A New Approach to Pipeline FFT Processor

Shousheng He and Mats Torkelson
Department of Applied Electronics, Lund University
S-22100 Lund, SWEDEN
email: he@tde.lth.se; torkel@tde.lth.se

***Abstract*— A new VLSI architecture for real-time pipeline FFT processor is proposed. A hardware oriented radix-2^2 algorithm is derived by integrating a twiddle factor decomposition technique in the divide and conquer approach. Radix-2^2 algorithm has the same multiplicative complexity as radix-4 algorithm, but retains the butterfly structure of radix-2 algorithm. The single-path delay-feedback architecture is used to exploit the spatial regularity in signal flow graph of the algorithm. For length-N DFT computation, the hardware requirement of the proposed architecture is minimal on both dominant components: $\log_4 N - 1$ complex multipliers and $N - 1$ complex data memory. The validity and efficiency of the architecture have been verified by simulation in hardware description language VHDL.**

I. Introduction

Pipeline FFT processor is a specified class of processors for DFT computation utilizing fast algorithms. It is characterized with real-time, non-stopping processing as the data sequence passing the processor. It is an AT^2 non-optimal approach with $AT^2 = O(N^3)$, since the area lower bound is $O(N)$. However, as it has been speculated [1] that for real-time processing whether a new metric should be introduced since it is *necessarily non-optimal* given the time complexity of $O(N)$. Although asymptotically almost all the feasible architectures have reached the area lower bound [2], the class of pipeline FFT processors has probably the smallest "constant factor" among the approaches that meet the time requirement, due to its least number, $O(\log N)$, of Arithmetic Elements (AE). The difference comes from the fact that an AE, especially the multiplier, takes much larger area than a register in digital VLSI implementation.

It is also interesting to note the at least $\Omega(\log N)$ AEs are necessary to meet the real-time processing requirement due to the computational complexity of $\Omega(N \log N)$ for FFT algorithm. Thus it has the nature of "lower bound" for AE requirement. Any "optimal" architecture for real-time processing will likely have $\Omega(\log N)$ AEs.

Another major area/energy consumption of the FFT processor comes from the memory requirement to buffer the input data and the intermediate result for the computation. For large size transform, this turns out to be dominating [3, 4]. Although there is no formal proof, the area lower bound indicates that the the "lower bound" for the number of registers is likely to be $\Omega(N)$. This is obviously true for any architecture implementing FFT based algorithm, since the butterfly at first stage has to take data elements separated N/r distance away from the input sequence, where r is a small constant integer, or the "radix".

Putting above arguments together, a pipeline FFT processor has necessarily $\Omega(\log_r N)$ AEs and $\Omega(N)$ complex word registers. The optimal architecture has to be the one that reduces the "constant factor", or the absolute number of AEs (multipliers and adders) and memory size, to the minimum.

In this paper a new approach for real-time pipeline FFT processor, the Radix-2^2 Single-path Delay Feedback, or R2^2SDF architecture will be presented. We will begin with a brief review of previous approaches. A hardware oriented radix-2^2 algorithm is then developed by integrating a twiddle factor decomposition technique in divide and conquer approach to form a spatially regular signal flow graph (SFG). Mapping the algorithm to the cascading delay feedback structure leads to the the proposed architecture. Finally we conclude with a comparison of hardware requirement of R2^2SDF and several other popular pipeline architectures.

II. Pipeline FFT processor architectures

Before going into details of the new approach, it is beneficial to have a brief review of the various architectures for pipeline FFT processors. To avoid being influenced by the sequence order, we assume that the real-time processing task only requires the input sequence to be in normal order, and the output is allowed to be in digit-reversed (radix-2 or radix-4) order, which is permissible in such applications such as DFT based communication system [5]. We also stick to the Decimation-In-Frequency (DIF) type of decomposition throughout the discussion.

The architecture design for pipeline FFT processor had been the subject of intensive research as early as in 70's when

real-time processing was demanded in such application as radar signal processing [6], well before the VLSI technology had advanced to the level of system integration. Several architectures have been proposed over the last 2 decades since then, along with the increasing interest and the leap forward of the technology. Here different approaches will be put into functional blocks with unified terminology, where the additive butterfly has been separated from multiplier to show the hardware requirement distinctively, as in Fig. 1. The control and twiddle factor reading mechanism have been also omitted for clarity. All data and arithmetic operations are complex, and a constraint that N is a power of 4 applies.

