Designing Pipeline FFT Processor for OFDM (de)Modulation

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*— FFT processor is one of the key components in the implementation of wideband OFDM systems. Architectures with structured Pipeline have been used to meet the fast, real-time processing demand and low-power consumption requirement in a mobile environment. Architectures based on new forms of FFT, the radix-2^2 algorithm derived by cascade decomposition, is proposed. By exploiting the spatial regularity of the new algorithm, the requirement for both dominant elements in VLSI implementation, the memory size and the number of complex multipliers, have been minimized. Progressive wordlength adjustment has been introduced to optimize the total memory size with a given signal-to-quantization-noise-ratio (SQNR) requirement in fixed-point processing. A new complex multiplier based on distributed arithmetic further enhanced the area/power efficiency of the design. A single-chip processor for 1K complex point FFT transform is used to demonstrate the design issues under consideration.**

I. Introduction

Mobile communication faces a particularly hostile environment, simultaneously containing multi-path, interference and impulsive parasitic noise. The problem is further complicated by the scarcity of available spectrum resources and power supply for the mobile devices. Robust, bandwidth efficient systems with good performance in such an environment are therefore desirable for digital transmission.

Orthogonal Frequency Division Multiplexing (OFDM), the most spectrum efficient multi-carrier modulation technique, has been recently proposed to overcome the adverse effects of communication channels [1, 2, 3]. The OFDM technique transforms a highly-selective wide-band channel into a large number of non-selective narrow-band slices which are frequency multiplexed. as shown in Fig. 1. The extended symbol duration together with the employment of

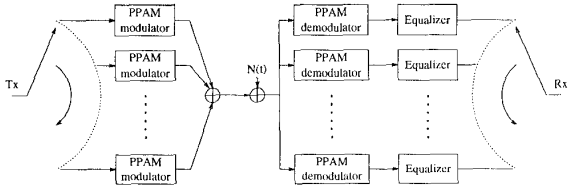

Figure 1: Principle of multi-carrier modulation

cyclic prefix reduces inter-symbol interference caused by delay spread. The narrow bandwidth of sub-channels has also made sophisticated channel equalization unnecessary. OFDM systems have been proposed for audio signal transmission to mobile receivers [4, 5], data intensive HDTV broadcasting programs [6] and cellular mobile communications [7].

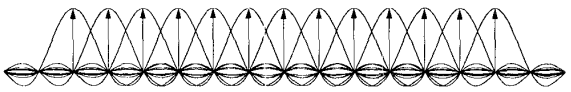

Figure 2: Carrier spectrum overlapping in OFDM systems

Unlike traditional frequency division multiplexing schemes, the spectra of sub-carriers in a OFDM system are mutually overlapped sinc functions, as shown in Fig. 2. The frequency division is achieved not by bandpass filtering, but by baseband processing utilizing the orthogonal property of the base function of the carriers. The implementation of wide-band OFDM system is only feasible when the equipment complexity and power consumption are greatly reduced by utilizing a real-time Fast Fourier Transform (FFT) processor to replace the bank of (de)modulators for each sub-carrier, eliminating individual pulse shaping mechanism [1, 3]. A simplified block diagram of an OFDM system utilizing the DFT transform is shown in Fig. 3.

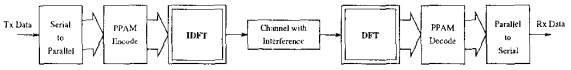

Figure 3: A simplified scheme for system utilizing OFDM modulation

FFT operation has been proven to be both computational intensive, in terms of arithmetic operations, and communicational intensive, in terms of data swapping/exchanging in the storage. For real-time processing of FFT transform, $O(\log N)$ arithmetic operations are required per sample cycle, where N is the length of the transform. High speed real-time processing can be accomplished in two different ways. In a conventional, general purpose DSP approach, the computation is carried out by a single processor driven to a very high clock frequency, which is $O(\log N)$ times the data sampling frequency. While in an application specific approach, parallel or concurrent/pipelined processors, operating on a clock frequency close or equivalent to the sampling frequency, are used to attain the performance. Analysis has shown that the second approach is more preferable

0-7803-4900-8/98/$10.00 © 1998 IEEE

when power consumption is limited by the application environment, such as in mobile communication [9].

