Low-Cost Variable-Length FFT Processor for DVB-T/H Applications

Kisun Jung and Hanho Lee
School of Information and Communication Engineering, Inha University, Incheon, 402-751, Korea
hhlee@inha.ac.kr

Abstract—This paper presents a low-cost variable-length FFT processor for digital video broadcasting – terrestrial/handheld (DVB-T/H) systems employing pipelined shared-memory architecture, in which a radix-2/23/24 FFT algorithm, multi-path delay commutator (MDC), a novel data scaling approach are exploited. Based on this architecture, novel low cost index block scaling approach was proposed to increase area efficiency. Also, an elaborate memory configuration scheme applied to make single-port SRAM without degrading throughput rate. The SQNR performance of this FFT processor has signal-to-quantization noise ratio (SQNR) of 8K-point FFT is about 46.8 dB at 11-bit internal word length for QPSK/16QAM modulation. The maximum clock frequency of proposed design is 110 MHz.

Keywords—DVB, FFT, variable length, scaling, pipelined shared memory.

I. INTRODUCTION

Orthogonal frequency division multiplexing (OFDM) attracts much attention in wireless communication systems, because it is feasible to many diverse requirements. In the typical OFDM systems, the complex symbols are modulated by means of inverse fast Fourier transform (IFFT). After guard interval insertion, the resulting OFDM symbol is serially transmitted over various channels, and at the receiver end the data stream is recovered by the inverse operation. Therefore, the FFT and IFFT blocks are the key computational blocks in OFDM systems. The large-size FFT is commonly adopted in OFDM system to increase transmission bandwidth or efficiency in many wireless applications such as digital video broadcasting (DVB), digital audio broadcasting (DAB), digital media broadcasting (DMB), very high-speed digital subscriber line (VDSL) and other mobile applications.

The 2k/4k/8k point FFT should be performed in 224µ/448µs/896µs, respectively, according to DVB standards. On designing a long-size FFT processor, one still has to consider its power consumption and hardware cost. Furthermore, the power consumption of both data access in memory and operation of complex multipliers is more than 75% of total power consumption in an FFT processor [1]. To reduce the power consumption of the FFT memories and complex multipliers, the useful method is reducing the memory access times, internal word length and the number of operations in complex multipliers.

The memory-based architecture provides lower hardware complexity and high flexibility. However, it has disadvantages such as low memory access speed and high controller complexity.

In this paper, a low-cost variable-length FFT processor architecture is proposed to optimize the memory-based processor design. This work is based on the modified memory-based FFT processor architecture called pipelined shared memory architecture. It locally employed the pipelined architecture to realize the butterfly unit and globally utilize the pipeline in memory-based architecture to reduce the hardware complexity. But this architecture still has two disadvantages such as low data throughput and high controller complexity. Therefore, the proposed FFT processor employed multi-path delay commutator (MDC) architecture in [2]. Also, using radix-24 algorithm reduces the memory access times. In order to maintain data accuracy in fixed-point FFT, the internal word length of the FFT processor is usually larger than the word length of input data to achieve a higher SQNR, especially in a long-size FFT processor. This causes high hardware costs. To resolve this issue, novel index block scaling approach is proposed to improve signal-to-noise ratio (SQNR) dramatically by increasing the number of scale factor and index.

II. FFT ALGORITHM

The N-point discrete Fourier transform (DFT) of a sequence x(n) is defined as

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

(1)

where x(n) and X(k) are complex numbers. The twiddle factor is defined as

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

(2)

Thus, the computational complexity in Eq. (1) is \(O(N^2)\) through directly performing the required computation. By using the FFT algorithm, the computational complexity can be reduced to \(O(N \log_2 N)\), where \(r\) means the radix-\(r\) FFT in [3].

