Reduction of Delay Propagation in Parallel Architecture Based on FNT for High Speed Cyclic Convolution

M.Vyshnava Datta[*], Ankit Kumar Jain[**], G. Mohan Bala Krishna[***]

[*] Mechanical, KL Univeristy
[**] Electrical and Electronics, KL University
[***] Mechanical, KL University

Abstract- This project deals with a cyclic convolution of high speed parallel architecture based on Fermat number transform (FNT). Cyclic convolution architectures are implemented for operands in diminished-1 representation. The code conversion (CC) method is mandatory to convert normal binary numbers into their diminished-1 representation. In this paper we discuss about the FNT (Fermat Number Transform) and IFNT (Inverse Fermat Number Transform) operations which are performed by CCWA (Code Conversion without Addition) & BOWA (Butterfly Operation without Addition). The convolution which is the point wise multiplication is implemented by modulo 2n+1 partial product multipliers (MPPM) and these output partial products are the inputs to the IFNT and modulo 2n+1 carry propagation additions are avoided in the FNT and the IFNT except in their final stages

The reduction of modulo 2n+1 carry propagation addition reduces the execution delay of the parallel architecture. The proposed one has better throughput performance and involves less hardware complexity than the existing cyclic convolution architecture.

Index Terms- Fermat Number Transform (FNT), Code Conversion Method without

-Addition (CCWA), Butterfly Operation without -Addition (BOWA), Modulo 2n+1 Partial Product Multipliers (MPPM).

I. Introduction

The cyclic convolution based on FFT is a widely used operation in signal processing, which needs to be performed in a complex domain even if both of the sequences to be performed would be real. Additionally, the dynamic range of the numbers varies widely so that one need to use floating point numbers to avoid scaling and quantization problems. Some architecture for efficient cyclic convolution has been developed to overcome the problems based on Number Theory Transform (NTT). They replace the complex domain with a finite field or a finite residue ring and can be defined by the FFT-like formula. All arithmetic operations are performed modulo *m* and the convolution results are exact without rounding errors. When the modulus in NTT is a Fermat number ($Ft=2^{2t}+1$, tth Fermat; *t* is an integer), the NTT turns into the Fermat Number Transform (FNT). The multiplication in the FNT and its inverse (IFNT) can be converted into bit shifts when the transform kernel is 2 or its integer power. Though the modulus of the FNT has a strict relationship with its maximum transform, the cyclic convolution based on FNT is more attractive than the conventional method in some areas.

1.1 Novel Cyclic Convolution Structure

Cyclic convolution architectures based on FNT are implemented for the operands in the diminished-1 representation. Thus the code conversion (CC) stage which converts the normal binary numbers into their diminished-1 representation is compulsory. Other arithmetic operations described originally by Leibowitz includes modulo 2^n+1 negation, addition ion, subtraction, multiplication operations in the dimis-nished-1 number system. These operations constitute the butterfly operation (BO) which is the most important element in the FNT.

The CC and the BO are both mainly composed of modulo 2^n+1 adders of which the fastest one in the diminished-1 number system is proposed by Vergos s so far. The fast modulo 2^n+1 adder involves the carry-propagation addition computation and is used in the recent FNT implementations.

In this paper, a code conversion method without addition (CCWA) and a butterfly operation method without addition (BOWA) which take full advantage of the carry-save adder are proposed to accomplish the cyclic convolution with the unity root 2 or its integer power.The modulo 2^n+1 partial product multiplier (MPPM) is used to accomplished the point wise multiplication so that the final carry-propagation addition of two partial product in the multiplier is avoided. Thus the execution delay of the architecture is reduced evidently. Model estimations and experiment results show that the proposed architecture is faster than the existing one when the modulus of the FNT is no less than 2^8+1. For wider modulus, the proposed parallel architecture leads to considerably faster hardware implementations than those presented.

Systolic arrays and distributed arithmetic are widely used for hardware implementation of cyclic convolutions. Distributed arithmetic is slow because of its inherent bit-serial nature. Many different systolic array based designs for the implementation of convolution were presented in, which can be used for implementation of cyclic convolutions.

These designs vary in terms of moving and staying parts (inputs, outputs, and weights) and the number of required delay elements. N-length cyclic convolution, all these designs will need multiplications, additions, and clock cycles.

1.2 Area efficient fault tolerant convolution

An efficient fault tolerant number theoretic transform (NTT) implementation of a convolution using the Redundant

Residue Number System (RRNS). The system is based on a recent method by Conway where a Modified Overlap Save (MOS) technique allows use of transform lengths of differing sizes in an NTT implementation of a convolution without fault tolerance. The proposed scheme adds fault tolerance and shows significant area improvements over RRNS methods combined with the traditional OS method.. The MOS system is then described and then the proposed RRNS MOS system is detailed. The final main Section compares the area of the proposed system with that of the conventional RRNS OS system across a range of convolution sizes.

