Volume 4, Issue 11, November 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Efficient Architecture for Implementation of Discrete Sine Transform

[1]Anjali Sahu[*], [2]Priyabrata Mohapatra, [3]S S Nayak
[1]Department of Physics, Temple City Institute of Technology and Engineering, Khordha, India
[2]Department of Basic Science GIET, Gunpur, India
[3]Department of Physics, Centurian University of Technology and Management, Paralakhemundi, India

Abstract— ***This paper presents an efficient architecture for implementation of the discrete sine transform (DST) which realizes the input data bit-serially. The DST equation is split into a few groups of equations by using some mathematical techniques and index tables are constructed to facilitate efficient data permutations. A new formulation of DST is then derived. We represent the sine co-efficient in binary form and realize multiplications using a new series-parallel multiplier architecture, which results in a simple structure for VLSI implementation.***

Keywords— ***DCT, DST, VLSI***

I. INTRODUCTION

Amongst the orthogonal transforms, the discrete cosine transforms (DCT)[1] and the discrete sine transform(DST)[2] are the most popular transforms and considered as the best substitute of the Karhunen-Loeve transform(KLT), not only for their near optimal performance but also for computational convenience. Many algorithms have been proposed for realization of the DCT and DST. These algorithms can be classified in to following categories: 1. Butterfly structure[3-6], 2. Convolution based implementation [7,8], 3. Bit- serial and bit-parallel data structure realization[9,10], 4. Systolic array implementation [11,12] and 5. Recursive filter implementation [13,14].
The butterfly structures involve irregular architectures and complicated routing. The convolution based implementations require some pre processing steps. The bit- serial and bit parallel data structure realizations require memory look-up tables. The systolic array implementations require large number of processing elements for the VLSI chip. The recursive filter structures are suitable for the parallel and pruning realizations.
Various parallel multiplication algorithms [16] have been proposed to meet the demand for high speed. The structures require large amount of hardware and consume large amount of power. To achieve hardware simplicity, shift-add method for multiplication is used[17] which results in slow speed.
In this paper, we propose a new formulation for the DST and implement the inner product by a proposed serial-parallel multiplier. The resulting structure is simple and most suitable for VLSI implementation.

II. DERIVATION OF THE DST ALGORITHM

In this section, we derive a new formulation for the DST, which is suitable for implementation using serial-parallel multipliers. The DST of a sequence $\{x_0(i)$: i=0, 1, …….,$N)$ can be written as

$$Y(k) = \sum_{i=1}^{N} x(i) \sin\frac{(2i-1)\pi k}{2N} \tag{1}$$

` for $K = 1, 2, 3,\ldots\ldots\ldots\ldots\ldots, N$.

The DST can be split into n groups of equations for the 2^n point DST. A general formulation is given in eqn. 2, where m is the group number starting from zero to $n-1$.

$$Y\left(2^m(2r+1)\right) = \sum_{i=1}^{\frac{N}{2^{m+1}}} \tilde{x}_{m+1}(i) \sin\left[(2i-1)2^m \frac{(2r+1)}{2N}\pi\right] \tag{2}$$

Where

$$x_{m+1}(i) = x_m(i) - x_m\left(\frac{N}{2^m}+1-i\right) \tag{3}$$

$$\tilde{x}_{m+1}(i) = x_m(i) + x_m\left(\frac{N}{2^m}+1-i\right) \tag{4}$$

We use an example with $N = 8$ here to clarify this idea. eqn. 2 represents different groups of equations for the realization, i.e.