Pipeline FFT processor is a class of architectures for application specific real-time DFT computation utilizing fast algorithms. It is characterized by non-stopping processing on a clock frequency of the input data sampling. A lower clock frequency is a clear advantage for pipeline architectures, when either a high speed processing or a low power solution is sought. In addition, pipeline structure is highly regular, which can be easily scaled and parameterized when Hardware Description Language (HDL) is used in the design. It is also flexible when forward/inverse transforms of different lengths are to be computed with the same chip.

In this paper architectures of pipeline FFT processor based on new hardware oriented FFT algorithms are presented. In the following section, previous approaches of pipeline FFT processors are briefly reviewed. Then hardware oriented FFTs, the Radix-2^i algorithms, are derived with a novel cascade decomposition approach. New architectures, compared favorably with previous ones, are conceived by mapping the algorithms to the pipeline structures. Finally we conclude by presenting a single chip implementation for 1K FFT processor, where functional and physical design issues are taken into consideration.

II. Pipeline FFT processor architectures

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 applications as radar signal processing [10]. Several architectures have been proposed over the last 2 decades. Here different approaches are put into functional blocks with unified terminology. The additive butterfly has been separated from multiplier to show the hardware requirement distinctively, as in Fig. 4. The control and twiddle factor reading mechanism have been omitted for clarity. All data and arithmetic operations are complex, and a constraint that N is a power of 4 applies.

R2MDC: Radix-2 Multi-path Delay Commutator [10] was probably the most classical 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 the 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 [11] uses the registers more efficiently by storing the one 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 [12] was proposed as a radix-4 version of R2SDF, employing CORDIC iterations. The utilization of multipliers has been increased to 75% by storing 3 out of 4 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 $N - 1$.

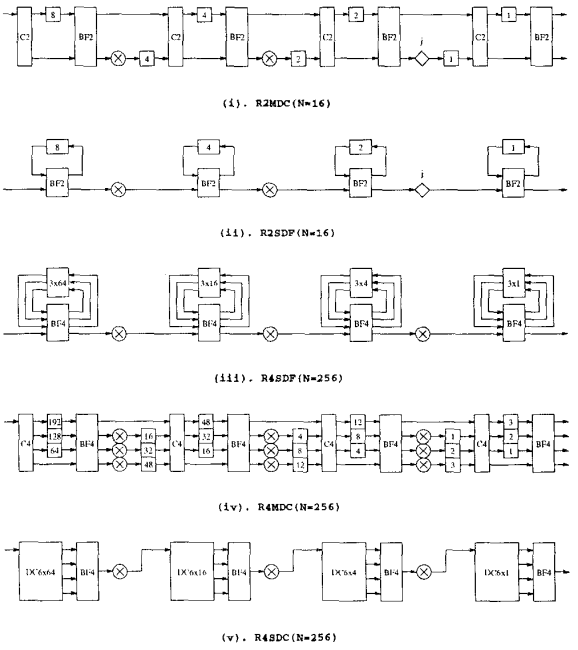

Figure 4: Various schemes for pipeline FFT processor

R4MDC: Radix-4 Multi-path Delay Commutator [10] is a radix-4 version of R2MDC. It has been used as the architecture for the initial VLSI implementation of pipeline FFT processor [13] and massive wafer scale integration [14]. 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 [15] uses a modified radix-4 algorithm with programable 1/4 radix-4 butterflies to achieve higher, 75% utilization of multipliers. A multiplexed 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 [16].

A swift skimming through of the architectures listed above reveals the distinctive merits and common requirements 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 butterfly output share the same storage with its input. Second, pipeline architectures require FFT al-

gorithms to be formulated in a "hardware-oriented" form, where spatial regularity is preserved in the Signal Flow Graph (SFG) so that arithmetic operations can be tightly scheduled for efficient hardware utilization. The new approach developed in the following sections is highly motivated by these observations.

