DESIGN OF DELAY COMPUTATION METHOD FOR CYCLOTOMIC FAST FOURIER TRANSFORM

Tejaswini Deshmukh[1], Prashant Deshmukh[2] Monica Kalbande[3] and Pravin Dakhole[4]

[1, 3&4]Department of Electronics Engineering, Y.C.C.E Nagpur University, India

[2]Department of Electronics, GCOE, Nagpur, India

ABSTRACT

In this paper the Delay Computation method for Common Sub expression Elimination algorithm is being implemented on Cyclotomic Fast Fourier Transform. The Common Sub Expression Elimination algorithm is combined with the delay computing method and is known as Gate Level Delay Computation with Common Sub expression Elimination Algorithm. Common sub expression elimination is effective optimization method used to reduce adders in cyclotomic Fourier transform. The delay computing method is based on delay matrix and suitable for implementation with computers. The Gate level delay computation method is used to find critical path delay and it is analyzed on various finite field elements. The presented algorithm is established through a case study in Cyclotomic Fast Fourier Transform over finite field. If Cyclotomic Fast Fourier Transform is implemented directly then the system will have high additive complexities. So by using GLDC-CSE algorithm on cyclotomic fast Fourier transform, the additive complexities will be reduced and also the area and area delay product will be reduced.

KEYWORDS

Cyclotomic Fast Fourier Transform, Gate Level Delay, Common Sub Expression Elimination, Critical Path Delay

1. INTRODUCTION

The Discrete Fourier Transform is the most important discrete transform, used to perform Fourier analysis in many practical applications. Since it deals with a finite amount of data, it can be implemented in computers by numerical algorithms or even dedicated hardware. These implementations usually employ efficient Fast Fourier transform (FFT) algorithms. Fast Fourier transforms are widely used for many applications in engineering, science, and mathematics. The cyclotomic fast Fourier transform (CFFT) is a type of fast Fourier transform algorithm over a finite field. The purpose of the cyclotomic fast Fourier transform is to decompose a discrete fast Fourier transform (DFT) into a several circular convolution and from the result of circular convolution, the result of discrete Fourier transform is obtained [3]. The cyclotomic fast Fourier transform are of great interest due to their low multiplicative complexity and this the advantage of CFFT where as it have very high additive complexity and this a issue with CFFT. Cyclotomic fast Fourier transform is being obtained by the matrix and vector product [3]. The DFT have a widespread application in error correction coding used in digital communication and storage system. And therefore, Reed-Solomon code can be constructed using cyclotomic fast Fourier transform. The CFFT is extensively used in cryptography applications and channel coding.

A gate level delay computing method has been applied to the existing common sub-expression elimination algorithm. In this the CSE algorithm is combined with the delay computing method and is known as Gate Level Delay computing with Common Sub expression Elimination (GLDC-CSE) [1]. In this paper the GLDC-CSE algorithm is being implemented on cyclotomic fast Fourier transform. If CFFT is implemented directly then the system will have high additive complexities. So by using gate level delay computing with common sub-expression (GLDC-CSE) algorithm on cyclotomic fast Fourier transform (CFFT), the additive complexities will be reduced and also the size of hardware will be reduced. Multiplication with a constant is commonly used in digital signal processing (DSP) circuits, such as digital filters and is called as Multiple Constant Multiplications (MCM). The main focus in a MCM problem is to find a configuration with the minimal number of adders and subtracters. To solve the MCM problem i.e. to reduce the adders a common sub-expression elimination method (CSE) has been adopted. The main idea of the CSE algorithm is to identify common sub-expressions, which appear more than once, and to replace them with a single variable [1]. With this replacement, each of these patterns need only be computed once, thereby reducing the area of the hardware required in VLSI implementation. Despite the large number of CSE algorithms proposed for minimizing the area, there are few algorithms that deal with both the area and the delay. Method has been proposed to determine the critical path delay (CPD) using restriction graph method [5] but the disadvantage is that it has complicated operations. A gate level delay computing method has been applied to the existing common sub-expression elimination algorithm. In this the CSE algorithm is combined with the delay computing method and is known as GLDC-CSE [1]. The gate level delay computing with common sub-expression elimination (GLDC-CSE) algorithm is being used as the running time of this algorithm is less and also the delay of the system is reduced.

