Low Power Complex Multiplier based FFT Processor

V. Sarada, Dr. T. Vigneswaran
1ECE, SRM University, Chennai, India
saradasaran@gmail.com
2ECE, VIT University, Chennai, India
vigneshvlsi@gmail.com

Abstract- High speed processing of signals has led to the requirement of very high speed conversion of signals from time domain to frequency domain. Recent years there has been increasing demand for low power designs in the field of Digital signal processing. Power consumption is the most important aspect while considering the system performance. In order to design high performance Fast Fourier Transform (FFT) and realization, efficient internal structure is required. In this paper we present FFT Single Path Delay feedback (SDF) pipeline architecture using radix -2^4 algorithm. The complex multiplier is realized by using Digit Slicing Concept multiplier less architecture. To reduce computation complexity radix 2^4 algorithms is used. The proposed design has been coded in Verilog HDL and synthesizes by Cadence tool. The result demonstrates that the power is reduced compared with complex multiplication used CSD (Canonic Signed Digit) multiplier.

Keywords- Fast Fourier Transform, Multiplier less multiplier, radix 2^4

1. INTRODUCTION

The Fast Fourier Transform and Inverse Fast Fourier Transform (IFFT) have played important roles in communication applications. The demand for variable length, high speed and low power FFT has increased, especially Orthogonal Frequency Division Multiplexing (OFDM) application such as 3GPP, LTE(Long Term Evaluation), WiMax(Worldwide Interoperability for Microwave Access). There are various architecture to implement FFT such as memory based, pipeline based. In memory based architecture hardware cost and power consumption are lower but has long latency. In pipeline based architecture are acceptable hardware cost, high throughput and low latency. Mostly Preferred design in pipelined architecture are SDF[1] and Multipath Delay Commutator (MDC). The SDF is good, it requires less memory space about (N-1) delay element and a multiplication computation less than 50% also design control unit is simple based on these reason SDF Architecture is adopted in this paper. Various FFT algorithm start with Cooley-Tukey algorithm can reduce computation complexity from O(N^2) to O(N log2N). Higher radix FFT algorithm reduces computation complexity [2]. The regularity of the algorithm makes it suitable for VLSI implementation. In this paper we propose radix 2^4 algorithm, SDF pipeline Architecture based FFT Processor with Digit slicing multiplier for complex multiplication [3].

This paper is organized as follows; Section II explains the FFT algorithms, section III shows FFT architecture, Section IV complex constant multiplier. In section V the proposed design are compared with existing one. Finally the contribution of this work is summarized.

II. FFT ALGORITHMS

An N-point DFT of a sequence \( x[n] \) is defined as

\[
X (k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad k = 0, 1, ..., N - 1
\]

\[
w_N^{nk} = e^{-j2\pi nk / N} = \cos\left(\frac{2\pi nk}{N}\right) - j \sin\left(\frac{2\pi nk}{N}\right)
\]

(1)

Where \( k \) is frequency index, \( n \) is time index and the twiddle factor \( W_N^{nk} \) are the complex roots of unity, they are symmetric and equally spaced on the unity circle.

The straightforward implementation of this algorithm is impractical due to high hardware requirement. Therefore FFT was developed [2] to reduce computation time and hardware cost. Our work uses DIF (Decimation In Frequency) decomposition because it facilitate SDF. The radix-2\(^4\) algorithm has the same butterfly structure regardless of \( k \) value. The radix 2\(^4\) was formulated [1] using \( K=4 \) dimension linear index mapping. Radix 2\(^4\) algorithm can be expressed as various formula using Common factor algorithm. The Radix 2\(^4\) algorithm is given as follows.

Applying a 5 dimensional linear index mapping
\[ X(k) + 2k_2 + 4k_4 + 8k_8 + 16k_{16} \]
\[ = \sum_{n_0=n_1=n_2=n_3=n_4=0}^{N} x(n_0) \cdot N \cdot n_1 + N \cdot n_2 + N \cdot n_3 + N \cdot n_4 \cdot W^{n_{16}} \]

With the cascade decomposition the twiddle factor can be expressed in the form
\[ W_{N}^{ak} = \left\{ (-1)^{n_1 + j} \right\}^{n_2} \cdot (k_1 + 2k_2 + 4k_4) \cdot W_{8}^{n_4} (k_1 + 2k_2 + 4k_4) \cdot W_{16}^{n_4} (k_1 + 2k_2 + 4k_4 + 8k_8) \]
\[ W_{N}^{ak} = \left\{ (-1)^{n_1 + j} \right\}^{n_2} \cdot (k_1 + 2k_2 + 4k_4 + 8k_8) \cdot W_{8}^{n_4} (k_1 + 2k_2 + 4k_4 + 8k_8) \cdot W_{16}^{n_4} (k_1 + 2k_2 + 4k_4 + 8k_8) \]

The signal flow graph for 16 point radix 2^4 algorithm is shown in Fig. 1.