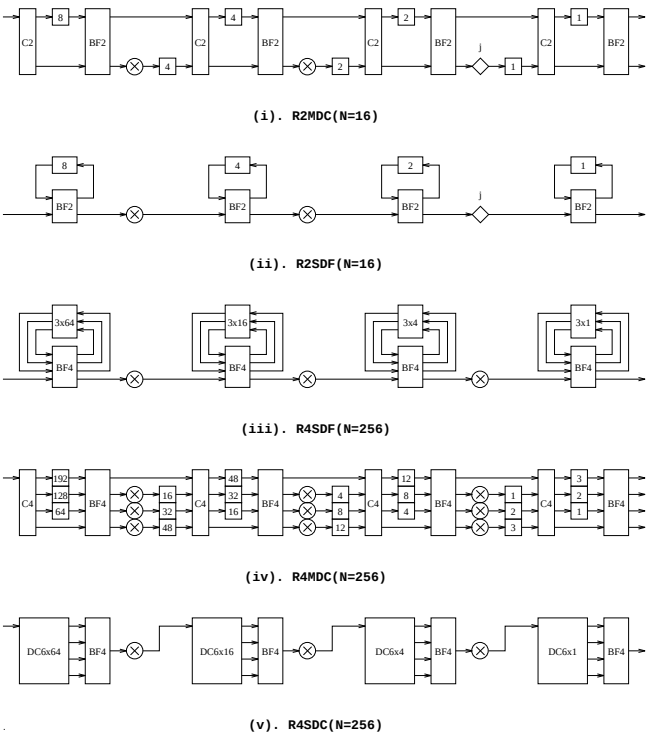


Figure 1: Various schemes for pipeline FFT processor

R2MDC: Radix-2 Multi-path Delay Commutator [6] was probably the most straightforward approach for pipeline implementation of radix-2 FFT algorithm. The input sequence has been broken into two parallel data stream flowing forward, with correct "distance" between data elements entering the butterfly scheduled by proper delays. Both butterflies and multipliers are in 50% utilization. $\log_2 N-2$ multipliers, $\log_2 N$ radix-2 butterflies and $3/2N-2$ registers (delay elements) are required.

R2SDF: Radix-2 Single-path Delay Feedback [7] uses the registers more efficiently by storing the butterfly output in feedback shift registers. A single data stream goes through the multiplier at every stage. It has same number of butterfly units and multipliers as in R2MDC approach, but with much reduced memory requirement: $N-1$ registers. Its memory requirement is minimal.

R4SDF: Radix-4 Single-path Delay Feedback [8] was proposed as a radix-4 version of R2SDF, employing CORDIC[1] iterations. The utilization of multipliers has been increased to 75% due to the storage of 3 out of radix-4 butterfly outputs. However, the utilization of the radix-4 butterfly, which is fairly complicated and contains at least 8 complex adders, is dropped to only 25%. It requires $\log_4 N-1$ multipliers, $\log_4 N$ full radix-4 butterflies and storage of size $N-1$.

R4MDC: Radix-4 Multi-path Delay Commutator [6] is a radix-4 version of R2MDC. It has been used as the architecture for the initial VLSI implementation of pipeline FFT processor [3] and massive wafer scale integration [9]. However, it suffers from low, 25%, utilization of all components, which can be compensated only in some special applications where four FFTs are being processed simultaneously. It requires $3\log_4 N$ multipliers, $\log_4 N$ full radix-4 butterflies and $5/2N-4$ registers.

R4SDC: Radix-4 Single-path Delay Commutator [10] uses a modified radix-4 algorithm with programable $1/4$ radix-4 butterflies to achieve higher, 75% utilization of multipliers. A combined Delay-Commutator also reduces the memory requirement to $2N-2$ from $5/2N-1$, that of R4MDC. The butterfly and delay-commutator become relatively complicated due to programmability requirement. R4SDC has been used recently in building the largest ever single chip pipeline FFT processor for HDTV application [4].

