research in Science, Engineering and Technology,* Vol 8 Issue 9 September, (2019).

[4] R. Tanner, "*A recursive approach to low complexity codes,"* IEEE Trans. Inf. Theory, vol. IT-27, no. 5, pp. 533–547, Sep. (1981).

[5] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov, "*Iterative decoding threshold analysis for LDPC convolutional codes," IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. (2010).

[6] C. Fougstedt and P. Larsson-Edefors, "*Energy-efficient high-throughput VLSI architectures for product-like codes*," J. Lightw. Technol, vol. 37, no. 2, pp. 477–485, Jan. 15, (2019).

[7] Y. Cai, W. Wang, W. Qian, J. Xing, K. Tao, J. Yin, S. Zhang, M. Lei, E. Sun, H. Chien, Q. Liao, K. Yang, and H. Chen, "*FPGA Investigation on Error-Flare Performance of a Concatenated Staircase and Hamming FEC Code for 400G Inter-Data Center Interconnect*," J. Lightwave Technol. 37, 188-195 (2019).

[8] L. M. Zhang and F. R. Kschischang, "*Low-complexity soft-decision concatenated LDGM-staircase FEC for high-bit-rate fiber-optic communication*," J. Lightw. Technol., vol. 35, no. 18, pp. 3991–3999, Sep. 15, (2017).

[9] M. Barakatain and F. R. Kschischang, "*Low-complexity concatenated LDPC-staircase codes*", J. Lightw. Technol., vol. 36, no. 12, pp. 2443–2449, Jun. 15, (2018).

[10] Y. Hilewitz, C. Lauradoux and R. B. Lee, "*Bit matrix multiplication in commodity processors,"* 2008 International Conference on Application-Specific Systems, Architectures and Processors, Leuven, Belgium, pp. 7-12, d, (2008).

[11] Safonov, I.; Kornilov, A.; Makienko, D. An Approach for Matrix Multiplication of 32-Bit Fixed Point Numbers by Means of 16-Bit SIMD Instructions on DSP. Electronics, 12, 78 (2023).

[12] Qasim, Syed Manzoor & Abbasi, Shuja & AlMashary, Bandar, "*A Survey of FPGA Based Design of Matrix Multipliers: Fixed- and Floating-Point Realizations",* (2008).

[13] By Prabir Saha, Arindam banerjee, partha Bhattacharya, Anup Danapat, "*Improved matrix multiplier design for high-speed digital signal processing applications",* (2013)