1.3 Proposed RRNS MOS system:

The proposed system uses the MOS method to realize a convolution in the RRNS. The flexibility with respect to transform length provided by the MOS method allows design of a system where a convolution is realized using a combination of Mersenne and generalized Fermat moduli implemented using WSCA and NTTs, respectively. Table 1.1 shows the moduli and transforms lengths used. The outputs are reconstituted using the Chinese Remainder Theorem. A computer program was developed in Matlab to identify the minimum area implementation for the convolution lengths and bit widths shown in Table 1.2 The number of multiplications and additions required for the WSCA is calculated using the method described by Agarwal and Cooley and using the performance figures given in. The areas of the Fermat transforms are calculated on the basis of the number of additions in the NTT and inverse NTT together with the area of the post-multiplication stage. It is assumed that the filter coefficients are pre-computed.

All NTT kernels, v, used are either of the form 2k requiring two additions per butterfly or ±2k ($2^{M/2}$ + 1) which requires three additions per butterfly. In the latter case, even powers of the kernel are powers of 2 and only one stage in an NTT requires odd powers of v, involving three additions per butterfly.

A sequence of length N has log2 N stages; each stage has N/2 butterflies. Thus, the number of adds for an NTT is calculated as the product of the number of adds per butterfly, the number of stages and the number of butterflies per stage. For example, to implement an NTT with modulus the generalized Fermat number 2^{12} +1 and transform length 16, the more complex kernel, $2^4(2^6+ 1)$ is needed. There are four stages and eight butterflies per stage but one stage has three additions per butterfly and three stages have two additions per butterfly. This gives 3 X 1 X 8 + 2 X 3 X 8= 72 adds for this NTT and a total of 144 adds for the convolution together with 16 multiplications.

The gate equivalents (GE) for addition and multiplication modulo 2^n ±1 as given in Table 3 are used and the area is calculated on the basis of GEs per unit output. Following, the area per unit output for a channel is calculated as the total area for the channel 1/(N - Q + 1), where N is the transform length associated with the particular modulus used in that channel and Q is the filter length.

Moduli	Transform lengths	Moduli	Transform lengths
2^4+1	8,16	2^{28}+1	8,16
2^8+1	8,16,32	2^{32}+1	8,16,32,64,128
2^{12}+1	8,16		
2^{16}+1	8,16,32,64	2^5-1	6,10,15,30
2^{20}+1	8,16	2^7-1	7,14,21,28,35, 42
2^{24}+1	8,16,32	2^{13}-1	15,30,35,42

2^n+1	Adder.9n/2log$_2$n +n/2+6	Multiplier $8n^2$+ n-2+adder
2^n-1	Adder. 3nlog$_2$n +4n	Multiplier $8n^2$-14n+adder

Table 1.1 Moduli transform lengths and kernel types for proposed system

Word length	Sequence length	Reduction (%)
8	15,19,23,27,31	64,60,64,69,80
12	15,19,23,27,31	59,59,64,72,86
16	15,19,23,27,31	46,49,53,63,86
20	15,19,23,27,31	82,100,100,100,100
24	15,19,23,27,31	84,100,100,100,100
28	15,19,23,27,31	100,100,100,100,100
32	15,19,23,27,31	100,100,100,100,100

Table 1.2 Percentage area reduction achieved in MOS RRNS system.

II. Fermat Number Theoretic Transform

The cyclic convolution via the FNT is composed of the FNTs, the point wise multiplications and the IFNT. FNTs of two sequences $\{ai\}$ and $\{bi\}$ will produce two sequences $\{Ai\}$ and $\{Bi\}$. Modulo 2^n+1multipliers are employed to accomplish the point wise multiplication between $\{Ai\}$ and $\{Bi\}$ and produce the sequence $\{Pi\}$. The final resulting sequence $\{pi\}$ can be obtained by taking the inverse FNT of the product sequence $\{Pi\}$.Each element in the $\{pi\}$ is in the diminished-1 representation.

The FNT of a sequence of length N $\{x_i\}$ (i= 0,1, …N- 1) is defined as :

$$X_k = \sum_{i=0}^{N-1} x_i \alpha_N^{(ik)} \bmod(n) F_t (k = 0,1....N-1)$$
… (2.1)

where Ft=2^{2t}+1, the tth Fermat; N is a power of 2 and α is an Nth root unit (*i.e.* α^N_N mod Ft=1 and α^M_N mod Ft ≠1 ,1≤m <N). The notation < ik > means ik modulo N.