A swift skimming through of the architectures listed above reveals the distinctive merits of the different approaches: First, the delay-feedback approaches are always more efficient than corresponding delay-commutator approaches in terms of memory utilization since the stored butterfly output can be directly used by the multipliers. Second, radix-4 algorithm based single-path architectures have higher multiplier utilization, however, radix-2 algorithm based architectures have simpler butterflies which are better utilized. The new approach developed in following sections is highly motivated by these observations.

III. Radix-2^2 DIF FFT algorithm

By the observations made in last section the most desirable *hardware oriented* algorithm will be that it has the same number of non-trivial multiplications at the same positions in the SFG as of radix-4 algorithms, but has the same butterfly structure as that of radix-2 algorithms. Strictly speaking, algorithms with this feature is not completely new. An SFG

[1]The Coordinate Rotational Digital Computer

with a complex "bias" factor had been obtained implicitly as the result of constant-rotation/compensation procedure using restricted CORDIC operations [11]. Another algorithm combining radix-4 and radix-'$4+2$' in DIT form has been used to decrease the scaling error in R2MDC architecture, without altering the multiplier requirement [12]. The clear derivation of the algorithm in DIF form with perception of reducing the hardware requirement in the context pipeline FFT processor is, however, yet to be developed.

To avoid confusing with the well known radix-$2/4$ split radix algorithm and the mixed radix-'$4+2$' algorithm, the notion of *radix*-2^2 algorithm is used to clearly reflect the structural relation with radix-2 algorithm and the identical computational requirement with radix-4 algorithm.

The DFT of size N is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad 0 \le k < N \tag{1}$$

where W_N denotes the Nth primitive root of unity, with its exponent evaluated modulo N. To make the derivation of the new algorithm clearer, consider the first 2 steps of decomposition in the radix-2 DIF FFT together. Applying a 3-dimensional linear index map,

$$\begin{aligned} n &= < \tfrac{N}{2} n_1 + \tfrac{N}{4} n_2 + n_3 >_N \\ k &= < k_1 + 2k_2 + 4k_3 >_N \end{aligned} \tag{2}$$

the Common Factor Algorithm (CFA) has the form of

$$\begin{aligned} & X(k_1 + 2k_2 + 4k_3) \\ &= \sum_{n_3=0}^{\frac{N}{4}-1} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x(\tfrac{N}{2} n_1 + \tfrac{N}{4} n_2 + n_3) W_N^{(\frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3)(k_1 + 2k_2 + 4k_3)} \\ &= \sum_{n_3=0}^{\frac{N}{4}-1} \sum_{n_2=0}^{1} \{ B_{\frac{N}{2}}^{k_1} (\tfrac{N}{4} n_2 + n_3) W_N^{(\frac{N}{4} n_2 + n_3) k_1} \} W_N^{(\frac{N}{4} n_2 + n_3)(2k_2 + 4k_3)} \end{aligned} \tag{3}$$

where the butterfly structure has the form of

$$B_{\frac{N}{2}}^{k_1} (\frac{N}{4} n_2 + n_3) = x(\frac{N}{4} n_2 + n_3) + (-1)^{k_1} x(\frac{N}{4} n_2 + n_3 + \frac{N}{2})$$

If the expression within the braces of eqn. (3) is to be computed before further decomposition, an ordinary radix-2 DIF FFT results. The key idea of the new algorithm is to proceed the second step decomposition to the remaining DFT coefficients, *including the "twiddle factor"* $W_N^{(\frac{N}{4} n_2 + n_3) k_1}$, to exploit the exceptional values in multiplication before the next butterfly is constructed. Decomposing the composite twiddle factor and observe that

$$\begin{aligned} & W_N^{(\frac{N}{4} n_2 + n_3)(k_1 + 2k_2 + 4k_3)} \\ &= W_N^{N n_2 k_3} W_N^{\frac{N}{4} n_2 (k_1 + 2k_2)} W_N^{n_3 (k_1 + 2k_2)} W_N^{4 n_3 k_3} \\ &= (-j)^{n_2 (k_1 + 2k_2)} W_N^{n_3 (k_1 + 2k_2)} W_N^{4 n_3 k_3} \end{aligned} \tag{4}$$

Substituting eqn. (4) in eqn. (3) and expand the summation with index n_2. After simplification we have a set of 4 DFTs of length $N/4$,

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \Big[H(k_1, k_2, n_3) W_N^{n_3 (k_1 + 2k_2)} \Big] W_{\frac{N}{4}}^{n_3 k_3} \tag{5}$$

where $H(k_1, k_2, n_3)$ is expressed in eqn. (6).