$$Y(8) = x_3(1), \tag{5}$$

$$Y(4) = \sin\left(\frac{4\pi}{16}\right)\tilde{x}_3(1) \tag{6}$$

$$\begin{pmatrix} Y(2) \\ Y(6) \end{pmatrix} = \begin{pmatrix} \sin\frac{2\pi}{16} & \sin\frac{6\pi}{16} \\ \sin\frac{6\pi}{16} & -\sin\frac{2\pi}{16} \end{pmatrix} * \begin{pmatrix} \tilde{x}_2(1) \\ \tilde{x}_2(2) \end{pmatrix} \tag{7}$$

$$\begin{pmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{pmatrix} = \begin{pmatrix} \sin\frac{\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{5\pi}{16} & \sin\frac{7\pi}{16} \\ \sin\frac{3\pi}{16} & \sin\frac{7\pi}{16} & \sin\frac{\pi}{16} & -\sin\frac{5\pi}{16} \\ \sin\frac{5\pi}{16} & \sin\frac{\pi}{16} & -\sin\frac{7\pi}{16} & \sin\frac{3\pi}{16} \\ \sin\frac{7\pi}{16} & -\sin\frac{5\pi}{16} & \sin\frac{3\pi}{16} & -\sin\frac{\pi}{16} \end{pmatrix} * \begin{pmatrix} \tilde{x}_1(1) \\ \tilde{x}_1(2) \\ \tilde{x}_1(3) \\ \tilde{x}_1(4) \end{pmatrix} \tag{8}$$

Obviously, the multiplicands and multipliers in equation 6-8 are signed numbers. Our objective is to compute the matrix multiplication in an effective way. Eqn. 8 is converted into a new matrix equation 9, such that the new formulation will result in easier hardware design.

$$\begin{pmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{pmatrix} = \begin{pmatrix} \tilde{x}_1(1) & \tilde{x}_1(2) & \tilde{x}_1(3) & \tilde{x}_1(4) \\ \tilde{x}_1(3) & \tilde{x}_1(1) & -\tilde{x}_1(4) & -\tilde{x}_1(2) \\ \tilde{x}_1(2) & \tilde{x}_1(4) & \tilde{x}_1(1) & -\tilde{x}_1(3) \\ -\tilde{x}_1(4) & \tilde{x}_1(3) & -\tilde{x}_1(2) & \tilde{x}_1(1) \end{pmatrix} * \begin{pmatrix} \sin\frac{\pi}{16} \\ \sin\frac{3\pi}{16} \\ \sin\frac{5\pi}{16} \\ \sin\frac{7\pi}{16} \end{pmatrix} \tag{9}$$

The computation of eqn.9 requires sixteen multiplications. Our purpose is to minimize the implementation effort needed to find the multiplication results. Input data multiplexing techniques are used to reduce the multiplications. Now only four multipliers, namely $\sin\frac{\pi}{16}$, $\sin\frac{3\pi}{16}$, $\sin\frac{5\pi}{16}$ and $\sin\frac{7\pi}{16}$ are required to compute sixteen multiplications. The inputs are multiplexed with their corresponding multiplication sequences as shown in eqn.9. Let us represent the multipliers $\sin\frac{\pi}{16}$, $\sin\frac{3\pi}{16}$, $\sin\frac{5\pi}{16}$ and $\sin\frac{7\pi}{16}$ in the binary form.

$$\sin\frac{\pi}{16} = 0.00110 \tag{10}$$

$$\sin\frac{3\pi}{16} = 0.100011 \tag{11}$$

$$\sin\frac{5\pi}{16} = 0.110101 \tag{12}$$

$$\sin\frac{7\pi}{16} = 0.1111111 \tag{13}$$

With the above binary representation, the product of $\tilde{x}(i)$ and $\sin\left(\frac{(2i+1)\pi}{16}\right)$ can easily be computed by using the proposed serial-parallel multiplier. The addition and subtraction operations that follow the multiplications, as shown in eqn. 9, can therefore be implemented by using serial adders and subtractors.