**III. FFT ARCHITECTURE**

There are many pipeline architecture to design FFT Processor for reducing complexity, power and overall to improve the performance. Among the structure three kind of pipeline architecture are used such as Multipath Delay Commutator(MDC),Single path delay Commutator(SDC) and Single path Delay feedback(SDF). In proposed FFT SDF architecture hardware and control circuit complexity is reduced. The proposed FFT processor shown in Fig. 2. The hardware requirements of different pipelined SDF architecture are listed in Table 1.

**TABLE 1. Comparison of Hardware requirement for N-length with different SDF architecture**

<table>
<thead>
<tr>
<th>Radix2 SDF</th>
<th>Complex multiplier</th>
<th>Complex adder</th>
<th>Register control circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radix2^1 SDF</td>
<td>Log_4N-1</td>
<td>4(Log_4N)</td>
<td>N-1</td>
</tr>
<tr>
<td>Radix2^2 SDF</td>
<td>Log_8N-1</td>
<td>4(Log_4N)</td>
<td>N-1</td>
</tr>
<tr>
<td>Radix2^3 SDF</td>
<td>Log_16N-1</td>
<td>4(Log_4N)</td>
<td>N-1</td>
</tr>
</tbody>
</table>
Fig. 2 16 point radix $2^4$ SDF architecture

It is composed of Radix-2 butterfly unit, Delay Buffer (FIFO) for support time multiplexing, -j for non-trivial multiplication, complex multiplier, and a control unit.

The Dual port FIFO function as shift register, it is used to accept data from butterfly unit and then shift left and feed back to the same butterfly unit.

The operation of radix-2 butterfly is shown in Fig 3

IV. COMPLEX MULTIPLIER

The complex multiplication is regarded as most important for FFT operation as it affects the speed performance of the digital signal processing system.

The complex multiplication with three multiplier is given by expression (4) and shown in Fig. 4

$$(Ar+jAi)*(Br+jBi) = \{Br (Ar-Ai) + Ai (Br-Bj)\} + j \{Bi (Ar+Ai) + Ai (Br-Bj)\}$$

IV. COMPLEX MULTIPLIER

The complex multiplication is regarded as most important for FFT operation as it affects the speed performance of the digital signal processing system.

The complex multiplication with three multiplier is given by expression (4) and shown in Fig. 4

$$(Ar+jAi)*(Br+jBi) = \{Br (Ar-Ai) + Ai (Br-Bj)\} + j \{Bi (Ar+Ai) + Ai (Br-Bj)\}$$

IV. COMPLEX MULTIPLIER

The complex multiplication is regarded as most important for FFT operation as it affects the speed performance of the digital signal processing system.

The complex multiplication with three multiplier is given by expression (4) and shown in Fig. 4

$$(Ar+jAi)*(Br+jBi) = \{Br (Ar-Ai) + Ai (Br-Bj)\} + j \{Bi (Ar+Ai) + Ai (Br-Bj)\}$$

IV. COMPLEX MULTIPLIER

The complex multiplication is regarded as most important for FFT operation as it affects the speed performance of the digital signal processing system.

The complex multiplication with three multiplier is given by expression (4) and shown in Fig. 4

$$(Ar+jAi)*(Br+jBi) = \{Br (Ar-Ai) + Ai (Br-Bj)\} + j \{Bi (Ar+Ai) + Ai (Br-Bj)\}$$

A. Digit Slicing multiplier

The concept behind the digit slicing architecture is any binary number can be sliced into a few blocks of shorter binary numbers, each having a shorter word length. [4] Depending on the word length, the fundamental sliced algorithm is applied to the binary number.