The inverse FNT is given by

$$x_t = \frac{1}{N}\sum_{k=0}^{N-1} x_k \alpha_N^{-(ik)} \bmod F_t (i = 0,1....N-1)$$
… (2.2)

Where 1/N is an element in the finite field or ring of integer and satisfies the following condition:
(N.1/N) mod F_t=1 …..(2.3)

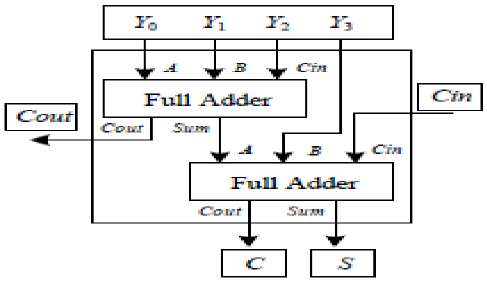
Parameters α, Ft, N must be chosen carefully and some conditions must be satisfied so that the FNT possesses the cyclic convolution property. In this project, α=2, Ft=2^{2t}+1 and N=2.2^t where t is an integer.
Table 1.3 Gate equivalents for implementation of arithmetic modulo 2^n ±1.

III. Important Operations In Cyclic Convolution Based On FNT

Important operations of the cyclic convolution based on FNT with the unity root 2 include the CCWA, the BOWA and the MPPM. The CCWA and the BOWA both consist of novel modulo 2^n+1 4-2 compressors mainly which are composed of the 4-2 compressor introduced by Nagamatsu. The 4-2 compressor, the novel modulo 2^n+1 4-2 compressor and the BOWA are shown in Fig.3.2 (b). In the figure, "X*" denotes the diminished-1 representation of X, i.e. X* =X-1.

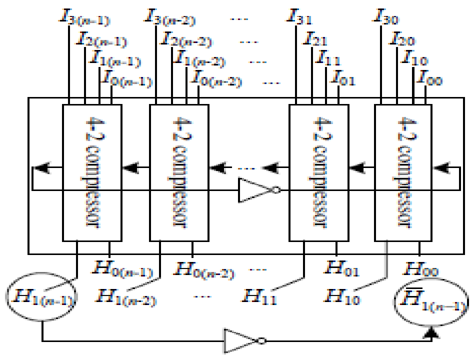
3.1 Code Conversion without Addition

The CC converts normal binary numbers (NBCs) into their diminished-1 representation. It is the first stage in the FNT. Delay and area of CC of a 2n-bit NBC are no less than the ones of two n-bit carry propagation adders.



(a)

Fig.3.1 Elementary operations of FNT architecture with unity root 2, (a) 4-2 compressor


$I_{3(n-1)}$ $I_{3(n-2)}$ I_{31} I_{30}
$I_{2(n-1)}$ $I_{2(n-2)}$ I_{21} I_{20}
$I_{1(n-1)}$ $I_{1(n-2)}$ I_{11} I_{10}
$I_{0(n-1)}$ $I_{0(n-2)}$ I_{01} I_{00}
4-2 compressor
4-2 compressor
4-2 compressor
4-2 compressor
$H_{0(n-1)}$ $H_{0(n-2)}$ H_{01} H_{00}
$H_{1(n-1)}$ $H_{1(n-2)}$ H_{11} H_{10} $\bar{H}_{1(n-1)}$

(b)

Fig.3.2 (b) modulo 2^n+1 4-2 compressor

To reduce the cost, we propose the CCWA that is performed by the modulo 2^n+1 4-2 compressor. Let A and B represent two operands whose widths are no more than 2n bits. We define two new variables:

$$A = 2^n A_H + A_L$$
$$B = 2^n B_H + B_L \qquad (3.1)$$

And

$$M_0 = (2^n - 1) - A_H = \bar{A}H$$
$$M_1 = (2^n - 1) - B_H = BH$$
$$M_2 = (2^n - 1) - B_L = B_L \qquad (3.2)$$

If the subsequent operation of CC is modulo 2^n+1 addition, assign A_L, M_0, B_L and M_1 to I0, I1, I2, I3 in the modulo 2^n+1 4-2 compressor respectively. I0, I1, I2, I3 are defined as follows:

$$I_0 = I_{0(n-1)} I_{0(n-2)} \ldots\ldots I_{01} I_{00}$$
$$I_1 = I_{1(n-1)} I_{1(n-2)} \ldots\ldots I_{11} I_{10}$$
$$I_2 = I_{2(n-1)} I_{2(n-2)} \ldots\ldots I_{21} I_{20}$$
$$I_3 = I_{3(n-1)} I_{3(n-2)} \ldots\ldots I_{31} I_{30} \qquad (3.3)$$

