Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2007, Article ID 67360, 8 pages
doi:10.1155/2007/67360

Research Article

Fast Discrete Fourier Transform Computations Using the Reduced Adder Graph Technique

Uwe Meyer-Bäse,[1] Hariharan Natarajan,[1] and Andrew G. Dempster[2]

[1] *Department of Electrical and Computer Engineering, Florida State University, 2525 Pottsdamer Street, Tallahassee, FL 32310-6046, USA*
[2] *School of Surveying and Spatial Information Systems, University of New South Wales, Sydney 2052, Australia*

Received 28 February 2006; Revised 23 November 2006; Accepted 17 December 2006

Recommended by Irene Y. H. Gu

It has recently been shown that the n-dimensional reduced adder graph (RAG-n) technique is beneficial for many DSP applications such as for FIR and IIR filters, where multipliers can be grouped in multiplier blocks. This paper highlights the importance of DFT and FFT as DSP objects and also explores how the RAG-n technique can be applied to these algorithms. This RAG-n DFT will be shown to be of low complexity and possess an attractively regular VLSI data flow when implemented with the Rader DFT algorithm or the Bluestein chirp-z algorithm. ASIC synthesis data are provided and demonstrate the low complexity and high speed of the design when compared to other alternatives.

1. INTRODUCTION

The discrete Fourier transform (DFT) and its fast implementation, the fast Fourier transform (FFT), have both played a central role in digital signal processing. DFT and FFT algorithms have been invented (and reinvented) in many variations. As Heideman et al. [1] have pointed out, we know that Gauss used an FFT-type algorithm we now call the Cooley-Tukey FFT.

We will follow the terminology introduced by Burrus [2], who classified FFT algorithms according to the (multidimensional) index maps of their input and output sequences. We will therefore call all algorithms which do *not* use a multidimensional index map DFT algorithms, although some of them, such as the Winograd DFT algorithms, enjoy an essentially reduced computational effort.

In a recent EURASIP paper by Macleod [3], the adder costs were discussed of rotators used to implement the complex multiplier in fully pipelined FFTs for 13 different methods, ranging from the direct method and 3-multiplier methods to the matrix CSE method and CORDIC-based designs. It was determined that not a single structure gave the best results for all twiddle factor values. On average the CORDIC-based method gave the best results for single multiplier costs. In this paper, we restrict our design to the two most popular methods ($4 \times 2+$ and $3 \times 5+$) used in FFT cores [4, 5] by FPGA vendors.

The literature provides many FFT design examples. We found implementations with programmable signal processors and ASICs [6–10]. FFTs have also been developed using FPGAs for 1D [11, 12] and 2D transforms [13, 14].

This paper deals with the implementation of two alternatives of fast DFTs via a transformation into an FIR filter. The methods are called a Rader DFT algorithm and a Bluestein chirp-z transform. We will present latency data (measured in clock cycles) when the FFT-block is used in a microprocessor coprocessor configuration. The design data are compared with direct matrix multiplier DFT methods and radix-2 and radix-4 type Cooley-Tukey based FFTs as used by FPGA vendors [5]. The provided area data are measured in equivalent gates as typical for cell-based ASIC designs.

2. CONSTANT COEFFICIENT MULTIPLICATIONS

DSP algorithms are MAC intensive. Essential savings are possible if the multiplications are constant and not variable. Statistically, half the digits will be zero in the two's complement coding of a number. As a result, if a constant coefficient is realized with an array multiplier,[1] on average 50% of the partial products will also be zero. In the case of a *canonic signed*

[1] An array multiplier is usually synthesized by an ASIC tool in a binary adder tree structure.

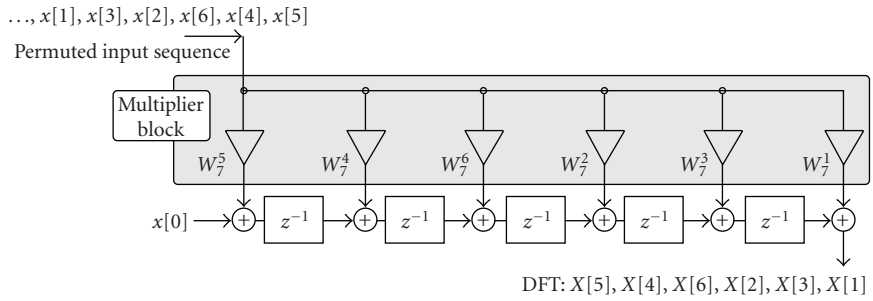


Figure 1: Length $p = 7$ Rader prime factor DFT implementation.

digit (CSD) system, that is, digits with the ternary values $\{0, 1, -1\} = \{0, 1, \bar{1}\}$, and no two adjacent nonzero digits, the density of the nonzero elements becomes 33%. However, sometimes it can be more efficient to first factor the coefficient into several factors, thus realizing the individual factors in an optimal CSD sense [15–18]. This multiplier adder graph (MAG) representation reduces, on average, the implementation effort to 25% when compared to the number of product terms used in an array multiplier [3, 19].