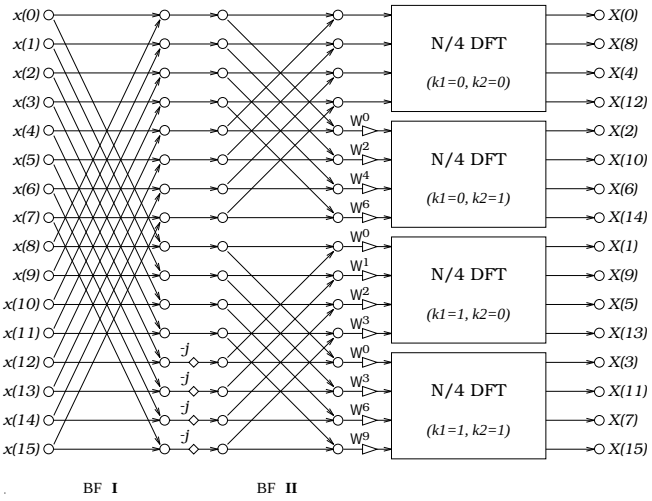


Figure 2: Butterfly with decomposed twiddle factors.

eqn. (6) represents the first two stages of butterflies with only trivial multiplications in the SFG, as BF **I** and BF **II** in Fig. 2. After these two stages, full multipliers are required to compute the product of the decomposed twiddle factor $W_N^{n_3 (k_1 + 2k_2)}$ in eqn. (5), as shown in Fig. 2. Note the order of the twiddle factors is different from that of radix-4 algorithm.

Applying this CFA procedure recursively to the remaining DFTs of length $N/4$ in eqn. (5), the complete radix-2^2 DIF FFT algorithm is obtained. An $N = 16$ example is shown in Fig. 3 where small diamonds represent trivial multiplication by $W_N^{N/4} = -j$, which involves only real-imaginary swapping and sign inversion.

Radix-2^2 algorithm has the feature that it has the same multiplicative complexity as radix-4 algorithms, but still retains the radix-2 butterfly structures. The multiplicative operations are in a such an arrangement that only every other stage has non-trivial multiplications. This is a great structural advantage over other algorithms when pipeline/cascade FFT architecture is under consideration.

IV. R2^{2}SDF ARCHITECTURE

Mapping radix-2^2 DIF FFT algorithm derived in last section to the R2SDF architecture discussed in section II,a new architecture of Radix-2^2 Single-path Delay Feedback (R2^{2}SDF) approach is obtained.

$$H(k_1,k_2,n_3)=\overbrace{\underbrace{\left[x(n_3)+(-1)^{k_1}x(n_3+\frac{N}{2})\right]}_{\text{BF I}}+(-j)^{(k_1+2k_2)}\underbrace{\left[x(n_3+\frac{N}{4})+(-1)^{k_1}x(n_3+\frac{3}{4}N)\right]}_{\text{BF I}}}^{\text{BF II}} \quad (6)$$

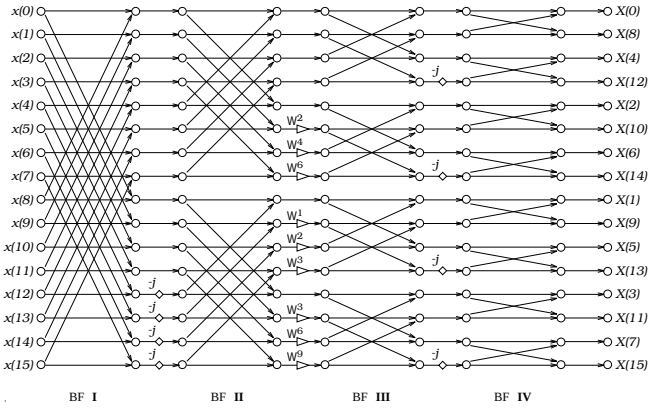


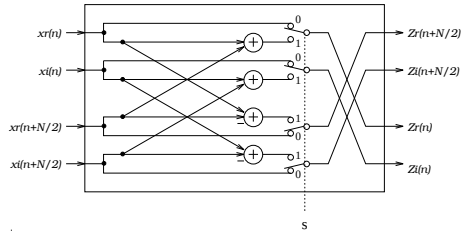
Figure 3: Radix-2^2 DIF FFT flow graph for $N = 16$