III. PROPOSED SERIAL PARALLEL MULTIPLIER

In general the multipliers for sine-bit numbers are complicate and require more hardware compared to multipliers using unsigned bit. So, in this paper we proposed Zero-MSB-of-Multiplier (ZMM) scheme of multiplication of sine-bit numbers to achieve considerable simplicity in architecture in addition to less hardware requirement and high throughput. In this scheme, the MSB of the multiplier should be ‘0’.If the MSB of the multiplier is ‘1’, the number is converted into two’s complement form before feeding serially into the multiplier . Again, if the MSB of the multiplier is ‘0’ the product is obtained directly. However, if MSB of the multiplier is ‘1’, the output from the multiplier is reconverted into two’s complement form to get the real product

IV. MATHEMATICAL BACKGROUND

By multiplying arbitrary n-bittwo’s complement numbers A and B, the product P can be expressed by

$$P = A.B = \left(-2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i\right)\left(-2^{n-1}b_{n-1} + \sum_{j=0}^{n-2} 2^j b_j\right) \tag{14}$$

Since $b_{n-1} = 0$ in ZMM scheme

$$P = -2^{n-1}a_{n-1}\sum_{j=0}^{n-2} 2^j b_j + \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} 2^{i+j} a_i b_j \tag{15}$$

$$= 2^{n-1}\sum_{j=0}^{n-2} 2^j b_j\left(\overline{a_{n-1}b_j} - 1\right) + \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} 2^{i+j} a_i b_j \tag{16}$$

$$= \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} 2^{i+j} a_i b_j + 2^{n-1}\sum_{j=0}^{n-2} 2^{j}\left(\overline{a_{n-1} b_j} - 2^{2n-1} + 2^{2n-2} + 2^{n+1}\right) \qquad (17)$$

Eqn. 15 may be also be expressed as

$$P = -2^{n}\sum_{j=0}^{n-2} 2^{j} a^{n-1} b_j + 2^{n-1}\sum_{j=0}^{n-2} 2^{j} a_{n-1} b_j + \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} 2^{i+j} a_i b_j \qquad (18)$$

Equation 17 and 18 can be easily applied to serial-parallel multipliers.

V. IMPLEMENTATION OF TWO'S COMPLEMENT BIT-LEVEL MULTIPLICATION

Multiplier-1

Equation 17 is implemented in the proposed multiplier-1 shown is fig-3. An example of 4×4 two's complement multiplications of numbers 3 and -5 is shown in fig.1. First the multiplier -5 is converted into +5 according to ZMM scheme and the product of 3 and 5 is implemented in a multiplier

```
        0 0 1 1  (3)
        0 1 0 1  (5)
        ----------
        1 0 1 1
      1 0 0 0
    1 0 1 1
  1 0 0 0
1       1
---------------
0 0 0 0 1 1 1 1  (+15)
```

Fig. 1

```
                0 0 1 1  (3)
                0 1 0 1  (5)
                ----------
            0 * 0 0 1 1
            0   0 0 0
            ----------
        0 * 0   0 0 1 1
        0   0   1 1
------------------------
0 * 0 * 0   0   1 1 1 1
    0   0   0   0
------------------------
0   0   0   0   1 1 1 1  (+15)
------------------------
```

Fig .2

* -sign –bit extension.

Fig. 1 and Fig .2: Examples of two's complement multiplication.

Each bit of multiplicand is multiplied by each bit of multiplier arriving serially from LSB. The last bit of each partial product is complemented. The final product is computed by adding '1' in the fourth column. As four zeros are appended to the left of the MSB of the multiplier, the '1' in the eighth column is obtained by NANDing the fourth bit of the fifth partial product. Since final product could be a 2 n-bit two's complement number, all excessive carry-outs after eighth bit should be ignored. The output of the multiplier is then converted into two's complement from as 11110001 (-15), which is the product of 3 and -5.

The proposed multiplier-1 consists of four units – two conversion units, a logic unit and an adder unit at the input and output consists of a XOR gate and a full adder. Both conversion units are controlled by the signal Q which is '00000000' if MSB of the multiplier is '0' and '11111111' if MSB is '1'. The multiplier bits and the control signal Q are fed simultaneously to the XOR gate of input-conversion unit. However the same control signal Q is fed to the XOR gate of the output-conversion unit staggered by 3 time-steps. Another control signal Q^1 is 00000000 if the MSB of the multiplier is '0' and 00000001 if MSB is '1'. However, Q^1 is fed to the full adder of the output-conversion unit staggered by 3 time-steps. So that if the multiplier is a positive number, it is available as such at the logic gates and the final product is available directly at the output. If the multiplier is a negative number, its two's compliment form is available at the logic gates and the final product is available at the output after two's compliment conversion of multiplier output. The logic unit consists of (n-1) AND gates and one NAND gate. Each bit from conversion unit is available to all the logic gate simultaneously. The bits of the multiplicand are stored at the individual gates. The duration of the clock cycle is T_A, which is the full adder delay, since the gate delay is comparatively smaller. The extra '1' for the fourth column is provided to the right most adder of the adder unit with the help of Q^{11} signal. Four MS bits are appended to the left of MSB of the multiplier for input/output synchronization.