The radix-2 FFT algorithm is simplest form in all FFT algorithms. But its computational complexity of multiplication is higher than any other higher radix. In order to save the number of complex multiplications, we mainly employed radix-24 decimation in frequency (DIF) algorithm. Table I shows computational complexity of complex multipliers in 8K mode. Also, radix-2424 DIF algorithms are used in 2K/8K mode additionally. Radix-2424 algorithm further decomposes...
radix-8/16 butterflies into three/four steps of radix-2 units, respectively.

A. Radix-2 DIF algorithm

Final butterfly stage of 8k mode employed radix-2 DIF algorithm. Radix-2 algorithm has a disadvantage of computational complexity of complex multiplication. However, in final stage of 8k mode, FFT processor is not performing multiplication. Thus, it hadn’t any disadvantages from a radix-2 algorithm.

B. Radix-2\(^3\) DIF algorithm

Final butterfly stage of 2k mode employed Radix-2\(^3\) DIF algorithm. In order to derive the radix-2\(^3\) algorithm, consider the following first 3 steps of decomposition in [4]. Applying a 4-dimensional linear index map by

\[
\begin{align*}
\alpha &= \lfloor \frac{n}{2} + \frac{n}{4} + \frac{n}{8} \rfloor, \\
\beta &= \lfloor \frac{n}{4} + \frac{n}{8} \rfloor,
\end{align*}
\]

The Common Factor Algorithm (CFA) takes the form of

\[
X(k_1 + 2k_2 + 4k_3 + 8k_4) = \sum_{n_1=0}^{N/2-1} \sum_{n_2=0}^{N/4-1} \sum_{n_3=0}^{N/8 -1} \sum_{n_4=0}^{N/16-1} X \left( \frac{N}{2} n_1 + \frac{N}{4} n_2 + \frac{N}{8} n_3 + n_4 \right) W_N^\alpha W_N^\beta W_N^\gamma W_N^\delta,
\]

The twiddle factors can be expressed in the form of

\[
W_N^\alpha = (-1)^{\alpha} n_1 (j n_1 k_1) W_N^{n_1 k_1 + 2n_1 k_2 + 4n_1 k_3 + 8n_1 k_4}
\]

and

\[
W_N^\beta = (-1)^{\beta} n_2 (j n_2 k_2) W_N^{n_2 k_2 + 2n_2 k_3 + 4n_2 k_4 + 8n_2 k_4}
\]

The twiddle factors can be expressed in the form of

\[
W_N^\gamma = (-1)^{\gamma} n_3 (j n_3 k_3) W_N^{n_3 k_3 + 2n_3 k_4 + 4n_3 k_4 + 8n_3 k_4}
\]

and

\[
W_N^\delta = (-1)^{\delta} n_4 (j n_4 k_4) W_N^{n_4 k_4}
\]

C. Radix-2\(^3\) DIF algorithm

Radix-2\(^3\) DIF algorithm is mainly used in all (2k/4k/8k) modes. In order to derive the radix-2\(^3\) algorithm, consider the following first 4 steps of decomposition in [4].

Applying a 5-dimensional linear index map by

\[
\begin{align*}
\alpha &= \lfloor \frac{n}{4} + \frac{n}{8} \rfloor, \\
\beta &= \lfloor \frac{n}{8} \rfloor,
\end{align*}
\]

The CFA takes the form of

\[
\begin{align*}
X(k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5) &= \sum_{n_1=0}^{N/4-1} \sum_{n_2=0}^{N/8 -1} \sum_{n_3=0}^{N/16 -1} \sum_{n_4=0}^{N/32 -1} \sum_{n_5=0}^{N/64 -1} X \left( \frac{N}{4} n_1 + \frac{N}{8} n_2 + \frac{N}{16} n_3 + \frac{N}{32} n_4 + n_5 \right) W_N^\alpha W_N^\beta W_N^\gamma W_N^\delta W_N^\epsilon,
\end{align*}
\]