Fig. 5 outlines an implementation of the R2^{2}SDF architecture for $N = 256$, note the similarity of the data-path to R2SDF and the reduced number of multipliers. The implementation uses two types of butterflies, one identical to that in R2SDF, the other contains also the logic to implement the trivial twiddle factor multiplication, as shown in Fig. 4-(i)(ii) respectively. Due to the spatial regularity of Radix-2^2 algorithm, the synchronization control of the processor is very simple. A $(\log_2 N)$-bit binary counter serves two purposes: synchronization controller and address counter for twiddle factor reading in each stages.

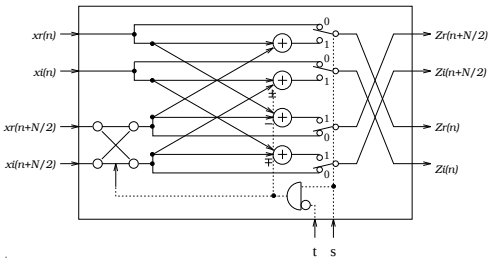
With the help of the butterfly structures shown in Fig. 4, the scheduled operation of the R2^{2}SDF processor in Fig. 5 is as follows. On first $N/2$ cycles, the 2-to-1 multiplexors in the first butterfly module switch to position "0", and the butterfly is idle. The input data from left is directed to the shift registers until they are filled. On next $N/2$ cycles, the multiplexors turn to position "1", the butterfly computes a 2-point DFT with incoming data and the data stored in the shift registers.

$$\begin{array}{ll} Z1(n) & = x(n)+x(n+N/2) \\ Z1(n+N/2) & = x(n)-x(n+N/2) \end{array} \quad ,0 \le n < N/2 \quad (7)$$

The butterfly output $Z1(n)$ is sent to apply the twiddle factor, and $Z1(n+N/2)$ is sent back to the shift registers to be "multiplied" in still next $N/2$ cycles when the first half of the next frame of time sequence is loaded in. The operation of the second butterfly is similar to that of the first one, except the "distance" of butterfly input sequence are just $N/4$ and the trivial twiddle factor multiplication has been implemented by real-imaginary swapping with a commutator and controlled add/subtract operations, as in Fig. 4-(ii), which requires two bit control signal from the synchronizing counter. The data then goes through a full complex multiplier, working at 75% utility, accomplishes the result of first level of radix-4 DFT word by word. Further processing repeats this pattern with the distance of the input data decreases by half at each consecutive butterfly stages. After $N-1$ clock cycles, The complete DFT transform result streams out to the right, in bit-reversed order. The next frame of transform can be computed without pausing due to the pipelined processing of each stages.



(i). BF2I



(ii). BF2II

Figure 4: Butterfly structure for R2^{2}SDF FFT processor

In practical implementation, pipeline register should be inserted between each multiplier and butterfly stage to improve the performance. Shimming registers are also needed for control signals to comply with thus revised timing. The latency of the output is then increased to $N-1+3(\log_4 N-1)$ without affecting the throughput rate.

V. Conclusion

In this paper, a hardware-oriented radix-2^2 algorithm is derived which has the radix-4 multiplicative complexity but retains radix-2 butterfly structure in the SFG. Based on this algorithm, a new, efficient pipeline FFT architecture, the R2^{2}SDF architecture, is put forward. The hardware requirement of proposed architecture as compared with various approaches is shown in Table 1, where not only the number of complex

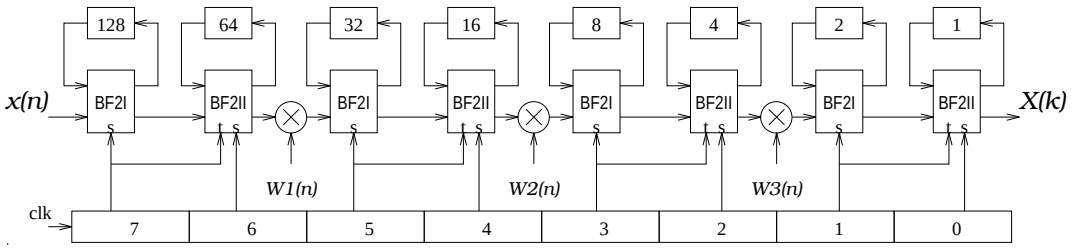


Figure 5: R2^{2}SDF pipeline FFT architecture for $N = 256$

multipliers, adders and memory size but also the control complexity are listed for comparison. For easy reading, base-4 logarithm is used whenever applicable. It shows R2^{2}SDF has reached the minimum requirement for both multiplier and the storage, and only second to R4SDC for adder. This makes it an ideal architecture for VLSI implementation of pipeline FFT processors.

Table 1: Hardware requirement comparison

	multiplier #	adder #	memory size	control
R2MDC	$2(\log_4 N - 1)$	$4\log_4 N$	$3N/2 - 2$	simple
R2SDF	$2(\log_4 N - 1)$	$4\log_4 N$	$N - 1$	simple
R4SDF	$\log_4 N - 1$	$8\log_4 N$	$N - 1$	medium
R4MDC	$3(\log_4 N - 1)$	$8\log_4 N$	$5N/2 - 4$	simple
R4SDC	$\log_4 N - 1$	$3\log_4 N$	$2N - 2$	complex
R2^{2}SDF	$\log_4 N - 1$	$4\log_4 N$	$N - 1$	simple

The architecture has been modeled with hardware description language VHDL with generic parameters for transform size and word-length, using fixed point arithmetic and a complex array multiplier implemented with distributed arithmetic. The validity and efficiency of the proposed architecture has been verified by extensive simulation.

REFERENCES

[1] C. D. Thompson. Fourier transform in VLSI. *IEEE Trans. Comput.*, C-32(11):1047–1057, Nov. 1983.

[2] S. He and M. Torkelson. A new expandable 2D systolic array for DFT computation based on symbiosis of 1D arrays. In *Proc. ICA^{3}PP'95*, pages 12–19, Brisbane, Australia, Apr. 1995.

[3] E. E. Swartzlander, W. K. W. Young, and S. J. Joseph. A radix 4 delay commutator for fast Fourier transform processor implementation. *IEEE J. Solid-State Circuits*, SC-19(5):702–709, Oct. 1984.

[4] E. Bidet, D. Castelain, C. Joanblanq, and P. Stenn. A fast single-chip implementation of 8192 complex point FFT. *IEEE J. Solid-State Circuits*, 30(3):300–305, Mar. 1995.

[5] M. Alard and R. Lassalle. Principles of modulation and channel coding for digital broadcasting for mobile receivers. *EBU Review*, (224):47–69, Aug. 1987.

[6] L.R. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing*. Prentice-Hall, Inc., 1975.

[7] E.H. Wold and A.M. Despain. Pipeline and parallel-pipeline FFT processors for VLSI implementation. *IEEE Trans. Comput.*, C-33(5):414–426, May 1984.

[8] A.M. Despain. Fourier transform computer using CORDIC iterations. *IEEE Trans. Comput.*, C-23(10):993–1001, Oct. 1974.

[9] E. E. Swartzlander, V. K. Jain, and H. Hikawa. A radix 8 wafer scale FFT processor. *J. VLSI Signal Processing*, 4(2,3):165–176, May 1992.

[10] G. Bi and E. V. Jones. A pipelined FFT processor for word-sequential data. *IEEE Trans. Acoust., Speech, Signal Processing*, 37(12):1982–1985, Dec. 1989.

[11] A.M. Despain. Very fast Fourier transform algorithms hardware for implementation. *IEEE Trans. Comput.*, C-28(5):333–341, May 1979.

[12] R. Storn. Radix-2 FFT-pipeline architecture with raduced noise-to-signal ratio. *IEE Proc.-Vis. Image Signal Process.*, 141(2):81–86, Apr. 1994.