Multiplier -- II

Eqn. 18 is implemented in the proposed multiplier – II, shown in Fig. 4. The first term of the eqn. 18 necessitates sign- bit extension. For multiplication 3 and -5, the multiplier -5 is converted into +5 according to ZMM scheme. The multiplication of 3 and 5 based on sign-bit extension is shown in Fig. 2. The sign – bit extension is effected either as a direct extension of partial product or created as a result of carry generated by the addition of the previous row. The output of the multiplier is converted into two's complement form as 11110001 (-15) which is product of 3 and -5. The proposed multiplier-II has same four units as multiplier-I. the conversion units function in the same manner with the help of same control signals Q and Q^1 as in multiplier-I. However both signals are fed to the XOR gate of the output conversion unit and output adder, respectively, staggered by 2 time-steps. The logic unit consists n AND gates. The OR gate in the left most adder of the adder unit provides necessary sign-bit extension.

VI. ARCHITECTURE

A straightforward implementation of the proposed algorithm using the serial-parallel multiplier method is efficient enough. However, in order to reduce further the hardware complexity, the transform matrix is redefined as shown in eqn. 9. Note that some additions and subtractions are required in the pre-processing as well. Thus the data have to be multiplexed and recorded before being sent to the serial-parallel multipliers.

Figure 5 shows the data pre-processing stage as defined in eqns. 3 and 4. The outputs of this stage are $\tilde{x}_1(i), \tilde{x}_2(i), \dots\dots$, and Y(8) respectively. The input data sequence $x_0(1)$, $x_0(2)$, ... $x_0(N)$ is shifted sequentially from a bit-parallel structure into *N* stage input registers. The data for the input registers are bit-parallelly loaded into the shift register, i.e. S register. The data in the S register are then serially shifted out to the serial adders and subtractors.

The structures of the serial adder are shown in Figure 6. The serial adder adds two numbers bit by bit, and requires *n* clock cycles to complete the adding process for two *n*-bit numbers. The output bit, i.e. Y_{in}, is performed, and the carry –in , i.e.C_{in}, of the full adder is set to one during the first clock cycle to perform the two 's complement subtraction for the inputs as shown in Figure 7.

The second stage is to implement the matrix multiplication a shown in Figure 8. The final stage includes serial adders and output registers. The data are serially shifted into the output registers. The data in the output registers are shifted out by the bit parallel structures.

VII. TIMINING ANALYSIS OF THE PROPOSED PIPELINE STRUCTURE

The DCT equation can be converted into different groups as shown in eqn. 2, which form a full pipeline architecture according their group numbers. Let us use the 8 points DCT again to illustrate the process of this pipeline architecture. Initial addition and subtractions are done as shown in fig. 3, and multiplexing and multiplication are then carried out after the pre-processing stage as shown in fig. 6, hence the output for the group 0 data are obtained sequentially. Similarly, the outputs with group number (m) which is equal to 1 and 2 can be obtained by using a similar hardware architecture, where figure 7 shows the timing diagram of the pipeline architecture for the implementation of the 8 points DCT.

In fig.7, the time axis is the number of clock cycles and P is the pre-processing time before the data are sent to the serial multipliers. M is the data precision, i.e. the word length, of the input data and N is the precision of the multiplier which is in general greater than M.

