Reconfigurable Architecture for Efficient and Scalable Orthogonal Approximation of DCT Using Verilog HDL

Vunnava Chaitanya Kumari, PG Scholar.Dept of ECE, Email Id: chaitanya.vunnava99@gmail.com
M.Nageswara Rao, M.Tech,(Ph.D),Asst.Professor,Dept of ECE, Email Id: nagmedikonda@gmail.com
Dr.V.S.R.Kumari, M.E.,Ph.D,[Professor & HOD Dept of ECE], Email Id: vsrk46@gmail.com
Sri Mittapalli College Of Engineering, Tummalapalem(V), Prathipadu(M), Guntur(Dt),Pin:522019

Abstract: A high speed word level finite field multiplier using redundant representation is proposed. Approximating the DCT is very worth full for decreasing its complexity without any effect on its performance. The new architecture has sign for higher speed compared to previously proposed architectures using either normal basis or redundant representation. One of the unique features of the proposed approximation is no need to redesign the architecture, for a given N input, it is used for one N , two N/2 , four N/4 length DCT. It is shown that the new approximation algorithm out-performs all the other techniques in comparison when considering the product of area and delay as a measure of performance. VLSI implementation of the proposed multiplier in a 0.18nm CMOS process is also presented.

I. Introduction

Finite fields have important applications in error control coding and cryptography [1]. Finite fields of characteristic two are most commonly used because they are inherently suitable for VLSI implementation. In practice, finite field multiplier is the key arithmetic unit for many systems based on finite field computations since more complicated finite field operations such like division and in version can be broken down into a series of consecutive multiplication operations. Efficiency of finite field multiplication depends on the choice of the basis to represent finite field elements.

Bases that have been used for realizing finite field multipliers include polynomial basis, normal basis (NB), dual bases, triangular basis, redundant representation or redundant basis, and their variations (i.e., shifted polynomial basis) [2], [3], [7], [8], [9], [10], [11]. Redundant representation is especially interesting because it not only offers almost free squaring as NB does but also eliminates modular operation for multiplication. The main drawback for the redundant representation is that it uses more bits to represent a finite field element, where the number of representation bits depends on the size of the cyclotomic ring in which the underlying layer is embedded.

Efficient computations in finite fields and their architectures are important in many applications including coding theory, computer algebra systems and public-key cryptosystems (e.g., elliptic curve cryptosystems). Although all finite fields of the same cardinality are isomorphic, their arithmetic efficiency depends greatly on the choice of bases for field element representations. The most commonly used bases are polynomial bases (PB) and normal bases (NB), sometimes combined with dual bases (DB)[15]. A major advantage of normal bases in the fields of characteristic two is that the squaring operation in NBs simply a cyclic shift of the coordinates of elements, so these are useful for computing large exponentiations and multiplicative inverses [13, 11, 1]. Also, the multiplication table of a normal
basis is symmetric, so suitable for hardware implementation.

We are mainly interested in finite fields of characteristic two, i.e. $F_{2^m}$, which are one of the two types of fields used most commonly in practice (the other one is $F_p$ where $p$ is a prime). We show how to find the smallest cyclotomic ring in which $F_{2^m}$ can be embedded. Since “embedding” is not unique, each element in the ring can be represented in more than one way, i.e., the representation contains a certain amount of redundancy. In this article, we also discuss how this redundant representation of a field element can be efficiently converted to a normal basis and vice versa.

II. Literature Survey

A representation of finite fields that has proved useful when implementing finite field arithmetic in hardware is based on an isomorphism between sub rings and fields. In this paper, we present an unified formulation for multiplication in cyclotomic rings and cyclotomic fields in that most arithmetic operations are done on vectors. From this formulation we can generate optimized algorithms for multiplication. For example, one of the proposed algorithms requires approximately half the number of coordinate-level multiplications at the expense of extra coordinate-level additions. Our method is then applied to the finite fields $GF(q^m)$ to further reduce the number of operations. We then present optimized algorithms for multiplication in finite fields with type-I and type-II optimal normal bases.

This article presents simple and highly regular architectures for finite field multipliers using a redundant representation. The basic idea is to embed a finite field into a cyclotomic ring which has a basis with the elegant multiplicative structure of a cyclic group. One important feature of our architectures is that they provide area-time trade-offs which enable us to implement the multipliers in a partial-parallel/hybrid fashion. This hybrid architecture has great significance in its VLSI implementation in very large fields. The squaring operation using the redundant representation is simply a permutation of the coordinates. It is shown that when there is an optimal normal basis, the proposed bit-serial and hybrid multiplier architectures have very low space complexity. Constant multiplication is also considered and is shown to have advantage in using the redundant representation. The elements of $-point$ DCT matrix are given by:

A new $GF(2^n)$ redundant representation is presented. Squaring in the representation is almost cost-free. Based on the representation, two multipliers are proposed. The XOR gate complexity of the first multiplier is lower than a recently proposed normal basis multiplier when $CN$ (the complexity of the basis) is larger than $3n$.

III. Scalable and Reconfigurable Architecture for DCT Computation

we discuss the proposed scalable architecture for the computation of approximate DCT of and 32. We have derived the theoretical estimate of its hardware complexity and discuss the reconfiguration scheme.

A. Proposed Scalable Design

The basic computational block of algorithm for the proposed DCT approximation, is given in [6]. The block diagram of the computation of DCT based on is shown in Fig. 1. For a given input sequence $\{X(n)\}, n \in [0, N-1]$, the approximate DCT coefficients are obtained by $F = C_N n X^t$.

An example of the block diagram of is illustrated in Fig. 2, where two units for the computation of are used along with an input adder unit and output permutation unit. The functions of these two blocks are shown respectively in (8) and (6). Note that structures of 16-point DCT of Fig. 2
could be extended to obtain the DCT of higher sizes. For example, the structure for the computation of 32-point DCT could be obtained by combining a pair of 16-point DCTs with an input adder block and output permutation block.

![Fig. 1. Signal flow graph (SFG) of (C_8^8). Dashed arrows represent multiplications by -1](image)

**FIG. 2.** Block diagram of proposed DCT for N=16 (c^16)

**B. Complexity Comparison**

To assess the computational complexity of proposed N–point approximate DCT, we need to determine the computational cost of matrices quoted in (9). As shown in Fig. 1 the approximate 8-point DCT involves 22 additions. Since has no computational cost and requires additions for –

**C. Proposed reconfiguration scheme**

![Fig. 3. Proposed reconfigurable architecture for approximate DCT of lengths N=8 and 16](image)

As specified in the recently adopted HEVC [10], DCT of different lengths such as, 16, 32 are required to be used in video coding applications. Therefore, a given DCT architecture should be potentially reused for the DCT of different lengths instead of using separate structures for different lengths.

We propose here such reconfigurable DCT structures which could be reused for the computation of DCT of different lengths. The reconfigurable architecture for the implementation of approximated 16-point DCT is shown in Fig. 3. It consists of three computing units, namely two 8-point approximated DCT units and a 16-point input adder unit that generates a(i) and b(i) he
input to the first 8-point DCT approximation unit is fed through 8 MUXes that select either \([a(0)\ldots a(7)]\) or \([x(0)\ldots x(7)]\) depending on whether it is used for 16-point DCT calculation or 8-point DCT calculation. Similarly, the input to the second 8-point DCT unit (Fig. 3) is fed through 8 MUXes that select either \([b(0)\ldots b(7)]\) or \([x(8)\ldots x(15)]\), depending on whether it is used for 16-point DCT calculation or 8-point DCT calculation. On the other hand, the output permutation unit uses 14 MUXes to select and reorder the output depending on the size of the selected DCT. is used as control input of the MUXes to select inputs and to perform permutation according to the size of the DCT to be computed. Specifically sel16=1 enables the computation of 16-point DCT and sel16=0 enables the computation of a pair of 8-point DCTs in parallel. Consequently, the architecture of Fig. 3 allows the calculation of a 16-point DCT or two 8-point DCTs in parallel.

Simulation Results:
Design summary:

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Sizes</td>
<td>1256</td>
<td>4096</td>
<td>20%</td>
</tr>
<tr>
<td>Number of 4-input LUTs</td>
<td>2257</td>
<td>3272</td>
<td>25%</td>
</tr>
<tr>
<td>Number of inv. bdds</td>
<td>923</td>
<td>292</td>
<td>32%</td>
</tr>
</tbody>
</table>

Top Module:

RTL Schematic:

Fig. 4. Proposed reconfigurable architecture for approximate DCT of lengths \(N = 8, 16\) and \(32\).

A reconfigurable design for the computation of 32-, 16-, and 8-point DCTs is presented in Fig. 4. It performs the calculation of a 32-point DCT or two 16-point DCTs in parallel or four 8-point DCTs in parallel. The architecture is composed of 32-point input adder unit, two 16-point input adder units, and four 8-point DCT units. The reconfigurability is achieved by three control blocks composed of 64 2:1 MUXes along with 30 3:1 MUXes. The first control block decides whether the DCT size is of 32 or lower. If , the selection of input data is done for the 32-point DCT, otherwise, for the DCTs of lower lengths.
Technology schematic:

IV. Conclusion

In this we have introduced a new recursive technique for orthogonal approximation of DCT where approximation does not affect its performance and it can be calculated by half the size of the original input length. The proposed reconfigurable approximation algorithm has several advantages like regularity, simplicity in its structure, scalability and the least computational complexity. When compared to the recently proposed methods our proposed approximation method is efficient in the fields of error energy, compressed image quality and hardware resource utilization. The multiplication result is computed after m clock cycles. The advantages of the proposed architecture are the high order of flexibility, which allows it to operate for all, half and one fourth of the length of the given input. In addition, the proposed multiplier has low power consumption features, which are achieved by using the gated clock technique. Comparing with previous published implementations, the used Wallace tree multiplier architecture is suitable for devices with limited silicon area.

Extension Work:

In Proposed system we are using Input Adder Unit, now it can be replaced by Wallace Tree Multiplier. By doing this we can get less power consumption, high accuracy and reduced delay.

V. References


