Presented at GSPx 2006, Santa Clara, California.

A High Performance Block Floating Point Systolic FFT Not Limited to Powers of Two

J. Greg Nash
Centar
2027 Linda Flora Dr., Los Angeles, CA 90077
(310) 765-4088
jgregnash@centar.net (www.centar.net)

Abstract:

A new high-performance systolic fast Fourier transform (FFT) circuit is described which supports transform lengths that aren't powers of two, provides low latency as well as high throughput and can do both 1-D and 2-D discrete Fourier transforms (DFTs). It is scalable so that any implementation can do any size FFT and it is inherently faster than most commercial pipelined FFTs because it uses fewer clock cylces per transform and runs at higher clock rates. Design, testing and maintainability are simplified because the architecture is based primarily on arrays of identical simple processing elements that contain a couple of registers and an adder. A design example is provided of 256-point streaming circuit that can run at a complex sample rate of 399MHz.

I. INTRODUCTION

The DFT is one of the prominent signal processing algorithms, being found in applications such as telecommunications, radar, beamforming, acoustics, seismic analysis, speech processing (spectrograms), medical signal processing, multi-media, and image processing [5],[6]. Many of these applications require "real-time" processing, so that special purpose parallel circuitry for computing the DFT is necessary. Here a high performance FFT circuit is described that offers the considerable functionality needed to meet the needs of such a wide variety of applications.

In Section II previous work related to systolic and parallel computation of the FFT is summarized. In Section III a new computationally simpler matrix based representation of the DFT is discussed and is mapped in Section IV into a block based systolic implementation of the 1-D DFT. Section V summarizes how the "row/column factorization" technique can be used to increase computationally efficiency and leads to the "base-4" architecture. Section VI provides a parameterized formulation of compuational performance. Section VII compares the base-4 design to other systolic and pipelined FFT implementations and Section VIII describes the design of a 256-point streaming FFT.

II. RELATED WORK

Past systolic array designs that have been proposed for computation of the DFT [11] typically offer very high performance in terms of throughput and transform sizes aren't limited to powers of 2. However, they are inherently inefficient and require substantial hardware. Approaches using linear arrays have been based on direct algorithm implementations so that the number of (complex) multiplies per DFT is $O(N^2)$, where N is the transform size, and the throughput is $O(N)$ arithmetic cycles per DFT [1]. A 2-D systolic array can improve efficiency when N can be expressed as the product of two cofactors so that a "row/column" DFT computation method can be used [12]. If this is done, the throughput becomes $O(n)$ for an N-point 1-D DFT where $N = n^2$, and the number of multiplies is reduced to $O(n^3)$. For both 1-D and 2-D systolic arrays the number of multipliers required is N, so that for a 1024 point transform a prohibitive number of multipliers (1024) would be necessary.

Most high performance 1-D FFT designs have appeared in the form of direct or modified implementations of decimation-in-time or frequency flow graphs with $O(\log_r N)$ stages of computation, where r is the radix, that use $O(N)$ delay stages that are used to match outputs and inputs of the different computational stages. These "pipelined" FFTs are very computationally efficient and make the effective use of hardware, particularly multipliers. However, these designs have disadvantages because for optimal designs often each butterfly, delay/commutator, and twiddle factor ROM has a different circuit design and/or its operation varies from stage to stage. Also, the multipliers do not always work at 100% efficiency, the designs are limited to transform lengths that are powers of 2 or 4, they are architecturally suited only for a 1-D DFT or 2-D DFT but not both, and it is difficult to build scalable designs because of their irregularity and larger granularity. Finally, the latency (number of clock cycles to do the first DFT in a series) is high because of the deep pipeline depths ($O(N)$) used. A good summary of these designs can be found in [9]. Some effort has been devoted to building systolic pipelined versions of these designs to improve circuit modularity and uniformity, many of which are summarized in [4].

The base-4 FFT design described here is intended to provide a performance level better than that of traditional pipelined FFTs, yet maintain the design/implementation simplicity and functionality of systolic arrays, e.g., the capapbility to perform non-power-of-two DFT comptuations. An additional motivation is that new FPGA

hardware changes previously established design tradeoffs. For example, recent FPGA chips are offered with large numbers of hardwired complex multipliers (>20) which consume less than 10% of the overall floor-plan area. This trend of embedding additional functionality in FPGA hardware will continue and consequently will require different tradeoffs in the way circuits are designed.

