A Fast Single-Chip Implementation of 8192 Complex Point FFT

Article *in* IEEE Journal of Solid-State Circuits · April 1995

DOI: 10.1109/4.364445 · Source: IEEE Xplore

4 authors, including:

Damien Castelain
Mitsubishi Electric France MERCE
53 PUBLICATIONS **798** CITATIONS

SEE PROFILE

Brief Papers

A Fast Single-Chip Implementation of 8192 Complex Point FFT

E. Bidet, D. Castelain, C. Joanblanq, and P. Senn

Abstract—**Large-scale single-frequency networks are now being considered in Europe as very promising network topologies to achieve drastic savings in spectrum usage for digital terrestrial television transmission. Such networks are possible using the COFDM system, with large guard intervals (more than 200 μs) to absorb long echoes. In order to limit the spectral efficiency loss to about 20%, very long size fast Fourier transforms (up to 8K complex points) have to be performed in real time for the demodulation of every COFDM symbol (every 1 ms). This paper presents the first VLSI single chip dedicated to the computation of direct or inverse fast Fourier transforms of up to 8192 complex points. Due to its pipelined architecture, it can perform an 8K FFT every 400 μs and a 1K FFT every 50 μs. All the storage is on-chip, so that no external memories are required. A new internal result scaling technique, called convergent block floating point, has been introduced in order to minimize the required storage for a given quantization noise. The chip, 1 cm^2 large for a 1.5 million transistor, has been designed in 3.3 V–0.5 μm triple-level metal and is fully functional. The 8K complex FFT function could therefore be introduced in the next years in digital terrestrial TV receivers at low cost.**

I. Digital TV and COFDM

MANY countries in the world have shown increasing interest in the development of a new digital television terrestrial broadcasting service that could be complementary to the equivalent satellite and cable services on the point to be introduced. In Europe, this led to the setup of several projects such as ^{H}DTV-T, HD-Divine and dTTb, with the common goal under the DVB project umbrella to provide a preliminary specification before the end of 1995. However, the introduction of a digital system is subject to the requirement to provide a better service than the existing analog system! In particular, and besides the quality aspect, this means the possibility to offer a much larger number of programs, taking into account the constraints inherent to network frequency planning. This feature is obtained by the association of two processes. The first one consists of associating sophisticated video compression techniques that allow one to code a conventional quality TV program within 5–6 Mb/s, and powerful coded modulation schemes (e.g., 64QAM and a rate-2/3 channel code) allowing one to transmit about 20–24 Mb/s in a 7.5 MHz bandwidth, i.e., four TV programs in one TV band: the frequency efficiency is therefore multiplied by four compared to analog TV broadcasting. The second process consists of introducing the single-frequency network (SFN) concept. In a classical network, each transmitter must use a different frequency band from those used by adjacent area transmitters: this leads to a huge waste of frequency. In an SFN, each transmitter emits the same signal at the same frequency and at the same time; the drawback is that each TV receiver will receive strong echoes, coming from different transmitters, with relative delay up to 250 μs if the distance between main transmitters is around 80 km. The difficulty is therefore to design a transmission scheme able to deal with long (up to 250 μs) and/or strong (up to 0 dB relative to the strongest received path) echoes. One answer to this problem is the COFDM (coded orthogonal frequency-division multiplex) system.

The principles of COFDM have already been extensively treated in several publications [1]. The first basic idea is to share the data among many orthogonal subcarriers. Each of them is therefore modulated at a low bit rate, i.e., the corresponding symbol length can be made much larger (e.g., 1 ms) than the maximum echo delay. This allows one to insert between each symbol a temporal guard interval, the duration of which approximately equals the maximum echo delay, without decreasing the spectral efficiency too much. With the above parameters, 80% of the total data length is used to transmit data. All the echoes, falling then within the guard interval, will help retrieve the signal instead of interfering. The second basic idea of COFDM, which will not be detailed in this paper, is to combat the frequency selectivity (i.e., the fact that a part of the signal spectrum is attenuated by the channel) by introducing an efficient channel code that will be able to take benefit from the knowledge of the channel frequency response (maximum likelihood decoder). In addition, an interleaving is added to ensure independence of the input values of the decoder, and therefore optimum use of the so-called "frequency diversity." The number of parallel subcarriers can be derived easily from the signal bandwidth ($W = 7.5$ MHz) and the useful symbol duration ($ts = 1$ ms) by the formula $Ns = W \times ts = 7500$.

