A NEW CLASS OF HIGH PERFORMANCE FFTS

J. Greg Nash

Centar (jgregnash@centar.net)

ABSTRACT

FPGA implementations of block floating point (BFP), streaming, 256-point and 1024-point fast Fourier transform (FFT) circuits are described as examples of a new architectural approach that provides better performance, flexibility, and functionality than commercially available pipelined FFTs. It is based on a matrix formulation of the discreet Fourier transform (DFT) that converts the direct transform into structured sets of arithmetically simple 4-point transforms that are computed on a systolic array. This circuit architecture permits transform lengths that are not a power-of-two, can do 2-D as well as 1-D transforms, is scalable, has low computational latency and utilizes BFP and floating point (FP) features to provide high dynamic range. Circuit comparisons are made with a commercially available pipelined FFT.

Index Terms— Discrete Fourier transforms, High-speed electronics, Field programmable gate arrays, Systolic arrays

1. INTRODUCTION

The DFT appears prominently throughout a large number of signal processing, communications, radar, acoustics, and electromagnetic applications [1]. Consequently, there is a considerable literature related to techniques for rapid computation of the DFT and these have been effectively exploited in new digital signal processing programmable chips such as the Texas Instruments TMS320 series. However, for high throughput applications it is still difficult to find parallel circuit implementations that combine functionality, flexibility and speed.

At the same time proposed and future systems will need considerably more implementation options for computing a DFT. For example, the recently announced Chinese DMB-T (Digital Media Broadcasting Terrestrial) transmission standard is based orthogonal frequency division multiplexing and uses a number of sub-carriers that is not a power of two [2]. The custom FFT circuit that was developed provided better overall system performance even though it was slower and less efficient in terms of memory usage. Also future FFT implementations will need to offer scalability (orthogonal frequency division multiple access), high dynamic range (spectrum analysis, radar), and 2-D processing capabilities (image processing). The "base-4" design proposed here addresses these needs.

This paper is organized as follows: Section 2 describes related work, Section 3 summarizes a new matrix based DFT formulation, Section 4 discusses the overall circuit architecture, Section 5 covers circuit implementation features, Section 6 compares base-4 designs to a commercial implementation, and finally Section 7 provides concluding remarks.

2. RELATED WORK

The most flexible parallel circuit approaches to calculating the DFT for arbitrary transform lengths, N, are based the on "direct" computational methods. A number of systolic approaches have been proposed to do this [3]; however, they are inherently inefficient and require substantial hardware [3]. For both 1-D and 2-D systolic arrays the number of (complex) multipliers required is N, so that for a 1024-point transform a prohibitive number of multipliers (1024) would be necessary. Alternatively, the base-4 approach strikes a balance between computational efficiency and limitations on reachable values of N in that it can calculate the DFT for any transform size that is a multiple of 256.

There are a few pipelined FFT implementations that permit multiple transform sizes (power-of-two) to be calculated [4][5]. Typically this is done by picking off smaller transform results earlier in the pipeline. However, in each case the choices are limited to a maximum transform size. In contrast the base-4 architecture can do any transform size on any implementation as long as the required amount of memory is available. This is possible because the architecture is essentially a matrix multiplication circuit and that considerably simplifies the partitioning issues.

In order to achieve high dynamic range in pipelined FFT circuits the better approaches are based on convergent BFP [6] or dynamic scaling [7] methodologies. In each of these cases the word length was reduced by four bits compared to conventional scaling approaches. The base-4 design also uses a form of dynamic scaling and achieves a four-bit improvement, yet achieves this goal with a simpler design and less overhead.

3. MATRIX EQUATION FOR THE DFT

The algorithm described here makes use of two levels of factorization. The first is the well-known row/column factorization, $N= N_r\, N_c$, where N is the desired transform length and N_c and N_r are the number of columns/rows [3]. This approach requires calculation of two sets of DFTs, N_c transforms of length N_r (referred to as "column" transforms) and N_r transforms of length N_c (referred to as "row" transforms). In between column and row transforms it is necessary to multiply each of the N points by a corresponding twiddle factor, $W_N^{n,k}$, n=0,1,..,N_c-1, k=0,1,..N_r-1. (Without the twiddle multiplication a 2-D DFT is performed.)