The twiddle factors can be expressed in the form of

\[
W_N^\alpha = (-1)^{\alpha} n_1 n_2 n_3 n_4 n_5 W_N^{n_1 k_1 + 2n_1 k_2 + 4n_1 k_3 + 8n_1 k_4 + 16n_1 k_5}
\]

The CFA takes the form of

\[
\begin{align*}
X(k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5) &= \sum_{n_1=0}^{N/4-1} \sum_{n_2=0}^{N/8 -1} \sum_{n_3=0}^{N/16 -1} \sum_{n_4=0}^{N/32 -1} \sum_{n_5=0}^{N/64 -1} X \left( \frac{N}{4} n_1 + \frac{N}{8} n_2 + \frac{N}{16} n_3 + \frac{N}{32} n_4 + n_5 \right) W_N^\alpha W_N^\beta W_N^\gamma W_N^\delta W_N^\epsilon,
\end{align*}
\]

The CFA takes the form of

\[
\begin{align*}
X(k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5) &= \sum_{n_1=0}^{N/4-1} \sum_{n_2=0}^{N/8 -1} \sum_{n_3=0}^{N/16 -1} \sum_{n_4=0}^{N/32 -1} \sum_{n_5=0}^{N/64 -1} X \left( \frac{N}{4} n_1 + \frac{N}{8} n_2 + \frac{N}{16} n_3 + \frac{N}{32} n_4 + n_5 \right) W_N^\alpha W_N^\beta W_N^\gamma W_N^\delta W_N^\epsilon,
\end{align*}
\]

The CFA takes the form of

\[
\begin{align*}
X(k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5) &= \sum_{n_1=0}^{N/4-1} \sum_{n_2=0}^{N/8 -1} \sum_{n_3=0}^{N/16 -1} \sum_{n_4=0}^{N/32 -1} \sum_{n_5=0}^{N/64 -1} X \left( \frac{N}{4} n_1 + \frac{N}{8} n_2 + \frac{N}{16} n_3 + \frac{N}{32} n_4 + n_5 \right) W_N^\alpha W_N^\beta W_N^\gamma W_N^\delta W_N^\epsilon,
\end{align*}
\]

In Eq. (9), second and third stage employed twiddle factors \((W_k, W_{k5})\) that used in complex constant multiplication. In order to reduce second and third stage’s computational complexity, Eq. (9) is modified to Eq. (10).

\[
W_N^{k5} = (-1)^{n_2} n_3 (j n_3 k_3) W_N^{n_3 k_3 + 2n_3 k_4 + 4n_3 k_4 + 8n_3 k_4 + 16n_3 k_4}
\]

In Eq. (10), third stage employed twiddle factors that used to only rotate factor \((-1)^{n_3} W_k\). Thus, Eq. (10) relatively has lower computation complexity than Eq. (9). While FFT processor computed four stages, go through one complex constant multiplication stage and one complex Booth multiplication stage.

III. PROPOSED FFT ARCHITECTURE

The block diagram of the proposed pipelined shared-memory FFT architecture is shown in Fig. 2 and Fig. 3. It mainly consists of 8k-word (22bit) single-port SRAM, control unit and PE. Through the control of input sequence of PE, it can employ only half of buffer compared to conventional one. FFT processor is performed input/output (I/O) operation and PE operation, repeatedly. This architecture has different iterations of PE operation in each other modes. The proposed FFT processor performs PE operation three times in the 2K/4K mode and four times in the 8K mode. 2K-point FFT can be performed using two iterations of radix-2\(^2\) operation and radix-2\(^3\) operation. 4K-point FFT can be performed using three iterations of radix-2\(^2\) operation. 8K-point FFT can be performed using three iterations of radix-2\(^2\) operation and radix-2 operation.
A. Multipliers