The last point to consider is the implementation of a COFDM receiver (see Fig. 1). It would be rather difficult to implement in parallel 7500 demodulators. In fact, it is easy to show that modulation and demodulation can be implemented with one analog demodulator associated with a FFT of size N, where N is chosen slightly greater than the number Ns of subcarriers, and for practical reasons, equal to a power of 2.

Manuscript received August 1, 1994; revised November 1, 1994.

E. Bidet, C. Joanblanq, and P. Senn are with France Télécom CNET Grenoble, Meylan Cedex, France.

D. Castelain is with the Centre Commun d'Etudes de Télécommunication et Télédiffusion, Cesson-Sévigné, France.

IEEE Log Number 9408731.

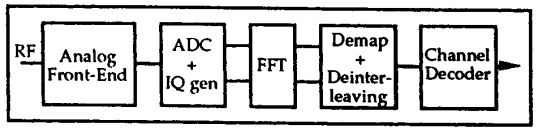

Fig. 1. A simplified COFDM receiver.

The cheap availability of FFT VLSI chips able to process up to 8K complex points every 1 ms then becomes a prerequisite for the cost-efficient implementation of these techniques. The purpose of the next sections is to detail the architecture and the design of such a chip, which has been developed by CNET in order to validate the large SFN concept.

II. FFT Architectural Considerations

A. FFT Algorithms

The $N-$Point discrete Fourier transform $[X(k)$ of an $N-$point sequence $x(n)]$ is, by definition,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad \text{for } k = 0, 1, \cdots, N-1$$

$$\text{with } W_N = e^{-j(2\pi/N)}.$$

Many computationally efficient FFT algorithms [2] have been published since Cooley and Tukey in 1965 [3]; they all use divide-and-conquer approaches to reduce the number of operations from N^2 (direct implementation of the DFT formula) down to a number in $O(N \log N)$. Few of those, however, have the regularity, flexibility, and in-place characteristic of the original algorithm, which makes it the best candidate for large FFT VLSI realizations.

B. Conventional Architectures

In classical implementations of the Cooley and Tukey algorithm, we find an optimized arithematic unit (containing several complex multipliers and adders) to compute the FFT butterfly and the multiplications by the twiddle factors W_N^{nk}, a ROM to store the twiddle factors, an $N-$word RAM to store intermediate results, and an $N-$word input buffer (to store the input symbol while processing the previous one). See Fig. 2.

Strong speed limitations occur, however, with such architectures: for long FFT's, the large intermediate results storage area needed (N complex words) implies the use of an in-place algorithm using only one memory. If this memory is not partitioned, the number of R/W accesses to perform the FFT creates a bottleneck: an $N-$point FFT computation actually requires $N/r \log_r N$ radix-r butterfly computations, which implies $2N \log_r N$ read or write RAM accesses (which would imply an R/W access to the internal RAM every 9 ns to perform an 8K FFT in 1 ms using a radix-4 approach. This clearly exceeds the capacity of current technology).

In order to speed up such an implementation, more advanced solutions have been proposed using either an increasing of the radix [4] to reduce the overall number of accesses, at the price of an expansion of the arithmetic complexity, or a partitioning of the memory [5] in r banks accessed simultaneously, at the price of a complex addressing and higher area. In either case, such single butterfly architectures need a fast clock which may be running at a considerably higher rate than the signal sampling rate in a COFDM system: the provision of multiple clocks would surely create difficulties in the design of a receiver.