The second level of factorization is applied separately to each row or column DFT. The index remapping for each column or row DFT starts with the DFT defined as

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

where M is the row or column transform length, $X(n)$ are the time domain input values, $Z(k)$ are the frequency domain outputs and $W_M = e^{-j(2\pi/M)}$. If M can be factored as $M = bN_1$ (b is the base, equal to 4 here) with N_1 divisible by 4, then using the reindexings $n = n_1 + N_1 n_2$ and $k = k_1 + N_1 k_2$ with $n_1 = 0,1,\cdots,N_1-1$, $k_1 = 0,1,\cdots,N_1-1$, $n_2 = 0,1,2,3$, $k_2 = 0,1,2,3$, then $Z(k)$ can be obtained from the matrix equations

$$\begin{aligned} Y &= W_b \bullet C_1 X \\ Z &= C_2 Y^t \end{aligned} \qquad (2)$$

where W_b is an $N_1 \mathrm{x} N_1$ matrix with elements $W_b[k_1,n_1]=W_N^{n_1k_1}$, C_1 is an $N_1 \mathrm{x} 4$ coefficient matrix with elements $C_1[k_1,n_2]=W_4^{n_2k_1}$, X is a $4 \mathrm{x} N_1$ matrix with elements $X[n_2,n_1]=X(n_1+N_1n_2)$, C_2 is a $4 \mathrm{x} N_1$ coefficient matrix with elements $C_2[k_2,n_1]=W_4^{n_1k_2}$, Z is an $4 \mathrm{x} N_1$ matrix containing the transform outputs $Z[k_2,k_1]=Z(k_1+k_2N_1)$, and "$\bullet$" means element-by-element multiply [3].

The computational advantages of the manipulation leading to the matrix algorithm form (2) are evident when compared to the traditional direct form (1). First, the matrix products C_1X and C_2Y^t involve only exchanges of real and imaginary parts plus additions because the elements of C_1 and C_2 contain only ± 1 or $\pm j$, whereas the product in (1) requires complex multiplications. Also, the size of the coefficient matrix W_b in (2) is $(M/4)\times(M/4)$ vs. the $M \times M$ size of W_M in (1) ; consequently the number of overall direct multiplications is reduced by a factor of x16 compared to the direct form on which past systolic FFT implementations are based. Note that distribution of the elements in C_1 and C_2 does not impose significant bandwidth requirements because full complex numbers are not used. More details are provided in [3].

The algebra leading to (2) restricted $N_1/4$ to integer values. Therefore, in the column/row factorization it follows that $N_c\, N_r = (16\, n_{col})\,(16\, n_{row})$, where n_{col} and n_{row} are integer constants, so that transform lengths are restricted to multiples of 256. However, this restriction still allows far more reachable transform values than power-of-two implementations and a further benefit is that they are uniformly distributed. Other options are possible.

4. ARCHITECTURE

The basic operation is described logically in Fig. 1 for a 256 point transform (N_c=N_r=16) and a corresponding implementation for this is shown in Fig. 2. During the first stage the column transforms are performed on the 16 columns (X_{ci}, i=1..16) to produce 16 DFT results (Z_{ci},i=1..16). (Argument and result values are shown inside the boxes.) The second stage multiplies the Z_{ci} by the twiddle factors $W_N^{n,k}$. The output of this stage are the row DFT inputs (X_{ri}, i=1..16). These inputs are used in stage 3 (logically identical to the stage 1), to produce the final FFT result (Z_{ri},i=1..16). As shown in Fig. 2, the boxes on each end contain a 4x4 array of small processing elements (PEs) which do the 4x4 matrix multiplies and a 4x1 PE array in between that does the element-by-element multiplies. The only difference in the two arrays is that the LHS array produces the matrix output on it's right edge, whereas the RHS array stores its matrix output in the PE array. (Each PE in the array has a small memory associated with it to store intermediate results.) The final results Z_{ri},i=1..16, can be read from the array as soon as they are computed.