We obtain the sum vector H_O^* and carry vector H_1^* in the diminished-1 number system. The most significant bit of H_1^* is complemented and connected back to its least significant bit. That is to say

$$H_O^* = H_{0(n-1)} H_{0(n-2)} \ldots\ldots H_{01} H_{00}$$
$$H_1^* = H_{1(n-2)} \ldots\ldots H_{11} H_{10} H_{1(n-1)} \qquad \ldots\ldots(3.4)$$

The result of modulo 2^n+1 addition of A* and B* is equal to the result of modulo 2^n+1 addition of H_O^* and H_1^* in this way, A and B are converted into their equivalent diminished-1 representations H_O^* and H_1^* .

Let $|A^* + B^*|_{2^n+1}$, $|\bar{A}^*|_{2^n+1}$, $|A^* - B^*|_{2^n+1}$, and $|A^* + 2^i|_{2^n+1}$ denote modulo 2^n+1 addition, negation, subtraction and multiplication by the power of 2 respectively which are proposed by Leibowitz originally. The CCWA for subsequent modulo 2^n+1 addition can be described as follows.

$$|A^*+B^*|_{2n+1} = |AL+M0+BL+M1|_{2n+1} = |H_O^* + H_1^*| \quad \ldots (3.5)$$

If the subsequent operation is modulo 2^n+1 subtraction, we assign A_L, M_0, M_2 and B_H to I_0, I_1, I_2, I_3 respectively. Then H_O^* and H_1^* in the modulo 2^n+1 4-2 compressor constitute the result of the CCWA. The conversion is described as follows:

$$|A^*-B^*|_{2^n+1} = |A-B|_{2^n+1} = |A+B|_{2^n+1} = |A_L+M_0+B_L+M_1|_{2^n+1} \quad \ldots (3.6)$$
$$= |H_O^* + H_1^*|_{2^n+1}$$

After CCWA, we obtain the result consisting of two diminished-1 numbers. The result also includes the information of modulo 2^n+1 addition or subtraction in the first stage of previous BO.

3.2 Diminished-One modulo 2^n +1 Adder Design

Modulo arithmetic has been used in digital computing systems for various purposes for many years. In particular, modulo 2^n +1 arithmetic appears to play an important role in many algorithms. A first application field is in Residue Number Systems (RNS) . In an RNS based application, every number X is represented by a sequence of residues {X1, X2 . . . XM.} where Xi=X mod pi. The pies, 1 $\leq$ i$\leq$ M, comprise the base of the RNS and are pair wise relative prime integers. A two operand RNS operation, suppose}, is defined as (Z1, Z2 . . . ZM.)=(X1, X2 . . . XM.) ◊ (Y1, Y2. YM) where Z_i = (Xi ◊ Yi). Mod pi. For most RNS applications ◊ is addition, subtraction, or multiplication. Since the computation of Zi only depends upon Xi, Yi, and pi, each Z_i is computed in parallel in a separate arithmetic unit, often called channel. Moduli choices of the form{2^n -1,2^n,2^n +1} have received significant attention because they offer very efficient circuits in the area x time^2 product sense Addition in such systems is performed using three channels that, in fact, are a modulo 2^n -1 (equivalently, one's complement), a modulo 2^n , and a modulo 2^n +1 adder . The addition delay in an RNS application which uses the above moduli is dictated by the modulo 2^n +1 channel. The latter means that, if we can cut down the time required for modulo 2^n +1 addition, we also cut down the addition time in an RNS application.

Modulo 2^n +1 adders are also utilized as the last stage adder of modulo 2^n +1multipliers. Modulo 2^n +1multipliers find applicability in pseudorandom number generation, cryptography, and in the Fermat number transform, which is an effective way to compute convolutions Leibowitz has proposed the diminished-one number system. In the diminished-one number system, each number X is represented by X^* =X-1. The representation of 0 is treated in a special way. Since the adoption of this system leads to modulo 2^n +1 adders and multipliers of n bits wide operands, it has been used for many residue number system implementations; Efficient VLSI implementations of modulo 2^n +1 adders for the diminished-one number system have recently been presented.
The adders although fast, are, according to the comparison presented, still slower than the fastest modulo 2^n adders or the fastest modulo 2^n -1 adders.. Therefore, their use in an RNS application would still limit the performance of the system.
In this paper, we derive two new design methodologies for modulo 2^n +1 adders in the diminished-one number system. The first one leads to traditional Carry Look-Ahead (CLA), while the second to parallel-prefix adder architectures. Using implementations in a static CMOS technology, we show that the proposed CLA adder design methodology leads to more area and time efficient implementations than those presented, for small operand widths. For wider operands, the proposed parallel-prefix design methodology leads to considerably faster adder implementations than those presented and as fast as the integer or the modulo 2^n +1 architecture presented.