As mentioned earlier that there are not much method to determine the critical path delay with the adders of the circuit. One of the methods proposed to determine both critical path delay as well as adder of the circuit is the 'restriction graph' method. Restriction Graph is translated into XOR tree which divided into fixed vertices and flexible vertices. It has two algorithms

Shrink Graph- To remove flexible vertices

Minimum Delay- To determine the minimum feasible CPD

CSE with restriction graph delay have been proposed (i.e DACSE algorithm) to determine area and delay of the architecture. The restriction graph method has very complicated operation for the implementation which is not that suitable with computers. The implementation of DACSE algorithm mainly depends upon condition. The saving in additive complexity and whether the given CPD requirement is satisfied. A transformation is carried out only when both conditions are met. If the condition is not satisfied than optimized area and delay will not be obtained.

2. GATE LEVEL DELAY COMPUTATION ALGORITHM

Establish an initial delay matrix based on the primitive constant matrix before running a CSE algorithm. First we have to consider a constant matrix M, in which the rows of the matrix will be the output variable and the columns of the matrix will be the input variable. The row of matrix M will be represented as'1' which means the participation of the input variable for the computation of output variable, where as '0' represents there is no participation of the input variable. Now for the initial delay matrix M_d, the input variables will be represented by '0'. The initial delay matrix is being obtained by the formula as

$$M_d = M - M_1$$

Where, M_1 is the matrix with all the elements are '1'. In initial delay matrix '-'will be considered as invalid value. The algorithm consist of two main steps

1) Update delay matrix during a CSE algorithm
2) Compute delay value based on the delay matrix and determine the CPD.

2.1. Update delay matrix during a CSE algorithm

The constant matrix *M* has been changed during the running period of a CSE algorithm, therefore the delay matrix should be updated accordingly. During a CSE algorithm running, suppose that a pattern "$x_i+x_j+\ldots+x_k$" is replaced by a new variable "x_{new}" and appended to the bottom of constant matrix as an additional row. Suppose the delay value of "x_{new}" is represented by "d_{new}" and the delay value of "x_i", "x_j" , ... , and "xk" is represented by "d_i", "d_o" , ... , and "d_k"

respectively. Correspondingly, an additional row should be appended to the bottom of delay matrix according to the "d_i", "d_j" , ... , and "d_k". Then "*dnew*" can be figured out by running algorithm 1 according to the additional row of delay matrix.

Algorithm 1 : Compute delay value

a) Appending a row of delay value d_0, d_1 ,……,d_n ($n\geq 2$).
b) Sort the rows in increasing order.
c) Select the smallest two positive delay value d_1 and d_{i+1}.
d) The new value of d_i and d_{i+1} is being updated as

$$d_i = -1,$$

$$d_{i+1} = \max(d_i, d_{i+1})+1;$$

e) The steps from b-d are repeated until there is one positive value in that row.

Algorithm 2 : Update delay value

a) Append a new row according to transformation of constant matrix M.
b) Append a new column with d_{new} according to transformation of constant matrix M.
c) Detect the row number of position of d_{new} if the row number 'l' is greater than "m-1" of original matrix.
d) Repeat step 3, until there is no row-number of position of d_{new} is greater than "m-1".

2.2. Compute delay value based on delay matrix and determine the CPD

After running the CSE algorithm, the delay value of each output variable can be determined by running Algorithm 1 for first *m* rows of delay matrix. As the GLDC-CSE works simultaneously with CSE, so before performing GLDC-CSE first the CSE is being implemented on the circuit.

3. COMMON SUB EXPRESSION ELIMINATION ALGORITHM

Common sub expression elimination (CSE) is an optimization method that search for happening of identical expressions and replacing them with a single variable that holds the computed value. With this, the identical expressions will be computed only once and hence leading to a savings in terms of the hardware required. CSE algorithm [1] involves the following steps

1) Identify patterns (common factors) that are present in the transformation
2) Select a pattern for elimination.
3) Remove all the occurrences of the selected pattern.

4) The eliminated pattern is computed only once.
5) Repeat Step 1-4 until none of multiple patterns is present.

3.1 Illustration of GLDC-CSE

Consider random matrix for GLDC CSE illustration

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Now GLDC method will be applied to the matrix M.
1) Set up the initial delay matrix

$$Md1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$Md1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & -1 & 0 & -1 \\ -1 & -1 & 0 & -1 & -1 \end{bmatrix}$$

2) Update Delay Value

In this step, mainly the algorithm 1 and algorithm 2 has the importance. This step is dependent on both the algorithms.