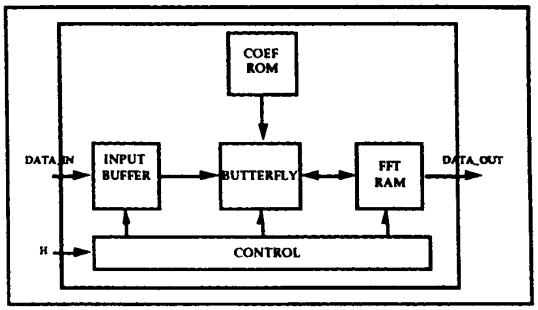

Fig. 2. Conventional FFT implementation (single butterfly).

C. Pipelined Architectures

We preferred to choose an alternative solution by adopting a pipelined architecture (Fig. 3), such as the ones already proposed 20 years ago [6] to reach high-speed operation. To derive the required hardware in such pipelined architectures, the FFT signal flow graph is projected perpendicularly to the data flow and the necessary hardware is allocated (Fig. 3 gives an example with a radix-4 16 point FFT). The hardware consists of several butterfly units (one per stage of the FFT) with associated complex multipliers, separated by delay commutator units. The goal of the delay commutator is to reorder the data coming from one stage butterfly so as to present them in the right order in the next stage butterfly. This architecture allows one to break the recursivity inherent to the FFT algorithm by having a unidirectional processing of the data. The speed can then be enhanced almost at will by some pipelining of the arithmetic units, without deeply modifying the sequencing. Another advantage of the pipelined architectures is that the power consumption spikes can be limited by adopting opposite directions for clock and data propagation. One obvious disadvantage is, however, that the arithmetic parts are replicated $\log_r N$ times compared to a single butterfly approach.

The major reported realizations of such a pipelined FFT architecture generally use different chips [7] or even different wafers [8] for the commutator element and the butterfly element. Such an approach, although fast enough, is clearly not adapted for consumer products, and some improvements have to be found.

If we split radix-r butterflies into r simplified radix-r butterflies (each one computing only one of the r original outputs of the radix-r butterfly), we obtain a much more complicated flow graph (Fig. 4). After projection onto the same axis as before,

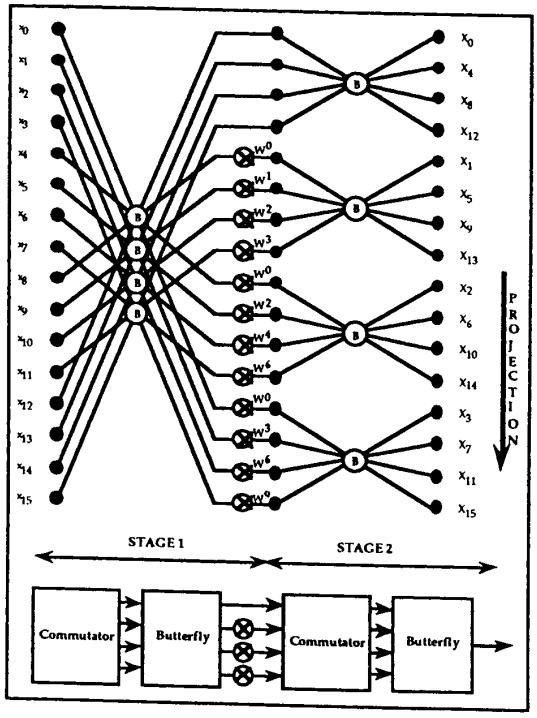

Fig. 3. 16 point FFT flow graph and pipelined architecture.