III. DIRECT FORM DFT DERIVATION

The base-4 architecture is derived from the direct form DFT representation

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

where *X(n)* are the time domain input values, *Z(k)* are the frequency domain outputs and $W_N = e^{-j(2\pi/N)}$. In matrix terms (1) can be represented as

$$Z = CX \qquad (2)$$

where C is a coefficient matrix containing elements $W_N^{nk} = e^{-j(2\pi nk/N)}$. If N can be factored as $N = N_1 N_2$, then using the reindexings $n = n_1 + N_1 n_2$ and $k = k_1 + N_1 k_2$ with $n_1 = 0,1,\cdots,N_1-1$, $k_1 = 0,1,\cdots,N_1-1$, $n_2 = 0,1,\cdots,N_2-1$, $k_2 = 0,1,\cdots,N_2-1$, it can be shown [15] that, if it is assumed that N_1/N_2 is an integer value, (1) becomes

$$\begin{aligned} Y_b &= W_M \bullet C_{M1}\ X_b \\ Z_b &= C_{M2}\ Y_b^t \end{aligned} \qquad (3)$$

where W_M is an $N_1 \text{ x } N_1$ matrix with elements $W_M[k_1,n_1]=W_N^{n_1 k_1}$, C_{M1} is an $N_1 \text{ x } N_2$ coefficient matrix with elements $C_{M1}[k_1,n_2]=W_{N_2}^{n_2 k_1}$, X_b is an $N_2 \text{ x } N_1$ matrix with elements $X_b[n_2,n_1]=X(n_1+N_1 n_2)$, Y_b is a $N_1 \text{ x } N_1$ matrix with elements $Y_b[k_1,n_1]=Y(k_1,n_1)$, C_{M2} is an $N_2 \text{ x } N_1$ coefficient matrix with elements $C_{M2}[k_2,n_1]=W_{N_2}^{n_1 k_2}$, and Z_b is an $N_2 \text{ x } N_1$ matrix containing the transform outputs $Z_b[k_2,k_1]=Z(k_1+k_2 N_1)$. and "$\bullet$" means element-by-element multiply. In (3) C_{M1} and C_{M2} contain N/N_2^2 sub-matrices $C_B = \left[c_1 \mid c_2 \mid ... \mid c_{N_2}\right]^t$ with the form $C_{M1} = \left[C_B^t \mid C_B^t \mid ...\right]^t$ and $C_{M2} = \left[C_B \mid C_B \mid ...\right]$ due to the periodicity of W_{N_2}, where c_i are constant vectors.

The "base" *b* for the architecture corresponds to the value of N_2 that is chosen for reindexed formulation (3). Here, we have chosen N_2=4 because it represents a good tradeoff between circuit performance and circuit complexity. This selection results in

$$c_1 = \begin{bmatrix}1\\1\\1\\1\end{bmatrix},\ c_2 = \begin{bmatrix}1\\-j\\-1\\j\end{bmatrix},\ c_3 = \begin{bmatrix}1\\-1\\1\\-1\end{bmatrix},\ c_4 = \begin{bmatrix}1\\j\\-1\\-j\end{bmatrix}$$

and

$$C_B = \begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -j & -1 & j\\ 1 & -1 & 1 & -1\\ 1 & j & -1 & -j\end{bmatrix}$$

where C_B above is the coefficient matrix for a 4-point DFT and also describes a radix-4 decimation in time butterfly.

The reindexed direct form expresssion (3) leads to several advantageous computational features, compared to previous systolic implementations, that are exploited in realizing the base-4 architecture described in the next sections:

1. Since it has been assumed that N_2=4 and that $N_1/N_2 = m$, where m is an integer, it follows that $N = N_1 N_2 = m N_2^2 = 16m$, e.g., transform sizes are only constrained to be integer multiples of 16.
2. In (3) Y_b and Z_b are both obtained from a series of simple 4- point transforms.
3. In comparing (3) with (2), significant computational advantages of the reindexed form (3) can be seen. In (3) the matrix products $C_{M1}X$ and $C_{M2}Y^t$ involve only addition/subtraction because the elements of C_{M1} and C_{M2} contain only ± 1 or $\pm j$, whereas the product *CX* in (2) requires complex multiplications.
4. The size of the coefficient matrix W_M in (3) is $(N/4)\times(N/4)$ vs. the $N \times N$ size of *C* in (2), leading to a reduction in the number of overall multiplications in by x16 compared to (2).
5. Systolic implementations that involve flows of coefficient data throughout the structure benefit because the elements, $C_{M1}[i,j]$ and $C_{M2}[i,j]$ do not impose significant bandwidth requirements (full complex numbers are not used).