The maximum operation frequency depends on the longest part between two adjacent flip-flops as shown in fig. 2. The adder, which has five inputs and three outputs, has the longest delay path in the proposed structure. The chip has a full pipeline architecture and can receive 8 data for P+4M+N clock cycles. If we let the pre-processing time be 4 clock cycles, the input data precisions be 8 bits, and the multiplier precision be 11 bits, the resultant outputs are obtained after 80 clock cycles

VIII. CONCLUSION

In this paper, a new multiplier implementation scheme for the discrete cosine transform is proposed. By using various mathematical techniques, a new equation which computes the DST effectively is generated. We have also proposed a serial-parallel multiplier algorithm that can be used to multiply an unsigned number by a two-s complement number .These two numbers have different precisions. But using this novel realization technique, the multiplication can be implemented in an efficient way. An example with request to the realization of an 8 points DST has also been provided to illustrates our design. The resultant design has been verify in software simulation and by using different sets of input sequences.

REFERENCES

[1] Ahmed N., Natarajan T. and Rao K.R., "Discrete cosine transform," IEEE Transactions on Computer, Vol, 23, No. 1, pp. 90-93, Jan. 1974

[2] Jain, A. K. :"A tast Karhunen-Loeve transform for a class of stocastic process", IEEE Trans. on commun., Vol. 24, pp. 1023-1029, 1976.

[3] Lee B.G., "A new algorithm to compute the discrete cosine transform,"IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 32, No. 6, pp. 1243-1245, Dec. 1984

[4] Loeffler C., Ligtenberg A. and Moschytz G.S., "Practical fast 1-D DCT algorithms with 11 multiplications," IEEE International Conference on Acoustics, Speech & Signal Processing, pp.988-991, 1989

[5] Cvetkovic Z. and Popovic M.V., "New fast recursive algorithms fro the computation of discrete cosine and sine transforms," IEEE transactions on signal processing, Vol. 40, No. 8, pp. 2083-2086, Aug. 1992

[6] Skodras A.N. and Christppoulous C.A, "Split-radix fast cosine transform algorithm," International journal of Electronics, Vol. 74, No. 4, pp. 513-522, 1993

[7] Duhamel P. and H'Mida H., "New 2nd DCT algorithms suitable for VLSI implementation,"Proceedings, IEEE International Conference on Acoustics, Speech & Signal Processing, pp.1850-1808, 1987

[8] Chan Y.H. and Siu W.C., "On the realization of discrete cosine transform using the distributed arithmetic," IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 39, No. 0, pp. 705-712, Sept. 1992

[9] Sun M.T., Wu L., "A concurrent aerchitecture for VLSI implementation of discrete cosine transform," IEEE Transactions on Circuits and Systems, Vol. 34, No. 8, pp. 992-994, 1987

[10] Sun M.T., Chen T.C. and Gottileb A.M., "VLSI implementation of a 16 X 16 discrete cosine transform," IEEE Transactions on Circuits and Systems, Vol. 36, No. 4, pp. 610-617, Apr. 1989

[11] Cho N.I. and Lee S.U., "DCT algorithms for VLSI parallel implementations," IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 38, No. 1, pp. 121-127, Jan. 1990

[12] Chan, L.W. and Wu, M.C.:"A unified systolic array for discrete cosine and sine transform", IEEE Trans. on Signal processing, Vol. 39, No. 1, pp. 192-194, Jan. 1991.

[13] Chan Y.H., Chau L.P. and Siu W.C., " Efficient implementation of discrete cosine transform using recursive filter structure," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 4, No. 6, pp. 550-552, Dec. 1994

[14] Wang Z., Jullien G.A. and Miller W.C., "Recursive algorithms for the forward and invderse discrete cosine transform with arbitrary length," IEEE signal processing letters, Vol. 1, No. 7, pp.101-102, Jul. 1994

[15] Chau L.P. and Siu W.C.," Direct formulation for the realization of discrete cosine transform using recursive structure, "IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal processing, Vol. 42, No. 1, pp.50-52, Jan. 1995

[16] Ma G. K. and Taylor F.J., "Multiplier policies for digital signal processing," IEEE ASSP Magazine, Vol. &, pp. 6-20, Jan. 1990

[17] White S.A., "Applications of distributed arithmetic to digital signal processing: A tutorial review," IEEE ASSP magazine, Vol. 6, pp. 4-19, 1989

[18] Sunder S., EI-Guibaly F. and Antaniou A., "Two's-compliment fast serial-parallel multiplier," IEE Proceedings Circuits, Devices and Systems, Vol. 142, No. 1, pp. 41-44, Feb. 1995

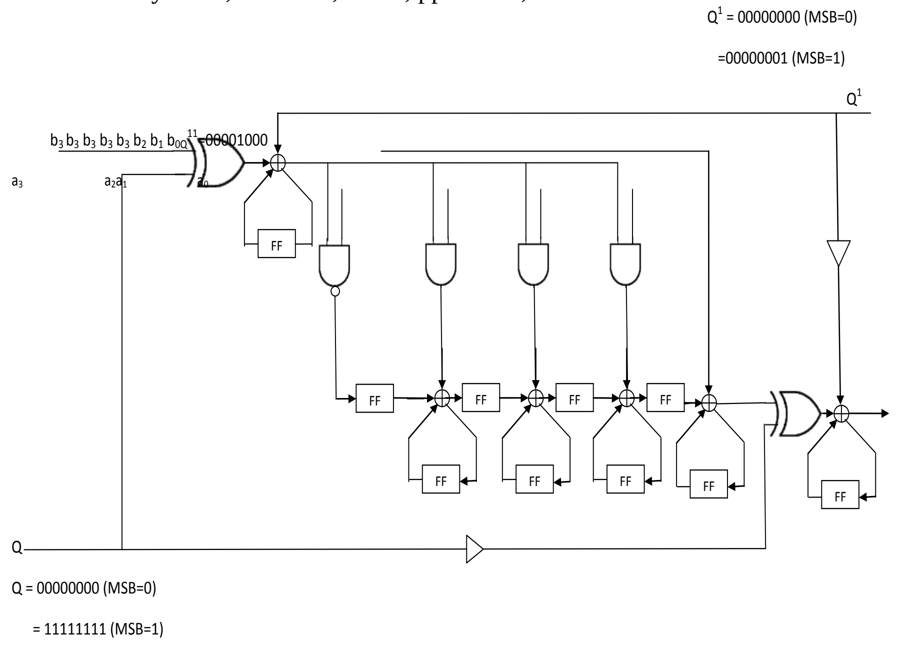

Fig. 3 Multiplier - I

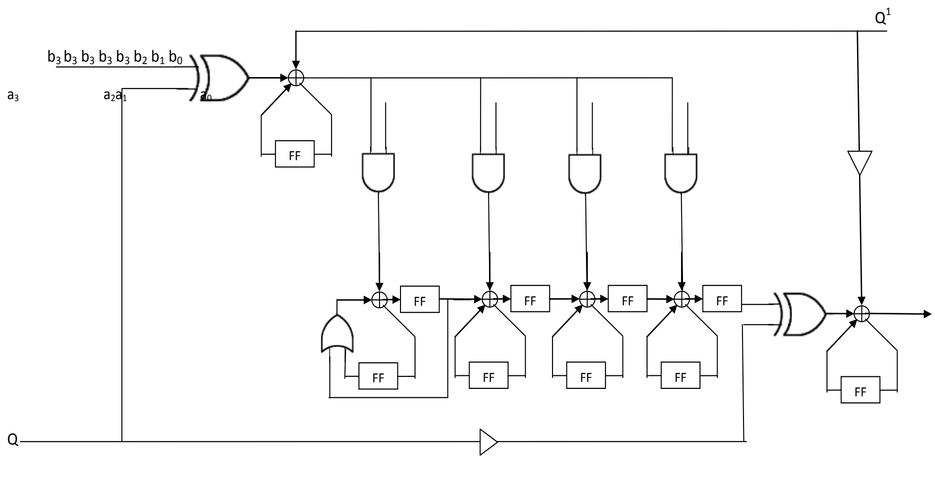

Fig. 4 Multiplier - II

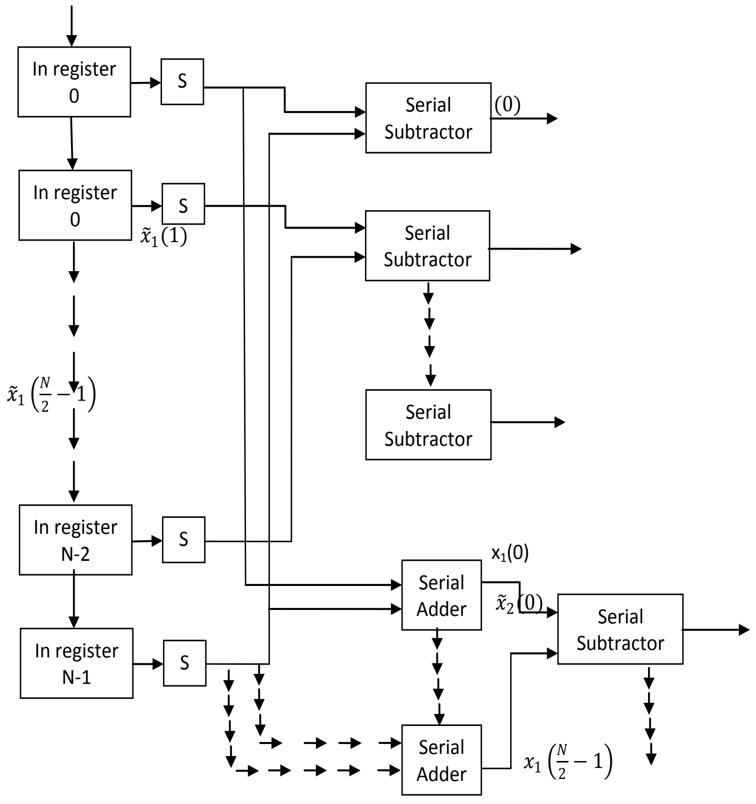

Figure 5: Data pre-processing stage of the proposed algorithm

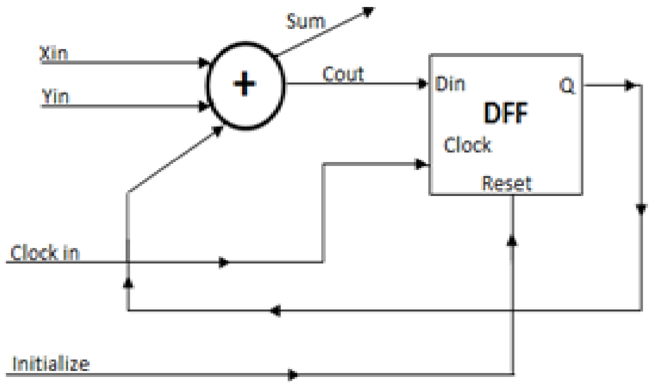

Figure 6: Serial adder

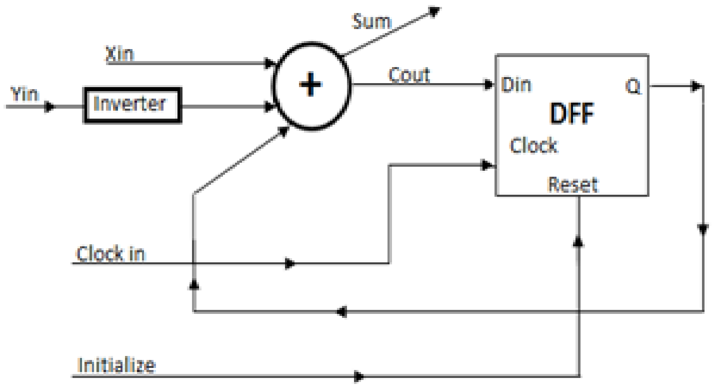

Figure 7: Serial subtractor

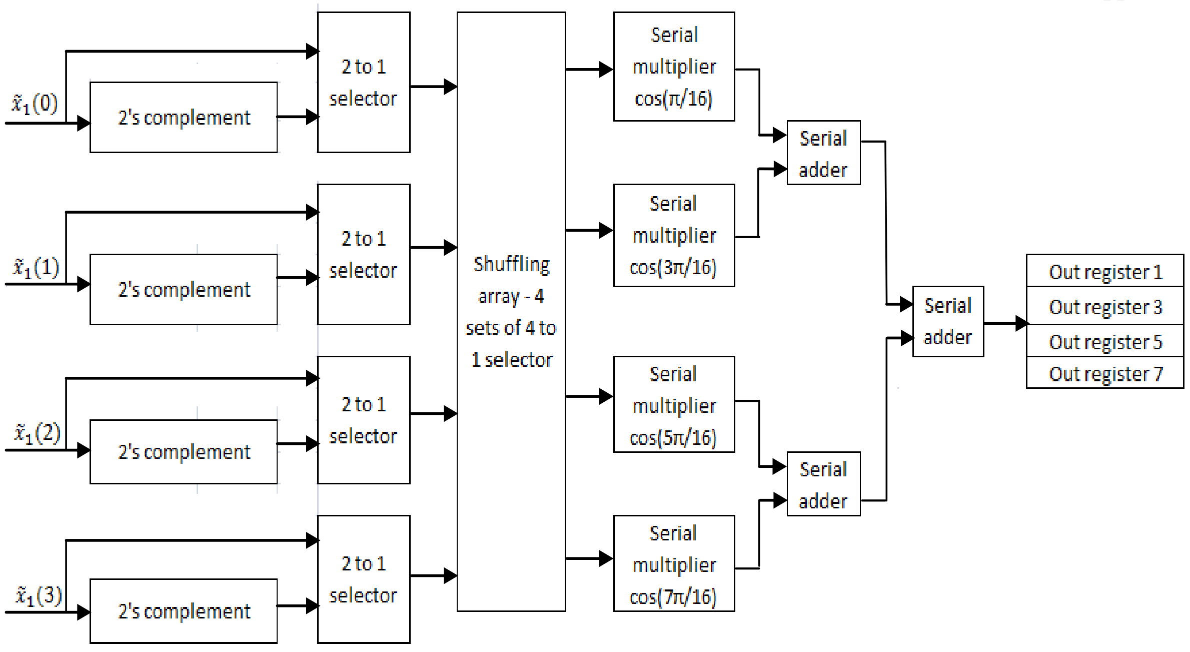

Figure 8: Block diagram for implementation of the matrix multiplication for eqn. 14

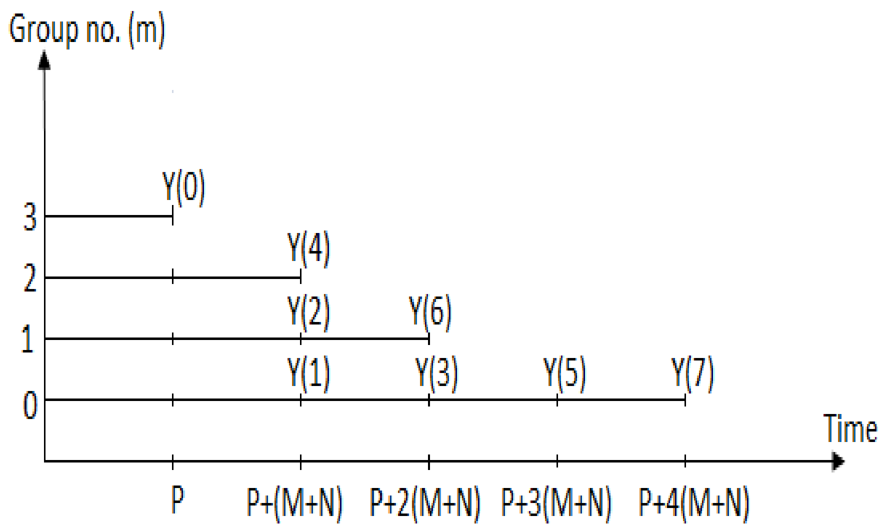

Figure 9 : Timing diagram for the proposed pipelined architecture