III. Hardware-oriented FFT by Cascade Decomposition

The DFT of length N is defined by

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

where W_N, the so called "twiddle factor", denotes the N-th primitive root of unity, with its exponent evaluated modulo N. Most fast algorithms share the same general strategy, i.e. mapping the one-dimensional transform into a two or multi-dimensional representation, then exploiting the congruence property of its coefficients to simplify the computation. Unlike traditional step-by-step decomposition of the twiddle factors, it will be shown in this section that by cascading the twiddle factor decomposition, new forms of FFT with high spatial regularity can be derived. Throughout the derivation, we stick to Decimation In Frequency (DIF) type of decomposition. Decimation In Time type algorithms can be easily obtained by applying the transposition theorem to the algorithm in DIF form [18].

A. Radix-2^2 DIF FFT

In the classical divide and conquer procedure leading to a radix-2 DIF FFT, consider the first 2 steps of decomposition together. Applying a 3-dimensional linear index map,

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

the Common Factor Algorithm (CFA) [19] takes the form of

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

where the first butterfly structure can be written as

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

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 approach is to proceed the decomposition to the remaining short DFTs, *cascading the "twiddle factor" $W_N^{(\frac{N}{4} n_2 + n_3) k_1}$ into the next step decomposition* to exploit the exceptional values in the multiplication, before the butterflies are constructed. Decompose the cascade twiddle factor and observe that

$$\begin{aligned} & W_N^{(\frac{N}{4} n_2 + n_3) k_1} W_N^{(\frac{N}{4} n_2 + n_3)(2k_2 + 4k_3)} \\ &= W_N^{N n_2 k_3} W_N^{\frac{N}{4} n_2 (k_1 + 2k_2)} W_N^{n_3 (k_1 + 2k_2)} W_N^{4 n_3 k_3} \\ &= (-j)^{n_2 (k_1 + 2k_2)} W_N^{n_3 (k_1 + 2k_2)} W_N^{4 n_3 k_3} \end{aligned} \qquad (5)$$

Substitute eqn. (5) into eqn. (3) and expand the summation with regard to index n_2. After simplification we have a set of 4 DFTs of length $N/4$,

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \left[H_{\frac{N}{4}}(n_3, k_1, k_2) W_N^{n_3 (k_1 + 2k_2)} \right] W_{\frac{N}{4}}^{n_3 k_3} \qquad (6)$$

where a secondary butterfly structure can be expressed as

$$H_{\frac{N}{4}}(n_3, k_1, k_2) = \overbrace{B_{\frac{N}{2}}(n_3, k_1)}^{\text{BF I}} + (-j)^{(k_1 + 2k_2)} \overbrace{B_{\frac{N}{2}}(n_3 + \tfrac{N}{4}, k_1)}^{\text{BF I}} \qquad \text{BF II} \qquad (7)$$

Eqn. (4) and eqn. (7) represent the first two columns of butterflies with only trivial multiplications in the SFG of the radix-2^2 algorithm. After these two columns, full multiplications are used to apply the decomposed twiddle factor $W_N^{n_3 (k_1 + 2k_2)}$ in eqn. (6). Note that the order of the twiddle factors is different from that of a radix-4 algorithm.

Applying CFA with this cascade decomposition recursively to the remaining DFTs of length $N/4$ in eqn. (6), the complete radix-2^2 DIF FFT algorithm is obtained. An $N = 64$ example is shown in Fig. 5 where small diamonds represent trivial multiplications by $W_N^{N/4} = -j$, which involves only real-imaginary swapping and sign inversion.