IV. DIRECT FORM DFT ARCHITECTURE

This section describes direct form systolic designs for calculating the DFT based on the reindexed expression (3). Because there a large number of possible systolic architectures that can be obtained from (3), an automated CAD tool, Symbolic Parallel Algorithm Development Environment (SPADE), was used to make the choices. SPADE was developed to allow a designer to easily and rapidly explore the design space of various systolic algorithm implementations so that system tradeoffs can be efficiently analyzed. SPADE allows a user to specify his algorithm with traditional high-level code, set some architectural constraints and then view the results in a meaningful graphic form. More details on mapping and use of constraints within SPADE can be found in[13] [14].

The main constraint imposed using SPADE in making design choices was that the systolic array be linear or pseudo-linear (a 2-D array which is fixed in one dimension) because the target hardware was the new generation of FPGA devices which are constructed with linear arrays of multiplier and memory cells embedded in their logic fabric. This hardware organization also minimizes the necessity to keep all memory structures at the boundary of the array. With this basic constraint designs were looked for that maximize throughput.

SPADE found two unique throughput and latency optimal systolic arrays. These are shown in Fig. 1a and b and labeled according to the elements of (3). (The systolic array designs in Fig. 1 can be mathematically characterized via transformation matrices and data flow directions for variables mentioned earlier as described in more detail in [15]). Each of the two array designs has three basic components:

a) A *4* x *N/4* array of adder/subtractor processing elements (PEs) that perform the computation $C_{M1}\ X_b$ by systolic matrix-matrix multiplication.
b) A linear array of *N/4* multipliers that performs the element-by-element multiplies leading to $Y_b = W_M \bullet C_{M1}\ X_b$
c) A *4* x *N/4* array of adder/subtractor PEs that performs the computation $C_{M2}\ Y_b$, again by systolic matrix-matrix multiplication.

In each pseudo-linear design data flows from a) to b) to c). The main difference between the two designs is that for one (Fig. 1a) the input source is in a FIFO memory at the edge of the array and the transform result remains within the array after processing, whereas for the other design (Fig. 1b) the input source is internal to the array and the output is collected in a FIFO memory at the edge of the array. Each PE associated with a) and c) contains an adder/subtractor and nominally two registers, whereas each PE in component b) contains a complex multiplier and an amount of coefficient memory that depends on the transform size. Each design in Fig. 1 each has a latency $L = 2N/b + b + 4$ cycles and a throughput $T = N/b$ cycles per DFT, where a cycle is the time for each PE to perform either an addition or multiplication.

The architectures in Fig. 2 were chosen by SPADE because of their suitability to FPGA structures. For example, most of the PEs in Fig.1 consist of a couple of registers and an adder/subtractor. This type of element is part of the basic hardware fabric in FPGAs made by popular vendors such as Altera and Xilinx. Also, hardwired complex multipliers now embedded in the FPGA fabric result in multiply times that are comparable to adder times so that the cycle times in a) and c) vs b) are balanced. Finally, the localized computations and data movement of the arrays place minimal demands on valuable interconnect fabric, a critical issue in designing FPGA circuitry.