3.3 Butterfly operation without addition

After the CCWA, we obtain the results of modulo 2^n+1 addition and subtraction in the diminished-1 representation. Each result consists of two diminished-1 values. The subsequent butterfly operation involves four operands. The proposed BOWA involves two modulo 2^n+1 4-2 compressors, a multiplier and some inverters as shown in Fig. 3.3.

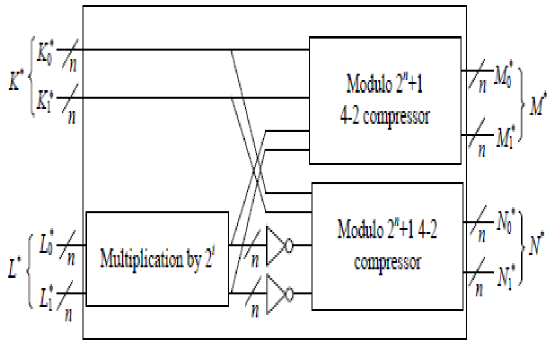


Fig.3.3 Butterfly operation without addition

The multiplication by an integer power of 2 in the diminished-1 number system in the BOWA is trivial and can be performed by left shifting the low-order *n-i* bits of the number by *i* bit positions then inversing and circulating the high order *i* bits into the *i* least significant bit positions. Thus the BOWA can be performed without the carry-propagation chain so as to reduce the delay and the area obviously. *K**, *L**, *M**, *N** are corresponding to two inputs and two outputs of previous BO in the diminished-1 number system respectively and given by

M*=$|M_0^* + M_1^*|_{\cdot 2^n+1}$=$|K_0^* + K_1^* + L_0^* X 2^t + L_1^* X 2^t|_{\cdot 2^n+1}$=
$|K_\cdot^* + L_0^* X 2^t|_{\cdot 2^n+1}$

N*=$|N_0^* + N_1^*|_{\cdot 2^n+1}$=$|K_0^* + K_1^* - L_0^* X 2^t - L_1^* X 2^t|_{\cdot 2^n+1}$=
$|K_\cdot^* - L_0^* X 2^t|_{\cdot 2^n+1}$
=$|K_\cdot^* + L_0^* X 2^t|_{\cdot 2^n+1}$
… (4.1)
Where$K_\cdot^* = |K_0^* + K_1^*|_{\cdot 2^n+1}$, $L_\cdot^* = |L_0^* + L|_{\cdot 2^n+1}$

3.4 Modulo 2^n+1 Partial Product Multiplier

For the modulo 2^n+1 multiplier proposed by Efstathiou, there are *n*+3 partial products that are derived by simple AND and NAND gates. An FA based Dadda tree that reduces the *n*+3 partial products into two summands is followed. Then a modulo 2^n+1 adder for diminished-1 operands is employed to accept these two summands and produce the required product.

In the proposed parallel architecture for cyclic convolution based on FNT, the BOWA can accept four operands in the diminished-1 number system. Every point wise multiplication only needs to produce two partial products rather than one product. The operation can be accomplished by taking away the final modulo 2^n+1 adder of two partial products in the multiplier. Thus the final modulo 2^n+1 adder is omitted and the modulo 2^n+1 partial product multiplier is employed to save the delay and the area.

IV. The FNT Architecture

In the previous sections, we have presented the reconfiguration at a rather low level. The Butterfly constitutes a high parameterized function level. The fact to have this parameterized function allows designing a reconfigurable operator who's Butterfly forms the highest level operator.