The multipliers can be divided into two types. First is the constant multiplier and the other is the complex Booth multiplier. In Fig. 3, constant multiplier is located next to BU 2. The proposed constant multiplier performs multiplication operation about two input symbols at the same time. The constant multiplier has three input and three output ports, which include feedback loop input and output ports. One of the inputs is passed by MUX and the other two inputs are used to multiplication. Also, Compared with conventional constant multiplier, proposed constant multiplier performs slightly different operation. In order to reduce the constant multiplier’s hardware area, we controlled scheduling of twiddle factor. The proposed constant multiplier delays one clock cycle for twiddle factor in a 2\textsuperscript{nd} data-path, when the time sequences are

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
Time Sequence & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
1st Data Path & 1 & 1 & 1 & 1 & w(1) & w(2) & w(3) \\
\hline
2nd Data Path & w(9) & w(2) & -j & w(6) & 1 & 1 & w(3) & w(9) \\
\hline
0.9239(a) & 2 & X & X & X & 2 & 2 & 2 \\
0.3827(b) & 2 & X & X & X & 2 & 2 & 2 \\
0.7071(c) & 2 & X & X & X & 2 & 2 & 2 \\
\hline
\end{tabular}
\caption{Scheduling of the twiddle factor $W_N^k$ (Constant Multiplier)}
\end{table}

at 4 ~ 7 time units. That twiddle factors being delayed are used in multiplication at next clock. TABLE II shows a detailed scheduling of the twiddle factor.

B. Indexed Block Scaling Approach

Although the proposed indexed block scaling (IBS) approach is based on block scaling (BS) method in [5], IBS uses different scaling techniques with BS method and has different characteristic. IBS method uses indexes instead of buffers in BS method. Increasing the number of scaling blocks, total index size is increased proportionally.

Fig. 4 shows performance comparisons of various scaling methods. Fig. 5 shows memory usage of IBS and BS methods. The IBS method is effective if block size is higher than 128 as shown in Fig. 5. For BS method, the elements inside the block are determined sequentially. However, in the IBS method, the elements inside the block are efficiently determined regardless of order of operations. Thus, IBS method can be gained higher SQNR than BS method at same block size. In the proposed method, we used 512 block size. As a result, the SQNR performance of 8K-point FFT achieved 46.8 dB for QPSK...
/16QAM signals. This result is almost same as BS method using 64 block size. As a result, IBS method has the advantage of memory element usage compared with BS method.

C. Memory Unit

As mentioned in Section I, the major part of FFT processor’s power consumption is resulted from the main memory blocks. So, in order to reduce chip area and power consumption, single-port SRAM and low internal word length was used in the proposed architecture. Single-port SRAM is operated with lower memory access and occupied lower area than dual-port SRAM.

The total memory size in our design is 176K bit, consisting of 11bit real and imaginary parts. The read/write scheduling scheme of the four memory banks and prefetch/prewrite buffers are shown in Fig. 6. The memory banks are performed write and read operation per each clock, iteratively.

IV. PERFORMANCE EVALUATION AND COMPARISON

This proposed architecture was designed in Verilog HDL and simulated to verify its functionality. Both simulation and synthesis steps were carried out using SYNOPSYS design tools and 180-nm CMOS technology optimized for a 1.8 V supply voltage. Also, this work is verified via MATLAB and C simulation.

Table II and Table III illustrated comparison of this work with several long-length FFT designs in [5]-[7]. We figure out that the optimal bit number for data representation is 11bits and twiddle factor is 10bits in our work. The SQNR performance of proposed FFT processor is about 46.8 dB. It completes the 8K point FFT in 819.8 μs at 20MHz which meets DVB-T standard (i.e., 896 μs in 8K mode). The maximum frequency of proposed design is 110 MHz.

V. CONCLUSION

In this paper, we present a low-cost variable-length FFT processor for DVB-T/H systems, in which the pipelined shared-memory architecture based on MDC architecture was employed. Also, in order to reduce the internal word length, it employed new scaling approach. And, single-port SRAM with minimal word length is used to reduce the hardware complexity and power consumption. As a result, proposed processor achieves lower hardware complexity, small memory size and sufficient SQNR performance more than 45db.