In many DSP algorithms, we can achieve additional cost reduction if we combine several multipliers within a *multiplier block*. The transposed FIR filter shown in Figure 1 is a typical example for a multiplier block. It has been noted by Bull and Horrocks [15, 16] that such a multiplier block can be implemented very efficiently. Later, Dempster and Macleod [20] introduced a systematic algorithm, which produces an *n-dimensional reduced adder graph* (RAG-*n*) of a block multiplier. In general, however, finding the *optimal* RAG-*n* is an NP-hard problem. RAG-*n* determines when the design is optimal; for the suboptimal case, heuristics are used. The full 10-step RAG-*n* algorithms can be found in [20].

Another alternative to implementing multiple constant multiplication is to use the subexpression technique first introduced by Hartley [21]. Here, common patterns in the CSD coding are identified and successively combined. For random coefficients, minor improvements were observed compared with RAG-*n*. For multiplier blocks with redundancy, RAG-*n* generally offered the best performance [23].

3. FIR FILTER STRUCTURES USED TO COMPUTE THE DFT

FIR filters are widely studied DSP structures. Their behavior in terms of quantization error, BIBO stability, and the ability to build fast-pipelined structures make FIR filters very attractive. Two algorithms have been used to compute the DFT via the FIR structure. These two are the Rader algorithm, which requires an I/O data permutation and a cyclic convolution, and the Bluestein chirp-*z* algorithm, which uses a complex I/O multiplication and a linear FIR filter. These two algorithms are briefly reviewed below. Details can be found in the DSP textbooks [24, 25], as well as in a wide variety of FFT books [26–30].

The DFT is defined as follows:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad k, n \in \mathbb{Z}_N,\ W_N = e^{j2\pi/N}. \tag{1}$$

The Rader algorithm [31, 32] used to compute the DFT is defined only for prime length N. Because $N = p$ is a prime, we know that there is a primitive element, a *generator* g, that generates all elements of n and k in the field $\mathbb{Z}_p$, excluding zero. We substitute n with $g^n \bmod N$ and k through $g^k \bmod N$ and get the following index transform:

$$X[g^k \bmod N] - x[0] = \sum_{n=0}^{N-2} x[g^n \bmod N] W_N^{g^{n+k \bmod (N-1)}} \tag{2}$$

for $k \in \{1, 2, 3, \ldots, N-1\}$. We notice that the right-hand side of (2) is a cyclic convolution, that is,

$$[x[g^0 \bmod N], x[g^1 \bmod N], \ldots, x[g^{N-2} \bmod N]] \circledast \left[W_N, W_N^g, \ldots, W_N^{g^{N-2 \bmod (N-1)}}\right]. \tag{3}$$

The DC component must be computed separately as

$$X[0] = \sum_{n=0}^{N-1} x[n]. \tag{4}$$

Figure 1 shows the Rader algorithm for $N = 7$ using the multiplier block technique.

The second algorithm that transforms a DFT into an FIR filter is the Bluestein chirp-z transform (CZT) algorithm. Here the DFT exponent nk is a quadratic expanded to

$$nk = -\frac{(k-n)^2}{2} + \frac{n^2}{2} + \frac{k^2}{2}. \tag{5}$$

The DFT therefore becomes

$$X[k] = W_N^{k^2/2} \sum_{n=0}^{N-1} \left(x[n] W_N^{n^2/2}\right) W_N^{-(k-n)^2/2}. \tag{6}$$

The computation of the DFT is therefore done in three steps:

(1) N multiplications of $x[n]$ with $W_N^{n^2/2}$;
(2) linear convolution of $x[n] W_N^{n^2/2} * W_N^{-n^2/2}$;

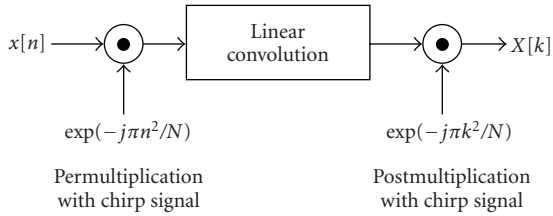


Figure 2: The Bluestein chirp-z algorithm.

Table 1: Number of coefficients and costs of Rader multiplier block implementation for 12-bit plus sign coefficients.

DFT length	7	17	31	61	127	257
C_N	6	16	30	60	126	256
R_N	6	16	30	60	124	253
CSD	21	59	100	201	428	810
MAG	18	51	85	175	360	688
RAG-n	11	23	35	61	124	237

(3) N multiplications with $W_N^{k^2/2}$.

This algorithm is graphically interpreted in Figure 2.

For a complete transform, we need a length N linear convolution and $2N$ complex multiplications. The advantage, compared with the Rader algorithms, is that there is no restriction to primes in the transform length N. CZT can be defined for every length.

3.1. RAG-n implementation of DFTs

Because the Rader algorithm is restricted to prime lengths, there is less redundancy in the coefficients compared with the Bluestein chirp-z DFT algorithms, which can be defined for any length. Table 1 shows, for the primes next to length 2^n, the implementation effort of the circular filter in transposed form. The numbers of adders required to implement the 12-bit filter coefficients are shown for CSD, MAG [17], and RAG-n [20].