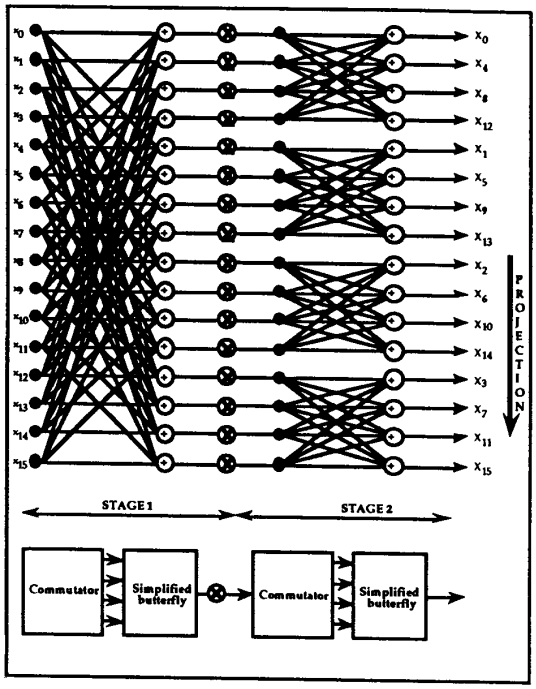

Fig. 4. 16 point FFT modified flow graph and architecture.

we save, however, a great amount of arithematic hardware and reach a much better computational efficiency. We actually replaced the original butterfly computing r intermediate results at a time (but being active one rth of the time because of the need to rearrange the data in the commutators) by one simplified butterfly working full time and sequentially computing the r previous results. This is the architecture proposed by Bi and Jones [9] in 1989, which we took as a basis for our implementation.

Each radix -r stage now requires one complex multiplier only (instead of $r-1$ for conventional radix-r stage), a simplified butterfly, a ROM to store twiddle factors, and a delay commutator. For $r=2$ and $r=4$, butterflies are formed by complex adders/subtractors (one for $r=2$ and three for $r=4$) and multiplexers. Commutators have to present data in the correct order to the butterfly: they are made of $2^*(r-1)$ delay lines of length Q/r (if Q is the input length for the current stage). Thus, if $N=\prod_{i=1}^{q} r_i$, for an $N-$length FFT, we need $q-1$ multipliers, q simplified butterflies (whose complexity depends on the radix r_i), and $D=\sum_{i=0}^{q} 2*(r_i-1)*(N/\prod_{j=0}^{i} r_j)=2*(N-1)$ complex data stores (as for the conventional approach).

This modified architecture achieves a quasi-optimal hardware utilization (75% for the multiplier and 100% for the adders) and permits a serial processing of data, which is well adapted to COFDM symbol demodulation in a receiver. No extra buffer memory is required to store one symbol while processing the previous one. In this architecture, no fast clock is needed, and all the computations are performed at the input data sampling frequency. Furthermore, the latency (around $1.2N$ clock cycles for an $N-$point FFT) is almost divided by 2 compared to conventional structures because the computation can begin without waiting for the whole symbol to be stored in the input buffer.

D. Area Tradeoffs

It is still clear that, compared to a conventional approach, arithmetic units are more important: $q-1$ multipliers and q simplified butterflies. For example, an 8K-point radix-4 FFT requires 6 complex multipliers instead of 3 and 19 complex adders/subtractors instead of 8 (with reference to a single radix-4 ALU architecture). The operators have, however, many fewer constraints than in the conventional case where pipelining and sequencing can be difficult for reaching high sampling rates. Furthermore, for long FFT's as in our case, the arithmatic units are much smaller than storage units, so that this extra hardware is not very significant.

For the storage units, at a first glance, the conventional approach and the pipelined approach have the same requirements in terms of capacity ($2N$ points). But we must notice that, for the pipelined approach, the first stage contains $(r_0 - 1/r_0)(N/N-1)$ of the total storage and that, for this stage, the word length is minimal and equal to the input word length. In our case, $r_0=4$ and three-quarters of the total storage has a minimum word length (to be compared to exactly one-half for a conventional approach).

