Abstract: This paper described for computing the generalized cyclic convolution is also applied for the linear convolution by the modified Fermat number transform (FNT) number system. The Code conversion method and butterfly operation are proposed to perform both FNT and its inverse (IFNT) to their final stage in the convolution. The crosswise and vertical multiplication in the convolution is accomplished by modulo $2^{m+1}$ partial product multipliers and output partial product which are input to the IFNT. The execution delay of parallel architecture is reduced due to the decrease of modulo $2^{m+1}$ carry propagation addition. We can compare with existing convolution architecture; the proposed one has a better throughput performance and involves less hardware complexity. In this result using 60nm FPGA technology, and also comparisons with 90nm, 130nm over the report solution.

1. INTRODUCTION

The cyclic convolution is one of the most important digital signal processing problems. It can be implemented more efficiently based on the Fermat number transform (FNT) than based on the fast Fourier transform (FFT) since the FNT possesses the property of cyclic convolution and low computational complexity. The expensive multiplication in the FFT and inverse (IFFT) can be replaced by bit shifts in the FNT and its inverse (IFNT) with the transform kernel 2 or its integer power. The introduction of the fast Fourier transform (FFT) allowed a tremendous saving in calculating effectively the discrete Fourier transform (DFS). This algorithm reduces the number of real multiplication for direct DFT computation. Additionally dynamic range of the numbers varies widely, so that we need to use floating point numbers to avoid quantization problems. The cyclic convolution developed to be overcome the problems based on number theory transform (NTT). They replace the linear complex domain with a finite residue ring can be defined by FFT. All the arithmetic operations are done in modulo m and then get exact result of convolution with out rounding or quantization errors. The number of multiplication for DFT computations can be further by using number theoretic transform (NTT). Of the various versions of the NTT proposed, the Fermat numbers transform (FNT) investigated by Agarwal and Burrus and the mersenne number transform (MNT) which was introduced by radar appear to offer the greater promise vis-à-vis speed and complexity. If we take the number of shift-adds into account for the evaluation of the entire computational effort, it turns out that because there is no fast algorithm for a computation of discrete Fourier transforms using MNT. The transform length of the Fermat number transform is highly composite, and hence fast Fourier transform types of algorithms can be used for the computation. In this it would appear that the FNT is the most prominent condition for the computation of DFTs. For
actual implementation, all number theoretic transforms are limited by sequence length constraints. The modulo $2^{m+1}$ partial product multiplier is used to accomplished the crosswise and vertical 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 faster than the existing one when the modulus of the FNT is no less then $2^{8}+1$. The proposed architecture lead to considerably faster hardware implementations then those presented applications.

The foundations of cyclic convolution based on FNT are formulated in the next chapter. The architecture for cyclic convolution based on FNT is illustrated. We get results that show the efficiency of the proposed architecture against result.

**OPERATIONS IN CYCLIC CONVOLUTION BASED ON FNT:**

Important operation of the cyclic convolution based on FNT with the unity root 2 includes the code conversion, butterfly and MPPM. The code conversion and butterfly both consists of novel modulo $2^{m+1}$ 4-2 compressors mainly which are composed of the 4-2 compressor introduced. The 4-2 compressor, the novel modulo $2^{m+1}$ 4-2 compressor and the butterfly as shown.

**CODE CONVERSION METHOD:**

The code conversion converts normal binary numbers into their diminished-1 representation. It is the first stage in the FNT. Delay and area of cc of 2n-bit NBC are no less than the ones of two n-bit carry propagation adders. It can be reduce the cost, we proposed the CC that is performed by the modulo $2^{m+1}$ 4-2 compressor.

Let $A$ and $B$ represent two operand whose widths are no more then 2n bits. We define two new variables:

$A = 2^{n}A_{H} + A_{L}$

$B = 2^{n}B_{H} + B_{L}$

and

$M0 = (2^{m} - 1) - AH = A'_{H}$

$M1 = (2^{m} - 1) - BH = B'_{H}$

$M2 = (2^{m} - 1) - BL = B'_{L}$

If the subsequent operation of CC is modulo $2^{m}+1$ addition assign $AL, M0, BL$ and $M1$ to $K0, K1, K2, K3$ in the modulo $2^{m}+1$ 4-2 compressor respectively. $K0, K1, K2, K3$ are defined as follows:

$K_{0} = K_{0(m-1)} K_{0(m-2)} \ldots K_{01} K_{00}$

$K_{1} = K_{1(m-1)} K_{1(m-2)} \ldots K_{11} K_{10}$

$K_{2} = K_{2(m-1)} K_{2(m-2)} \ldots K_{21} K_{20}$

$K_{3} = K_{3(m-1)} K_{3(m-2)} \ldots K_{31} K_{30}$

Than we obtain the sum vector $H_{0}^{*}$ and carry vector $H_{1}^{*}$ in the diminished-1 number system. The most significant bit of $H_{1}^{*}$ is complimented and connected back to its least significant bit.
Figure 1: (i)4-2 compressor, (ii) modulo $2^m+1$ 4-2 compressor (iii) butterfly operation

$$H_0' = H_{0(m-1)} H_{0(m-2)} \ldots \ldots H_{0} H_{00}$$

$$H_1' = H_{1(m-2)} \ldots \ldots H_{1} H_{0} H_{1'(m-1)}$$

The result of modulo $2^m + 1$ addition of $A'$ and $B'$ is equal to the result of modulo $2^m + 1$ addition of $H'_0$ and $H'_1$. In this way, A and B are converted into their equivalent diminished-1 representations $H'_0$ and $H'_1$.

Let $[A' + B']$ mod $2^m + 1$, $[A' - B']$ mod $2^m + 1$ and $[A' \times 2]$ mod $2^m + 1$, denote modulo $2^m + 1$ addition, negation, subtraction and multiplication by the power of 2 respectively which are proposed by Leibowitz. The CC without addition for subsequent modulo $2^m + 1$ addition can be described as follows:

$$|A' + B'| = |A_0 + M_0 + B_1 + M_1|2^m + 1 = |H'_0 + H'_1|2^m + 1$$

If the subsequent operation is modulo $2^m + 1$ subtraction, we assign $A_0$, $M_0$, $M_2$, and $B_1$ to $I_0$, $I_1$, $I_2$, respectively. Then $H'_0$ and $H'_1$ in the modulo $2^m + 1$ 4-2 compressor constitute the result of the CC without addition. The conversion is described as follows:

$$|A' - B'| = |A - B|2^m + 1 = |A - B_1 + M_0 + B_1 + M_0|2^m + 1 = |H'_0 + H'_1|2^m + 1$$

After CC without addition, we get the result consisting of two diminished-1 numbers. The result also includes the information of modulo $2^m + 1$ addition or subtraction in the first stage of previous Butterfly operation.

After completion of the code conversion without addition, we find the result of modulo $2^m + 1$ addition and subtraction in diminished-1 representation. Getting of each result consists of two diminished-1 values. The subsequent butterfly operations use four operands. The proposed butterfly operation without addition use two modulo $2^m + 1$ 4-

2 compressors, a multiplier and some inverters as shown in fig.

Once the different elements of the Butterfly are defined, one can implement them to obtain the reconfigurable Butterfly. The switch from an operating mode to another requires a change of the Fourier kernel and the reconfiguration of connection inter-operators. To do this, the Butterfly should download the primitive element activate the different logic gate (AND, OR and the multiplexers) and configure the connection inter-operators. The multiplication by an integer power of 2 in diminished-1 number system in the butterfly operation without addition is trivial and can be performed by left shifting the low-order m-i bits of the number by 1 bit positions then inversing circulating the high-order l bits in to the least significant bit position of i. the Butterfly without addition can be performed without the carry-propagation chain so as to reduce the delay and the occupation of area obviously, $K', L', M', N'$ are corresponding to the diminished-1 number system respectively and then given by

$$M' = |M_0 + M_1|2^m + 1 = |K_0 + K_1 + L_0 \times 2^i + L_1 \times 2^i|2^m + 1 = |K + L \times 2|2^m + 1$$

$$N' = |N_0 + N_1|2^m + 1 = |K_0 + K_1 - L_0 \times 2^i - L_1 \times 2^i|2^m + 1 = |K - L \times 2^i|2^m + 1$$

Where $|K_0 + K_1|2^m + 1$, $L' = |L_0 + L_1|2^m + 1$.

