High-Throughput and Compact FFT Architectures Using the Good-Thomas and Winograd Algorithms

Nikhilesh Bhagat, Daniel Valencia, Amirhossein Alimohammad, and fred harris
Department of Electrical and Computer Engineering
San Diego State University, San Diego, U.S.A.

***Abstract*—This article presents two hardware architectures for the efficient implementation of fast Fourier transform (FFT) based on the combined Good-Thomas and Winograd Fourier transform algorithms. The combined algorithms require fewer multiplications compared to the other FFT algorithms by eliminating twiddle factor multiplications. Two hardware architectures are presented: one is a fully-pipelined architecture using multi-dimensional Chinese remainder theorem for a high-throughput implementation and the other is a time-multiplexed architecture, which utilizes the folding transformation for the compact realization of short-length Winograd Fourier transforms. Both designs are fully-parameterizable and have been verified against their synthesizable Verilog descriptions. The characteristics and the implementation results of the two hardware architectures on a Xilinx field-programmable gate array (FPGA) are presented. To the best of our knowledge, this article presents the first hardware realization of FFT based on the combined Good-Thomas and Winograd FFT algorithms. The application of the proposed FFT architectures in the physical layer of the long-term evolution (LTE) wireless standard is discussed.**

I. Introduction

While fast Fourier transform (FFT) algorithms have reached maturity, their efficient hardware implementation is still of interest for a wide range of applications. Variants of FFT algorithms use a divide-and-conquer strategy to map the original computation into several sub-computations in such a way that the cost of subproblems (including computational complexity and memory accesses) and the mapping overhead is less than the cost of the original solution [1]. Various FFT algorithms have been developed over the decades to find a balance between the cost of subproblems and the mapping overhead.

The FFT algorithms can be broadly divided into two main categories: (i) the composite (non-prime) length FFT algorithms, such as the Cooley-Tukey (CT) [2] and the Good-Thomas (GT) algorithms [3], [4], where the length of the Fourier transform N can be any length that can be subdivided into smaller transforms of sizes N_1 and N_2 (N_1 and N_2 are integer factors of N, i.e., $N = N_1 \times N_2$); and (ii) the prime-length FFT algorithms, such as the Winograd [5] and Rader algorithms [6], where the Fourier transform length is a prime number.

The breakthrough of the Cooley-Tukey FFT comes from the fact that it brings the N^2 computational complexity of discrete Fourier transform down to an order of $N \log N$ operations. Special cases of the Cooley-Tukey algorithm for $N = 2^n$ or 4^n, or r^n in general, can be derived for what is called the mixed-radix FFT algorithms [7]. Radix 2 and 4 are widely utilized in many signal processing applications. Higher radices, such as Radix 8 and up, decreases the number operations, but they generally require a relatively complex hardware for small enhancements. Split-radix algorithms [8] use a different radix for the even and odd parts of the transform and is known for achieving the minimum known number of operations for powers-of-two FFTs. The mixed-radix generalization may use different algorithms depending on whether the factors satisfy certain restrictions.

The Cooley-Tukey algorithm and its radix-based variants are attractive due to their simple structures and their relatively low arithmetic complexity for lengths equal to powers of 2 or 4, which has led to many relevant published works [9] [10] [11] [12] [13] [14]. However, when the initial transform length is divided into sublengths which are not relatively prime, these groups of algorithms lead to unavoidable auxiliary complex multiplications by roots of unity, known as twiddle factors. When the factors of the transform lengths are co-prime (i.e., if $N = N_1 \times N_2$ and $gcd(N_1, N_2) = 1$), Good and Thomas proposed an alternative divide-and-conquer strategy based on the index transformations to factorize the Fourier transform into multi-dimensions without the twiddle factor multiplications. In fact, only real multiplications are utilized. This algorithm is also known as the prime-factor algorithm (PFA). These multi-dimensional shorter length sequences can be transformed using the Winograd Fourier transform algorithm (WFTA), which further reduces the number of required multiplications.

This article presents two architectures for compact and also high-throughput (HT) implementations of Fourier transforms using the combined Good-Thomas and Winograd algorithms. The rest of this article is organized as follows. Section II briefly discusses the combined Good-Thomas and Winograd algorithms. The high-throughput and compact FFT hardware architectures of the combined PFA and WFTA algorithms are explained in Sections III and IV, respectively. Section V discusses the application of the proposed architectures in the physical layer of the long-term evolution (LTE) wireless standard. Finally, section VI makes some concluding remarks.

II. Combined Good-Thomas and Winograd FFT Algorithms

The Good-Thomas algorithm involves an indexing of the input array by exploiting the Chinese remainder theorem (CRT), which transforms the one-dimensional input sequence of length N into two dimensions $N_1 \times N_2$, and computing the two-dimensional transform along each dimension, and finally re-indexing of the results back to one dimension using the Ruritanian correspondence mapping (RCM) [15]. Note that these smaller transforms of size N_1 and N_2 can be evaluated by applying PFA recursively. The input data is stored in a two-dimensional array by starting in the upper left corner and listing the indices down the extended diagonal. Because the number of rows and the number of columns are co-prime, the extended diagonal passes through every element of the array. The input indices can be described by the CRT as:

$$n_1 = n \bmod N_1,$$

$$n_2 = n \bmod N_2,$$

where $n_1 = 0, 1, 2, \ldots, N_1 - 1$; $n_2 = 0, 1, 2, \ldots, N_2 - 1$; and

$$n = [n_1 N_2 M_2 + n_2 N_1 M_1] \bmod N,$$

where $n = 0, 1, 2, \ldots, N - 1$. The values of M_1 and M_2 can be solved through $[N_1 M_1 + N_2 M_2] = 1$ by using the Euclidean algorithm, where N_1 and N_2 are scalar and integer coefficients of the given linear Diophantine equations, $N_2 M_1 = 1 \bmod N_1$, and $N_1 M_2 = 1 \bmod N_2$.

The output indices are defined using the RCM as follows:

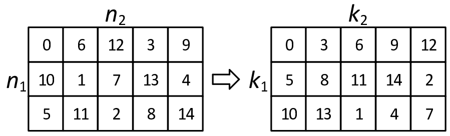
$$k_1 = k M_2 \bmod N_1,$$

$$k_2 = k M_1 \bmod N_2,$$

where $k_1 = 0, 1, 2, \ldots, N_1 - 1$; $k_2 = 0, 1, 2, \ldots, N_2 - 1$; and

$$k = [k_1 N_2 + k_2 N_1] \bmod N,$$

where $k = 0, 1, 2, \ldots, N - 1$, and M_1 and M_2 are integers determined during the input mapping. As an example, the input and output mapping for a 15-point FFT, where $N = 15$, $N_1 = 3$, and $N_2 = 5$, as shown in Fig. 1, can be solved using the above equations.



0	6	12	3	9
10	1	7	13	4
5	11	2	8	14

0	3	6	9	12
5	8	11	14	2
10	13	1	4	7

Fig. 1. The block diagram of the Good-Thomas mapping for a 15-point Fourier transform.

The DFT $X[k]$ of a sequence $x[n]$ of N terms is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] \omega^{nk}, \tag{1}$$

where the $x[n]$ is viewed as N consecutive samples, $x[nT]$, of a continuous signal $x(t)$, and $\omega^{nk} = e^{-j2\pi nk/N}$ are the twiddle factors. By substituting these indices in Equation (1), we can write:

$$X_{k_1 k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} W_{N_1 N_2}^{(n_1 M_2 N_2 + n_2 M_1 N_1)(k_1 N_2 + k_2 N_1)} x[n_1 n_2],$$

where the product in the exponent can be written as:

$$W_N^{nk} = W_N^{n_1 k_1 M_2 N_2^2} W_N^{n_2 k_2 M_1 N_1^2} \times W_N^{n_1 k_2 M_2 N_1 N_2} W_N^{n_2 k_1 M_1 N_1 N_2}.$$

From the CRT/RCM equations, we can write:

$$M_2 N_2 = (1 - M_1 N_1),$$

$$M_1 N_1 = (1 - M_2 N_2),$$

$$N_1 N_2 = N,$$

$$W^{N_1 N_2} = 1.$$

Therefore, the exponent will reduce to:

$$W_N^{nk} = W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2}.$$

The Good-Thomas Fourier transform can thus be written as:

$$X_{k_1 k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2} x[n_1 n_2]. \tag{2}$$

Equation (2) is a decoupled two-dimensional $N_1 \times N_2$-point transform, i.e., either the rows or columns can be transformed in any order. An important property of the Good-Thomas Fourier transform is that there are no twiddle factor multiplications involved. The pseudo-code of the GT FFT algorithm is given in Algorithm 1.

Algorithm 1 Good-Thomas Fourier Transform Algorithm

```
procedure INPUT MAPPING:
  for $i \in N_1$ do
    for $j \in N_2$ do
      CRT($i, j$) = $(j-1) \times N_2 \times M_2 + (i-1) \times N_1 \times M_1) \mod N$;
    end for
  end for
end procedure
procedure COLUMN TRANSFORMATION:
  for $i \in N_2$ do
    Select all the elements of column $i$;
    Perform $N_1$-point FFT for the $N_1$ elements of column $i$;
    Returns a 2D array with the FT of all $N_2$ column vectors;
  end for
end procedure
procedure ROW TRANSFORMATION:
  for $i \in N_1$ do
    Select all the elements of row $i$;
    Perform $N_2$-point FFFT for the $N_2$ elements of row $i$;
    Returns a 2D array with the FT of all $N_1$ row and $N_2$ column vectors;
  end for
end procedure
procedure OUTPUT MAPPING:
Initialize $k = 0$
  for $i \in N_1$ do
    for $j \in N_2$ do
      $k$ = $(j-1)$ × $N_2$ + $(i-1)$ × $N_1 \bmod N$;
      $k = k + 1$;
    end for
  end for
end procedure
```

The limitation of the PFA algorithm is that it requires that the lengths along each dimension N_1 and N_2 to be co-prime. It is reasonable to assume that these lengths are relatively small since Good's mapping can provide a full multi-dimensional factorization when N is highly composite. In fact, a set of small Fourier transforms (e.g., N_i = 2, 3, 4, 5, 7, 8, 16) is sufficient to provide a set of feasible longer lengths. While PFA requires a more complicated indexing and re-indexing of data, even though the mapping requires no arithmetic operation and only permutations, the index mapping can be precomputed and stored for indirect indexing.

To efficiently compute these set of relatively small lengths FFTs that are co-prime, we utilize the WFTA [16], which requires only $O(N)$ real irrational multiplications, leading to a proven achievable lower bound on the number of multiplications $2N$. Note that WFTA has failed to replace the popular Cooley-Tukey algorithm and its radix-based variants for FFTs of sequences with a relatively large lengths. This is due to the fact that replacing twiddle factor complex multiplications by a smaller number of real multiplications may increase the number of additions for relatively large transform lengths.

The combined PFA and WFTA algorithms can be summarized in the following four steps: (i) mapping the input sequence based on the CRT into two dimensions $N_1 \times N_2$, where N_1 and N_2 are co-prime; (ii) calculating N_1-point WFTAs of N_2 columns. The result is a two-dimensional array with the Fourier transform of all N_2 column vectors; (iii) calculating N_2-point WFTAs of N_1 rows. The result is a two-dimensional array with the Fourier transform of all N_1 row vectors; (iv) mapping the two-dimensional output sequence back onto a one-dimensional sequence of N samples based on the RCM [15]. Note that the row and column transforms can be performed in either orders. The above four steps can be repeated iteratively to calculate FFTs with smaller composite co-prime transform lengths. For example a 1008-point FFT can be decomposed into 63×16-point FFTs, and a 63-point FFT can be decomposed into 7×9-point FFTs.

To compare the number of multiplications and additions required by various FFT algorithms, let's write Equation (2) as:

$$X_{k_1k_2} = \sum_{n_1=0}^{N_1-1} W_{N_1}^{n_1k_1}\, x[n_1] \quad \sum_{n_2=0}^{N_2-1} \quad W_{N_2}^{n_2k_2}\, x[n_2], \tag{3}$$

where $x[n_1]$ and $x[n_2]$ denote row/column mapped input samples. Assuming that the row and column Fourier transforms are computed using DFTs, this results in N_1 column transforms, each requiring approximately N_2^2 multiplications and additions, and N_2 row transforms, each requiring N_1^2 multiplications and additions. The GT FFTs require $N_1(N_2)^2 + N_2(N_1)^2$ multiplications and $N_1(N_2)^2 + N_2(N_1)^2$ additions while the CT FFTs require $N_1(N_2)^2 + N_2(N_1)^2 + N_1N_2$ multiplications and $N_1(N_2)^2 + N_2(N_1)^2$ additions. Winograd transforms for N_1 and N_2 point FFTs convert the prime length FFTs into cyclic convolutions without using any complex twiddle multiplications while using the minimum number of multiplications. The number of additions and multiplications for different transform lengths using the Winograd algorithm are given in Table I. Thus, the total number of multiplications required for an N-point FFT, implemented using the combined Good-Thomas and Winograd FFT algorithms is $N_2(N_{1w}) + N_1(N_{2w})$, where N_1 and N_2 are the two prime factors of N and N_{1w} and N_{2w} are the number of multiplications for Winograd transforms lengths of N_1 and N_2, respectively.

Fig. 2 and Fig. 3 show the number of multiplications and additions required by various FFT algorithms. Table II gives the total number of additions and real multiplications for a 1008-point GT+WFT and also 1024-point CT, radix-2, and radix-4 FFT algorithms. It can be seen that the GT+WFTA approach requires the minimum number of multiplications at the cost of more additions, while the total number of arithmetic operations is still significantly lower for GT+WFT. This makes it a suitable candidate for an efficient hardware implementation, which can ultimately lead to a less power-consuming design.

TABLE I
NUMBER OF ADDITIONS AND MULTIPLICATIONS FOR DIFFERENT TRANSFORM LENGTHS USING THE WINOGRAD ALGORITHM

N	Multiplications	Additions
2	0	2
3	2	6
4	0	8
5	5	17
7	8	38
8	2	26
9	10	44
16	10	74

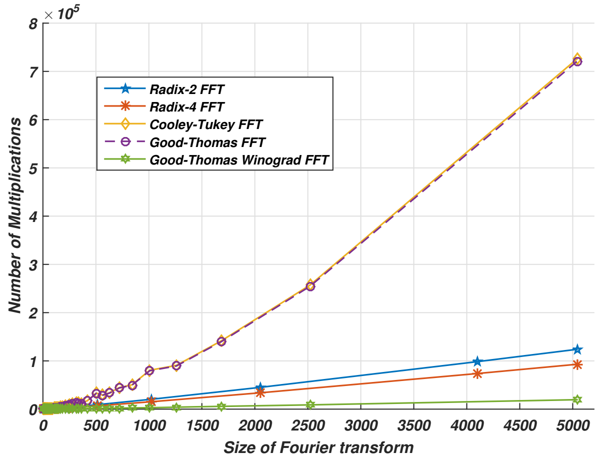


Fig. 2. Number of multiplications for different FFT algorithms.

TABLE II
NUMBER OF ADDITIONS AND REAL MULTIPLICATIONS REQUIRED FOR A 1008-POINT FFT USING VARIOUS FFT ALGORITHMS

FFT	GT+WFT	CT	Radix-2	Radix-4
Multiplications	2902	80640	20480	15360
Additions	34416	79632	30720	28160
Total	37318	160272	51200	43520

III. A HIGH-THROUGHPUT ARCHITECTURE FOR THE COMBINED GOOD-THOMAS AND WINOGRAD FFT

The top-level block diagram of our high-throughput architecture is shown in Fig. 4. The architecture consists of four computational stages: input mapping, N_1-point and N_2-point Winograd Fourier transforms (WFTs), and output mapping. The architecture reads the real and imaginary parts of N complex-valued input samples in sequence and stores them in an input block memory (BRAM). The outputs are produced in sequence and stored in an output BRAM. The FFT architecture uses a finite state machine (FSM) to control the sequence of FFT operations. When the input data is valid (indicated by the

TABLE III
CHARACTERISTICS AND IMPLEMENTATION RESULTS OF THE HIGH-THROUGHPUT FFT ARCHITECTURE

FFT points	2	3	4	5	7	8	9	16	1008
Multipliers	0	2	0	5	8	2	10	10	28
Adders	2	6	8	17	35	26	44	74	153
Registers	467	876	984	1123	2061	1385	2764	3576	8513
LUTs	598	978	1126	1605	3029	2330	3949	6003	14615
DSP48s	0	4	0	10	16	4	20	20	56
Pipeline Stages	1	8	3	10	11	8	13	10	34
Clock Frequency (MHz)	530	530	530	530	530	530	530	530	386
Computational Latency (Cycles)	1	8	3	10	11	8	13	10	2084
Overall Latency (Cycles)	3	11	7	15	18	16	22	26	4100

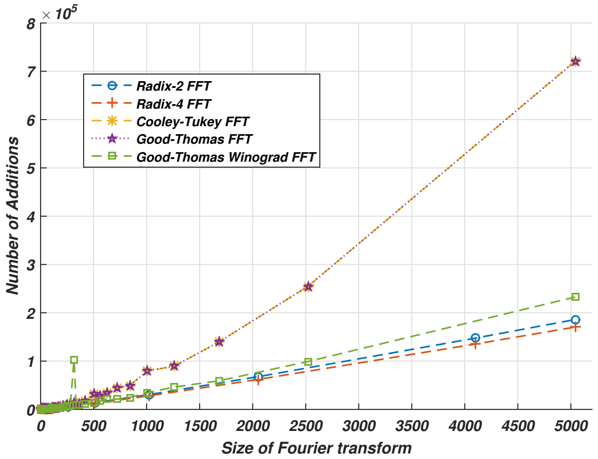


Fig. 3. Number of additions for different FFT algorithms.

validIn control signal), the controller goes to the *CRT mapping* state to map the one-dimensional input sequence onto a two-dimensional sequence based on the CRT. Let N denote the length of the transform and N_1 and N_2 be its co-prime factors. The index value of the element in the (i, j) index is given by $(j-1) \times N_2 \times M_2 + (i-1) \times N_1 \times M_1) \mod N$, where M_1 and M_2 are calculated using the linear Diophantine equation [7]. These N CRT index values are pre-computed and stored in the CRT lookup table (LUT) at initialization. We utilize two and three-dimensional CRT mappings when converting a one-dimensional sequence into a two-dimensional sequence. The three-dimensional CRT mapping is the repetitive use of the two-dimensional CRT mappings. To explain the differences between the two and three-dimensional realizations, assume a 1008-point FFT. The CRT mapping starts with the mapping of 1008 inputs into a 16×63 two-dimensional matrix of indices. Then each 63 elements in a row of this matrix is CRT-mapped into a 7×9 two-dimensional matrix of indices. In the two-dimensional CRT mapping, WFTs are applied to 7×9 matrices. That is, for the first 63 samples in a row, we perform 9-point transforms 7 times and also 7-point transforms 9 times (row- and column-based). The process continues with the CRT-mapping of the 63 elements in the second row and so on. In the three-dimensional implementation, we also map 1008 inputs into a 16×63 two-dimensional matrix of indices and then each 63 elements in a row of this matrix is CRT-mapped into a 7×9 two-dimensional matrix of indices. However, in the three-dimensional CRT mapping, we replace these indices into the corresponding row in the original 16×63 matrix of indices. To apply WFT, we first perform 9-point row-based transforms $1008/9 = 112$ times, 7-point row-based transforms $1008/7 = 144$ times and then 16-point column-based transforms $1008/16 = 63$ times. The three-dimensional CRT mapping especially reduces the latency required to perform higher order FFTs.

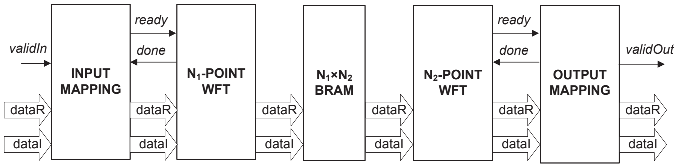


Fig. 4. High-throughput architecture of the combined PFA and WFTA.

After N_1 clock cycles, which is how long it takes to map the input samples onto the first column, the control goes to the *column transformation* state while the *input mapping* module starts mapping the input samples onto the second column. In this state, WFT of the N_1-point column sequence is performed. We derived and implemented WFT of various small block lengths using their signal flow graphs (SFGs). As an example, the SFG of a 3, 5, and 8-point WFTs are shown in Fig. 5. The multiplier coefficients of different SFGs are stored in a LUT at initialization. For high-throughput operations, the feedforward SFGs of relatively small transform lengths are pipelined, utilizing three-stage and one-stage pipelined multipliers and adders, respectively. The output of N_1-point WFT will be stored in an $N_1 \times N_2$ BRAM. The process of input mapping and N_1-point WFT will be repeated N_2 times so that all N input samples are column-transformed.

After performing column transformation, the control unit moves to the *row transformation* state. In this state, samples at the distance of N_1 points are read from the BRAM to form an N_2-point row sequence. After every N_2-point WFT, the control unit moves to the *output mapping* state. The output mapping is performed using RCM technique to convert a given two-dimensional sequence back to a one-dimensional sequence. According to the RCM index mapping, the index value of the element in the (i, j) index is given by $(j-1) \times N_2 + (i-1) \times N_1 \mod N$. These N index values are pre-computed and are stored in the RCM LUT at initialization. The process of N_2-point WFT and output mapping will be repeated N_1 times for all N_1 rows. Once the FFT computation is completed, the *validOut*

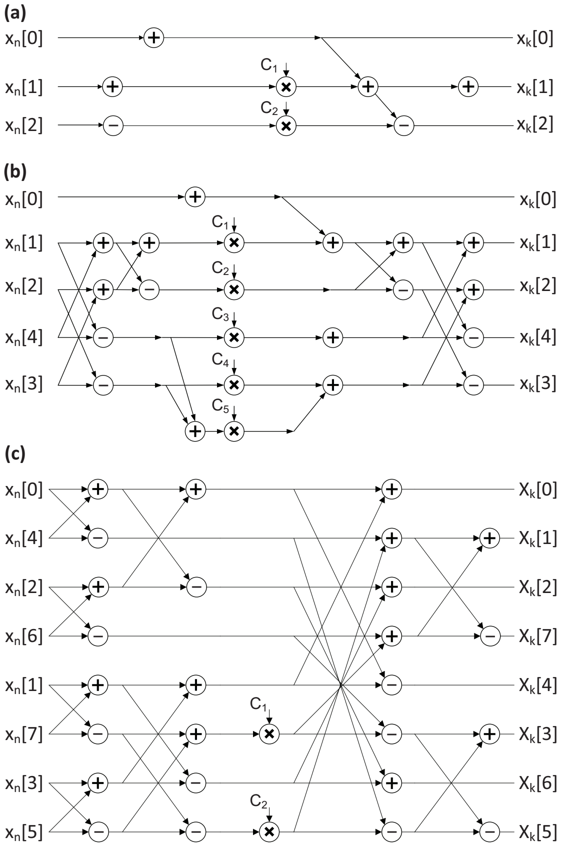


Fig. 5. The SFGs for the (a) 3-point, (b) 5-point, and (c) 8-point WFTs.

signal is asserted indicating that the transformed sequence is ready at the output. The high-throughput realization of FFTs is supported by the pipelined implementation of the Winograd's SFGs and also the temporal parallelization among the sub-blocks of the FFT architecture. The FFT computational blocks are operating in a time-overlapped fashion using *ready* and *done* hand-shaking control signals for inter-communications between different blocks. As soon as data is ready at a stage, it is passed on to the next module and it accepts new data for processing.

We developed a comprehensive library of bit-true fixed-point and floating-point arithmetic and logical operations in Mex-C (for fast simulation). The library includes parameterizable modules with adjustable bit-widths, that provides a flexible simulation environment for the bit-true comparison of approximated fixed-point values with function values in double precision. Our fixed-point FFT designs are fully-parameterizable in which the fixed-point representation of inputs, outputs, intermediate signals, and multiplier coefficients, can be adjusted. The characteristics and implementation results of the 16-bit high-throughput FFT architecture for different block lengths on a Xilinx Virtex-6 CX6VLX240T field-programmable gate array (FPGA) are given in Table III. We verified our design in fixed-point Matlab against its synthesizable Verilog description. The number of DSP48s is twice as many as the number of multipliers as the multipliers multiply complex-valued data with real-valued coefficients. The 1008-point FFT implementation utilizes the three-dimensional CRT mapping. The overall latency includes the number of clock cycles required to write the input samples onto an input BRAM and reading the transformed samples from an output BRAM. The input and output samples are in-order.

IV. Compact Architectures for the Combined Good-Thomas and Winograd FFT

For the compact realization of the FFT architecture, we utilize folding transformation [17] to implement the WFT of different transformation lengths (i.e., 2, 3, 4, 5, 7, 8, 9, and 16 points in this work) using only one shared stage of computation. For example, the SFG of an 8-point WFT shown in Fig. 5(c) is realized by reusing a single stage of adders, subtractors, and multipliers to perform its required operations, iteratively. Then we use the folded Winograd SFGs of different block lengths in the time-multiplexed FFT architecture, shown in Fig. 6. In this architecture, the four operations of input mapping, column transformation, row transformation, and output mapping are performed iteratively. The *N-point WFT* block implements various short-length WFT SFGs, each realized using a single stage of adders and multipliers. Similar to the high-throughput approach, the N CRT index values and N RCM index values are pre-computed and stored in the CRT and RCM LUTs, respectively, at initialization. The architecture consists of two input multiplexers *M1* and *M2* and one output demultiplexer *D1*. The intermediate samples are stored in the $N_1 \times N_2$ BRAM. The *address* line of the BRAM can be a CRT index for an input sample, an RCM index for an output sample, or an address from the N_1-point or N_2-point WFTs, selected by *M1*. The *data* line of the BRAM can be from the input or from output of the *N-point WFT module*. The output demultiplexer *D1* passes the transformed samples to the output or to the WFT modules. The select control signals of the multiplexers and the demultiplexer are generated by the FSM controller. Table IV gives the characteristics and the implementation results of the compact FFT architecture on the same FPGA device used for the high-throughput FFT architecture implementation.

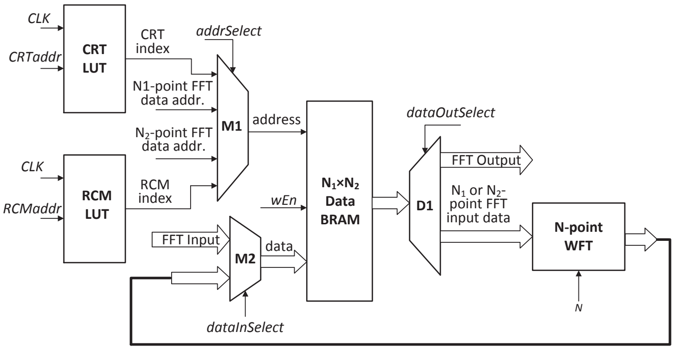


Fig. 6. The top-level architecture of the time-multiplexed FFT architecture.

To realize an even more compact implementation of the FFT architecture, we use folding transformation to implement the Winograd SFGs of different block lengths using a single stage of adders and multipliers only, as shown in Fig. 8. The number of adders and multipliers in this folded architecture is

TABLE IV
CHARACTERISTICS AND THE IMPLEMENTATION RESULTS OF THE COMPACT FFT ARCHITECTURE

Length	2	3	4	5	7	8	9	16
Mults.	0	2	0	5	8	2	10	10
Adders	2	2	4	4	8	8	8	16
Regs.	67	102	131	205	424	271	533	597
LUTs	198	503	404	1008	2433	2304	4739	5703
DSP48s	0	4	0	10	16	4	20	20
Freq.	404	216	293	224	222	191	170	180
Cmp. L.	1	5	3	7	8	5	10	7
Ovr. L.	3	8	7	12	15	13	19	23

TABLE V
RESOURCE UTILIZATION OF THE GOOD-THOMAS FFT ARCHITECTURE USING COMPACT WINOGRAD SINGLE STAGE FFT ARCHITECTURE FOR 1008-POINT FFT

Design	Time-multiplexed FFT arch.	Folded FFT arch.
Multipliers	28	10
Adders	32	16
Registers	1635	681
LUTs	13054	15119
DSP48s	56	20
BRAMs	8	7
Clock Frequency (MHz)	182	150
Computational Latency	11017	7962
Overall Latency	12027	9978

determined by the maximum number of adders and multipliers required in different stages of various Winograd SFGs. The eight adders, eight subtractor, and ten multipliers used in the single-stage WFT architecture are not pipelined. *MA00* to *MA15* are 16 multiplexers for the adders' inputs, *MS00* to *MS15* are 16 multiplexers for the subtractors' inputs, and *MM0* to *MM9* are 10 multiplexers for the multipliers' inputs. Note that one of the inputs of each multiplier is a real value and all the multipliers' coefficients are stored in a LUT at initialization. The select signals for the multiplexers, collectively denoted as *ControlWord*, is a 268-bit word and is generated by the FSM controller. The first 178 bits of the *ControlWord* are connected to the select lines of the multiplexers, which control the inputs of the adders and multipliers. The remaining 90 bits are connected to the select lines of the input multiplexers of registers $R_0, \ldots, R_{N-1}$. Reusing the array of adders, subtractors, and multipliers will result in a significantly more compact realization compared to our high-throughput implementation. Table V gives the characteristics and the implementation results of the time-multiplexed FFT architecture utilizing an array of single-stage Winograd SFGs and the folded FFT architecture using a single-stage Winograd SFG for the realization of all small block lengths as shown in Fig. 8. Note that the number of computational resources such as adders and multipliers reported in Table V may not necessarily be the same as the theoretical number of numerical operations inherent to the transform, as given in Table II. For example, in some FPGA designs, a 32-bit multiplication may be implemented using two 18-bit DSP48 units. The reduced clock frequency is due to the fact that multipliers and adders are not pipelined and the wide multiplexers add to the critical path delay of the compact design. The shorter latency of the folded architecture is due the utilization of three-dimensional CRT mapping instead of using two-dimensional CRT mapping twice.

Table VI gives the characteristics and the implementation results of various published FFT implementations along with our implementation results. The designs in [9] and [10] present radix-2^2 single-path delay feedback (SDF) FFT architecture, which implements a FFT with the same multiplicative complexity of radix-4 FFT but maintains the simple structure of the radix-2 butterfly. The design in [10] also presents a radix-4 single-path delay commutator FFT architecture. Note that the designs in [9] and [10] implemented on spartan FPGA devices do not report the number of registers, look-up tables (LUTs), DSP48s, and block RAMs utilized. Memory-based radix-2 FFT architectures are presented in [11] and [12]. The implementation results of the intellectual property from Xilinx [13] are also presented. The design in [14] implements a 12-point to 1296-point FFT using a combination of high-radix FFTs such as 25- and 16-point FFTs using the Winograd Fourier transform algorithm. Unfortunately, they do not directly report the number of utilized registers, DSP48s, and signals' wordlengths in their manuscript.

We have also implemented various memory-based FFTs, including radix-2, radix-4, radix-8, mixed-radix, and split-radix algorithms. The designs perform computation via recursive use of radix-n (i.e. n = 2, 4, 8, etc) butterfly units. Once a stage has been computed, the outputs are passed to the following stage or the output ports, allowing the FFT to be performed on a new dataset. The twiddle factor coefficients are pre-computed and stored in block RAMs. One can see that the number of hardware resources required for the implementation of GT+WFTA FFT algorithm is generally greater. This is due to the irregularity of the SFGs of various WFTs, which makes it less suitable for compact VLSI implementation compared to the regular butterfly structures of radix-based FFT algorithms. Additionally, our designs use LUTs to store memory addresses, shown in Fig. 6 as *CRT LUT* and *RCM LUT*, as opposed to BRAM resources. Our design, however, works at a higher clock frequency and has higher throughput. The compact GT+WFTA design also reduces the number of register resources utilized. By using block RAMs for the storage of memory addresses, the LUT count can also be remedied. Note that the compact architecture is designed to reduce the number of computational units utilized for implementing the FFT at the expense of a longer latency and a lower throughput while the high-throughput architecture is designed to process a large number of FFT computations per second at a relatively small latency, which is required in various applications, such as in the LTE physical layer algorithms.

V. APPLICATION OF PROPOSED FFT ARCHITECTURE IN THE LTE PHYSICAL LAYER

In the time domain, different time intervals within LTE standard are expressed as multiples of a basic time unit $T_s = 1/30720000$ [18]. Fig. 7 shows the frame structure for LTE in frequency division duplex (FDD) mode. The radio frame has a length $T_{frame} = 307200 \times T_s = 10\, ms$. Each frame is divided

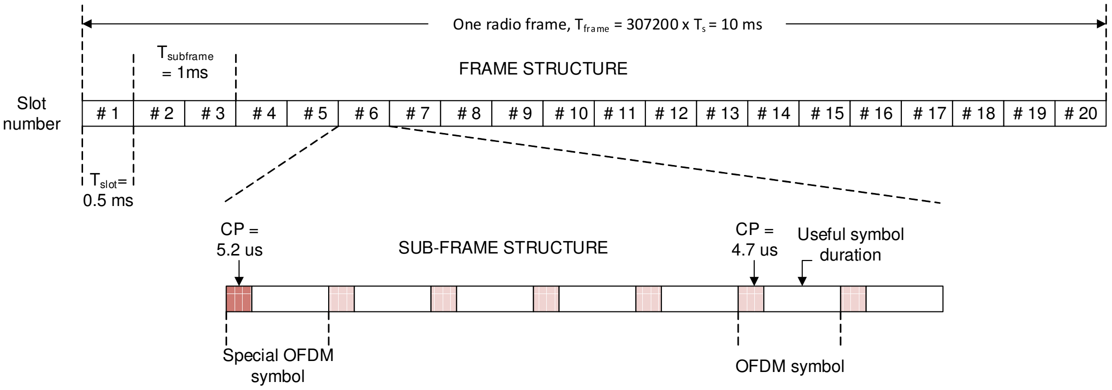


Fig. 7. LTE symbol and frame structure.

TABLE VI
CHARACTERISTICS AND THE IMPLEMENTATION RESULTS OF DIFFERENT FFT IMPLEMENTATIONS ON VARIOUS FPGAS

Design	FFT	N	W_L	Device	Regs.	LUTs	BRAMs	DSP48s	Clock (MHz)	Latency (Cycles)	Time μsec	Throughput MSamp/sec
[9]	Radix-2^2	1024	16	Spartan 3	−−	5916	−−	16	92	6086	65.89	–
[10]	R4SDC	1024	16	Spartan	3064	−−	8	−−	219	1041	4.67	219
[10]	R2^2SDF	1024	16	Virtex	2256	−−	8	−−	235	1042	4.35	235
[11]	Radix-2	1024	32	–	−−	27403	19	32	250	2850	11.4	90
[12]	Radix	var	−−	–	2372	4278	19	−−	200	–	–	200
[13]	Radix-2	1024	16	Virtex-6	4699	6298	17	16	366	12453	34.02	366
Ours	Radix-2	1024	16	Virtex-7	5786	5800	10	72	317	6663	21.01	317
Ours	Radix-4	1024	16	Virtex-7	9517	7247	5	96	317	6918	21.82	317
Ours	Radix-8	4096	16	Virtex-7	16033	14100	16	100	317	36870	116.31	317
[14]	Mixed-Radix 25/16/9	1296	−−	Virtex-5	−−	23807	−−	−−	122	1188	−−	122
Ours	Mixed-Radix 2/4/8	1024	16	Virtex-7	10126	8050	5	84	317	8200	25.86	317
Ours	Split-Radix	1024	16	Virtex-7	6184	6392	10	64	317	8198	25.86	317
Ours-HT	GT+WFTA	1008	16	Virtex-6	8513	14615	10	56	386	4100	25.4	386
Ours-Compact	GT+WFTA	1008	16	Virtex-6	681	15119	7	20	150	9978	66.52	0.015

into 10 equally sized sub-frames of $T_{subframe} = 30720 \times T_s = 1\ ms$ in length. Scheduling is done on a sub-frame basis for both the downlink and uplink. Each sub-frame consists of two equally sized slots of $T_{slot} = 15360 \times T_s = 0.5\ ms$ in length. Each slot in turn consists of a number of orthogonal frequency-division multiplexing (OFDM) symbols, which can be either seven (normal cyclic prefix) or six (extended cyclic prefix). The useful symbol time is $T_u = 2048 \times T_s \approx 66.7\ \mu s$. For the normal mode, the first symbol has a cyclic prefix (CP) of length $T_{CP} = 160 \times T_s \approx 5.2\ \mu s$. The remaining six symbols have a cyclic prefix of length $T_{CP} = 144 \times T_s \approx 4.7\ \mu s$. The reason that the first symbol has a different CP length is to make the overall slot length, in terms of time units, divisible by 15360. For the extended mode, the cyclic prefix is $T_{CP-e} = 512 \times T_s \approx 16.7\ \mu s$. The CP is longer than the typical delay spread of a few microseconds typically encountered in practice. The normal cyclic prefix is used in urban cells and high data rate applications, while the extended cyclic prefix is used in special cases, such as multi-cell broadcast and in very large cells.

Table VII summarizes some of the main physical layer parameters specified in the LTE standard, where the sub-carrier spacing Δf is assumed to be 15 kHz [19], the sampling rate

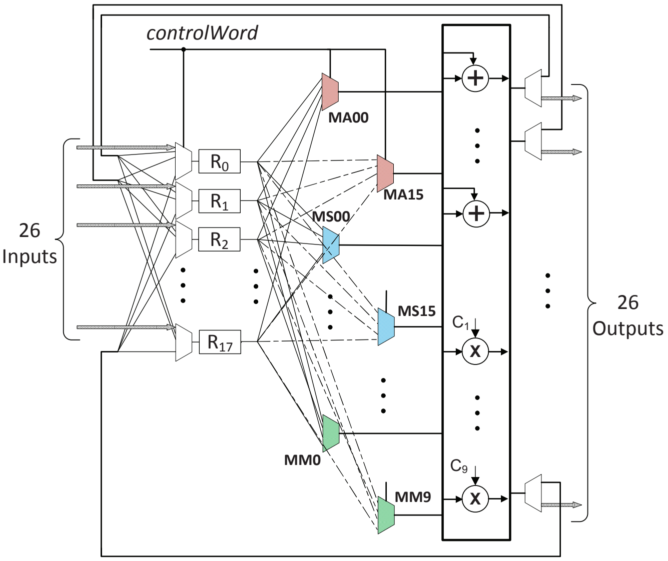


Fig. 8. Folded architecture for the implementation of various WFTs using a single stage of adders and multipliers.

is $f_s = \Delta f \times K$, where the number of sub-carriers K ranges from 128 to 2048, depending on the channel bandwidth with 512 and 1024 being most commonly used in practice for 5 and 10 MHz, respectively. It can be seen that the FFT lengths are chosen to be powers-of-2 as the Cooley-Tukey-type FFT algorithms have been widely utilized.

TABLE VII
PHYSICAL PARAMETERS FOR THE LTE STANDARD [19]

Channel BW. (MHz)	1.25	2.5	5	10	15	20
No. of Res. Blocks	6	12	25	50	75	100
Occup. Sub-carr.	76	151	301	601	901	1201
Min. Occup. Sub-carr.	72	144	300	600	900	1200
Guard Sub-carr.	52	105	211	423	635	847
FFT Length	128	256	512	1024	1536	2048
Sampling Freq. (MHz)	1.92	3.84	7.68	15.36	23.04	30.72

To use PFA, the sampling frequency should be adjusted based on the chosen FFT length. For a given channel bandwidth, the number of resource blocks and the minimum number of occupied sub-carriers are fixed. For instance, the 1.25 MHz channel will have 6 resource blocks and 72 occupied sub-carriers. Note that to avoid interference, the required guard sub-carriers should be at least 10% of the minimum number of occupied sub-carriers. Therefore, the required number of guard sub-carriers can be calculated for a chosen FFT length. For example, if we use a FFT of size 80 samples for a 1.25 MHz bandwidth, then the number of occupied sub-carriers is 72. Therefore, we can have 8 sub-carriers to give 80 sub-carriers, which is equal to the chosen FFT length. Table VIII gives the minimum, the possible, and the maximum FFT lengths using the Good-Thomas and Winograd FFT algorithms for sub-carrier spacing of 15 KHz and sub-frame duration of 6 *ms* over different channel bandwidths. One can see that for 10 MHz channel bandwidth, 720-point FFT can be used assuming 120 guard sub-carriers and the sampling frequency of 10.8 MHz or for 15 MHz channel bandwidth, 1008-point FFT can be utilized assuming 108 guard sub-carriers with the sampling frequency of 15.12 MHz.

TABLE VIII
THE MINIMUM, POSSIBLE, AND MAXIMUM FFT LENGTHS USING THE GOOD-THOMAS AND WINOGRAD FFT ALGORITHMS FOR VARIOUS LTE PARAMETERS

Channel BW. (MHz)	1.25	2.5	5	10	15	20
Occup. Sub-carr.	72	144	300	600	900	1200
Min. FFT lengths	80	160	330	660	990	1320
Guard Sub-carriers	8	16	30	60	90	120
Sampling Freq. (MHz)	1.2	2.4	4.95	9.9	14.85	19.8
Possible FFT lengths	90	180	360	720	1008	2520
Guard Sub-carriers	18	36	60	120	108	1320
Sampling Freq. (MHz)	1.35	2.70	5.4	10.8	15.12	37.8
Max. FFT length	120	240	504	1008	1680	2520
Guard Sub-carriers	48	96	204	408	780	1320
Sampling Freq. (MHz)	1.80	3.60	7.56	15.12	25.2	37.8

VI. CONCLUSIONS

The computation of the Fourier transform using the combined Good-Thomas or prime-factor algorithm (PFA) and Winograd Fourier transform algorithm (WFTA) proved to require fewer multiplications compared to Cooley-Tukey type algorithms, because it does not require twiddle factor multiplications and the use of the small Winograd FFT algorithms inside the Good-Thomas fast Fourier transform (FFT) algorithm significantly reduces the arithmetic complexity due to the cyclic convolution properties of the Winograd FFT algorithm. We presented a high-throughput and a compact architecture for efficient implementation of FFT based on the combination of PFA and WFTA. Fixed-point simulation results have been validated by the implementation of the proposed architectures on a field-programmable gate array (FPGA). Our future work will focus on extending both architectures to supports larger block lengths, as well as using more efficient memory address generation or storage schemes.

ACKNOWLEDGMENT

This work was supported by the Center for Sensorimotor Neural Engineering (CSNE), a National Science Foundation Engineering Research Center (EEC-1028725).

REFERENCES

[1] P. Duhamel and M. Vetterli, *Fast Fourier transforms: a tutorial review and a state of the art*. Elsevier, 1990, vol. 19, no. 4, pp. 259–299.

[2] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” *Mathematics of Computation*, vol. 19, no. 1, pp. 297–301, 1965.

[3] I. J. Good, “The interaction algorithm and practical Fourier analysis,” *Journal of the Royal Statistical Society.*, pp. 361–372, 1958.

[4] L. H. Thomas, “Using a computer to solve problems in physics,” *Application of Digital Computers*, no. 5, 1963.

[5] S. Winograd, “On computing the discrete Fourier transform,” *Mathematics of Computation*, vol. 2, no. 141, pp. 175–199, 1978.

[6] C. Rader, “Discrete Fourier transforms when the number of data samples is prime,” *Proceedings of the IEEE*, vol. 56, no. 6, pp. 1107–1108, 1968.

[7] C. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms: theory and Implementation*. John Wiley & Sons, Inc., 1991.

[8] P. Duhamel and H. Hollmann, “Split radix FFT algorithm,” *Electronics Letters*, vol. 20, no. 4, pp. 14–16, 1984.

[9] K. Harikrishna, T. R. Rao, and V. A. Labay, “FPGA implementation of FFT algorithm for IEEE 802.16e mobile wimax,” *International Journal of Computer Theory and Engineering*, vol. 3, no. 2, pp. 197–202, 2011.

[10] B. Zhou, Y. Peng, and D. Hwang, “Pipeline FFT architectures optimized for FPGAs,” *International Journal of Reconfigurable Computing*, pp. 1–9, 2009.

[11] Sundance Technologies, “`http://www.sundance.com/`,” 2015.

[12] V. Gautam, K. C. Ray, and P. Haddow, “Hardware efficient design of variable length FFT processor,” in *IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, 2011, pp. 309–312.

[13] Xilinx Incorporation, “`http://www.xilinx.com/`,” 2015.

[14] J. Chen, J. Hu, S. Lee, G. Sobelman, “Hardware efficient mixed radix-25/16/9 FFT for LTE systems,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 2, pp. 221 – 228, 2015.

[15] E. Chu, *Discrete and continuous Fourier transforms: analysis, applications and fast algorithms*. CRC Press, 2008.

[16] D. P. Kolba and T. W. Parks, “A prime factor FFT algorithm using high-speed convolution,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 25, no. 3, pp. 281–294, 1977.

[17] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New Yourk: John Wiley & Sons, 1999.

[18] J. Zyren and W. McCoy, “Overview of the 3GPP long term evolution physical layer,” *Freescale Semiconductor, Inc., white paper*, 2007.

[19] T. Innovations, “LTE in a nutshell,” *White paper*, p. 6, 2010.