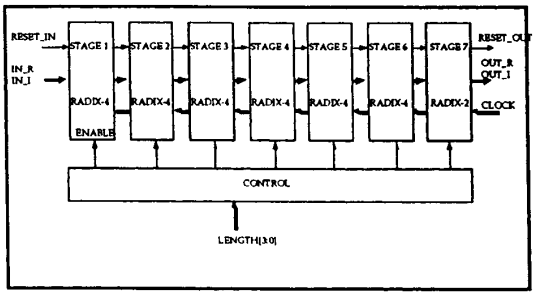

Fig. 5. Chip Architecture.

Moreover, and that is the main point, data inside the delay commutator are stored in delay lines which can be realized very efficiently using dedicated three-transistor per point delay line compilers such as the one presented in [10]. In our design, the hardware savings on the memory, thanks to delay lines (compared to dual-port RAM's needed otherwise), represents 30% of the total hardware. This consideration was essential for the design of an 8K chip were 16K words of memory are required.

E. Chip Architecture

To be able to perform an 8K FFT, the chip is made of six successive radix-4 stages and a last radix-2 stage (see Fig. 5). By bypassing several stages, a reduced size FFT can be performed (2K by skipping the first stage, 512 point by skipping the two first stages, etc.). This allows one to sort by the testing program chips working up to 8K, plus others working up to 2K, etc., in order to use silicon in the most efficient way as possible. Some changes have been introduced additionally in the radix-4 commutator and butterfly design in order to enable any radix-4 stage to work as a radix-2 stage, and therefore to obtain all the lengths between 2 points and 8K points.

III. Signal-to-Noise Ratio Optimization

Good data scaling techniques are of primary importance if one wishes to limit the word size of the internal memories of an FFT VLSI implementation. It is obvious that keeping the whole accuracy after the multiplications or working with a floating-point representation implies a too large internal word length. This is also true, but to a minor extent, with the semi-floating point solutions [11]. In scaled fixed-point representations, data have to be saturated and rounded after each stage. There is always a tradeoff between saturation and rounding to find, which can be different from stage to stage and is very dependent on the data.

The most efficient scheme proposed up to now is the block floating point (BFP), where each data block at the output of an FFT stage is stored into memory, then adaptively scaled depending on the maximum amplitude of the data inside the block. For each stage, there is a single exponent shared by all output data of this stage (see Fig. 6). This technique, unfortunately, cannot be used in the pipelined implementation,

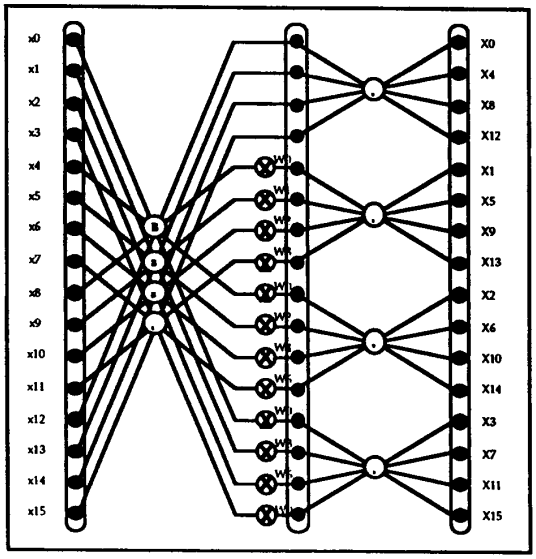

Fig. 6. BFP rescaling (one exponent for all data in each stage).

for the first internal result of one stage is computed before all the previous stage computations are achieved, and therefore before any knowledge of the common exponent of the previous stage is possible. All existing silicon implementations using a pipelined approach, therefore, have to use an oversized internal word length [7], [8], [12] or floating or semi-floating representations [11].