$$F_k = F_{Rk} + j F_{Ik}$$

$$F = \sum_{k=0}^{b-1} 2^{p-1} F_{Rk} + j \sum_{k=0}^{b-1} 2^{p-1} F_{Ik}$$

Where $F_{Rk} = \sum_{j=0}^{p-1} 2^j F_{Rkj}$ and $F_{Ik} = \sum_{j=0}^{p-1} 2^j F_{Ikj}$
In this equation \( F_{R_k} \) and \( F_{I_k} \) have values which are either zero or one. Any value whose absolute value is less than one can be represented in two's complement as \( x = \sum_{k=0}^{b-1} 2^k \ X_k \). Here \( x \) is any number with an absolute value less than one and \( x \) is sliced into \( b \) blocks, each block being \( p \) bits wide. \( x_k = \sum_{j=0}^{p-1} 2^j \ X_{k,j} \)

For example \( X=A*B \) In this multiplication one of the operand \( (A) \) divided into four parts as shown in Fig. 5

A divided into four parts Such as
- \( part1= A_3A_2A_1A_0 \)
- \( part2= A_7A_6A_5A_4 \)
- \( part3= A_{11}A_{10}A_9A_8 \)
- \( part4= A_{15}A_{14}A_{13}A_{12} \)

There are four different cases for the multiplication between the four bits and the twiddle factors. Fig 6 shows the block diagram of the digit-slicing multiplier less using the shift and addition technique. Shift-and-add multiplication is similar to the multiplication performed by paper and pencil.

\[
K_0=(A_3A_2A_1A_0)*B,
K_1=(A_7A_6A_5A_4)*B,
K_2=(A_{11}A_{10}A_9A_8)*B,
K_3=(A_{15}A_{14}A_{13}A_{12})*B
\]

\[
X=A*B=K_0 + 2^4K_1 + 2^8K_2 + 2^{12}K_3
\] (7)

\[\text{Fig. 6 Complex multiplication using Digit slicing}\]

**B. CSD Multiplier**

The Canonic signed digit (CSD) representation to realize the function of complex multiplier in non-trivial multiplication. The complex multiplier performs the computation of a constant factor and a non-constant number. The real and imaginary parts of the constant factor are represented in CSD form. The corresponding parts of the non-constant factor are represented in two's complement form. This method has potential advantage of easier circuit structures than the conventional complex multiplier.

### TABLE 2. Represent of binary and CSD

<table>
<thead>
<tr>
<th>Coefficient Value</th>
<th>16 Bit Expression</th>
<th>Binary</th>
<th>CSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.923828</td>
<td>01110110010000000</td>
<td>1001 10 1</td>
<td>0010000000</td>
</tr>
<tr>
<td>0.707092</td>
<td>01011010100000010</td>
<td>01101 1</td>
<td>010100000010</td>
</tr>
<tr>
<td>0.382629</td>
<td>00110001000000110</td>
<td>0101 000100000010</td>
<td>10</td>
</tr>
</tbody>
</table>
The CSD unit consists of many shifters and adders. It is used to realize the multiplication when the one factor is constant. The CSD unit is used to construct number that contains the minimum possible number of non-zero bits.

\[
0.707092 = [(x>>1 + x>>2) - (x>>4 + x>>6 + x>>8)]
\]

Where \(x\) denotes input data, can be shared by two or more constant multiplications. The values in the diamonds represent the times of right shift. So this realization only needs four adders/subtractors [5].

V. RESULT

The architecture of the proposed FFT processor was designed in Verilog and simulated to verify its functionality. The simulation and synthesis were performed using the cadence design tool 180 nm CMOS Technology. Table 3 shows the performance comparison between the proposed and FFT processor using existing CSD multiplier based FFT. The results shows the proposed FFT processor obtain low power when compared with CSD based processor with little increase in area.
TABLE 3. Power and area Analysis

<table>
<thead>
<tr>
<th></th>
<th>Word length</th>
<th>Power (mW)</th>
<th>Frequency (MHz)</th>
<th>Area (No. slice used)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT(using CSD multiplier based)</td>
<td>16</td>
<td>6.69</td>
<td>166.6</td>
<td>3892</td>
</tr>
<tr>
<td>Proposed</td>
<td>16</td>
<td>4.60</td>
<td>166.6</td>
<td>4022</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

In this paper, low power 16 point FFT processor using Digit slice based multiplier less multiplier has been designed and 16 point FFT processor using CSD multiplier has also been designed. The result shows that our design lowers hardware cost and power consumption. Our proposed scheme can also be adapted to high point FFT Fixed and variable FFT processor which is used in various OFDM based communication.

ACKNOWLEDGEMENT

The author would like to thank SRM University Research lab for supporting this work.

REFERENCES