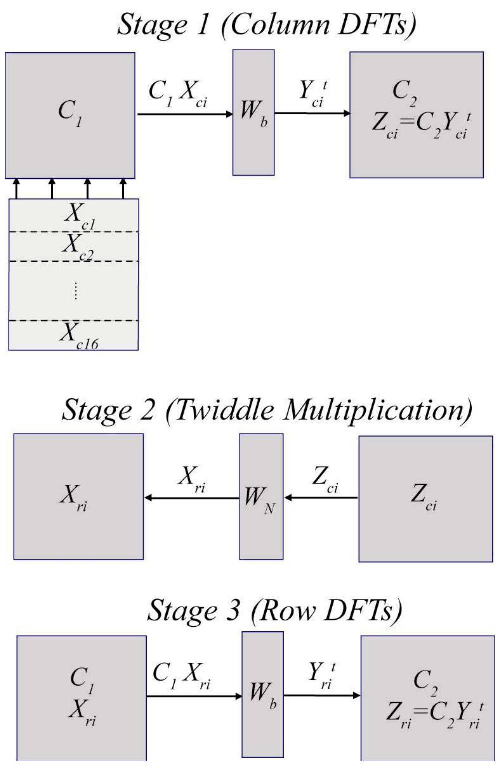

Figure 1. Functional description of the base-4 FFT circuit (N=256).

In general for the column DFTs, since C_1 is $N_1 \mathrm{x} 4$ and X is $4 \mathrm{x} N_1$, the matrix product C_1X can always be computed on an N_1 x 4 or (N_r/4) x 4 systolic array of PEs, each containing nominally two registers and an adder [8]. And since C_2 is $4 \mathrm{x} N_1$ and Y^t is N_1 x N_1, C_2Y^t can also be computed on an N_1 x 4 or (N_r/4) x 4 systolic array. Therefore, the basic column DFT architecture is two (N_r/4) x 4 PE arrays, with a single $N_r/4$ PE linear array in between the two to do the element by element complex multiplies by W_b and W_N. The array is two dimensional, but scales with transform size in only one dimension (vertically in Fig. 2). The transpose in between row and column DFTs is handled by appropriate shifts in C_2 and W_b [3].

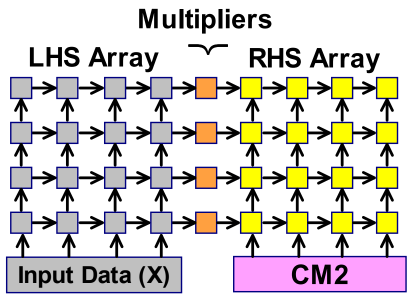

Figure 2. Base-4 array design corresponding to Fig. 1, showing PEs and data flow during Stage 1. A left hand side (LHS) and right hand side (RHS) 4x4 array of PEs perform matrix multiplication

It is necessary to implement the row computations differently because in general $N_r \neq N_c$, which means that $(N_c/4)x4$ matrix arrays would be required if the physical processing were identical to that in stage 1. Therefore, the computations associated with a row DFT, although logically similar to that for a column DFT, are physically mapped to one of the $(N_r/4)$ PE rows in used in stage 1. The row DFT processing within a PE row is done using systolic flows. The number of row DFTs to be performed is N_r so that with $(N_r/4)$ physical PE rows available, each PE row needs to perform four row DFTs.

From Fig. 2 it is clear that this architecture is very simple in that it avoids the stage-to-stage irregularities and the complex permutation networks, commutators and butterflies of conventional pipelined FFT implementations. Because each PE is simple and interconnections are local, higher clock speeds are possible. Finally, the mesh based layout and linear structure matches modern FPGA logic fabrics with their linear embedded memories and multipliers and is also very well suited to ASIC implementations.

5. CIRCUIT IMPLEMENTATION

5.1. Dynamic Range and Signal to Noise Enhancement

After the matrix multiplication and twiddle operations in stage 1 and 2 on the N_c columns of N, each LHS PE row will have computed the four sets of N_c row inputs needed to do the four row DFTs. With the important processing confined locally to a row, it is natural to provide separate hardware for each row to enhance dynamic range and signal-to-noise. Therefore, the base-4 circuit implementation provides a BFP operation for the column DFTs and a FP operation for the row DFTs as follows:

Stage 1: each RHS PE stores an exponent associated with an element of C_2Y^t (obtained after accumulation of the matrix-matrix partial products).