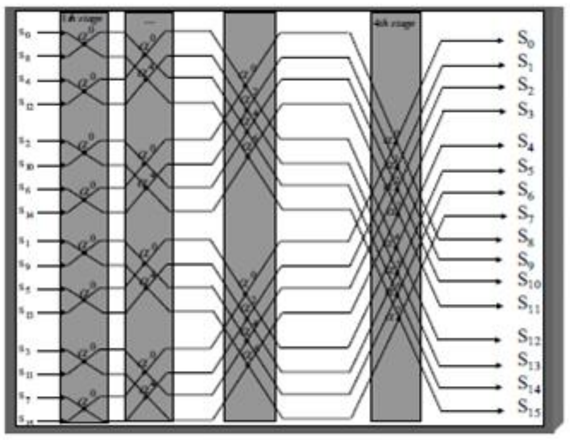


Fig 4.2 the architecture of FNT operator

V. Parallel Architecture For Cyclic Convolution

Based on the CCWA, the BOWA and the MPPM, we design the whole parallel architecture for the cyclic convolution based on FNT as shown in Fig.6.1. It includes the FNTs, the point wise multiplication and the IFNT mainly. FNTs of two input sequences {*ai*} and {*bi*} produce two sequences {*Ai*} and {*Bi*} (*i*=1, 2 …N- 1). Sequences {*Ai*} and {*Bi*} are sent to *N* MPPMs to accomplish the point wise multiplication and produce *N* pairs of partial products. Then the IFNT of the partial products are performed to produce the resulting sequence {*pi*} of the cyclic convolution.

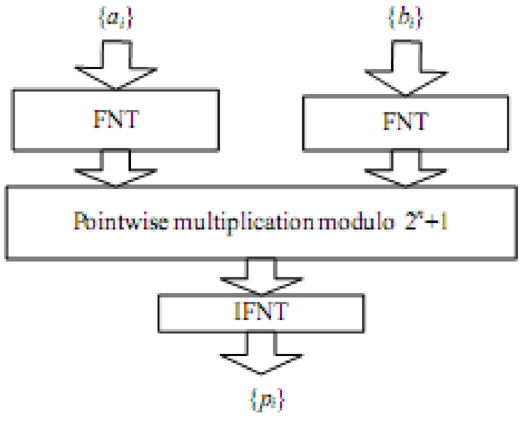
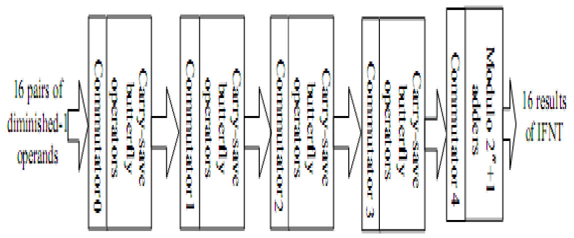


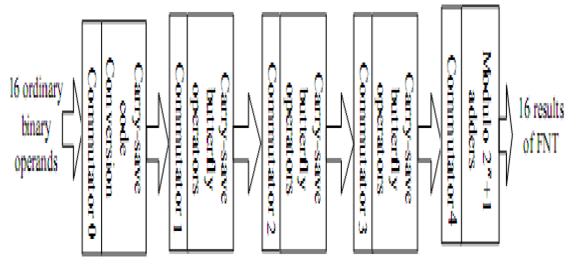
Fig.5.1 Parallel architecture for the cyclic convolution based on FNT

In the architecture, the radix-2 decimation-in-time (DIT) algorithm which is by far the most widely used algorithm is employed to perform the FNT and the IF*N*T.

Illustrative examples of the FNT and the IFNT are shown in Fig. 6.2(a) in the case the transform length is 16 and the modulus is 2^8+1. Commentators in Fig. 6.2 are used to adjust the operand order of every stage of FNT and IFNT according to the radix-2 DIT algorithm.


16 pairs of diminished-1 operands
Commutator 0
Carry-save butterfly operators
Commutator 1
Carry-save butterfly operators
Commutator 2
Carry-save butterfly operators
Commutator 3
Carry-save butterfly operators
Commutator 4
Modulo 2^n+1 adders
16 results of IFNT

(a)Parallel FNT structure


16 ordinary binary operands
Commutator 0
Carry-save code conversion
Commutator 1
Carry-save butterfly operators
Commutator 2
Carry-save butterfly operators
Commutator 3
Carry-save butterfly operators
Commutator 4
Modulo 2^n+1 adders
16 results of FNT

(b) Parallel IFNT structure
Fig. 5.2(b) Structures for FNT and IFNT

The efficient FNT structure involves $\log_2 N$+ 1 stages of operations. The original operands are converted into the diminished-1 representation in the CCWA stage, containing the information of modulo 2^n+1 addition or subtraction in the first butterfly operation stage of the previous FNT structure. Then the results are sent to the next stage of BOWA. After $\log_2 N$-1 stages of BOWAs, the results composed of two diminished-1 operands are obtained. The final stage of FNT consists of modulo 2^n+1 carry-propagation adders which are used to evaluate the final results in the diminished-1 representation. The CCWA stage, the BOWA stage and the modulo 2^n+1 addition stage in the FNT involves $N/2$ couples of code conversions including the information of modulo 2^n+1 addition and subtraction, $N/2$ butterfly operations and $N/2$ couple of modulo 2^n+1 additions respectively.