Partial product multiplier of modulo $2^m + 1$:

The partial product multiplier circuit allows users to efficiently pack multiplication logic into Altera’s FPGA devices when it is necessary to do so. The circuit features independent parameterized input widths, programmable pipeline stages and the selection between signed and unsigned multiplication. The programmable pipeline stage option allows you to find the most optimal setting between area and speed. For the modulo $2^m + 1$ multiplier proposed by Efstathiou, there $m+3$ partial products that are derived by simple AND and NAND gates. An FA based Daedal tree that reduces the $m+3$ partial products into two summands is followed. A modulo $2^m + 1$ adder for diminished-1 operands is employed to accept both summands and produce the required product.
Cyclic convolution based on FNT, the BOWA can accept more than four operands in the diminished-1 number system. Every crosswise and vertical multiplication only needs to produce two partial products rather than one product. The final modulo $2^{m+1}$ can adder is omitted and the partial product multiplier is employed to save the delay and the area.

Traditional architecture for cyclic convolution:

Depends upon the code conversion without addition (CCWA), the BOWA and MPPM we design the whole traditional architecture for cyclic convolution based on FNT. The crosswise, vertical multiplication can be used FNT and IFNT mainly.

The FNT of two input sequences $(p_i)$ and $(r_i)$ produce two sequences $(P_{i})$ and $(R_{i})$ $\{i=1, 2, 3...N\}$. These two sequences $(P_{i})$ and $(R_{i})$ are sent to N MPPMs to accomplish the crosswise or point wise multiplication and produce the N pair of partial products.

Then the IFNT of the partial products are performed to produce the resulting sequence $(Q_{i})$ of the cyclic convolution. In the traditional architecture, the radix-2 (DIT) algorithm which is by far the most widely used algorithm which is perform the FNT and the IFNT.

The FNT and IFNT are the case transform length is 16 and the modulus $2^8 + 1$ is the minimum transform length. In this Commutators are used to adjust the operand order of every stage of FNT and IFNT according to the radix-2 DIT algorithm.

Our work is focused on the application of the FNT to the channel coding –decoding. Indeed, the application of the discrete Fourier transform in the complex field occurs through the subject of the signal processing. By using the Fermat number transform, the principle of coding theory can be described in a setting that is much closer to the methods of signal processing. In this paper we will describe in details the practical realization of the FFT operator defined in C and which can be reconfigured to become the FNT operator with arithmetic carried out modulo Fermat numbers. This reconfiguration consists in reconfiguring each butterfly of the FFT structure. 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 information of modulo $2^{m+1}$ addition or subtraction in the first butterfly operation stage of the previous FNT structure.

After 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 result stage of FNT consists of modulo $2^{m+1}$ carry-propagation adders which are used to evaluate the final results in the dimenshed-1 representation. In the addition of all modulo stage in the FNT involves $N/2$ butterfly operations and $N/2$ couple of modulo $2^{m+1} + 1$ additions respectively. The difference between the FNT and the IFNT is normalization factor $1/N$ and sign of the phase factor alpha. Our architecture for cyclic convolution gives a better speed performance without requiring a complicated control. Furthermore it is more suitable for implementation of the overlap-save and overlap-add techniques which are used to reduce a long linear convolution to a series of short cyclic convolution.

**COMPARISON OF RESULT:**

In this section, we can compare the proposed traditional architecture for cyclic convolution against that introduced by Jain and Conway. The modulo $2^{m+1}$ addition for the diminished-1 number system is the crucial operation which contain a standard m-bit carry propagation computation such as a traditional –prefix adder with a carry-logic block and zero block and zero indicator of the diminished-1 operand to determine whether to perform sequent operations. In this technology produces the longest execution delay and requires large area in the previous solution. This paper proposed Code conversion (CC) and butterfly operation 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. The delay and area estimation of modulo $2^{m+1}$ adder and modulo $2^{m+1}$ multiplier in the cyclic convolution are given as a function of the operand size m. we proposed traditional architecture cyclic convolution in Verilog code is synthesized using a 0.6(micro meter) CMOS standard cells library in the worst operating condition.

**CONCLUSION:**

Traditional architecture for the cyclic convolution based on FNT and IFNT are proposed. This proposed case root of unity is equal to 2 or its integer power. The FNT and the IFNT are accomplished by the code conversion (CC) and the Butterfly operation (BO) s existing solution. When ever we are taking modulo is not less than the $2^k+1$. 
REFERENCES:


