The Serial Commutator FFT

Mario Garrido Gálvez, Shen-Jui Huang, Sau-Gee Chen and Oscar Gustafsson

Journal Article

N.B.: When citing this work, cite the original article.

©2016 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

http://dx.doi.org/10.1109/TCSII.2016.2538119
Postprint available at: Linköping University Electronic Press

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-132529
The Serial Commutator (SC) FFT

Mario Garrido, Member, IEEE, Shen-Jui Huang, Sau-Gee Chen and Oscar Gustafsson, Senior Member, IEEE

Abstract—This paper presents a new type of FFT hardware architectures called serial commutator (SC) FFT. The SC FFT is characterized by the use of circuits for bit-dimension permutation of serial data. The proposed architectures are based on the observation that in the radix-2 FFT algorithm only half of the samples at each stage must be rotated. This fact, together with a proper data management makes it possible to allocate rotations only every other clock cycle. This allows for simplifying the rotator, halving the complexity with respect to conventional serial FFT architectures. Likewise, the proposed approach halves the number of adders in the butterflies with respect to previous architectures. As a result, the proposed architectures use the minimum number of adders, rotators and memory that are necessary for a pipelined FFT of serial data, with 100% utilization ratio.

Index Terms—Serial Commutator (SC), FFT, Pipelined Architecture

I. INTRODUCTION

The fast Fourier transform (FFT) is one of the most important algorithms in signal processing. Many hardware FFT architectures have been proposed with the aims of speeding up the calculation of the FFT and reducing the amount of hardware resources.

Pipelined FFT architectures are the most common ones [1]–[9]. They process a continuous flow of data using a relatively small amount of resources. There are two main types of pipelined FFTs: Serial pipelined FFTs process one sample per clock cycle, whereas parallel pipelined FFTs process several samples in parallel per clock cycle.

Parallel FFT architectures have been widely developed. Nowadays, there exist multi-path delay commutator (MDC) FFT architectures that use the minimum amount of butterflies and memory, with 100% utilization ratio [1], as well as an efficient use of rotators.

Conversely, serial pipelined FFTs have not reached the efficiency of parallel ones yet. Typical radix-2 single-path delay feedback (SDF) FFTs [2] have a utilization ratio of 50% in butterflies and rotators. Other radices such as radix-4 [3], [4] and radix-$2^2$ [2] improve the use of rotators. However, they do not improve the efficiency of butterflies. Single-delay commutator (SDC) FFTs [5]–[8] improve the use of butterflies and rotators at the cost of larger memory. The same happens to the locally pipelined FFT [9]. Therefore, in all cases there is a trade-off among butterflies, rotators and memory.

II. THE FFT ALGORITHM

The $N$-point DFT of an input sequence $x[n]$ is defined as:

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

where $W_N^{nk} = e^{-j 2\pi nk / N}$.

In order to compute the DFT efficiently, the FFT based on the Cooley-Tukey algorithm [10] is most times used. The FFT reduces the number of operations from $O(N^2)$ for the DFT to $O(N \log N)$.

Figure 1 shows the flow graph of a 16-point radix-2 FFT decomposed according to decimation in frequency (DIF) [11]. The FFT is calculated in a series of $n = \log_2 N$ stages, where $\rho$ is the base of the radix, $r$, of the FFT, i.e. $r = \rho^s$. In the figure, the numbers at the input represent the index of the input sequence, whereas those at the output are the frequencies, $k$. This paper presents the serial commutator (SC) FFT. The SC FFT uses a novel data management based on circuits for bit-dimension permutation of serial data. The resulting SC FFT is the first one that requires the theoretical minimum amount of butterflies, rotators and memory with 100% utilization.

The paper is organized as follows. Section II reviews the FFT algorithm. Section III studies the theoretical boundaries of the hardware resources. Section IV presents in detail the SC FFT. Section V shows the case of natural I/O order. Section VI compares the proposed architectures to previous ones. Section VII presents experimental results. Finally, Section VIII summarizes the main conclusions of the paper.

M. Garrido and O. Gustafsson are with the Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, e-mail: mario.garrido.galvez@liu.se, oscar.gustafsson@liu.se

S.-J. Huang is with Novatek Corp., Hsinchu 300, Taiwan, e-mail: shen-ray.ee95@g2.nctu.edu.tw

S.-G. Chen is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, e-mail: sgchen@cc.nctu.edu.tw

This work was supported by the Swedish ELLIIT Program.

Copyright (c) 2016 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org
At each stage of the graph, \( s \in \{1, \ldots, n\} \), butterflies and rotations are calculated. Specifically, each number, \( \phi \), in between the stages indicates a rotation by:

\[
e^{-j \frac{2\pi}{N} \phi}
\]

As a consequence, if \( \phi = 0 \) no rotation must be carried out. Likewise, rotations by \( \phi \in \{N/4, N/2, 3N/4\} \) are trivial. This means that they can be carried out in hardware simply by interchanging the real and imaginary components and/or changing the sign of the data.

### III. Theoretical Boundaries

The serial commutator (SC) FFT is based on a simple observation. In Figure 1, each stage calculates \( N \) complex additions and \( N/2 \) rotations. Therefore, any radix-2 FFT architecture that processes one sample per clock cycle only needs a complex adder and half a rotator per stage. This leads to \( \log_2 N \) butterflies and \( \log_2 N - 1 \) rotators for the entire FFT, considering that the last stage does not have a rotator. These are the theoretical minimum number of resources for any radix-2 FFT that processes one sample per clock cycle.

Regarding memory, the theoretical minimum \( N - P \) [1] for \( P \)-parallel data also holds for serial data where \( P = 1 \). Thus, the minimum memory for a serial FFT is \( N - 1 \).

### IV. The Serial Commutator FFT

Figures 2(a) and 2(b) show the proposed serial commutator FFT for \( N = 16 \) points and radix-2, respectively for DIF and DIT. The architectures consist of \( n = \log_2 N = 4 \) stages that include butterflies, rotators and circuits for data management. Rotators that carry out trivial rotations are diamond-shaped whereas general rotators are represented by a circle.

Both butterflies and rotators are marked with 1/2. This means that they require half of the components in conventional butterflies and rotators: butterflies only use a real adder and a real subtracter instead of complex ones, and rotators use two real multipliers and one adder instead of four real multipliers and two adders. The half butterfly and half rotator form the processing element (PE) of the architecture, which is explained in detail in section IV-A.

The circuits for data permutation are circuits for elementary bit exchange. These circuits have already been used for the calculation of the bit reversal [12]. However, this is the first time that this type of circuits are used in an FFT architecture. The data management of the SC FFT is explained in section IV-B.

#### A. Processing Element

Figure 3 shows in detail the processing element (PE) used to calculate the butterflies and rotations of the radix-2 DIF SC FFT in Fig. 2(a). The processing element for the DIT SC FFT in Fig. 2(b) is analogous. The only difference is that the rotator is placed before the butterfly.

The PE is composed by the half butterfly and the half rotator. The PE does the calculation of a butterfly followed by a rotator:

\[
\begin{align*}
Y_0 &= X_0 + X_1 \\
Y_1 &= (X_0 - X_1)e^{j\alpha}
\end{align*}
\]  

with the particularity that the inputs \( X_0 = X_{R0} + jX_{I0} \) and \( X_1 = X_{R1} + jX_{I1} \), and outputs \( Y_0 = Y_{R0} + Y_{I0} \) and \( Y_1 = Y_{R1} + Y_{I1} \), are provided in consecutive clock cycles.

Table I shows the timing diagram of the PE in Fig. 3. It can be observed that the butterfly operates first on the real part of the inputs and then on the imaginary part. The rotator also multiplexes the calculations in time. This allows for halving the adders and multipliers in the butterfly and rotator.

#### B. Data Management

The PE calculates a butterfly and a rotation on pairs of data that arrive in consecutive clock cycles. In order to fulfill this, the data management of the SC FFT places samples that must be operated together in consecutive clock cycles. This happens at all the stages of the FFT.

Figure 4 shows the data management of the SC FFTs in Fig. 2. The data management is the same for both DIF and DIF cases. Each column in Fig. 4 represents the input order to the corresponding stage. The order of arrival is from top to bottom. Therefore, \( x[0] \) and \( x[8] \) are the first and second inputs to the first stage, respectively. The figure shows that, at all the stages, consecutive samples are operated together in a butterfly. This allows for the use of the PE with half of the resources.
In order to achieve the desired order, the SC FFT uses circuits for bit-dimension permutations of serial data, as shown in Fig. 5. These circuits interchange pairs of data delayed by \( L \) clock cycles. In Fig. 4, the first, second and third stages interchange data separated 3, 1 and 7 clock cycles, respectively. These are equal to the lengths of the buffers of the three first stages in Fig. 2.

In a general case, for a SC FFT of length \( N \), the length and delay of the buffers at stages \( s = 1, \ldots, n - 2 \) is

\[
L = 2^{n-s-1} - 2^0,
\]

and \( L = 2^{n-1} \) at stage \( s = n - 1 \).

The control of the architecture is simple and obtained directly from the bits of an \( n \)-bit counter \( c_{n-1}, \ldots, c_0 \) that counts from 0 to \( N - 1 \). For a buffer of length \( L = 2^i - 1 \), the control signal \( S_i \) is:

\[
S_i = c_i \text{ OR } c_0
\]

The control signals must be delayed according to the pipeline of the architecture, so that the count starts when the first sample arrives at the corresponding shuffling circuit.