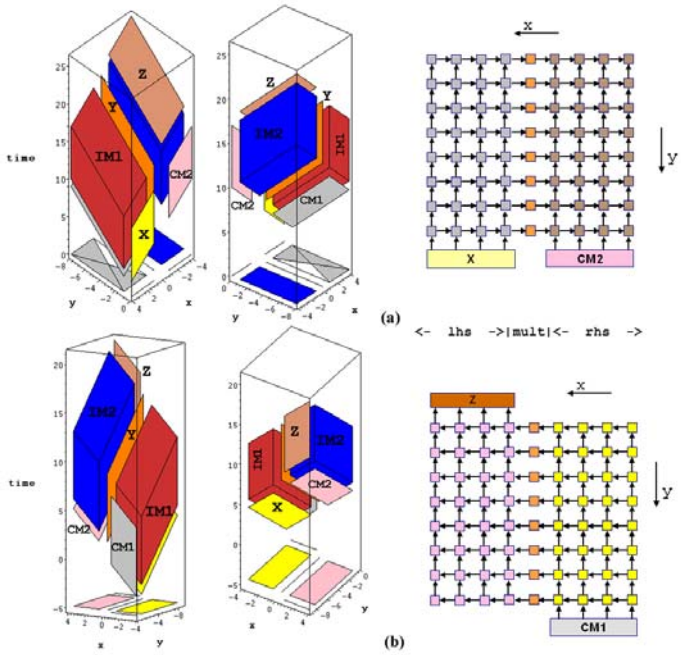

Figure 1. Two different space-time views of systolic designs, (a) and (b), and their corresponding PE arrays on the right (*N*=32). The labeling is consistent with (3). Here W_M has been mapped to the same locations as *Y* and is not shown. The "intermediate" variables *IM1* and *IM2* are created in SPADE to perform the running sums associated the matrix multiplication a) and c).

V. CIRCUIT DESIGN

The designs in Fig. 1a and 1b provide arithmetic and architectural efficiencies compared to other systolic arrays described in Section II; however, they still represent a direct DFT implementations because the transform time is proportional to *N*. To further increase performance while minimizing hardware penalties, a second level factorization, $N=N_3\ N_4$, is applied to the overall computation using the traditional row/column approach. This factorization requires computation of two sets of DFTs, N_3 transforms of length N_4 (stage 1) and N_4 transforms of length N_3 (stage 2). Each transform (of a row or column) in both stages is computed using the direct base-4 architecture described in the previous section.

Both the systolic designs from Fig. 1 are necessary for this factorization, that in Fig. 1a for stage 1 and that in Fig.1b for stage 2. In this way the stage 1 inputs are appropriately at the edges of the array, while the output of stage 1 matches the location of the inputs to stage 2. The main architectural addition is a problem size dependent amount of internal memory with which to store the stage 1 DFT results. Since the two array designs in Fig. 1 are very similar and use the same types of PEs, it is straightforward to just use one physical array in the circuit and have it mimic the operations of the arrays in Fig. 1a and 1b [15].

Since the computation time (Section III) of either a row or column is *16m*, where *m* is an integer, the computation time associated with the factorization $N=N_3\ N_4$ will be *N=n256*, where *n* is an integer. That is, this factorization

imposes the restriction that transform lengths N must be a multiple of 256.

In between stage 1 and stage 2 each of the N points is multiplied by an appropriate twiddle factor, $W_N^{i,k}$, i=1..N_3, k=1..N_4 in the multiplier PEs of Fig. 2. FPGA hardwired multipliers are pipelined and it will be assumed that a four stage pipeline is used, matching the four pipeline stages of the rhs array PEs. In this way every four cycles a new set of four twiddle updated coefficients are written while a new set of coefficients to be updated are read (assuming memories in the rhs array are dual ported). The total computation time for this twiddle step using the N_3/b multiplier PEs is then $N_3\,N_4/(N_3/\mathrm{b})+\mathrm{b} = \mathrm{b}(N_4+1)$ cycles.

VI. COMPUTATIONAL PERFORMANCE

A. Throughput

A throughput estimate can be determined from the computation time of the three basic operations: column DFTs, twiddle multiplication, row DFTs. There is also a time delay associated with the switch from solution of Fig. 1a to Fig. 1b. Since the combined processing requires N_3 column DFTs of length N_4 and N_4 row DFTS of length N_3, the overall throughput *Thrpt* in cycles per transform is

$$Thrpt = \underbrace{N_4(N_3/4)}_{column\ DFTs} + \underbrace{4(N_4+1)}_{twiddle\ \ multiplication} + \underbrace{N_4^2/4}_{row\ DFTs} + row/col\ delay$$

where the number of cycles $b(N_4+1)$ for twiddle multiplication from Section V has been inserted. The row DFT cycle count has a different form than the column DFT count because the data movement has been rearranged to accommodate the case where N_3 and N_4 are not equal and the two logical array sizes in Fig. 1a and 1b are not the same [15]. The delay in switching between column and row processing is twice the time to traverse the East-West direction of the array or $6b = 24$. Therefore, the approximage throughput becomes

$$Thrpt = N/4 + N_4^2/4 + 4N_4 + 28\ \ (cycles/DFT)\ .$$

The throughput for a variety of FFT calculations using this formula are shown in Table 1, along with the corresponding amount of hardware used.

