Presented at the 2014 International Conference on Computing, Networking and Communications (ICNC), CNC Workshop (ICNC'14 - Workshops - CNC), Honolulu, HI, Feb 2014.
(www.centar.net)

High-Throughput Programmable Systolic Array FFT Architecture and FPGA Implementations

J. Greg Nash, Senior Member, IEEE
Centar LLC
Los Angeles, CA USA
jgregnash@centar.net

***Abstract*—A small, fine-grained systolic FFT architecture is described that is fast, programmable, can do non-power-of-two DFTs, and provides a higher signal-to-noise ratio for a given fixed-point word length than traditional block floating point approaches. To demonstrate the basic architecture, several designs were implemented using 65nm FPGA technology: (1) fixed-size 256-point and 1024-point circuits; (2) a power-of-two variable FFT circuit for LTE OFDM; and (3) a non-power-of-two circuit for LTE SC-FDMA DFT computations, that is programmed by entering parameter values into a single ROM memory. These three circuits demonstrate >37%, 62% and >100% higher throughputs than the other pipelined and memory-based FFTs to which they are compared. These circuits run at clocks speeds as high as 566 MHz, 46% higher than any other circuit in the comparisons. Finally, the architecture provides scalable throughput by increasing the array size.**

***Index Terms*—Fast Fourier transform, discrete Fourier transform, systolic array, FPGA, pipelined FFT, non-power-of-two.**

I. INTRODUCTION

The discrete Fourier transform (DFT) is one of the most prominent signal processing algorithms. In particular it is has been proposed for use in a variety of wireless transmission protocols, as for example WiMax, LTE and DVB-T2. The protocols are complex and place significant demands on the DFT circuitry due to the need for

- run-time choice of DFT sizes (scalable OFDM)
- larger transform sizes (LTE: 2K points; DVB: 32K points)
- high throughputs as a result of MIMO data streams and carrier aggregation (LTE Advanced 100MHz bandwidths)
- non-power-of-two DFTs (SC-FDMA: 35 transform sizes; digital radio mondiale: 288, 256, 176,112 points)

Since these applications are real-time, special purpose parallel circuitry, coupled with fast algorithms for computing the DFT, is essential.

Most high performance 1-D FFT designs fall into one of two categories: either delay feedback (DF) and delay commutator (DC) ([1], [2]) pipelined designs or serial memory-based designs [3]. Pipelined designs are computationally efficient and fast, but lack programmability and are typically intended only for power-of-two-transforms. Memory-based designs are more flexible, but considerable complexity is required to achieve high speed and they are not inherently scalable. "Systolic" array (SA) designs have also been proposed for wireless fixed-size DFT computations [4][5] and are attractive because of their simplicity, regularity, scalability, locality of interconnections and suitability for non-power-of-two transforms. However, they require substantial hardware, $\sim N$ real multipliers, where N is the transform size.

In contrast the "base-b" architecture described here is also a SA (memory-based) with the beneficial attributes listed above; however, it is far smaller in size than traditional SAs because hardware usage is comparable to that of traditional FFT implementations, with which it also shares computational efficiencies. Additionally, it is programmable, can do non-power-of-two DFT sizes, provides higher throughput implementations, supports cyclic prefix insertion and includes a low overhead hybrid floating-point feature that increases dynamic range for a given fixed-point word size.

The SA memory-based design is architecturally very different than traditional memory-based approaches as can be seen in Fig. 1. Here a traditional high-performance memory-based design (Fig. 1(a)) contains a plurality of large arithmetic units, where typically butterfly computations are performed, connected to a similar plurality of memories. The goal in such designs is to sequence data to/from the memories in such a way that data I/O rates are maximized. Alternatively, the SA does the same thing, but at a finer level of granularity. From Fig. 1(b) the SA is seen to consist of many very small processing elements (PEs), each containing multiplier/adders and a few registers. Each PE reads and writes to a typically small "simple" dual-port memory and aggregate bandwidth is limited only by the number of PEs. The well-known scalability of systolic algorithms [6] means that high bandwidths, and thus performance, are achieved by simply increasing the array size. It is much more difficult to do the same for traditional memory based designs.

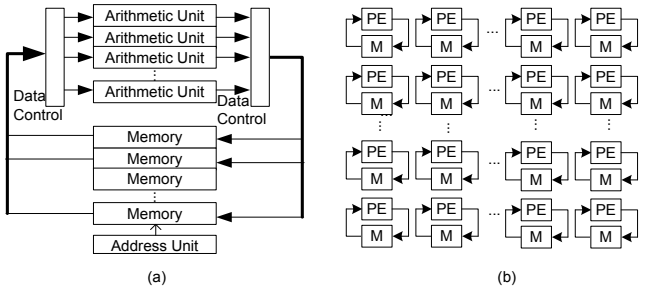

Fig. 1(a). Traditional memory-based FFT architecture and (b) fine-grained SA equivalent.

Although not shown in Fig. 1(b), each PE is locally connected to its neighbors which keeps interconnections short. This is particularly important as process technologies migrate to 22nm sizes and routing delays become an increasingly

This work was supported in part by the National Science Foundation under Grant IIP-0848285.

dominant part of the critical path. A good example of this trend are the Tabula ABAX[2] FPGAs with their new "Spacetime" architecture which adds a virtual vertical dimension to their LUT fabric. In this case finding localized circuit compilations will be essential to efficient use of such devices. The base-*b* architecture presented here was actually designed using a true space-time mapping tool that generates latency optimal solutions [7], so that implementations are fundamentally localized. Therefore, with good place-and-route compilers, high speeds are easily obtainable along with potentially lower power due to reduced interconnection usage. For example the power-of-two circuits presented here all run with worst-case clock speeds >500MHz, using 65nm FPGA technology, speeds which are higher than any other similar FPGA FFT circuits of which we are aware and approach the clock speed limits of the FPGA fabric.

Note that we have previously implemented a couple of base-4 concept demonstration, fixed-size FFT circuits in FPGA technology [8]; however, the circuit architecture described here is much improved. In particular, that in [8] (1) did not employ the simple PE-memory processing flow shown Fig. 1(b); (2) required an entirely separate processing step and support hardware to perform the twiddle operations (Fig. 1 in [8]), which has been eliminated here; (3) used inefficient PE memory management schemes, replaced here by "in-place" RAM read/write operations and agglomeration of output buffers, reducing the number of embedded PE RAMs by more than a factor of 2; (4) utilized some fixed scaling, whereas here there is no fixed scaling; and (5) was constrained to the fixed, non-programmable base-*b* processing described in [7], whereas here additional architectural support for programmability has been added.

The focus in this paper is on FPGA-based implementations, because of their rapidly growing use in semiconductor applications. Modern FPGAs have large numbers of embedded multipliers (as many as 868 on a single FPGA in the Altera Stratix III series) and embedded memories, which result in very different design tradeoffs compared to ASIC designs. Therefore, because FPGA chips are expensive, the main design objective here is to minimize the use of LUT logic and registers, while maximizing performance.

The algorithmic basis for the architecture is summarized in Section II-A, followed by a simplified architectural description in Section II-B. Section III introduces a very general implementation scheme, followed by several DFT implementation examples: fixed-size power-of-two circuits in Section IV-B; a variable power-of-two circuit in Section IV-C; and a non-power-of-two circuit in Section IV-D. (Each design was chosen with wireless applications in mind.) Section V concludes with a summary of base-*b* architectural features.

II. ARCHITECTURE

A. Algorithmic Basis

The overall architecture is based on two levels of factorization [7] of the transform size N starting with $N=N_3*N_4$.[1] Here the N_3xN_4 "DFT matrix" X contains input samples x_i that are arranged $x_1,x_2,...,x_{N4}$ on row 1, $x_{N4+1},x_{N4+2},...,x_{2*N4}$, on row 2, etc. Using the traditional "row/column" approach to compute the DFT, this factorization requires computation of two sets of smaller DFTs, N_4 transforms of length N_3 (referred to as "column" DFTs) and N_3 transforms of length N_4 (referred to as "row" DFTs). In between column and row transforms it is necessary to multiply each of the N points by the usual twiddle factor,

$$W_N^{i,k}=e^{-j(\frac{2\pi ik}{N})}\ i=0,1,..,N_3\text{-}1,\ k=0,1,..N_4\text{-}1.$$

A second level of factorization [7] can be used in computing an M-length row or column DFT based on the decomposition $M=N_1*N_2=N_1*b$, leading to

$$Y_b=W_M\cdot C_{M1}X_b \qquad (1)$$
$$Z_b=C_{M2}Y_b^t,$$

where "b" refers to the value of N_2 or "base". X_b is an input data matrix of bxN_1 points and Z_b is a DFT output matrix of bxN_1 points. Also, W_M is a small N_1 x N_1 coefficient matrix used in an element-by-element multiply and C_{M1}, C_{M2} are arrays of radix-b butterfly matrices.

B. Architecture

An abstraction of the architecture (Fig. 2) shows that it consists of two N_3/bxb processing element (PE) SAs connected by a N_3/bx1 array of complex multipliers ([7]). Each PE in the LHS and RHS SAs contains a few registers, multiplexors, a complex multiply/adder, and miscellaneous logic. Additionally, each PE in each SA array has access to a small simple dual-port RAM, to which RHS PEs write and LHS PEs read. The systolic matrix-matrix multiplications are carried out using well known mappings of signal flow graphs to PE arrays [6]. FFT input data in X_b are all fixed-point n-bit 2's complement words. The column and row DFT stages are described below based on (1).

Stage 1: Column DFTs

An input buffer feeds the LHS SA with columns X_{bci}, $i=1..N_4$, of the N_3xN_4 DFT matrix, and each X_{bci} is organized as a bxN_{1c} matrix ($M=N_3=b*N_{1c}$). Also, each PE in the LHS SA contains an element of the matrix C_{M1}. The systolic matrix-matrix multiplication result $C_{M1}X_{bci}$ then flows out of the LHS SA through the multiplier array shown in Fig. 2 where the coefficient multiplication by $W_M=W_{N3}$, stored in ROM, produces Y_{bci}^t. A second systolic matrix-matrix multiplication is then performed by the RHS SA with inputs Y_{bci}^t from the left and C_{M2} from the bottom (one C_{M2} matrix per column X_{bci}), producing the results Z_{bci}, which are stored in a distributed fashion in the RHS PE RAMs and become the X_{bri} for row DFTs after the twiddle multiplication by W_N.

Stage 2: Row DFTs

The processing in this step when based on (1) with $M=N_4=b*N_{1r}$ is identical to that for the column DFTs with two exceptions. First, there is a juxtaposition of the C_{M1} values with the X_{bri} row inputs. In this case the row X_{bri} values are

[1] Notation follows that in [7] to maintain consistency.

retrieved from the LHS SA PE internal RAMs, while the C_{M1} values now flow into the LHS SA from the bottom. Second, the Z_{bri} FFT outputs are stored in a separate RAM output buffer if normal order data output is desired.

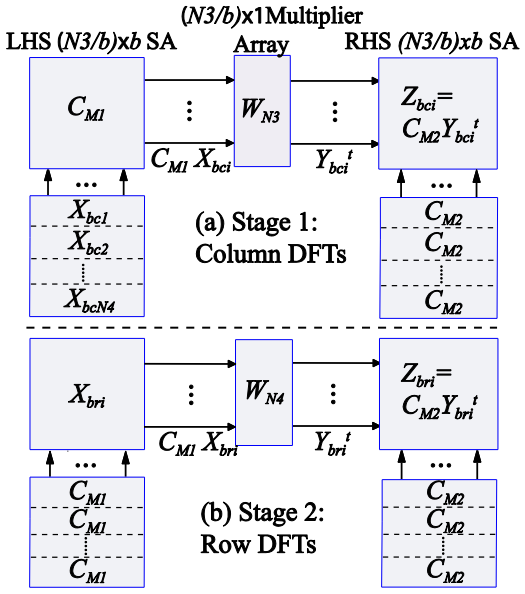

Fig. 2. Systolic processing flow for column and row DFTs. Subscript "*c*" and "*r*" refer to computing column and row DFTs of the N_3xN_4 DFT matrix.

III. Implementation

A. Introduction

The actual base-*b* circuit implementation is a direct mapping of the architecture shown in stage 1 of Fig. 2, i.e., the circuit contains the three array structures with distributed internal RAMs along with integrated input and output buffers if normal order data output is desired. However, the mapping of the row DFT operations in stage 2 of Fig. 2 onto this hardware is not direct and is described fully in Section V-D of [7].

A more detailed view of the circuit hardware is provided in Fig. 3, which shows the flow of X_b in one "PE row". A PE row contains one row each from the LHS and RHS SAs including the multiplier in between the two SAs. The particular PE row shown is the bottom row #1. Since the operations in all PE rows are identical it is only necessary to describe data flow in one of the N_3/b PE rows.

The input buffer contains addressing logic (not shown) that takes a continuous (normal order) serial stream of FFT input data, converts each FFT block into N_4 columns sets X_{bci}, and then distributes these column sets to the *b* RAMs labeled "B" so that each contains N_4*N_{1c} values prior to the start of FFT processing. (More input buffer detail is provided in [7].)

With reference to Fig. 3 the two processing stages are described again at another level of detail below.

Stage 1: Column DFTs

For this stage the "mux" in Fig. 3 supplying data to LHS PEs (Fig. 4) is set to select its input from registers labeled "R", which are there to permit systolic flow of X_{bci} from input buffer RAMs "B" up through the LHS array. The LHS PEs in row 1 use preloaded values of the first row of C_{M1} (Fig. 4) to perform vector-matrix products as a linear systolic array producing the first row of C_{M1} X_{bci}, i=1,2,.. N_4. Successively produced elements are multiplied by appropriate elements of W_M=W_{N3} (stored in a ROM) and become the first row of Y^t_{bci}, i=1,2,.. N_4. As the RHS PE row 1 receives elements of Y^t_{bci} it performs similar matrix-vector multiplications producing the first column of Z_{bci}, i=1,2,.. N_4. Here, the elements C_{M2} (not shown) flow up through registers in the RHS SA from a small ROM (not shown) at the array bottom edge.

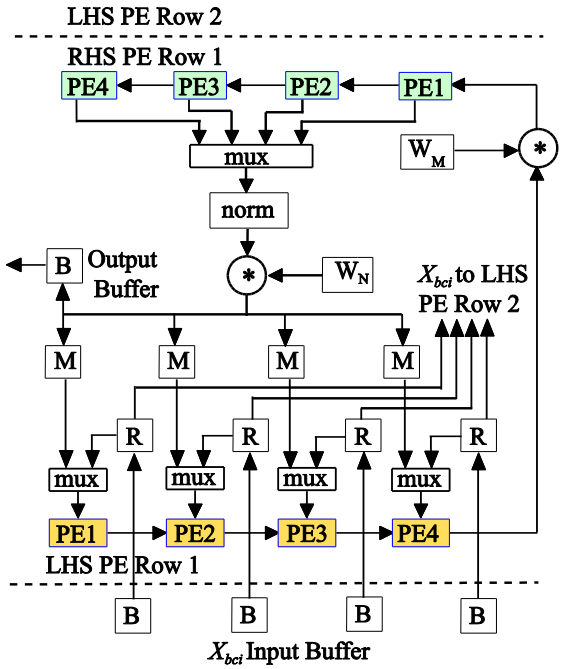

Fig. 3. Systolic processing flow for bottom PE row #1. To avoid clutter the C_{M1} and C_{M2} inputs and associated registers are not shown.

After each column element of Z_{bci} is computed in the RHS PEs (Fig. 4), a multiplexor ("mux" in Fig. 3) selects them and sends them sequentially to a normalizer ("norm" in Fig. 3). Here, the growth in the word length at that point from the original *n*-bit length is determined and a shift occurs so that the word is restored to *n*-bits with the shift amount saved as an exponent.

The final operation in the stage 1 column DFT systolic flow is the twiddle multiplication by values of W_N stored in a ROM. The output of the multiplier feeds a bus that connects the *b* PE internal memories "M". Each memory receives enable signals so that the elements of Z_{bci} flow to the correct memory.

Stage 2: Row DFTs

Conceptually, the processing during this step with M=N_4 based on (1) follows closely that for the column DFTs, with two minor differences. First, the LHS PE input multiplexors are set to take the X_{bri} inputs from the internal PE memories "M", so C_{M1} X_{bri} is computed using a systolic flow of C_{M1} values into the LHS SA from a ROM at bottom edge of the SA. Second, the Z_{bri} outputs (normalized *n*-bit values plus exponents) are stored in the output buffer "B" instead of the internal memories (twiddle multiplication by "1" here).

Control

Systolic data flow is naturally best served by a similar flow of control. Therefore for each clock cycle a control word is supplied by an external array controller to each column of PEs in both SAs and this flows upward in systolic registers providing values for addresses, read/write enables, etc.

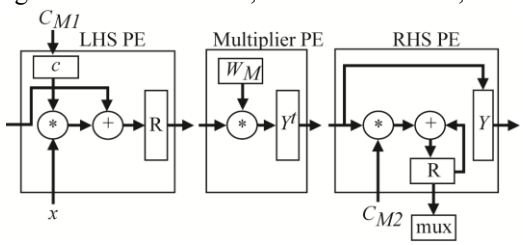

Fig. 4. PE structures depicted for the data flow of stage 1 in Fig. 2-3.

B. Dynamic Range

Signal-to-quantization-noise ratio (SQNR) performance is another important circuit characteristic, the goal being to maximize SQNR while minimizing fixed-point word lengths because longer words necessitate bigger memories and increase critical path lengths. A hybrid floating point (FP) alternative is used here that requires ~15% hardware overhead, an earlier version of which is described in more detail in [8]. It now involves no fixed scaling and has been modified here to be programmable. The associated control circuitry contains N_3 registers to keep track of the maximum exponent for each of the N_3 DFT matrix rows computed during stage 1 (Fig.2). These exponent values are added to results from the same row in the N_3xN_4 DFT matrix upon output. Comparisons show at least 4-bit (24db) improvement in SQNR over pure block floating point (BFP) approaches.

C. Programmability

The benefit of a programmable circuit is that the computational hardware can be highly optimized and then reused with different control circuitry to do different sets of computations. All that is necessary is that the DFT size be factorable into relatively small numbers. For example a 100-point transform could be factored as 4x(5x5). In this case it would be necessary to do 4 25-point transforms followed by 25 4-point transforms. Each 25-point transform would be most naturally done as 5 column transforms followed by 5 row transforms on the 5x5 matrix representing the 25-points. An array with b=5 could be used and a control circuit built that would be responsible for generating the addresses and coefficients supplied to the SA. For simple transforms, like the power-of-two examples (Section IV-B and C), finite state machines can be used. For more complex examples like the LTE example (Section IV-D), more general loop based RTL code could be used with loop parameter values specified in a small ROM memory.

D. Scaling

The base-b architecture throughput can be scaled upward in two ways: (1) by choosing N_3 to increase the linear length of the array [7] and (2) by simply replicating the array up to b times. Approach (1) is simpler but less efficient since the stage 1 processing time does not decrease, whereas for (2) throughput is increased by b. Both approaches can also be combined.

IV. Implementation Examples

A. Introduction

In this section three very different FPGA implementations of the base-b architecture are described to demonstrate its range of use. The first designs (Section IV-B) are fixed-size power-of-two FFTs that take computational advantage of the radix-4 butterfly matrix, in which case the LHS/RHS PE multipliers in Fig. 4 can be replaced by adders because $C_{M1/2}$ only contains $\{\pm 1, \pm j\}$. The second design (Section IV-C) is similar to the first, buts adds a requirement for run-time choice of FFT size. For these two design types the SA FFT is compared to Altera's commercially available FFT pipelined circuits (IP v10.1), since these have been optimized over 10 years and are representative of the fastest with the highest dynamic range of which we are aware.

Finally, for the third example (Section IV-D) the focus was on a more complex, non-power-of-two design, one that uses mixed-radices, offers run-time FFT choice and yet has the a very simple programming model.

B. 256-point and 1024-point "Streaming" (Normal Order In and Out) Fixed-size FFTs

For these two transform sizes the FFT circuits were compiled using Altera's software tools (Quartus II) with exactly the same settings (all synthesis options turned off) and the same target hardware (Stratix III EP3SE50F484C2 65 nm FPGA). The SQNR results for the Altera circuits were obtained from a bit-accurate Matlab model created along with the circuit by the Altera FFT generators. Altera's Timequest static timing analyzer was used to determine "worst case" maximum clock frequencies (Fmax) at 1.1V and 85°C.

Circuit comparisons shown in Table I are with Altera's mixed radix-4/2 20-bit pipelined FFTs that use BFP (single exponent per FFT block) to achieve a similar SQNR as the SA FFT circuits. Here the adaptive logic module (ALM) is the basic unit of a Stratix III FPGA (an 8-input "adaptive" LUT, two registers plus other logic). Fmax or clock rate is the same as the complex data sample rate. Table I shows that the SA FFTs use less logic and memory and still provide throughput rates that are 46%/37% higher for 256/1024-points.

TABLE I. COMPARISON OF ALTERA AND SA FIXED-SIZE FFT CIRCUITS.

	Altera 20-bits	SA FFT 16-bits	Altera 20-bits	SA FFT 16-bits
Transform Size	256	256	1024	1024
ALMs	4261	3982	4394	4331
Memory Bits (K)	49	40.6	195	145
Multipliers (18-bit)	24	33	24	33
Fmax (MHz)	387	566	382	524
SQNR	76.6	86.7	81.3	82.8
FFT time (μsec)	0.66	0.45	2.7	2.0

C. Variable FFT

Design Approach

In this section a variable streaming FFT circuit is described that provides run-time choice of 128/256/512/1024/2048

transform sizes as required for WiMax and LTE protocols. For this circuit with b=4 it was natural to choose a factorization with N_3=16 for all transform sizes. This choice leads to simple 4x4 SAs in Fig. 2-3 for the 16-point column transforms and processing using (1). Also all the desired twiddle factors W_N can be conveniently found among the elements of W_{2048}.

After the column DFTs, best efficiency is achieved by factoring the DFT differently for each transform size. The rows in Table II, "Stage 2a" and "Stage 2b", show how each DFT transform of a matrix row of length N_4 is performed in the SAs. The "4x2" means that an 8-point transform is done as a 4-point transform in the LHS SA followed by a 2-point transform in the RHS SA. The "4" means the LHS SA only does the 4-point transforms and "16" means processing follows (1) and is conceptually identical to the row DFT processing described in stage 2 of Fig. 2-3. For example to do a 2048-point transform, 128 16-point column DFTs are performed. Then for the row transforms, 16 8-point transforms are computed in stage 2a using both LHS and RHS SAs, followed by 8 16-point DFTs using (1) in stage 2b. (Both stage 2a and b use similar "row DFT" flow patterns.)

TABLE II. FACTORIZATIONS OF N_4 USED FOR FIVE TRANSFORM SIZES, N.

N	128	256	512	1024	2048
N_4	8	16	32	64	128
Stage 2a	4x2	16	4x2	4	4x2
Stage 2b	-	-	4	16	16

Circuit Comparisons

Table III, comparing variable FFT circuits (128-2048 points), was put together using the same design software and target hardware as the fixed point designs. Additionally, SA FFT circuit operation at 500MHz was demonstrated on an Altera Stratix III EP3SL150F1152C2 FPGA using a development board. Fmax (clock speed) is the same as the complex data sample rate.

TABLE III. COMPARISON OF VARIABLE SA AND ALTERA FFT CIRCUITS.

	SA FFT 16bits in/30 bits out	Altera 16bits in/out
ALMs/slices	4522	3826
RAM Memory (K)	290	208
Multipliers (18-bits)	33	36
Fmax (MHz)	510	315

As can be seen in Table III, the variable SA FFT circuit uses 18%/39% more ALMs/memory, but the relative throughput is a substantial 62% better. The Altera (non-programmable) single-path radix 2^2 delay-feedback (SDF) design is slower than Altera's fixed point FFT (Section IV B) because the 16-bit input word length grows to 30 bits at the output, so that the SQNR is as high as that for the SA FFT (~84db average). Processing these longer word lengths slows down the Altera SDF speed. Additionally, the SA FFT has built in support for the cyclic prefix generation and insertion needed for the targeted LTE/Wimax protocols, whereas the Altera circuit would require a completely separate 30-bit circuit to perform this function. Therefore, for the Altera circuit to support these protocols, more ALMs and memory would be required than indicated in Table III.

D. Non-power-of-two Circuit

1) Design Approach

Here a circuit capable of computing at run-time any one of the 35 different DFT sizes used in the LTE SC-FDMA uplink protocol (12 to 1296 points) is described. In this implementation the value b is chosen to be 6, so that SAs for b=5,4,3,2 are embedded as well. (The architecture is exactly the same as described for the power-of-two cases.) To make the circuit more directly comparable to other circuit implementations to which it is compared (Section IV-D-5), only a single linear PE row was used from the LHS and RHS SAs. In other words each LHS, RHS and multiplier PE columns in the array (Fig. 2) is projected (collapsed vertically) onto PE row #1 in Fig. 3. Thus, there are 6 RAM memories associated with LHS/RHS PEs and a single output buffer RAM. Also, a complex multiplier is used in each LHS/RHS PE (Fig. 4) because CM values are complex. This linear SA emulates the operations of the 2-D SA of Fig. 2, but completes a DFT computation a factor of b times slower.

2) On-the-fly-Twiddle Coefficient Calculation

The algorithmic discussion in Section II-A describes how a twiddle factor (W_N) computation must occur in between the column DFTs (stage 1) and the row DFTs (stage 2). For a single DFT, or small number of DFTs, the twiddle coefficients could be stored in a small ROM; however, for a large number of DFT sizes the ROM size needed could be prohibitively large and effectively limit the number of DFT sizes supported.

To address this twiddle storage issue we have adopted an approach based on a programmable on-the-fly-generation of the twiddle values for a particular DFT size. This twiddle coefficient engine uses a single complex multiplier, a table of twiddle seed values, and a set of size parameters as a basis for doing this. The iterative equation $W^{mn}=W^{(m-1)n}W^n$, where $W=e^{-2\pi i/N}$ and W^n is a seed value, is used to generate values for a particular DFT size given the starting seed.

For example in the implementation of the LTE SC-FDMA application the twiddle generation circuit required only 1024 words of memory to hold all twiddle seeds needed for 35 DFT sizes. The logic needed to implement the entire programmable twiddle circuit is only ~10% of the circuit hardware.

3) Example DFT Operation (N=540-point)

The circuit implemented, shown in Fig. 5a, consists of 6x6 (virtual) LHS and RHS SAs, so that any DFT of size $B\leq 6$ can be computed in either by multiplying the data vector by the coefficient matrix $C_B^{i,k}=e^{-j(\frac{2\pi ik}{B})}$, $i=0,1,..,B-1$; $k=0,1,..B-1$. The matrices C_{M1}, C_{M2} in (1) contain one or more arrays of these coefficient matrices C_B, e.g., $C_{M2}=[C_B|C_B|...]$ and $C_{M1}=[C_B^t|C_B^t|...]^t$ as shown in [7].

Consider the example N=540=N_3*N_4. Since our implementation consists of 6x6 SAs, it would be most efficient to choose the factorization N_3*N_4=36*15=6^2*(3*5) because this makes best use of all the hardware. In this case the processing consists of 15 36-point column DFTs followed by 36 15-point row DFTs. The input X_b is stored in the input

buffers in such a way that it is accessible as a sequence of blocks X_{bci}, i=1..15 of 6x6 column data. Then the 36-point column DFTs are done using (1) with $M=N_3=36=N_{1c}*N_2=N_{1c}*b=6*6$ and $C_{M1}=C_{M2}=C_B$ (B=6). Each X_{bci} enters the array at the bottom of the LHS SA (Fig. 2a and Fig.5a) and flows upward with systolic matrix-matrix multiplications performed as shown in Fig. 2a. As each of these 36-point column DFTs are computed, they are multiplied by elements in the 36x15 twiddle matrix W_{520} (Fig. 3) which are generated on-the-fly. During this processing stage all PEs are used with 100% efficiency. (As noted in Section IV-D-1, the implementation is a linear array that emulates this 2-D processing.)

After twiddle multiplication by W_{520}, the multiplexor in Fig. 3 is used to store data for the 15-point row DFTs in a way that they can be accessed as 3x5 data input blocks, X_{bri}, i=1..36, from the internal PE RAMs. In particular each of the 6 PE virtual rows is responsible for storing 6 3x5 blocks of DFT row data in associated internal RAMs. For the row DFTs not all the LHS/RHS PEs are used as shown in Fig. 5b. Rather, as shown in Fig. 5b, the LHS side SA reads from RAMs in five of the six PE columns to do the 5-point transforms by multiplication of these data blocks by C_5. The multiplier array then multiplies these transform values arriving from the LHS array by appropriate elements of a 3x5 twiddle matrix stored in a small ROM. Finally, only 3 PE columns are used on the RHS array to perform all the 3 point transforms. Results are stored in an output buffer and are output in normal order.

4) Programming

The base-6 circuit described here is "programmed" at a more abstract level than the base-4 circuits in Section IV B and C. It uses a single ROM or RAM memory to hold parameters that determine the specific factorizations and execution orderings used for loop index ranges in the verilog HDL coded control modules. Consequently, any transform size consistent with the circuit base b can be computed and the number of different DFT sizes that can be supported (including powers-of-2) is only limited by the size of this parameter memory.

5) Comparisons

To compute transforms for $N\neq 2^n$, a variety of mixed radix approaches have been used [10]-[16]) and systolic arrays (SAs) [4] have been proposed. However, of these only the memory-based approaches [12] [15] [16] provide high speed and are capable of run-time choice of different non-power-of-two DFTs. The speeds of the different designs are primarily related to the complexity of the butterfly unit design. For example [12] uses the mixed radix approach with radices {2,3,5} so that $N=2^n*3^m*5^p$, where n,m,p are integers, based on a single pipelined, parallel butterfly unit that can do radix-2,3,4 and 5 operations. Alternatively, [16] uses a maximum radix size of 16 to reduce computation cycle count.

Table IV is a technology independent comparison of these designs obtained from averages over the 35 different DFT LTE transform sizes (34 for the Altera [15] circuit). In addition to the usual latency (cycles) and throughput numbers (1/cycles), the average time in cycles to compute a single LTE resource block, assuming 7 symbols duration in an uplink slot, has been added. (This latter more practical measure combines times associated with both latency and throughput.) The SA FFT cycle counts are based on Modelsim gate level simulations.

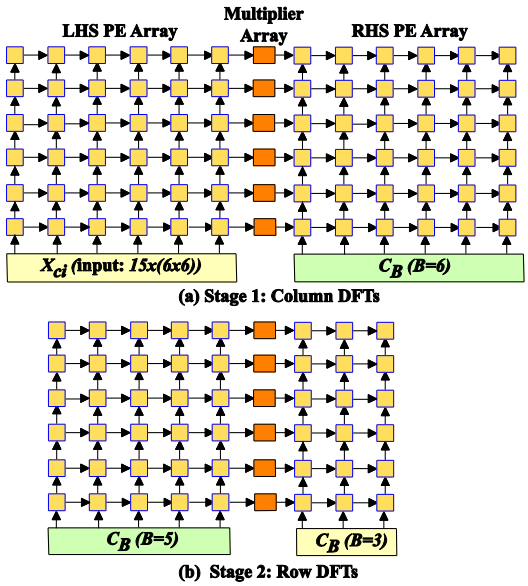

Fig. 5. Virtual 2-D arrays used for row and column DFTs, N=520.

TABLE IV. DFT PERFORMANCE BASED ON CYCLE COUNTS.

	Average Latency	Average Throughput	Resource Block Computation
Altera [15]	1.39	0.47	2.01
Xilinx [12]	0.86	0.65	1.50
SA FFT	1.00	1.00	1.00

For ease of comparison in Table IV the results have been normalized to the corresponding base-6 SA FFT value. As can been seen, the other designs are slower by ≥50% in the important resource block category.

The technology comparisons for all the FPGA LTE circuits of which we are aware are shown in Table V. For Virtex-5 FPGAs (also a 65nm technology) the basic logic unit is a logic element (LE) (one ALM≈1.5 LE's)[2]. The Virtex 18/36K BRAMs are counted as equivalent to 2/4 Stratix 9K RAMs. The SA FFT circuit uses a smaller word length, but has a much more efficient hybrid FP scaling scheme, providing 64 to 74db SQNR; the SQNR is not reported in the other designs. The SA FFT hardware entries came from Altera Quartus II design, synthesis and place-and-route tools with the Timequest static timing analyzer generating Fmax. Two Xilinx [12] designs, 8-bits and 18-bits, are used as these likely bracket the SA FFT SQNR values.

Table V is only intended to provide a rough guide to implementation costs because of the different FPGAs used, lack of reported SQNR values, and different functionality (the Altera design doesn't do all LTE sizes and doesn't provide normal order outputs which would require more logic/memory and increase latency). However, Table V shows the SA FFT

[2] Xilinx and Altera benchmark studies show 1 ALM=1.2 LEs (Xilinx white paper WP284 v1.0, December 19, 2007, and 1 ALM=1.8 LEs (Altera white paper WP-01007-2.1 Oct. 2007). Therefore, an average of 1.5 was used here.

implementation is likely to be more compact than [12][16] designs in terms of FPGA logic and registers. The Altera implementation uses less logic, but lacks complete functionality and is slower. The SA FFT clock speed is lower than design examples in Section IV B and C largely due to desire for enhanced programmability, e.g., the critical paths were in the control unit rather than the SA. It should be noted, however, that the SA LTE circuit was tested at 450 MHz on an Altera Stratix III EP3SL150F1152C2 FPGA using a development board. (Fmax in the tables here, at least for the Altera Quartus designs, represents a worst-case speed.)

If the clock speeds from Table V are combined with cycle counts in Table IV, a good measure of performance can be obtained as shown in Table VI. Here the average DFT computation times for a resource block are shown normalized to the SA FFT value and show speed-ups of >x2.

It wasn't possible to include [16] in Table VI because only DFT throughputs were reported there, not latency values. However, if only throughput rates are compared, the SA FFT provides rates that are ~35% faster on average based on the cycle counts reported in [16].

TABLE V. LTE CIRCUIT TECHNOLOGY COMPARISONS

Design	FPGA	Bits	Scaling	LUT	ALM /LE	RAM 9K eqv	Mult 18-bit	Fmax (MHz)
SA	Stratix III	12	BFP/FP	3582	2733	30	60	394
Xilinx [12]	Virtex-5	8	BFP	3447	2955	20	16	318
Xilinx [12]	Virtex-5	18	BFP	4707	3864	20	16	276
Altera [15]	Stratix III	18	BFP	2600	n.a.	17	32	260
Chen [16]	Virtex-5	18	Scaling	7791	n.a.	n.a.	44	123

TABLE VI. LTE CIRCUIT THROUGHPUT COMPARISONS

Design	Average LTE Resource Block Compute Time
SA FFT	1.0
Xilinx [12]	2.1
Altera [15]	3.0

Note that [15] and [16] rely on the use of the prime factor algorithm, which avoids the need for twiddle multiplications. The SA FFT design does not use the prime factor approach and therefore provides more flexibility in choice of DFT sizes. Also, the SA FFT is programmable compared to [12][15][16] as it relies simply on entering parameter values in a ROM or RAM for each desired transform size.

V. CONCLUSION

We have demonstrated a new class of FFT architectures that combines the simplicity, regularity and interconnection locality of SAs, with the speed of pipeline architectures and flexibility/programmability of memory based architectures. In particular the comparative benefits are:

- Improved throughput rates resulting from high clock speeds (>500MHz for 65nm FPGA technologies), made possible by the inherently localized circuit operation which reduces routing delays. For example the SA FFT compute times for LTE resource blocks are >x2 faster than the non-programmable circuits available from the two largest FPGA manufactures, Xilinx and Altera.
- Suitability for a wide range of power-of-two and non-power-of-two transform sizes.
- High dynamic range as a result of a combined block floating point and floating point scaling.
- Programmability.
- Throughput scalability due to the use of systolic algorithms.
- Built in cyclic prefix support for wireless protocols.

Finally, circuit performance and verification have been confirmed in FPGA hardware as well as simulations for both power-of-two and non-power-of-two classes of circuits. By providing both high performance and programmability, such circuits can be used to meet the demanding FFT requirements for future 4th generation (and beyond) wireless systems.

ACKNOWLEDGEMENT

We would like to acknowledge the help of Sarath Kallara for assistance in the design of the power-of-two circuits, in particular test benches, control circuitry, plus tools to enable parametric generation of circuits with different word lengths, and Wayne Fang for on-the-fly twiddle and address generation circuitry.

REFERENCES

[1] Gijun Yang and Yunho Jung, "Scalable FFT Processor for MIMO-OFDM Based SDR Systems", 2010 5th Int. Sym. on Wireless Pervasive Computing, pp. 517-521.

[2] Pei-Yun Tsai, Chia-Wei Chen, and Meng-Yuan Huang, "Automatic IP Generation of FFT/IFFT Processors with Word-Length Optimization for MIMO-OFDM Systems," EURASIP J. on Advances in Signal Processing, V. 2011, Article ID 136319, 15 pages.

[3] Chen-Fong Hsiao, Yuan Chen, and Chen-Yi Lee, "A Generalized Mixed-Radix Algorithm for Memory-Based FFT Processors", IEEE Trans. Circuits and Systems II: Express Briefs, V. 57, Issue 1, Jan. 2010, pp. 26-30.

[4] P. K. Meher, J. C. Patra, A. P. Vinod, "Efficient systolic designs for 1- and 2-D DFT of general transform-lengths for high-speed wireless communication applications", J.Sig. Process Sys.(2010), V.60, pp.1–14.

[5] H. Ho, V. Szwarc, and T. Kwasniewski, "A reconfigurable systolic array architecture for multicarrier wireless and multirate applications", Int. J. of reconfigurable computing, V. 2009, Article ID 529512, 14 pages.

[6] S. Y. Kung, VLSI Array Processors, Prentice Hall, 1988.

[7] J. Greg Nash, "Computationally efficient systolic architecture for computing the discrete Fourier transform", IEEE Trans Sig. Process., V. 53, Dec. 2005, pp. 4640-4651.

[8] J. G. Nash, "A new class of high performance FFTs", Proc. 2007 IEEE Conf. on Acoustics, Speech and Sig. Proc., pp. II-21 - II-24.

[9] S. Y. Kung, *VLSI Array Processors*, Prentice Hall, 1988.

[10] A. T. Jacobson, D. N. Truong, B. M. Baas, "The design of a reconfigurable continuous-flow mixed-radix FFT processor" Proc. 2009 IEEE Int. Symp. on Circuits and Systems, pp. 1133-1136.

[11] P.A. Milder, F. Franchetti, J.C. Hoe, M. Puschel, "Hardware implementation of the DFT with non-power-of-two problem size", Proc. 2010 IEEE Int.Conf. Acoustics Speech Sig. Proc., pp.1546-1549.

[12] Xilinx Discrete Fourier Transform v3.1, DS615 Mar. 1, 2011.

[13] Z.-X. Yang, et. al., "Design of a 3780-point IFFT processor for TDS-OFDM," IEEE Trans. Broadcast., vol. 48, pp.57–61, Mar. 2002.

[14] Shin-Chi Lai, et.al, "Low-computation-cycle, power-efficient, and reconfigurable design of recursive DFT for portable digital radio mondiale receiver," IEEE Trans. Circuits and Systems II, 2010, V. 57, pp. 647-651.

[15] Altera DFT/IDFT Reference Design, Application Note 464, May 2007.

[16] Jienan Chen, Jianhao Hu, and Shuyang Li, "High throughput and hardware efficient FFT architecture for LTE application", Proc. 2012 IEEE Wireless Communications and Networking Conf., pp. 826-831.