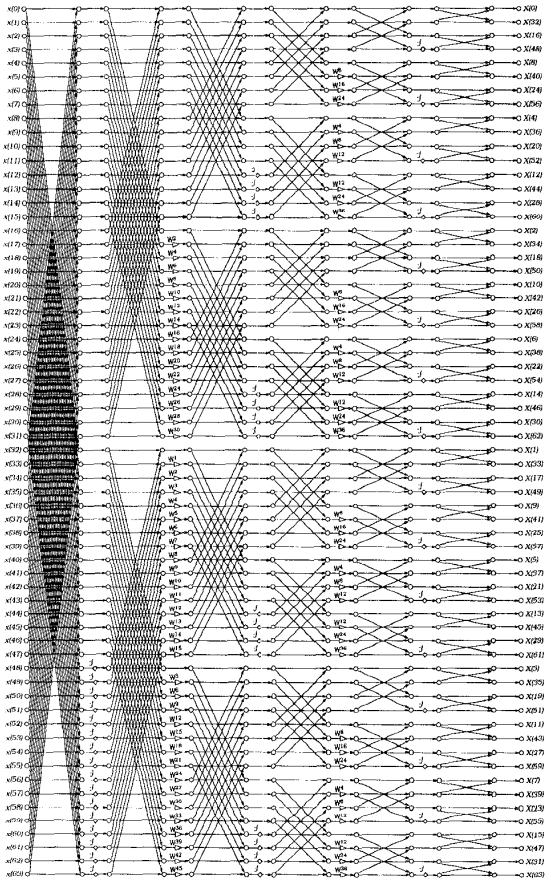

Figure 5: Radix-2^2 DIF FFT signal flow graph for $N = 64$

Radix-2^2 algorithm is characterized with the trait that it has 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 columns have non-trivial multiplications. This spatial regularity is a great structural advantage over other algorithms when pipeline/cascading architecture is under consideration for VLSI implementation.

B. Radix-2^3 DIF FFT

The approach with cascade decomposition can be further exploited. Let's consider the first 3 steps in a classical DIF decomposition. The linear index mapping becomes 4-dimensional:

$$\begin{aligned} n &= < \tfrac{N}{2}n_1 + \tfrac{N}{4}n_2 + \tfrac{N}{8}n_3 + n_4 >_N \\ k &= < k_1 + 2k_2 + 4k_3 + 8k_4 >_N \end{aligned} \tag{8}$$

the CFA takes the form of

$$\begin{aligned} &X(k_1 + 2k_2 + 4k_3 + 8k_4) \\ &= \sum_{n_4=0}^{\frac{N}{8}-1} \sum_{n_3=0}^{1} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x(\tfrac{N}{2}n_1 + \tfrac{N}{4}n_2 + \tfrac{N}{8}n_3 + n_4) W_N^{nk} \end{aligned} \tag{9}$$

With cascade decomposition, the twiddle factor can be expressed in the form of

$$\begin{aligned} W_N^{nk} &= W_N^{(\frac{N}{2}n_1 + \frac{N}{4}n_2 + \frac{N}{8}n_3 + n_4)(k_1 + 2k_2 + 4k_3 + 8k_4)} \\ &= W_N^{\frac{N}{2}n_1 k_1} W_N^{\frac{N}{4}(k_1 + 2k_2)} \\ &\quad \cdot W_N^{\frac{N}{8}n_3(k_1 + 2k_2 + 4k_3)} W_N^{n_4(k_1 + 2k_2 + 4k_3 + 8k_4)} \\ &= (-1)^{n_1 k_1} (-j)^{n_2(k_1 + 2k_2)} W_N^{\frac{N}{8}n_3(k_1 + 2k_2 + 4k_3)} \\ &\quad \cdot W_N^{n_4(k_1 + 2k_2 + 4k_3)} W_N^{8n_4 k_4} \end{aligned} \tag{10}$$

Substitute eqn. (10) into eqn. (9) and expand the summation with regard to index n_1, n_2 and n_3. After simplification we have a set of 8 DFTs of length $N/8$,

$$\begin{aligned} &X(k_1 + 2k_2 + 4k_3 + 8k_4) \\ &= \sum_{n_4=0}^{\frac{N}{8}-1} \left[T_{\frac{N}{8}}(n_4, k_1, k_2, k_3) W_N^{n_4(k_1 + 2k_2 + 4k_3)} \right] W_{\frac{N}{8}}^{n_4 k_4} \end{aligned} \tag{11}$$

where a third butterfly structure has the expression of

$$\begin{aligned} &T_{\frac{N}{8}}(n_4, k_1, k_2, k_3) \\ &= \underbrace{\overbrace{H_{\frac{N}{4}}(n_4, k_1, k_2)}^{\text{BF II}} + W_N^{\frac{N}{8}(k_1 + 2k_2 + 4k_3)} \overbrace{H_{\frac{N}{4}}(n_4 + \frac{N}{8}, k_1, k_2)}^{\text{BF II}}}_{\text{BF III}} \end{aligned} \tag{12}$$