Points	N3	N4	Throughput (cycles/DFT)	Latency (cycles/DFT)	Mults	Adders
512	32	16	284	292	8	64
1024	32	32	668	676	8	64
2048	64	32	924	940	16	128
2816	176	16	860	904	44	352
8192	128	64	3356	3388	32	256

Table 1. Estimated nominal throughput and hardware used in base-4 FFT.

B. Latency

The latency L is the time it takes to do the first FFT in a sequence FFTs. Consequently, it is obtained by adding to the throughput the number of cycles necessary to "fill" the pipeline. From Fig. 1a and 1b it can be seen that the maximum length data path in the array is the time to travel the length (of the array ($N_3/4$ cycles), so that

$$L = N_3/4 + Thrpt\ .$$

VII. DESIGN COMPARISONS

A. Systolic Arrays

As described in Section II, a variety of systolic array designs, characterized by use of uniform arrays of fine-grained PEs, have been proposed for computation of the DFT. The most common are of two basic types, linear systolic arrays and those based on use of a row/column factorization of N similar to that described in Section V. A comparison of the design parameters presented in the first three lines of Table 2 show that the base-4 FFT offers the following advantages:

1. *Smaller granularity*: Because PE size is defined primarily by additions rather than multiplier-adds, as is the case for most previous systolic designs, higher speeds are possible because the shorter wiring paths between PEs leads to lower gate delays and the arithmetic is simpler.
2. *Reduction in hardware complexity*: Table 2 shows that most past systolic architectures require one multiplier-adder combination per FFT point. Therefore, a 1024-point transform would require 1024 multipliers vs. only 8 for the base-4 FFT design (Table 1).
3. *Computational efficiency*: This feature derives from two different levels factorization. Because the designs in Table 3 are so structurally different, it useful to compare them based on an area-time product. In each case the PE structures are simple and tend to be dominated by arithmetic units, so an estimate of area can be made by using the number of multipliers and adders as a proxy for area. In order to determine the computing area an assumption was made that the area of an adder was ~¼ that of a multiplier. This is a good approximation for FPGA hardware; for example the floor-plan of the Altera Stratix™ FPGA assigns four reconfigurable logic array blocks (LABs) to a 20-bit complex adder, whereas the 18-bit hardwired complex multiplier consumes approximately the space of 16 LABs. For the measure of time throughput is the most useful for general signal processing. From Table 2, the base-4 FFT area-time product is $\sim 3N\sqrt{N}/8 + 3N + 21\sqrt{N}$ vs. N^2 and $5N\sqrt{N}/2 + 5N/4$ for the 1-D direct and 2-D FFT. Therefore, the base-4 design improves the area-throughput product by factors of approximately 4,5 and 6 for 256, 1024 and 4096 point transforms compared to the 2-D systolic FFT, and more for the direct methods.
4. *Latency and Throughput*: Most previous systolic designs are based on a computation performed by one pass through a single architecture. For large transforms this implies a large pipeline that takes significant time to fill

and empty, as reflected in Table 2. Alternatively, the base-4 FFT design is based on many passes through a smaller architecture, hence the fill/empty times are small.

There have been a variety of more course-grained systolic FFT implementations that are based on FFT flow graphs which permit a high degree of stage-to-stage regularity, two of which are shown in Table 2. Consequently, these have characteristics that are similar to other pipelined FFTs.

B. Pipelined FFTs

Comparison of the proposed base-4 FFT design to the many pipelined FFT designs that have been proposed is more difficult because they are typically comprised of far fewer, more course-grained modules, each with a larger number of different components and there are a large number of architectural variations. However, a few of the more recent designs that have been implemented in hardware are provided in Table 2. In general the base-4 design offers higher throughput and lower latency, but uses more hardware.

Compared to most commercially available pipelined FFTs which have *Thrpt=N* and *L=2*N*, the base-4 architecture possesses the following advantages:

- *It can compute the DFT for any N-point sequence divisible by 256.* In contrast traditional FFT circuits require *N* to be a power of two, which limits the number of reachable values of *N* and their spacing uniformity.
- *Any FFT implementation can do any FFT size*. This capability makes it possible to match circuit architecture to required application throughputs.
- *It uses fewer clock cycles per transform.* The number of clock cycles per DFT is $\sim N/2 + 4\sqrt{N} + 40$ compared to *N* for a typical pipeline FFT.
- *Higher clock rates are possible* because
  - the circuit consists of a regular array of locally connected small processing elements, each PE containing only a few registers, an adder and memory.
  - the circuit architecture matches that of FPGAs with their embedded linear arrays of hardwired multipliers and memory.
  - there are only three global lines (2 clocks and a global clear)
  - a large number of smaller, faster and more power efficient memories are used
- *It offers better dynamic range (DR) and signal-to-noise (S/N) due to an enhanced BFP circuitry*. This feature is a result of having many different block floating point circuit regions, i.e., each region has its own exponent.
- *It provides low latency as well as high throughput.* Computational latency in cycles/DFT is only slightly higher than the throughput measure.
- *The design is scalable and reconfigurable*. Larger FFTs are obtained by replicating identical blocks of PEs
- *It can do both 1-D and 2-D DFTs.*
- *Design, testing and maintainability are simplified* because the circuit is based on only two small PE types.

VIII. 256 POINT FFT EXAMPLE

To demonstrate this base-4 architecture an FFT design that accepts a continuous input stream *X(n)*, while generating a continuous output stream *Z(n)* at the same rate ("streaming") was chosen since this mode is common to many signal processing applications. To add application generality a BFP capability was added and a word size of 16-bits was chosen since this a common choice. The design has circuit pins for real and imaginary inputs/outputs, *Z/X*, a single global reset, and two clocks. Because the base-4 design computes FFTs using a number of clock cycles that is less than the transform size, a separate higher speed clock is used to read out the data.

An evaluation of the base-4 design is best done by comparison with another start-of-art pipelined FFT design built to the same specifications, using exactly the same underlying hardware. For this purpose a streaming BFP 256-point FFT design from Altera was chosen (FFT v2.2.0), since their pipelined streaming BFP FFT circuits are the fastest of which we are aware. Both designs were targeted to an Altera Stratix II EP2S15F484C3 FPGA. Altera Quartus II tools (v5.1) were used to design and evaluate the two FFT circuits. The base-4 circuit operation was verified by comparing the Quartus simulator result with a Centar bit-accurate simulation model. The Quartus II timing analyzer finds the critical path that determines the maximum clock frequencies.

Because the base-4 design provides higher dynamic range for a given bit-length, a 20-bit Altera FFT was used in the comparison. For example, a set of 62 256-point transforms on 16-bit "single tone" data (random phase and random frequency with no noise added) show that for the base-4 circuit the mean signal-to-noise ratio was 89.0 db and the dynamic range was 96.3db. This compares to 86.7db and 98.2db obtained on the same data using an equivalent Altera 20-bit streaming block floating point circuit. The results for the Altera circuit were obtained from a bit-accurate Matlab model that's created by the Altera FFT generator. There was no noise added to the single frequency inputs, so the "noise" represents only internally generated round-off noise.

For the Altera design the maximum complex sample rate was 302 MHz vs. 399 MHz for the base-4 circuit. The corresponding transform times are 0.85μsec vs 0.64μsec.

This comparison shows that for a 256-point transform the base-4 architecture can provide a better throughput than traditional pipelined FFTs for applications that require a high dynamic range. Similar comparisons can be expected for other FFT sizes. For example a base-4 1024-point circuit uses only ~680 clock cycles per DFT vs. 1024 for pipelined FFTs. Because the base-4 structure is local and regular, similar clock speeds should be expected, yielding a complex sample rate of ~542 MHz. Alternatively, timing

analyses of even a smaller 16-bit Altera 1024-pt FFT indicate the complex sample rate would only be ~325MHz.

IX. SUMMARY

The base-4 FFT design is intended to strike a balance between the flexibility of direct systolic designs and the computational efficiency of a pipelined designs, so that fast, but regular and scalable implementations are possible for performing both 1-D and 2-D DFTs. It should be noted that other base-4 FFT designs are possible and these depend on the application environment and target hardware. For example, modifications are possible to reduce power dissipation and/or reduce area by eliminating redundant computations and result in different tradeoffs. Here, the systolic designs proposed are biased to favor implementations that take advantage of the architectural richness of new generations of target FPGA hardware and applications that require very high performance. High performance is also aided by the regularity of the fine-grained, locally connected base-4 design.