ACKNOWLEDGMENT

This research was supported by the MKE, Korea, under the ITRC support program supervised by the NIPA (NIPA-2010- C1090-1011-0007)

REFERENCES


TABLE III

COMPARISON OF HARDWARE COMPLEXITY OF COMPUTATION BLOCKS.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of complex</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adders/subtractors</td>
<td>8</td>
<td>8</td>
<td>26</td>
</tr>
<tr>
<td>No. of constant</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>multiplier(W^L)</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>No. of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>multiplier(W^L)</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>No. of complex</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Booth multiplier</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Size of twiddle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>factor LUT size(bit)</td>
<td>45,056</td>
<td>90,112</td>
<td>25,200</td>
</tr>
</tbody>
</table>

TABLE IV

PERFORMANCE COMPARISON.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CMOS Technology</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.18 μm</td>
<td>1/2/4/8(K)</td>
<td>2/4/8(K)</td>
<td></td>
</tr>
<tr>
<td>0.18 μm</td>
<td>2/4/8(K)</td>
<td>2/4/8(K)</td>
<td></td>
</tr>
<tr>
<td>0.18 μm</td>
<td>2/4/8(K)</td>
<td>2/4/8(K)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Representaion</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pipelined memory</td>
<td>176x+0.53k</td>
<td>176x+2.53k (estimated)</td>
<td>192x+1.57k (estimated)</td>
</tr>
<tr>
<td>Cached memory Radix-2</td>
<td>Single-port SRAM</td>
<td>Single-port SRAM</td>
<td>Single-port SRAM</td>
</tr>
<tr>
<td>Pipelined SDF Balanced binary-tree</td>
<td>Single-port SRAM</td>
<td>Single-port SRAM</td>
<td></td>
</tr>
<tr>
<td>Pipelined memory Radix-2</td>
<td>Single-port SRAM</td>
<td>Single-port SRAM</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Block scaling</td>
<td>2 x 11 bit</td>
<td>2 x 12 bit</td>
<td>2 x 13 bit</td>
</tr>
<tr>
<td>Block scaling</td>
<td>2 x 11 bit</td>
<td>2 x 12 bit</td>
<td>2 x 13 bit</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory element</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>176x+0.53k</td>
<td>176x+2.53k (estimated)</td>
<td>192x+1.57k (estimated)</td>
</tr>
<tr>
<td>Memory type</td>
<td>Single-port SRAM</td>
<td>Single-port SRAM</td>
<td>Single-port SRAM</td>
</tr>
<tr>
<td>Single-port SRAM</td>
<td>Single-port SRAM</td>
<td>Single-port SRAM</td>
<td></td>
</tr>
<tr>
<td>Single-port SRAM</td>
<td>Single-port SRAM</td>
<td>Single-port SRAM</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Max. clock frequency (MHz)</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>56</td>
<td>122</td>
<td>50</td>
</tr>
<tr>
<td>56</td>
<td>112</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>50</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SQNR (dB)</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
</tr>
</thead>
<tbody>
<tr>
<td>46.8</td>
<td>41.2</td>
<td>55</td>
<td>52.7</td>
</tr>
<tr>
<td>41.2</td>
<td>55</td>
<td>52.7</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>52.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Execution time @ 20MHz</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
</tr>
</thead>
<tbody>
<tr>
<td>819.8 μs</td>
<td>717.35 μs</td>
<td>410.2 μs</td>
<td>1025.8 μs</td>
</tr>
<tr>
<td>717.35 μs</td>
<td>410.2 μs</td>
<td>1025.8 μs</td>
<td></td>
</tr>
<tr>
<td>410.2 μs</td>
<td>1025.8 μs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Memory element: main memory (SRAM, FIFO) + exponent & index table, cache.