As in the Radix-2^2 algorithm, equations eqn. (4) and eqn. (7) represent the first two columns of butterflies with only trivial multiplications in the Radix-2^3 algorithm. The third butterfly contains a special twiddle factor

$$W_N^{\frac{N}{8}(k_1 + 2k_2 + 4k_3)} = \left(\frac{\sqrt{2}}{2}(1 - j) \right)^{k_1} (-j)^{(k_2 + 2k_3)} \tag{13}$$

It can be easily seen that applying this twiddle factor requires only two real multiplications. Since $\frac{\sqrt{2}}{2}$ is the only non-trivial operand involved in the third butterfly of eqn. (12), an constant scaler can be used instead, which differs from a multiplier in that (Booth) encoding/decoding circuit is not necessary, which saves further 40% of the logic. Full complex multiplications are used to apply the decomposed twiddle factor $W_N^{n_4(k_1 + 2k_2 + 4k_3)}$ after the third column. Complete algorithm can be obtained by repeating the procedure. An $N = 64$ example is shown in Fig. 6 where a small square is used to represent the special twiddle factor $W^{\frac{1}{8}}$, for which only 2 real scalers are non-trivial.

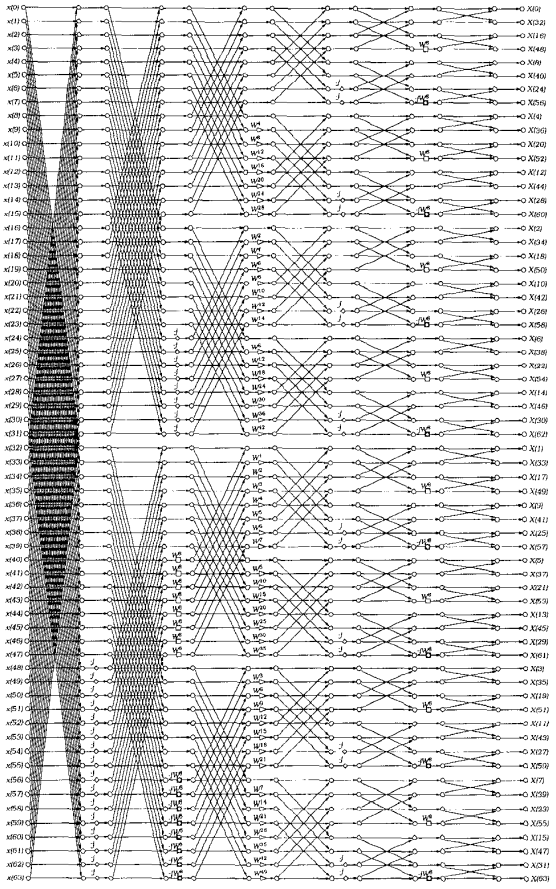

Figure 6: Radix-2^3 DIF FFT signal flow graph for $N = 64$

Radix-2^3 algorithm has the same computational complexity as the split-radix algorithm [20], yet with a much spatially regular SFG. The multiplicative operations are in such an arrangement that for every three columns, only one has non trivial full complex multiplication. The other two columns contains either pure trivial factor $-j$, or a combination of $-j$ and the special twiddle factor $W^{\frac{1}{8}}$, which can be implemented by 2 additive operation and 2 constant scaling. To utilize Radix-2^3 algorithm, carefully crafted scaler can be integrated into the third type butterfly. Due to lack of space, only architecture based on the Radix-2^2 algorithm is discussed in the remaining text.

IV. Radix-2^2 FFT Algorithm Based Architecture

Mapping radix-2^2 DIF FFT algorithm to R2SDF architecture discussed in section II, a new architecture of Radix-2^2

Single-path Delay Feedback ($R2^2SDF$) approach is obtained [21]. Fig. 7 outlines an implementation of the $R2^2SDF$ 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.

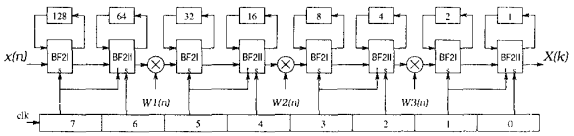

Figure 7: $R2^2SDF$ pipeline FFT architecture for $N = 256$

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 stage. The hardware requirement of proposed architecture as compared with various approaches is shown in Table 1, where not only the number of complex 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 $R2^2SDF$ 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
$R2^2SDF$	$\log_4 N - 1$	$4\log_4 N$	$N - 1$	simple

V. An Implementation Example

The architecture proposed in the above section has been modeled in hardware description language VHDL with generic parameters for transform length and word-length, using fixed point arithmetic. Implementation details, such as the control logic for forward/inverse transform, wordlength variation and scaling are taken into consideration. Data bypassing has been provided so that shorter transforms can be also carried out on a design for longer transform, as long as the transform lengths are of integer power of 2. Special scheduling has been also arranged for "loose" pipeline to allow an arbitrary gap between 2 successive data frames. This happens, for example, when a prefix insertion is required for OFDM system.

The area/power consumption in the pipeline architectures is dominated by the First-In First-Out (FIFO) register files at each stage and the complex multipliers at every other stage. To diminish the unnecessary data moving in the FIFO, reconstruct the storage is mandatory. A known approach is to construct FIFO with 2-port RAM with read a nd write addresses displaced by a constant [22], as shown in Fig. 8(i). The main drawback of using 2-port RAM is that each RAM cell takes an area 33% larger than the corresponding single port RAM cell [23], and it consumes more power due to constant read/write operation. In our implementation, single

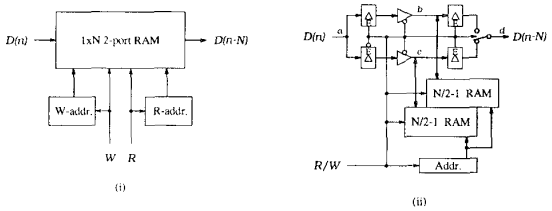

Figure 8: Implementing shift register with (i). 2-port RAM. (ii). single port RAM

port RAM modules are used to construct FIFOs. To avoid read/write conflict, a 2-words at a time scheme is delineated. For length-N FIFO an $(N/2-1)$ double-word RAM is used, as shown in Fig. 8(ii). The read and write operations are interleaved and each of them is active every other clock cycle.

For complex multiplier, the other dominant component in the architecture, an efficient structure based on distributed arithmetic has been used to reduce the area/power consumption for the operation [24].

For fixed point operation, wordlength in the processor must be sufficient to satisfy the Signal to Quantization Noise Ratio (SQNR) requirement. To reduce the power/area consumption, wordlength in the pipeline has been reconstructed to minimize the size of the memory and the wordlength of individual processor stages in the pipeline. Analysis of quantization effects in FFT algorithm has shown that quantization noise introduced by preceding stages in the SFG is reduced by scaling of succeeding stages [18]. Given the prespecified input wordlength of the processor, a progressive adjustment of the wordlength would gives satisfactory performance in terms of SQNR. This progressive adjustment is specially suitable for DIF type decomposition, which allocate the memory block in a exponentially decreasing size, as in Fig. 7. Fig. 9 shows memory requirement optimized by

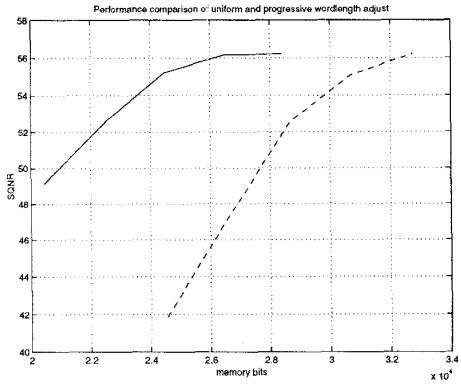

Figure 9: Performance optimization by progressive wordlength adjustment

progressive wordlength adjustment (solid line), as compared

with constant wordlength configuration (dashed line).

A design for 1024 complex point FFT has been synthesized using the VHDL model described above with a standard cell library and a RAM/ROM generator in 0.5μ CMOS technology and sent for fabrication. The synthesized chip consists 6 RAM modules of different sizes, 2 ROM modules and over 21K standard cells, in which 2K are flip-flops. A clock trunk driven by parallel drivers in the library is employed to reduce the clock skew. Placed and routed with a channelless router, the design takes an area of $5 \times 8\text{mm}^2$. Fig. 10 shows the final layout of the chip. Back annoted timing analysis and simulation shows that under worst-case condition, clock rate above 20MHz have been achieved.