Architecture	Multipliers	Adders	Data Memory	Throughput	Latency	Limits on N
1-D Systolic	N	N	$2N$	N	$2N-1$	none
2-D Systolic	N	N	$3N$	$2\sqrt{N}+1$	$4\sqrt{N}-1$	$N = N_3 N_4$
Base-4 FFT	$\sqrt{N}/4$	$2\sqrt{N}$	$1.75N + 3\sqrt{N}$	$N/2 + 4\sqrt{N} + 28$	$N/2 + 17\sqrt{N}/4 + 40$	$256n$
Systolic Pipelined[8]	$\log_2 N$	$2\log_2 N$	$N \log_2 N$	N	$N(1+\log_2 N/2)$	$N = 2^n$
Systolic Pipelined[4]	$3\log_2 N$	$3\log_2 N$	$5\log_2 N + 2N + 2$	N	$2N + \log_2 N$	$N = 2^n$
Pipelined FFT [9][10]	$\log_4 N$	$4\log_4 N$	$N-1$	N	$2N - 2 + 3(\log_4 N - 1)$	$N = 2^n$
Pipelined FFT[2][3]	$\log_4 N - 1$	$3\log_4 N$	$2N-2$	N	$2.2\,N$	$N = 4^n$
Pipelined FFT[7]	$\log_2 N$	$2\log_2 N$	$2N$	$N/2$	$2\,N$	$N = 2^n$

Table 3: Parameterized FFT comparisons assuming $N_3 = N_4 = \sqrt{N}$ for the 2-D systolic and base-4 FFTs. Here *n* is an integer greater than one. Latency is defined as the number of clock cycles to produce the first DFT in a sequence of DFTs.

X. References

[1] Beraldin, J. A., Aboulnasr, T., and Steenaart, W., "Efficient 1-D Systolic Array Realization for the Discrete Fourier Transform," IEEE Trans. Circuits and Systems, Vol. 36, pp. 95-100, 1989.

[2] Bi, G., and Jones, E.V., "A Pipelined FFT Processor for Word-sequential Data," IEEE Trans. Acoustics, Speech and Signal Processing, 37, Vol. 12, pp.1982-1985, Dec. 1989.

[3] Bidet, E., Castelain, D., Joanblanq, C., and Senn, P., "A Fast Single-Chip Implementation of 8192 Complex Point FFT," IEEE J. Solid-State Circuits, Vol. 30, pp. 300-305, (1995).

[4] Boriakoff, V., "FFT Computation with Systolic Arrays, A New Architecture", IEEE Trans. Circuits Syste. II, Vol. 41, pp.278-284, Apr. 1994.

[5] Bracewell, Ronald Newbold, The Fourier Transform & Its Applications, 3rd Edition, McGraw-Hill, 1999.

[6] Brigham, E. Oran, Fast Fourier Transform and Its Applications, 1st Edition, Prentice-Hall, 1977.

[7] Currie, Steven M. et. al., "Implementation of a Single Chip, Pipelined, Complex, One-Dimensional Fast Fourier Transform in 0.25 um Bulk CMOS," Proc. Int. Conf. on Application Specific Systems, (ASAP), 2002.

[8] Choi, J. and Boriakoff, V. , "A New Linear Systolic Array for FFT Computation," IEEE Trans. Circuits Syst. II, vol. 39, pp. 236–239, Apr. 1992.

[9] He, S. and Torkelson, M., "A New Approach to Pipeline FFT Processors," Proc. Int. Conf. Parallel Processing (IPPS 96), pp. 766-770.

[10] He., S., and Torkelson, M., "Design and Implementation of a 1024-point Pipeline FFT Processor", IEEE custom Integrated Circuits Conf., pp131-134, 1998.

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

[12] Lim, H. and Swartzlander, E. E., "Multidimensional Systolic Arrays for the Implementation of Discrete Fourier Transforms", IEEE Trans. Sig. Proc., Vol.47, No. 5, pp.1359-1370.

[13] Nash, J. Greg , "Automatic Generation of Systolic Array Designs For Reconfigurable Computing", 2002 Int. Conf. Engineering of Reconfigurable Systems and Algorithms (ERSA 02), June 24, Las Vegas, NV. (www.centar.net)

[14] Nash, J. Greg, "Constraint Directed CAD Tool For Automatic Latency-Optimal Implementation of 1-D and 2-D Fourier Transforms", Proc. SPIE ITCom 2002, Vol. 4867.

[15] Nash, J.G., "Computationally Efficient Systolic Architecture for Computing the Discrete Fourier Transform", IEEE Trans. Signal Processing, Vol. 53, pp. 4640-4651, Dec. 2005.