To solve this problem in the best possible way, we have implemented a new rescaling method attributing different rescaling factors to blocks of data during their transfer in the delay commutator between two different stages. The basic idea was found by trying to apply the BFP technique in a pipelined architecture. We can observe that the computation of the first $N/4$ outputs of the second stage (see Fig. 7) depends only on the first $N/4$ outputs of the first stage, and so on. Consequently, as in the classical BFP solution, we start from a fixed-point implementation with one global exponent per block of N data; however, as soon as the first $N/4$ results of the first stage are computed, a search for maximum amplitude is performed and an exponent e_1^1 is associated with this block of $N/4$ data. The computation of the second stage $N/4$ first results can then begin right away, without the need to wait for all N results of the first stage as in the classical BFP solution. This technique is therefore compatible with the selected pipelined approach (at the price of an extra complexity of 4% in the delay line part). A similar process is performed for all four blocks of $N/4$ results exiting the first stage. Four exponents e_1^i with i in $0 \cdots 3$ are then associated with blocks of $N/4$ data in the first stage.

The process is iterated from stage to stage, with magnitude detection and exponents associated with blocks of data of smaller lengths ($N/4$, $N/16, \cdots$ in the second, third stage, etc.). The whole scheme converges towards a representation where each one of the output data has its own exponent.

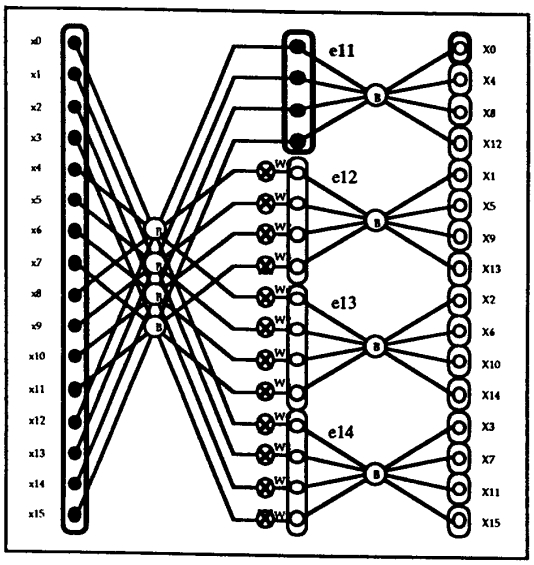

Fig. 7. CBFP technique (one exponent for each block of data).

The method is therefore called "convergent block floating point." As explained above, it is compatible with pipelined architectures, but also with conventional structures (with a change in the sequencing). Additionally, it understandably provides better results than fixed scaling or classical BFP techniques (its accuracy is, in fact, between the one of BFP and of floating point). The performance has been verified for many kinds of inputs (sine, diracs, wobulation, OFDM, @r@isine@i+@idiracs@i@r), and is always better by more than 0.5 dB than the BFP technique. The difference can reach several decibels or much more in cases where BFP fails (which happens, for example, when a sine and a dirac pulse are combined).

IV. Conclusion

A. Chip Features

The chip presented here (see Fig. 8) has been designed first in a 0.7 μm–5 V double-metal CMOS process than in a CMOS 0.5 μm–3.3 V triple-metal process. Its characteristics in terms of number of bits for input, output, and internal accuracy have been found after careful simulations of a COFDM transmission chain where several channel conditions have been applied (added Gaussian noise, added interferer, multipath propagation). They are the following: 2 × 10 b for input data and internal sine and cosine values, 2 × 12 b for internal results and output (provided in bit-reverse order). The performance achieved by the combination of the ADC and the FFT always exceeds an SNR of 40 dB compared to a floating-point FFT, which is enough for digital terrestrial TV broadcasting. Both direct and inverse FFT are possible, for any size of FFT up to 8K points and for any complex data sampling frequency up to 20 MHz. The 8K FFT chip can therefore compute an 8K FFT every 400 μs and a 1K FFT every 50 μs, which exceeds the performance and capacity of all current solutions available. Finally, it is packaged in a 100-pin PGA.