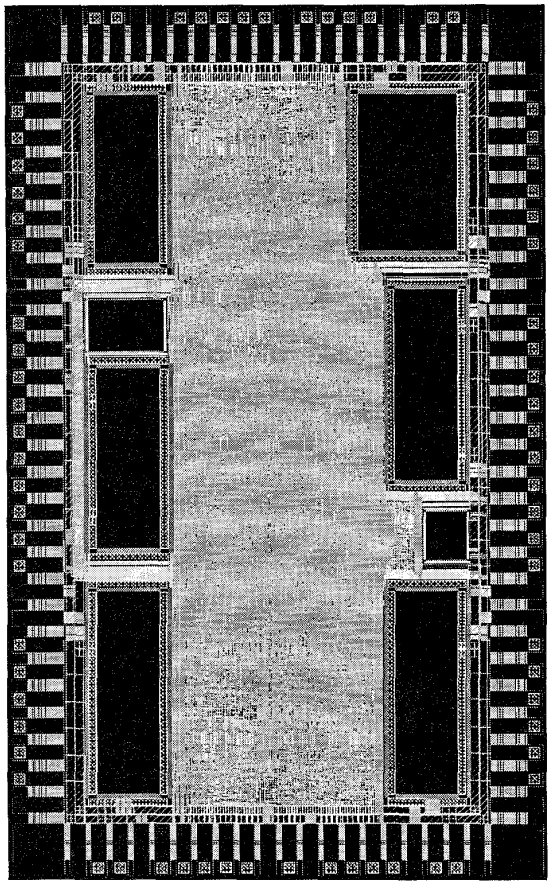

Figure 10: Layout of the 1K FFT processor

The chip can be used to compute forward and inverse FFT of $2^n, n = 0, 1, \ldots, 10$ complex points, with real time sampling frequency well above 20MHz under worst condition. This translates to above 30MHz sampling frequency under normal condition. The power consumption is expected to be low when voltage scaling is applied, owing to the low clock frequency. The chip will be available for testing by the time of the publication this paper.

REFERENCES

[1] S. B. Weinstein and P. M. Ebert. Data transmission by frequency-division multiplexing using the discrete fourier transform. *IEEE Trans. Commun.*, COM-19(5):628–634, Oct. 1971.

[2] B. Hirosaki. An orthogonally multiplexed QAM system using the discrete Fourier transform. *IEEE Trans. Commun.*, COM-29(7):982–989, July 1981.

[3] L. J. Cimini. Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing. *IEEE Trans. Commun.*, COM-33(7):665–675, Jul. 1985.

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

[5] B. Le Floch, R. Halbert-Lassalle, and D. Castelain. Digital sound broadcasting to mobile receivers. *IEEE Trans. Consum. Electro.*, 35(3):493–503, Aug. 1989.

[6] T. de Couasnon, R. Monnier, and J. B. Rault. OFDM for digital TV broadcasting. *Signal Processing*, 39:1–32, 1994.

[7] Y. Omori and H. Sasaoka. Multicarrier 16QAM system in land mobile communications. *IEICE Trans. Commun.*, E77-B(5):634–640, May 1994.

[8] E.A. Lee and D.G. Messerschmitt. *Digital Communication*. Kluwer Academic Publishers, 2nd edition, 1994.

[9] A.P. Chandrakasan and R.W. Brodersen. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.

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

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] R.E. Blahut. *Fast algorithms for digital signal processing*. Addison-Wesley, 1985.

[18] A.V. Oppenheim and R.W. Schafer. *Digital Signal Processing*. Prentice-Hall, Inc., 1975.

[19] C. S. Burrus. Index mapping for multidimensional formulation of the DFT and convolution. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-25(3):239–242, June 1977.

[20] P. Duhamel. Implementation of "split-radix" FFT algorithms for complex, real, and real-symmetric data. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-34(2):285–295, Apr. 1986.

[21] Swedish patent application No. 95/01371, Nov. 1995.

[22] E.E. Swartzlander Jr. *VLSI Signal Processing Systems*. Kluwer Academic Publishers, 1986.

[23] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design*. Addison-Wesley, 1985.

[24] S. He and M. Torkelson. A complex array multiplier using distributed arithmetic. In *Proc. IEEE CICC'96*, pages 71–74, San Diego, CA, May 1996.