From the definition of FNT and IFNT in section 2, the only difference between the FNT and the IFNT is the normalization factor $1/N$ and the sign of the phase factor α_N. If ignoring the normalization factor $1/N$, the above formula

(Ft=2^8+1) is the same as that given in the FNT except that all transform coefficients $\alpha_N{}^{ik}$ used for the FNT need to be replaced by $\alpha_N{}^{-(ik)}$ for the IFNT computation. The proposed FNT structure can be used to complete the IFNT as well with little modification as shown in Fig. 5.2(b). After the IFNT of N-point bit reversed input data, the interim results are multiplied by $1/N$ in the finite field or ring. Then $x[j]$ and $x[j+N/2]$ (j=1,2,…,N/2-1) exchange their positions to produce the final results of the IFNT in natural order. Our architecture for the cyclic convolution gives a good speed performance without requiring a complicated control. Furthermore, it is very suitable for implementation of the overlap-save and overlap adds techniques which are used to reduce a long linear convolution to a series of short cyclic convolutions.

VI. Comparison and Results

K	5….6	7…9	10…13	14…19
D(k)	3	4	5	6
K	20…28	29…42	43….63	64…49
D(k)	7	8	8	10

In this section, we compare the proposed parallel architecture for the cyclic convolution against that introduced by Conway. The modulo 2^n+1 addition for the diminished-1 number system is the crucial operation which contains a standard n-bit carry propagation computation such as a parallel-prefix adder with a carry-logic block and a zero indicator of the diminished-1 operand to determine whether to perform subsequent operations. It produces the longest execution delay and requires large area in the previous solution. The proposed CCWA and BOWA overcome the disadvantage of the carry-propagation adder and don't require a zero indicator. Thus our architecture is faster and more efficient than the existing one.

This model assumes that each two-input gate excluding XOR is equivalent to one elementary gate for both area and delay. An XOR gate counts for two gates for both area and delay. Thus, a full adder has an area of seven gates and a delay of four gates. This model does not involve the cost of buffering and routing, but achieve a reasonable accuracy for the purpose of comparison.

The delay and the area estimations of modulo 2^n+1 adder and modulo 2^n+1 multiplier in the cyclic convolution are given in Table 1 as a function of the operand size n. "$D(n+3)$" in Table 6.1 is defined as shown in Table 6.2

Table 6.1 Area and delay estimations for arithmetic modulo 2^n+1 "MA" and "MM" represent modulo 2^n+1 adder and multiplier respectively.

Ft	Area(μm^2)		Delay(ns)	
	This project	[3]	This project	[3]
2^8+1	3.5 x 10^5	3.5 x 10^5	8.9	9.9
$2^{16}+1$	1.86 x 10^6	2.05 x 10^6	11.6	14.4
$2^{32}+1$	1.08 x 10^7	1.24 x 10^7	15.1	20.4

Table 6.2 D (k) as a function of k

To obtain more accurate results, we describe the proposed parallel cyclic convolution in verilog for Ft=$2^8+1,2^{10}+1,2^{32}+1$.The validated Verilog code is synthesized using a 0.13-μm CMOS standard cells library in the worst operating condition by the Synopsys Design Compiler. The units of area and delay are μm^2 and ns respectively. Each design was recursively optimized for speed until the EDA software can't provide a faster design. The results for the fastest derived implementation are listed in Table 6.3.

Table 6.1 and 6.3 indicate that for values of Ft $\geq 2^8+1$ the proposed architecture comprising the CCWA and the BOWA require less delay and area than the previous one. The former results in a 12.6% reduction in area and a 26% reduction in delay respectively compared with the latter in the case Ft is $2^{32}+1$ and the transform length is 64. Moreover, our algorithm will be more and more advantageous with the growth of modulus width.

Table 6.3 Area and delay results of cyclic convolution based on FNT

VII. Conclusion and Future Scope

7.1 Conclusion

A novel parallel architecture for the cyclic convolution based on FNT is proposed in the case the principle root of unity is equal to 2 or its integer power. The FNT and the IFNT are accomplished by the CCWA and the BOWA mainly. The point wise multiplication is performed by the modulo 2^n+1 partial product multiplier. Thus there are very little modulo 2^n+1 carry-propagation addition compared to the existing cyclic convolution architecture.

operator	Area		Delay	
	This project	[3]	This project	[3]
MA	14n	9/2nlogn+n/2+6	8	2logn+3
MM	$8n^2$+n-1	9/2nlogn+8n^2+n/2+4	4D(n+3)+1	4D(n+3) $2\log_2 n$+3

A theoretical model was applied to access the efficiency independently of the target technology. VLSI implementations using a 0.13 um standard cell library show the proposed parallel architecture can attain lower area and delay than that of the existing solution when the modulus is no less than 2^8+1.

7.2. Future Scope

This is the High speed parallel architecture for cyclic convolution based on FNT. Here in this architecture we used modulo adders and modulo multipliers to reduce the round off errors and also to reduce the delay and also that CCWA and BOWA blocks are used for further reduction of the delay. Based on the number of operands as the butterfly stages generally increases there is no other alternative for finding the convolution.

The changes can be made either for the blocks like multiplication by 2^n or for modulo adder or for modulo multiplier in order to reduce the size. Also that this can be implemented in low power technology by carefully following the parameters and the circuit elements used for the implementation. So as this is the high speed architecture, we further work on for the size or power consumption and others. VLSI implementations using a 0.13 um standard cell library show the proposed parallel architecture can attain lower area and delay than that of the existing solution when the modulus is no less than 2^8+1.

References

[1] C. Cheng, K.K. Parhi, “Hardware efficient fast DCT based on novel cyclic convolution structures”, IEEE Trans. Signal processing, 2006, 54(11), pp. 4419- 4434

[2] H.C. Chen, J.I. Guo, T.S. Chang, et al., “ A memory efficient realization of cyclic convolution and its application to discrete cosine transform”, IEEE Trans. Circuit and system for video technology, 2005, 15(3), pp. 445-453

[3] R. Conway, “Modified Overlap Technique Using Fermat and Mersenne Transforms”, IEEE Trans. Circuits and Systems II: Express Briefs, 2006, 53(8), pp.632 – 636

[4] A. B. O'Donnell, C. J. Bleakley, “Area efficient fault tolerant convolution using RRNS with NTTs and WSCA”, Electronics Letters, 2008, 44(10), pp.648-649

[5] H. H. Alaeddine, E. H. Baghious and G. Madre et al., “Realization of multi-delay filter using Fermat number transforms”, IEICE Trans. Fundamentals, 2008, E91A(9), pp. 2571-2577

[6] N. S. Rubanov, E. I. Bovbel, P. D. Kukharchik, V. J. Bodrov, “Modified number theoretic transform over the direct sum of finite fields to compute the linear convolution”, IEEE Trans. Signal Processing, 1998, 46(3), pp. 813-817

[7] L. M. Leibowitz, “A simplified binary arithmetic for the Fermat number transform,” IEEE Trans. Acoustics Speech and Signal Processing, 1976, 24(5):356-359

[8] H. T. Vergos, C. Efstathiou, D. Nikolos, “Diminishedone modulo 2n + 1 adder design”, IEEE Trans. Computers, 2002, 51(12), pp. 1389-1399

[9] M. Nagamatsu, S. Tanaka, J. Mori, et. al. “15-ns 32 × 32-b CMOS multiplier with an improved parallel structure”, IEEE Journal of Solid-State Circuits, 1990, 25(2), pp. 494-497

[10] C. Efstathiou, H. Vergos, G. Dimitrakopoulos, et al., “Efficient diminished-1 modulo 2^n + 1 multipliers”, IEEE Trans. Computers, 2005, 54(4), pp. 491-496

[11] A. Tyagi, “A reduced-area scheme for carry-select adders”, IEEE Trans. Computers, 1993, 42(10), pp. 1163-1170

Books

[1] J. G. Proakis and D. G. Manolakis, Digital signal processing: principles, algorithms, and applications, Prentice Hall, New Jersey, 2007.

[2] Essentials of VLSI Circuits and systems by Kamran Eshraghian, Douglas A. Pucknell, Sholeh Eshraghian.

[3] Modern VLSI Design, System - on - Chip Design Third Edition Wayne Wolf, Pearson Education.

[4] Barbe, D.F. (Ed.) (1982) Very Large Scale Integration – Fundamentals and applications, Springer - Verlag, West Germany / USA.

Authors

First Author – M. Vyshnava Datta, Mechanical Engineering, 4th year, KL University, Email: vyshnavm@live.com
Second Author – Ankit Kumar Jain, Electrical and Electronics Engineering, 4th year, KL University, Email: ankith851@gmail.com
Third Author – G. Mohan Bala Krishna, Mechanical Engineering, 4th year, KL University, Email: urstrulynani1@gmail.com

Correspondence Author – M. Vyshnava Datta, Email: vyshnavm@live.com, Phone: +919940695295