The first row in Table 1 shows the cyclic convolution length N, which is also next to the number of complex coefficients $C_N = N - 1$, shown in row 2. Row 3 shows the number R_N of different real sin/cos coefficient multiplier that must be implemented. Comparing row 3 and the worst case with $2(N-1)$ real sin/cos coefficients, we see that redundancy and trivial coefficients reduce the number of nontrivial coefficients by a factor of 2. The last three rows show the costs (i.e., the number of adders) for a 12-bit multiplier precision implementation using CSD, MAG, or RAG-n algorithms, respectively. Note the advantage of RAG-n, especially for longer filters. RAG-n only requires about 1/3 the adder of CSD-type filters.

The effort for the CSD, MAG, and RAG-n methods for all the Rader DFTs up to a length of 257 is graphically interpreted in Figure 3.

Narasimha et al. [33] have noticed that in the CZT algorithm many coefficients of the FIR filter part are trivial or identical. For instance, the length-8 CZT has an FIR filter of length 15, $C(n) = e^{j2\pi((n^2/2 \bmod 8)/8)}$, $n = 1, 2, \ldots, 15$, but there are only four different complex coefficients. These four coefficients are 1, j, and $\pm e^{j\pi/8}$, that is, we have only two nontrivial real coefficients to implement in the length-8 CZT.

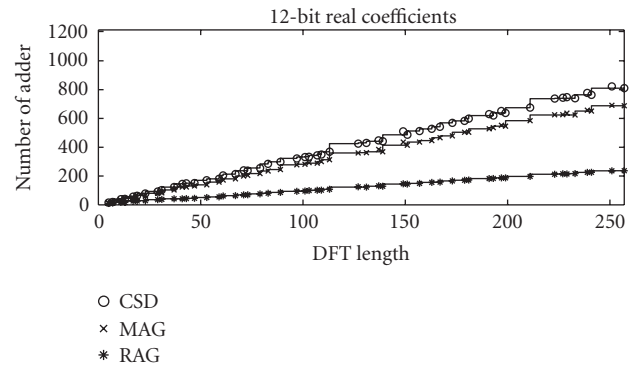


Figure 3: Effort for a complex multiplier block design in the Rader algorithm.

Table 2: Number of coefficients and costs of a CZT multiplier block implemented with 12-bit plus sign coefficients.

N	8	16	32	64	128	256
C_N	4	7	12	23	44	87
R_N	2	3	6	11	22	43
CSD	6	10	19	38	70	148
MAG	6	9	17	34	62	129
RAG-n	5	7	11	19	24	44

In general, power-of-two lengths are popular building blocks for Cooley-Tukey FFTs, so we use $N = 2^n$ in Table 2 for a comparison.

The comparison of Table 2 with the Rader data shown in Table 1 shows the advantages of the CZT implementation.

The effort for the CSD, MAG, and RAG-n methods for the CZT DFT up to a length of 256 is graphically interpreted in Figure 4. Note that the DFTs with a maximum transform length are connected through an extra solid line. Due to coefficient redundancy explored in the CZT design, we see that some longer transform lengths may have a lower implementation effort than some shorter transforms. For this reason, we might try to use the longer transform whenever possible.

3.2. Complex RAG-n DFT implementations

Thus far we have implemented a DFT of a real input sequence; the complex twiddle factor multiplication W_n^{nk} is implemented with two real multiplications. For complex input DFTs, we have two choices for how to implement the complex multiplication. We might use a straightforward approach with 4 real multiplications and 2 real additions:

$$(a + jb)(c + js) = a \times c - b \times s + j(a \times s + b \times c). \quad (7)$$

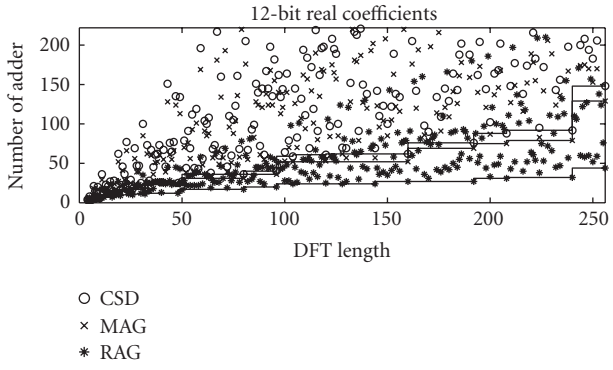


Figure 4: Effort for a real coefficient multiplier block design in the Bluestein chirp-z algorithm. The solid line shows the maximum transform length for a specific cost value.

Or, we might use a different factorization such as

$$\begin{aligned} s[1] &= a - b, \quad s[2] = c - s, \quad s[3] = c + s, \\ m[1] &= s[1]s, \quad m[2] = s[2]a, \quad m[3] = s[3]b, \\ s[4] &= m[1] + m[2], \quad s[5] = m[1] + m[3], \\ (a + jb)(c + js) &= s[4] + js[5], \end{aligned} \tag{8}$$

which uses 3 real multiplications and 5 real additions,[2] as shown in Figure 5.

Figure 7 shows that for a transform length of up to 257, the algorithm with $4 \times 2+$ is superior (for both Rader and CZT) when compared with the $3 \times 5+$ algorithms. This is due to the fact that with the $4 \times 2+$ algorithms for a filter with N complex coefficients, two multiplier blocks with size $2N$ are designed, while for the $3 \times 5+$ algorithms *three* real multiplier block filters with block size N must be used. To have cleaner results, we do not show the implementation effort for all CZT lengths; only the maximum transform lengths for the same implementation effort are shown.

The overall adder budget now consists of three parts: (a) the multiplier-block adders, used for CSD, MAG, or RAG coding; (b) the two output adders required to compute the complex multiplier outputs; and (c) the 2 structural adders used for each tap. Because CZT uses only a few different coefficients, the required number for (b) is much smaller than for the Rader transform. However, the filter structure for the CZT is about twice as long when compared with the Rader transform. Table 3 shows a comparison for the overall adder budget required for a CZT of length 64 and a Rader transform of length 61. Again, the direct comparison of Rader and CZT shows a reduced effort for CZT.

[2] Note that in the $3* \times 5+$ block multiplier architecture, the sum $s[2] = c - s$ and $s[3] = c + s$ is precomputed and is therefore sometimes called a $3 \times 3+$ algorithm.

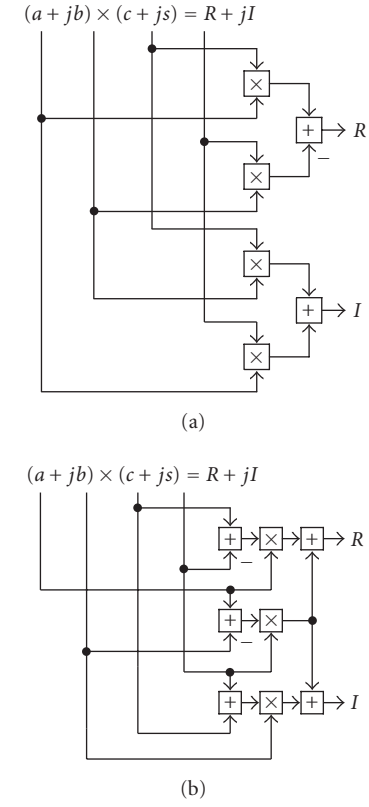


Figure 5: The two complex multiplier versions (a) $4 \times 2+$, (b) $3 \times 5+$.

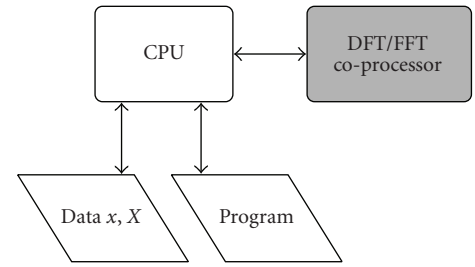


Figure 6: Co-processor configuration of FFT core.

3.3. Alternative DFT implementations and synthesis data

In a typical OFDM or DVB configuration [34], the FFT core is used as a coprocessor to speed up the host processor performance as shown in Figure 6. The computation of the DFT as coprocessor then has three stages.

(a) The serial data transfer to the coprocessor.
(b) The computation of the DFT, until the first output value is available.
(c) The data transfer back to the host processor.

While (a) + (c) are usually constants, the latency of the DFT (b) is a critical design parameter. Table 4 summarizes the equivalent gate count and the latency of different algorithms.

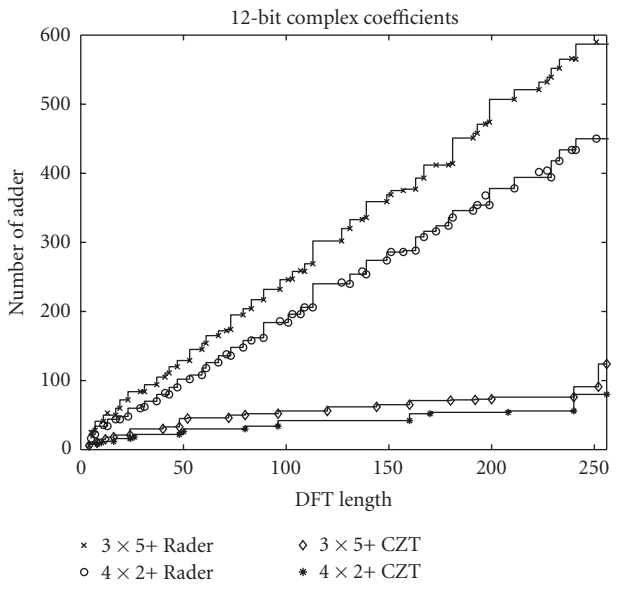


Figure 7: Comparison of complex multiplier block effort for the Rader and CZT algorithm.

Table 3: Total required adders for complex DFTs.

	CZT-64 points			Rader-61 points		
	CSD	MAG	RAG	CSD	MAG	RAG
Mul. block	76	68	38	402	350	120
Cmul	22	22	22	120	120	120
Structural	252	252	252	124	124	124
Total	350	342	312	646	594	366

The gate count is measured as equivalent gates as used in cell-based ASIC design. The latency is the number of clock cycles the FFT core needs until the first output sample is available (see (b) above).

Alternative DFT implementations of the CZT RAG-n design include a direct implementation via DFT matrix multiplication [22] using subexpression sharing. Here a length 8 DFT (8-bit) already requires 74 adders; a 16-point DFT in 16 bits requires 224 adders.

For short length DFTs, the Winograd algorithm seems to be an attractive alternative as well, because it reduces the number of multiplications to a minimum. Unfortunately, the number of structural adders in the Winograd algorithm increases more than is proportional to the length. For instance, a complex length 8 DFT requires 52 structural adders [32].

Another common approach uses radix-2 or 4 FFT processor elements [5, 35]. A fully pipelined Cooley-Tukey FFT (called Stream I/O by Xilinx) can benefit from MAG coefficient coding, but each butterfly in 12-bit precision will require, on average, $12 \times 4 \times 25\% + 2 = 14$ adders. A 64-point FFT therefore requires $32 \times 6 \times 14 = 2688$ adders if MAG coding is used. If we use the optimum rotator from [3], then the required adder can be further reduced to 1684 in a radix-2 scheme. A mixed radix-2/4 algorithm is reported with 1412 adders in [3]. In Table 3, the same transform is listed with 312 adders for the chirp-z algorithm.

Table 4: Size (measured via equivalent number of gates for combinational and noncombinational elements) and speed as latency (measured as clock cycles until first output value are available) for different DFT lengths sorted by latency.

Method		DFT length				
		4	8	16	32	64
Matrix Mult.∗ [22]	Size	—	26 640	80 640	—	—
	Latency	—	2	2	—	—
Winograd	Size	5129	14 137	36 893	—	—
	Latency	2	2	2	—	—
CSD-CZT	Size	10 349	14 192	23 630	41 426	78 061
	Latency	4	4	4	4	4
RAG-CZT	Size	9970	13 728	22 578	39 234	73 171
	Latency	4	4	4	4	4
Xilinx Radix-2 Min. Resource [5]	Size	—	—	29 535	30 455	32 255
	Latency	—	—	45	112	265
Xilinx Radix-4 Stream I/O [5]	Size	—	—	—	—	137 952
	Latency	—	—	—	—	64

∗ Estimated.

Minimum FFT resources are achieved with a single radix-2 Cooley-Tukey butterfly processor (called a minimum resource design by Xilinx) at the cost of high latency, shown as the radix-2 entry in Table 4. Faster but more resource intensive is a column processor that uses a separate butterfly processor in each stage, shown as the radix-4 streaming I/O in Table 4 [5].

Winograd, CSD, and RAG-n CZT circuits have been synthesized from their VHDL description and optimized for speed and size with synthesis tools from Synopsys. The `lsi_10k` standard-cell library under typical WWCOM operating conditions has been used. We used two pipeline stages for the multiplier and two for the RAG in the design.

From the comparison in Table 4, it can be concluded that the RAG-CZT provides better results in size compared to the Winograd DFT or the matrix multiplier for more than 16-point DFTs. Therefore, only CZT implementations were used for longer DFTs. When compared with a 64-point Cooley-Tukey FFT processor, only the single butterfly processor gives a smaller area, while a faster pipelined streaming I/O processor requires a 64 clock cycle latency and is twice the size of the RAG-CZT.

By providing a sufficient amount of extra buffer memory all of the above algorithms can be modified in such a way that the pipelined FFT computation is only limited by the data transfer time from host to FFT core. This is particularly useful in 2D FFT, when a large number of consecutive row/column FFTs need to be computed. However, in 1D DFT the latency, that is, the number of clock cycles will not change by adding buffer memory until a value is available at the core for the (waiting) host processor.

3.4. Alternative MCM arithmetic concepts

Other possible arithmetic modifications that can be used to implement the multiple constant multiplication (MCM) block in fast DFTs are the (exclusive) use of carry-save adders [36], distributed arithmetic [37], common subexpression sharing (CSE) [21], or the residue number system (RNS) [38].

It has also been suggested [3] that the MCM problem can be considered as a more general design of a $2N \times 2$ matrix multiply problem. This will then also cover the two cases $4 \times 2+$ and $3 \times 5+$ discussed in this paper. However, the conventional RAG-n algorithm used in this study with a single input and multiple outputs then needs to be modified to include such a CSE-like input permutation search. The same idea can also be applied to the 13 different methods discussed by Macleod [3]. We have also recently seen successful improvements of the RAG-n heuristic based on the HCUB metric [39] and the differential RAG [40], which will be especially beneficial for coefficient bit widths larger than the 12 bits used in this paper.

Some of the above-mentioned MCM arithmetic concepts may in fact further improve the implementation effort of the fast DFT algorithms for certain length or bit width and may be the basis for further studies. The main result of this paper, however, is that due to recent advances in MCM algorithms, Rader and chirp-z have become viable options over the conventional radix-2 FFT. This contrasts with previously accepted understanding, as expressed by Burrus and Parks [28, page 37], who state: "if implemented on digital hardware, the chirp-z transform does not seem advantageous for calculating the normal DFT."

3.5. Quantization noise of alternative DFT algorithms

Since fast DFTs and FFTs can be used, for instance, to implement a fast convolution, it is important to analyze and determine the required quantization error of the algorithms. To simplify our discussion let us make the following assumptions that are used in textbooks, like [25, 30].

(a) The quantization errors are uncorrelated.
(b) The errors are uniformly distributed random variables of $(B+1)$-bit signed fractions, such that the variance becomes $2^{-2B}/12$.
(c) The complex multiplication with 4 multiplications has a quantization error of $\sigma^2 = 4 \times 2^{-2B}/12 = 2^{-2B}/3$.
(d) The input signal x is random white noise with variance $\sigma_x^2 = 1/(3N^2)$.

With this assumption we can determine the quantization noise of the DFT since N source contributes to each output as

$$\mathbb{E}_{\mathrm{DFT}} = N \times \sigma^2. \tag{9}$$

From (d) we compute the output variance of the DFT/FFT as

$$\begin{aligned} \mathbb{E}_X = \mathbb{E}\{|X[k]|^2\} &= \sum_{n=0}^{N-1} \mathbb{E}\left\{|x[n]|^2\right\} |W_N^{nk}|, \\ \mathbb{E}_X &= N\sigma_x^2 = \frac{1}{3N}, \end{aligned} \tag{10}$$

and the noise-to-output ratio becomes

$$\frac{\mathbb{E}_{\mathrm{DFT}}}{\mathbb{E}_X} = 3N^2\sigma^2. \tag{11}$$

This results in a one-bit loss in the noise-to-signal ratio as the length doubles. If inside the DFT a double wide accumulator is used, the noise reduces to

$$\mathbb{E}_{\mathrm{DFT2accu}} = \sigma^2, \tag{12}$$

which provides the best performance of all algorithms. The same results occur with the Rader DFT if we use a double-width accumulator. For the chirp-z DFT, the input and output complex multiplications introduce another $2\sigma^2$ noise, and the overall output budget becomes

$$\mathbb{E}_{\mathrm{CZT}} = 3 \times \sigma^2 \tag{13}$$

assuming that we use a double width accumulator in the FIR part for the chirp-z DFT. For the FFT, let us have a look at the popular radix-2 Cooley-Tukey FFT. Here, a double-length accumulator does not help to reduce the round-off noise since the output of the butterfly must be stored in the same $(B-1)$-bit memory location. To avoid overflow, we can scale the input by N, but the quantization error

$$\mathbb{E}_{\mathrm{FFTinput}} = N \times \sigma^2 \tag{14}$$

will be essential. Double FFT length results in a loss of 1 bit in accuracy. A better approach is to scale at each stage by 1/2. Then each of the $N = 2^n$ output nodes is connected to 2^{n-s-1} butterflies and therefore to 2^{n-s} noise sources. Thus the output mean-square magnitude of the noise is

$$\begin{aligned} \mathbb{E}_{\mathrm{FFT}} &= \sigma^2 \sum_{s=0}^{n-1} 2^{n-s} \left(\frac{1}{2}\right)^{2n-2s-2} \\ &= 4\sigma^2\left(1 - 0.5^n\right) \approx 4 \times \sigma^2, \end{aligned} \tag{15}$$

and the noise-to-signal ratio becomes

$$\frac{\mathbb{E}_{\mathrm{FFT}}}{\mathbb{E}_X} = 12N \times \sigma^2. \tag{16}$$

Now we only have a 1/2-bit per stage reduction in the noise-to-signal ratio, as first shown by Welch [41]. Table 5 summarizes the results for the different methods.

The noise can be further reduced by using a higher radix in the FFT, more guard bits, or a block floating-point format, but these methods will usually require more hardware resources.

[3] The authors are grateful to an anonymous referee for this suggestion.

Table 5: Noise in length $N = 2^n$ DFT and FFT algorithms width $\sigma^2 = 2^{-2B}/3$.

Algorithm type	Noise variance	Noise-to-signal ratio
Direct DFT matrix multiply	$N\sigma^2$	$3N^2 \times \sigma^2$
DFT double width accumulator	σ^2	$3N\sigma^2$
Rader double width FIR accumulator	σ^2	$3N\sigma^2$
Chirp-z DFT	$3\sigma^2$	$9N\sigma^2$
Radix-2 FFT input scaling	$(N-1)\sigma^2$	$3N(N-1)\sigma^2$
Radix-2 FFT intermediate scaling	$4\sigma^2(1-0.5^n)$	$12N\sigma^2(1-0.5^n)$

4. CONCLUSION

This paper shows that both Rader and Bluestein Chirp-z DFTs are viable implement paths for DFT or large Radix FFTs when the multiplier block is implemented with a reduced adder graph technique. This paper shows that the CZT offers lower costs than the Rader design due to the larger number of redundant coefficients in the CZT, which is beneficial to RAG-n. The DFT hardware effort in an implementation via RAG-n CZT has only $O(N)$ effort (i.e., not quadratic $O(N^2)$ as for the direct DFT method) and provides a DFT with very short latency, which is attractive when the DFT is used as a coprocessor. For a 64-point RAG-CZT, 92% of the resources are used for the linear filter, 7% for the complex I/O multiplier, and 1% for coefficient storage.

From a quantization standpoint, both Rader and Bluestein Chirp-z DFTs perform better than the Radix-2 Cooley-Tukey FFT for fixed-point implementations. The Rader algorithm reaches the minimum quantization error of the direct matrix DFT algorithm.

ACKNOWLEDGMENTS

The authors would like to thank Xilinx and Synopsys (FSU ID 10806) for their support under the university program. Thanks also to the anonymous reviewers for their helpful suggestions for improving this paper.

REFERENCES

[1] M. T. Heideman, D. H. Johnson, and C. S. Burrus, "Gauss and the history of the fast Fourier transform," *IEEE Acoustic Speech & Signal Processing Magazine*, vol. 1, no. 4, pp. 14–21, 1984.

[2] C. S. Burrus, "Index mappings for multidimensional formulation of the DFT and convolution," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 3, pp. 239–242, 1977.

[3] M. D. Macleod, "Multiplierless implementation of rotators and FFTs," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 17, pp. 2903–2910, 2005.

[4] Altera Corporation, *FFT: MegaCore Function User Guide*, Ver. 2.1.3, 2004.

[5] Xilinx Corporation, "Fast Fourier Transform," LogiCore v3.1, November 2004.

[6] B. Baas, "SPIFFEE: an energy-efficient single-chip 1024-point FFT processor," 1998, http://nova.stanford.edu/~bbaas/fftinfo.html.

[7] G. Sunada, J. Jin, M. Berzins, and T. Chen, "COBRA: an 1.2 million transistor expandable column FFT chip," in *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '94)*, pp. 546–550, Cambridge, Mass, USA, October 1994.

[8] Texas Memory Systems, "TM-66 swifft chip," 1996, http://www.texmemsys.com.

[9] SHARP Microeletronics, "Bdsp9124 digital signal processor," 1997, http://www.butterflydsp.com.

[10] P. Lavoie, "A high-speed CMOS implementation of the Winograd Fourier transform algorithm," *IEEE Transactions on Signal Processing*, vol. 44, no. 8, pp. 2121–2126, 1996.

[11] G. Panneerselvam, P. Graumann, and L. Turner, "Implementation of fast Fourier transforms and discrete cosine transforms in FPGAs," in *Proceedings of the 5th International Workshop on Field-Programmable Logic and Applications (FPL '95)*, vol. 975 of *Lecture Notes in Computer Science*, pp. 272–281, Oxford, UK, August-September 1995.

[12] G. Goslin, "Using Xilinx FPGAs to design custom digital signal processing devices," in *Proceedings of the DSPX*, pp. 565–604, January 1995.

[13] N. Shirazi, P. M. Athanas, and A. L. Abbott, "Implementation of a 2-D fast Fourier transform on an FPGA-based custom computing machine," in *Proceedings of the 5th International Workshop on Field-Programmable Logic and Applications (FPL '95)*, vol. 975 of *Lecture Notes in Computer Science*, pp. 282–292, Oxford, UK, August-September 1995.

[14] C. Dick, "Computing 2-D DFTs using FPGAs," in *Proceedings of the 6th International Workshop on Field-Programmable Logic, Smart Applications, New Paradigms and Compilers (FPL '96)*, vol. 1142 of *Lecture Notes in Computer Science*, pp. 96–105, Darmstadt, Germany, September 1996.

[15] D. R. Bull and D. H. Horrocks, "Reduced-complexity digital filtering structures using primitive operations," *Electronics Letters*, vol. 23, no. 15, pp. 769–771, 1987.

[16] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *IEE Proceedings G: Circuits, Devices and Systems*, vol. 138, no. 3, pp. 401–412, 1991.

[17] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proceedings: Circuits, Devices and Systems*, vol. 141, no. 5, pp. 407–413, 1994.

[18] A. G. Dempster and M. D. Macleod, "Comments on "Minimum number of adders for implementing a multiplier and its application to the design of multiplierless digital filters"," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 2, pp. 242–243, 1998.

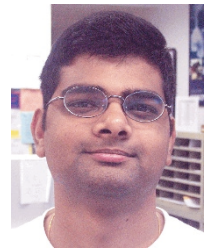
[19] O. Gustafsson, A. G. Dempster, and L. Wanhammar, "Extended results for minimum-adder constant integer multipliers," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 1, pp. 73–76, Phoenix, Ariz, USA, May 2002.

[20] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, no. 9, pp. 569–577, 1995.

[21] R. T. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 10, pp. 677–688, 1996.

[22] M. D. Macleod and A. G. Dempster, "Common subexpression elimination algorithm for low-cost multiplierless implementation of matrix multipliers," *Electronics Letters*, vol. 40, no. 11, pp. 651–652, 2004.
[23] M. D. Macleod and A. G. Dempster, "Multiplierless FIR filter design algorithms," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 186–189, 2005.
[24] S. D. Stearns and D. R. Hush, *Digital Signal Analysis*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1990.
[25] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1992.
[26] E. Brigham, *FFT*, Oldenbourg, München, Germany, 3rd edition, 1987.
[27] R. Ramirez, *The FFT: Fundamentals and Concepts*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1985.
[28] C. Burrus and T. Parks, *DFT/FFT and Convolution Algorithms*, John Wiley & Sons, New York, NY, USA, 1985.
[29] D. Elliott and K. Rao, *Fast Transforms Algorithms, Analyses, Applications*, Academic Press, New York, NY, USA, 1982.
[30] H. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*, Springer, Heidelberg, Germany, 1990.
[31] C. Rader, "Discrete Fourier transform when the number of data samples is prime," *Proceedings of the IEEE*, vol. 56, no. 6, pp. 1107–1108, 1968.
[32] J. McClellan and C. Rader, *Number Theory in Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1979.
[33] M. Narasimha, K. Shenoi, and A. Peterson, "Quadratic residues: application to chirp filters and discrete Fourier transforms," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '76)*, vol. 1, pp. 376–378, Philadelphia, Pa, USA, April 1976.
[34] U. Meyer-Bäse, D. Sunkara, E. Castillo, and A. Garcia, "Custom instruction set NIOS-based OFDM processor for FPGAs," in *Wireless Sensing and Processing*, vol. 6248 of *Proceedings of SPIE*, Kissimmee, Fla, USA, April 2006, article number 62480O.
[35] S. F. Gorman and J. M. Wills, "Partial column FFT pipelines," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, no. 6, pp. 414–423, 1995.
[36] O. Gustafsson, A. G. Dempster, and L. Wanhammar, "Multiplier blocks using carry-save adders," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '04)*, vol. 2, pp. 473–476, Vancouver, BC, Canada, May 2004.
[37] S. A. White, "Applications of distributed arithmetic to digital signal processing: a tutorial review," *IEEE Transactions on Acoustics, Speech and Signal Processing Magazine*, vol. 6, no. 3, pp. 4–19, 1989.
[38] M. Soderstrand, W. Jenkins, G. Jullien, and F. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press Reprint Series, IEEE Press, New York, NY, USA, 1986.
[39] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," to appear in *ACM Transactions on Algorithms*.
[40] O. Gustafsson, "A difference based adder graph heuristic for multiple constant multiplication problems," in *Proceedings Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '07)*, New Orleans, La, USA, May 2007, submitted.
[41] P. Welch, "A fixed-point fast Fourier transform error analysis," *IEEE Transactions on Audio and Electroacoustics*, vol. 17, no. 2, pp. 151–157, 1969.

Uwe Meyer-Bäse received his B.S.E.E., M.S.E.E., and Ph.D. "Summa cum Laude" degrees from the Darmstadt University of Technology in 1987, 1989, and 1995, respectively. In 1994 and 1995, he hold a Postdoctoral position in the "Institute of Brain Research" in Magdeburg. In 1996 and 1997, he was a Visiting Professor at the University of Florida. From 1998 to 2000, he was a Research Scientist for ASIC Technologies for The Athena Group, Inc., where he was responsible for development of high-performance architectures for digital signal processing. He is now a Professor in the Electrical and Computer Engineering Department at Florida State University. During his graduate studies, he worked part time for TEMIC, Siemens, Bosch, and Blaupunkt. He holds 3 patents, has supervised more than 60 master thesis projects in the DSP/FPGA area, and gave four lectures at the University of Darmstadt in the DSP/FPGA area. In 2003, he was awarded the "Habilitation" (venia legendi) by the Darmstadt University of Technology a requirement for attaining tenured Full Professor status in Germany. He received in 1997 the Max-Kade Award in Neuroengineering and the Humboldt Research Award in 2005. He is an IEEE, BME, SP, and C&S Society Member.



Hariharan Natarajan was born on 11th February 1980, in Chennai, India. After finishing high school in Hyderabad, India, he graduated from Madras University with B.S. degree in instrumentation and control engineering. He started his Masters of Science programme at Florida State University in fall 2001 and graduated in Summer 2004. His area of specialization is digital electronics and ASIC design.

Andrew G. Dempster is Director of Research in the School of Surveying and Spatial Information Systems at the University of New South Wales, Sydney, Australia. He holds B.E. and M.Eng.Sc. degrees from UNSW and a Ph.D. from the University of Cambridge. He worked for several years in telecommunications and satellite systems, leading the development of the first GPS receiver designed in Australia. For nine years, he held academic positions at the University of Westminster in London and has been at UNSW since 2004. His research interests are design of satellite navigation receiver systems, new positioning technologies, arithmetic circuits, and morphological image processing.