Algorithm 1 —> For computing the delay value
Algorithm 2 —> For updating delay value

Considering the matrix Md1

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & -1 & 0 & -1 \\ -1 & -1 & 0 & -1 & -1 \end{bmatrix}$$

Considering the first pattern elimination

X5 = X0 + X1 + X3 + X4

Now on the initial delay matrix delay value can be computed using algorithm 1
Appending the row to the bottom of Md1 matrix which is used for elimination, in this case

X0 + X1 + X3 + X4 will be eliminated

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & -1 & 0 & -1 \\ -1 & -1 & 0 & -1 & -1 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

[1] First the new row appended will be taken

0 0 -1 0 0

[2] Sorting the row in ascending order

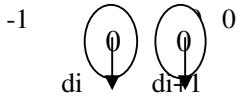
-1 0 0 0 0

[3] Smallest two positive value will be selected and the new values will be updated as;

di = -1

di+1 = max(0,0)+1 = 1

[4] Update the new value of di and di+1



The row will become

-1

-1	-1	1	0

Now repeat steps 2-4 until there is only the last positive delay value in the matrix. After repeating the steps the row will become

-1 -1 -1 -1 -1 -1 2

So, in 1st pattern elimination, the matrix will become

$$\begin{bmatrix} -1 & -1 & 0 & -1 & -1 & 2 \\ -1 & -1 & -1 & -1 & -1 & 2 \\ 0 & 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 & -1 & -1 \\ 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}$$

After this algorithm 1 will be applied

Now in the above matrix row no 1 and 2 is dnew. The rows of matrix M 'm=5', but according to this algorithm if the row no 'l' is greater than (m-1) than the matrix will be recomputed and re-updated. In the first iteration both row no 1 and 2 is smaller than m-1 = 4, so need to recomputed and re-update.

2nd pattern ▶ X0 + X1 + X3, it is replaced by 'X0' and the matrix will be

$$\begin{bmatrix} -1 & -1 & 0 & -1 & -1 & 2 \\ -1 & -1 & -1 & -1 & -1 & 2 \\ 0 & 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 & -1 & -1 \\ 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

By applying algorithm 1 to the new appended row, the row will become

-1 -1 -1 -1 -1 -1 2

Updating the delay value in the matrix, the matrix will become

$$\begin{bmatrix} -1 & -1 & 0 & -1 & -1 & 2 & -1 \\ -1 & -1 & -1 & -1 & -1 & 2 & -1 \\ 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 2 \\ -1 & -1 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 0 & -1 & 2 \\ 0 & 0 & -1 & 0 & -1 & -1 & -1 \end{bmatrix}$$

In the above matrix, it can be seen that row no '3' and row no '5' have the delays value. From which the row no is greater than 'm-1=4' so that row will be recomputed and re-updated.

For-computing algorithm 1 will be applied to the row which is greater than ‘m-1=4’. In this case algorithm 1 will be applied to row no ‘5’.

After applying algorithm 1 to the row will become

-1 -1 -1 -1 -1 -1 3

The new delay value is ‘3’. Therefore the matrix will be updated as

$$\begin{bmatrix} -1 & -1 & 0 & -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 2 \\ -1 & -1 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 0 & -1 & 2 \\ 0 & 0 & -1 & 0 & -1 & -1 & -1 \end{bmatrix}$$

Similarly, for remaining elimination pattern both algorithm 1 and algorithm 2 will be applied in the similar way as shown above.

So, the final matrix obtained after implementing both the algorithms is as follows

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & 4 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 3 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 2 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

In the above matrix last column calculates critical path delay and select highest value as a delay value.

3.2 GLDC-CSE Implementation on CFFT

In this paper the GLDC-CSE algorithm is being implemented on cyclotomic fast Fourier transform (CFFT). If CFFT is implemented directly then the system will have high additive complexities. So by using gate level delay computing with common sub-expression (GLDC-CSE) algorithm on cyclotomic fast Fourier transform (CFFT), the additive complexities will be reduced and also the size of hardware will be reduced.

The Fourier transform of a polynomial from [3]

$$f(x) = \sum_{i=0}^{n-1} f_i x^i$$

of degree f(x) = n−1, n | ($2^m - 1$),in the field GF(2^m) is the collection of elements,

$$F_j = f(\alpha^j) = \sum_{i=0}^{n-1} f_i \alpha^{ij}, j \in (0, n-1) \qquad j \in [0, n-1]$$

where α is an element of order n in the field GF (2m). The method used to design the architecture of FFT in this paper is decomposition of polynomials into a sum of linearized polynomials [2] , [6]. The polynomials are evaluated at a set of basis points.

The above equation can be represented in matrix form as, F = A L f. The multiplication Lf represents a set of cyclic convolutions [1] of each block Li by the corresponding cyclotomic coset of the vector f. Equation F = Alf can be rewritten as

F=AQ(C.(PF))

The given FFT algorithm is competent for small values of the DFT, while more efficient FFT algorithms for finite fields are known in the original field [4].The implementation of GLDC-CSE is applied on AQ matrix for CFFT GF (23) is

$$M = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

After applying GLDC CSE output matrix obtained is

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 2 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 3 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 3 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 3 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 3 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 3 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 3 \end{bmatrix}$$

Therefore critical path delay for GF (2^3) is 3. Similarly CPD will be calculated for various GF(2^m) elements.

4. RESULTS AND CONCLUSION

Gate Level Delay Computing method is used to determine the critical path delay in hardware implementation of constant matrix multiplication over Galois Field GF (2^m). Gate level delay computing method (GLDC) is being implemented simultaneously with common sub-expression elimination known as GLDC-CSE. GLDC-CSE is being implemented only when CSE is carried out for the circuit, if both area as well as delay is to be calculated. If only delay is to be determining of any circuit than only GLDC can be implemented. Here critical path delay is to determined using GLDC method for CFFT without CSE and CFFT with existing CSE.

Table 1. Critical Path Delay

CFFT for GF (2^m)	DFT	GLDC without CSE	GLDC with CSE	Elapsed time
2^3	7	3	3	0.002244 s
2^4	15	5	6	0.008139 s
2^5	31	6	8	0.034215 s
2^6	63	8	10	0.208692 s
2^7	127	10	11	1.656700 s

From table1 it is seen that the CPD of GLDC-CSE algorithm is increased this is because the algorithm computes reduced area as well as delay for the input matrix. In GLDC – CSE adders get reduced so the design becomes more complex and hence critical path delay increases. But area of the design gets reduced. The GLDC-CSE algorithm is implemented in MATLAB 2013 and evaluated its performance for various Galois field elements. Table 1 indicates critical path delay of GLDC without CSE and GLDC with CSE for various GF (2^m) CFFT.

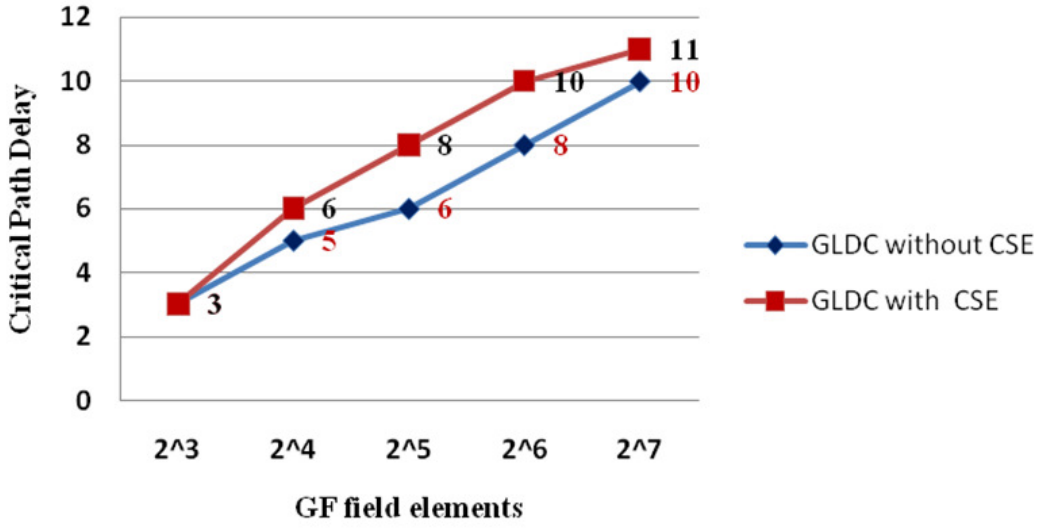


Figure1. Critical path delay comparison of Cyclotomic Fast Fourier Transform

Figure 1 presents critical path delay comparison between GLDC without CSE and GLDC with CSE of CFFT. Computation of area delay product categories into two parts. First part is of CFFT without CSE. Second part is of CFFT with GLDC CSE.

Table 2. Area Delay Product of CFFT without GLDC-CSE

CFFT for GF (2m)	Area without CSE	GLDC without CSE	Area Delay Product
2^3	46	3	138
2^4	278	5	1390
2^5	1169	6	7014
2^6	10366	8	82928
2^7	16686	10	166860

For all categories area delay product is being calculated Table 2 shows the computation of Area delay product of the CFFT without using CSE algorithm for various GF (2^m).

Table 3. Area Delay Product of CFFT with GLDC existing CSE

CFFT for GF (2m)	Area with existing CSE	GLDC with existing CSE	Area Delay Product
2^3	35	3	105
2^4	151	6	906
2^5	561	8	4488
2^6	870	10	8700
2^7	4253	11	46783

Table 3 shows Area delay product of CFFT with GLDC- CSE algorithm. The GLDC – CSE algorithm can determine critical path delay and reduced adders of cyclotomic fast Fourier transform. The percentage of area reduction is mentioned in figure 2.

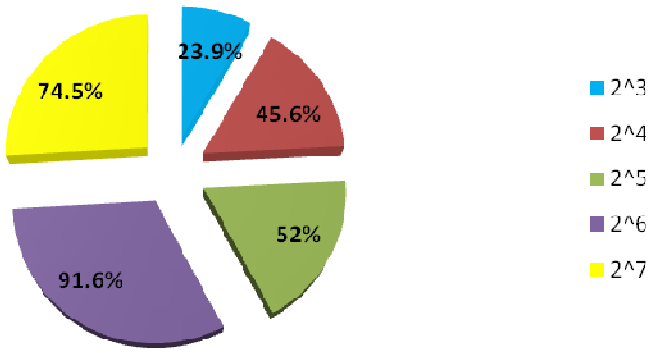


Figure 2. Percentage of area reduction

Table 4. Area Delay Product of comparison for GF (2^m) CFFT

CFFT for GF (2m)	N	ADP of CFFT without CSE	ADP of CFFT with existing CSE
2^3	7	138	105
2^4	15	1390	906
2^5	31	7014	4488
2^6	63	82928	8700
2^7	127	166860	46783

Area delay production (ADP) of cyclotomic fast Fourier transform using GLDC CSE is reduced and it is presented in table 4.

ACKNOWLEDGEMENT

The authors would like to thank Ning Wu, Xiaoqiang Zhang, Yunfei Ye and Lidong Lan for providing details of Gate Level Delay Computation and Common Sub expression Elimination Algorithm. They are grateful to Prof. P.Trifonov and Ali AL GHOUWAYEL for providing details of CFFTs.

REFERENCES

[1] Ning Wu, Xiaoqiang Zhang, Yunfei Ye, and Lidong Lan .Improving Common subexpression Elimination Algorithm with A New Gate-Level Delay Computing Method. Proceedings of the World Congress on Engineering and Computer Science. Vol II 23-25 October , San Francisco, USA, 2013.

[2] Fedorenko, S.V. and Trifonov, P.V. Finding Roots of Polynomials over Finite Fields. IEEE Trans. Commun , 2002, vol. 50 , no. 11 , pp. 1709-1711.

[3] S. V. Fedorenko and P. V. Trifonov . A method for Fast Computation of the Fast Fourier Transform over a Finite Field. Problems of Information Transmission, 39(3):231-238, July-September 2003.

[4] Y. Wang and X. Zhu. A fast algorithm for the Fourier transform over finite fields and its VLSI implementation. IEEE J. Sel. Areas Commun., vol. 6, no. 3, pp. 572–577.

[5] N. Chen, and Z. Y. Yan. High-performance Designs of AES Transformations. IEEE International Symposium on Circuits and Systems (ISCAS 2009), 2009, PP.2906 – 2906.

[6] Truong, T.-K., Jeng, J.-H., and Reed, I.S. Fast Algorithm for Computing the Roots of Error Locator Polynomials up to Degree 11 in Reed–Solomon Decoders. IEEE Trans. Commun., 2001, vol. 49, no. 5, pp. 779 – 783.

AUTHORS

Tejaswini P. Deshmukh, is an Assistant Professor at Yeshwantrao Chavan College of Engineering, Nagpur University. She is currently working towards the Ph.D. degree in Electronics Department of YCCE Research Centre, Nagpur University. Her research interest includes Digital Signal Processing, Digital Communication and Digital Image Processing. Her teaching interest includes Signals and Systems, Digital Signal Processing and Embedded Systems.

Prashant Deshmukh, is a Professor at Gov. College of Engineering Nagpur. He has teaching experience of 7 - 8 years. His research interest includes Signal Processing and VLSI. His teaching interest includes Advanced Digital System Design, and Verification and Testing.

Monica Kalbande, is an Assistant Professor at Yeshwantrao Chavan College of Engineering, Nagpur University.. Her research interest includes Digital Communication and VLSI design. Her teaching interest includes Digital Communication, Digital system design and Electronic devices.