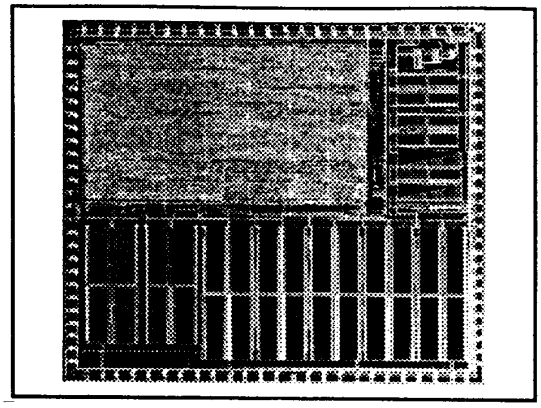

Fig. 8. FFT8K chip layout (0.5 μm).

B. Design

The 8K FFT chip has been designed using a video delay line generator for the delay commutators and logic synthesis for all the control and arithmetic parts. Referring to the target application, a sampling frequency of 20 MHz was found to be enough. However, it would be very easy to obtain frequencies of 50 MHz and more by very simple pipelining of the arithmetic parts. The chip includes all necessary data and coefficient storage elements (350 kb of delay lines). It contains 1.5M transistors over an area of 1 cm^2.

C. Results and Conclusion

The 8K FFT was tested in June and October 1994, and was found fully functional in the 8K mode up to 22–23 MHz at 3.3 V (in the 0.5 μm process). The power dissipation equals approximately 600 mW for an 8K FFT at 20 MHz, and 300 mW for a 2K FFT (where the first stage of memory containing half of the total number of transistors is not running).

These results confirm the feasibility of the 8K FFT function in a single chip, and therefore solves one of the key technical issues associated with large single-frequency networks.

References

[1] M. Alard and R. Lasalle, "Principles of modulation and channel coding for digital broadcasting for mobile receivers," presented at the ITU WARC–ORB Conf., Sept. 1988.
[2] P. Duhamel and M. Vetterli, "Fast Fourier transform: A tutorial review and a state of the art," *Signal Process.*, no. 19, 1990.
[3] J. W. Cooley and J. W. Tukey, "An algorithm for machine computation of complex Fourier series," *Math. Comput.*, vol. 19, 1965.
[4] Bhatia, Furuta, and Ponce, "A quasi radix-16 FFT VLSI processor," in *Proc. IEEE ICASSP*, July 1991.
[5] O'Brien, Mather, and Holland, "A 200 MIPS single-chip 1K FFT processor," in *Proc. ISCCC*, Feb. 1989.
[6] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing.* Englewood Cliffs, NJ: Prentice–Hall, 1975, pp. 600–614.
[7] Swartzlander, Young, and Joseph, "A radix-4 delay commutor for fast Fourier transform processor implementation," *IEEE J. Solid-State Circuits*, vol. SC-19, Oct. 1984.

[8] Swartzlander, Jain, and Hikawa, "A radix-8 wafer scale FFT processor," *J. VLSI Signal Process.*, vol. 4 Jan. 1992.
[9] G. Bi and E. V. Jones, "A pipelined FFT processor for word-sequential data," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, Dec. 1989.
[10] C. Joanblanq, F. Rothan, and P. Senn, "A video delay line compiler," in *Proc. ISCAS*, New Orleans, LA, May 1990.
[11] Blankenship and Hofstetter, "Digital pulse compression via fast convolution," *IEEE Trans. Acoust. Speech, Signal Process.*, vol. ASSP-23, pp. 191–199, Apr. 1975.
[12] Schirrmeister, Müller, Reventlow, Reimers, and Siebert, "A single chip solution for a high speed 128-point radix-2 FFT calculation," in *Proc. Int. Workshop HDTV'93*, Ottawa, Canada, Oct. 1993.