The total amount of memory for the shuffling circuits can be obtained by adding the delays at all the stages. This leads to a total memory of

\[
\sum_{i=1}^{n-1} 2^i - 2^0 = N - n - 1
\]
Therefore it only includes the permutations with buffer lengths \( L \).

The input and output orders of the SC FFT follow a sequence that is not in natural order, as shown in Fig. 4. In order to achieve natural I/O order, shuffling circuits can be added at the input and output. This is shown in Fig. 6 for a natural I/O 16-point SC FFT. The data management for the architecture in Fig. 6 is shown in Fig. 7.

In the general case of a natural I/O order, the number of elementary bit-exchanges have a buffer length \( \lfloor N/4 \rfloor \) for \( N \leq 64 \), and more for larger \( N \), being the differences small.

The output reordering circuit is more complex and requires \( \lceil (n+1)/2 \rceil \) elementary bit-exchanges in series, as shown in Fig. 9. The first two elementary bit-exchanges have a buffer length of \( L_a = 2^{n-2} - 2^0 = N/4 - 1 \) and \( L_b = 2^{n-1} - 2^{n-2} = N/4 \), respectively. The following \( \lceil (n-3)/2 \rceil \) elementary bit-exchanges have a buffer length \( L_i = 2^{n-i-2} - 2^i \) for \( i = 1, \ldots, \lceil (n-3)/2 \rceil \). In our example in Fig. 6 for \( N = 16 \) the number of elementary bit-exchanges in series is \( \lceil (n+1)/2 \rceil = \lceil (4+1)/2 \rceil = 2 \). Therefore it only includes the permutations with buffer lengths \( L_a = N/4 - 1 = 16/4 - 1 = 3 \) and \( L_b = N/4 = 16/4 = 4 \), as shown in Fig. 6.

In a general case, the shuffling circuits for natural I/O add an overhead to the total memory of approximately \( 5N/4 \), leading to a total memory of about \( 9N/4 \).

VI. COMPARISON AND ANALYSIS

Tables II and III compare pipelined FFT architectures for serial data. Table II does not impose any specific order of inputs and outputs, whereas table III compares architectures for natural I/O order.

In table II, the first column shows the type of architecture. The second, third and fourth columns show the resources used by the architecture: rotators, adders and data memory. The last two columns compares the performance in terms of latency and throughput. As all the architectures that are compared process serial data, the throughput of all of them is 1 sample per clock cycle.

In table II, it can be observed that previous architecture require the minimum of some of the hardware resources, but not all of them. Various SDF FFT architecture [2]–[4], [13] use the minimum amount of rotators and memory. Previous SDC FFTs [5], [6], [8] achieve the minimum number of adders. And the locally pipelined FFT [9] achieves the minimum number of rotators and adders. Finally, the proposed SC FFT is the first architecture that achieves the minimum amount in all hardware resources.

For natural I/O order, table III compares previous SDC architectures to the proposed SC FFT. Compared to [5]–[7] the proposed architecture reduces the number of rotators by 50%. Furthermore, up to \( N = 64 \) points the memory of the proposed approach is also smaller than that in [5]–[7]. Compared to [8], the proposed architecture has less memory for \( N \leq 64 \), and more for larger \( N \), being the differences small.

VII. EXPERIMENTAL RESULTS

The proposed SC FFT for \( N = 1024 \) points and word length 16 bits has been implemented on ASIC technology using the library UMC 55 nm process. Table IV compares the implementation with previous serial FFTs on ASICs. The proposed architecture improves the clock frequency of previous designs. At the same time it achieves less area than previous 2048-point [16] and 256-point [17] SDF FFTs, high SQNR and low power consumption. In the table, area and power are normalized to 55 nm and 0.9 V according to [18].

VIII. CONCLUSIONS

This paper has presented the serial commutator (SC) FFT architecture. This architecture is the first FFT to use circuits to calculate bit-dimension permutation on serial data. This creates a data management that allows for using the theoretical minimum amount of hardware resources for a serial FFT with 100% utilization. Compared to previous designs, the proposed
TABLE II
Comparison of Pipelined Hardware Architectures for the Computation of an N-point FFT on Serial Data.

<table>
<thead>
<tr>
<th>PIPELINED ARCHITECTURE</th>
<th>AREA</th>
<th>PERFORMANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Complex Rotators</td>
<td>Complex Adders</td>
</tr>
<tr>
<td>SDF Radix-2, [2]</td>
<td>2(log₂ N) - 1</td>
<td>4(log₂ N)</td>
</tr>
<tr>
<td>SDF Radix-2, [9]</td>
<td>log₂ N - 1</td>
<td>2(log₂ N)</td>
</tr>
<tr>
<td>SDF Radix-4, [3], [4]</td>
<td>log₂ N - 1</td>
<td>8(log₂ N)</td>
</tr>
<tr>
<td>SDF Radix-2², [2]</td>
<td>log₂ N - 1</td>
<td>4(log₂ N)</td>
</tr>
<tr>
<td>SDF Split-radix, [13]</td>
<td>log₂ N - 1</td>
<td>4(log₂ N)</td>
</tr>
<tr>
<td>SDC Radix-2, [6], [7]</td>
<td>2(log₂ N) - 1</td>
<td>2(log₂ N)</td>
</tr>
<tr>
<td>SDC Radix-2, [5]</td>
<td>2(log₂ N) - 1</td>
<td>2(log₂ N)</td>
</tr>
<tr>
<td>SDC Radix-4, [14]</td>
<td>log₂ N - 1</td>
<td>3(log₂ N)</td>
</tr>
<tr>
<td>SDC-SDF Radix-2, [8]</td>
<td>log₂ N - 1</td>
<td>2(log₂ N) + 1</td>
</tr>
<tr>
<td>Proposed SC Radix-2</td>
<td>log₂ N - 1</td>
<td>2(log₂ N)</td>
</tr>
</tbody>
</table>

TABLE III
Comparison of Pipelined Hardware Architectures for the Computation of an N-point FFT on Serial Data with Natural I/O.

<table>
<thead>
<tr>
<th>PIPELINED ARCHITECTURE</th>
<th>AREA</th>
<th>PERFORMANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Complex Rotators</td>
<td>Complex Adders</td>
</tr>
<tr>
<td>SDC Radix-2, [6], [7]</td>
<td>2(log₂ N) - 1</td>
<td>2(log₂ N)</td>
</tr>
<tr>
<td>SDC Radix-2, [5]</td>
<td>2(log₂ N) - 1</td>
<td>2(log₂ N)</td>
</tr>
<tr>
<td>SDC-SDF Radix-2, [8]</td>
<td>log₂ N - 1</td>
<td>2(log₂ N) + 1</td>
</tr>
<tr>
<td>Proposed SC Radix-2, even N</td>
<td>log₂ N - 1</td>
<td>2(log₂ N) + 1</td>
</tr>
<tr>
<td>Proposed SC Radix-2, odd N</td>
<td>log₂ N - 1</td>
<td>2(log₂ N) + 1</td>
</tr>
</tbody>
</table>

TABLE IV
Comparison of serial FFTs implemented on ASICs.

<table>
<thead>
<tr>
<th>Proposed</th>
<th>1024</th>
<th>64</th>
<th>2048</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radix</td>
<td>2</td>
<td>3²</td>
<td>2²</td>
<td>2²</td>
</tr>
<tr>
<td>Architecture</td>
<td>SC</td>
<td>SDF</td>
<td>SDF</td>
<td>SDF</td>
</tr>
<tr>
<td>Word length</td>
<td>16</td>
<td>18</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>Technology (nm)</td>
<td>55</td>
<td>180</td>
<td>350</td>
<td>180</td>
</tr>
<tr>
<td>Voltage (V)</td>
<td>0.9</td>
<td>2.7</td>
<td>2.7</td>
<td>1.8</td>
</tr>
<tr>
<td>Clk (MHz)</td>
<td>200</td>
<td>166</td>
<td>76</td>
<td>51.5</td>
</tr>
<tr>
<td>Area (mm²)</td>
<td>0.15</td>
<td>0.47</td>
<td>7.58</td>
<td>-</td>
</tr>
<tr>
<td>Norm. Area</td>
<td>0.15</td>
<td>0.04</td>
<td>0.19</td>
<td>-</td>
</tr>
<tr>
<td>Gate Count</td>
<td>134066</td>
<td>-</td>
<td>173875</td>
<td>-</td>
</tr>
<tr>
<td>SQRN (dB)</td>
<td>55</td>
<td>-</td>
<td>45.3</td>
<td>-</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>8.0</td>
<td>29.7</td>
<td>526</td>
<td>-</td>
</tr>
<tr>
<td>Norm. Power</td>
<td>8.0</td>
<td>-</td>
<td>9.18</td>
<td>-</td>
</tr>
</tbody>
</table>

SC FFT reduces either the number of rotator, or the number of adders or the memory of the design. A solution for natural I/O has also been presented, which offers comparable results to previous natural I/O FFTs. Finally, experimental results have been obtained to verify the architecture, leading to small area and low power consumption.

REFERENCES