Stage 2: during multiplication, inputs associated with the same row DFT are normalized to the same exponent.

Stage 3: same as stage 1

On output the stage 3 exponents are combined with the stage 1/2 exponents to produce a single exponent associated with each FFT output value. Thus, a BFP operation is performed on each column DFT and a FP operation on each row DFT.

To demonstrate this dynamic scaling feature both the dynamic range and a signal-to-noise ratio were obtained from bit-accurate circuit simulations and the results are shown in the Table 1. Each entry in the Table 1 is the mean value obtained from "single tone" full range input data sets with different frequencies (random frequency and phase). No noise was added to the single frequency inputs, so "noise" here represents only internally generated round-off noise. Dynamic range DR was determined from the ratio

$$DR = 10 * \log_{10}\Big(\sum_{signals} x_m^2(i) / \max_{noise}(x_m^2(i))\Big)$$

where x_m is the FFT output magnitude. Signal-to-quantization-noise ratio S/QN is based on taking the ratio of total signal power divided by total noise power or

$$S/QN = 10 * \log_{10}\Big(\sum_{signals} x_m^2(i) / \sum_{noise} x_m^2(i)\Big) .$$

A comparison was made to Altera BFP circuits (FFT v2.2.0), since these are the fastest with the highest dynamic range of which we were aware. The results for the Altera circuit were obtained from a bit-accurate Matlab model that is created by the Altera FFT generator. (The entries don't include results for the 256-point 16-bit circuits where there was no round-off error, four for the base-4 circuit and one for the Altera circuit). Here it can be seen that overall the base-4 circuit saves approximately four bits in word length with its BFP/FP operation. (More bits are saved compared to fixed point FFT circuits that use scaling.)

Table 1. Dynamic range and signal-to-noise characteristics

Circuit	Size	Bits	S/QN	DR
base-4	256	16	89.0	96.3
Altera	256	16	77.2	84.9
Altera	256	20	86.9	98.2
base-4	1024	16	86.9	96.2
Altera	1024	16	71.5	84.4
Altera	1024	20	84.4	99.5
base-4	4096	16	85.5	99.0
Altera	4096	16	65.6	84.3
Altera	4096	20	81.9	103.1
Altera	4096	24	87.3	107.4

5.1. Scaling

The elements of C_1 are stored internally in the LHS array, one value in each of the $(N_r/4)$ x 4 available PEs and repeat every four rows because $C_1 = [C_B^t | \ldots | C_B^t]^t$, where

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

Therefore, each set of four PE rows containing C_B is generating the same input to the multiplier array. This makes it possible to use just four PE rows to generate all values of Y^t by recycling X_c sequentially through them. The multiplier PEs in Fig.2 would function the same as in a full sized array except that different values of W_b would be used for each set of four PE rows. The matrix multiply operations C_2Y^t on the RHS would be unchanged. During the row stage computations, as noted in Section 4, all row DFT computations are confined to one physical PE row. In this way a simple partitioning scheme is possible, whereby the entire computation can be processed by any set or sets of four PE rows.

Partitioning in this way permits two important scaling options. First, it allows a small array to compute any size transform as long as there is enough local PE memory to hold intermediate results. Second, it allows an option to add hardware in the form of sets of four PE rows to increase overall throughput (as long as the total number of rows is not more than the nominal array length of $N_r/4$). These rows are added to extend the array vertically from the minimal array size in Fig.2. This latter feature is particularly important when it is necessary to match system throughput to hardware throughput. Finer partitioning options are also possible.

5.1. Computational latency

It is often important to rapidly calculate a DFT with minimum latency (time to compute as single DFT), as opposed to a sequence of DFTs with a maximum throughput. A difficulty arises in doing this for most traditional pipelined FFTs because the pipeline depth can be significant, typically equal to N [5]. In contrast the base-4 design has a pipeline depth of only $N_r/4$. Using a 1024-point DFT for example, a traditional FFT pipeline depth would be 1024 clock cycles vs. 8 for the base-4 design.

6. FFT PERFORMANCE

Performance characteristics were obtained from two FFT circuits, (256 and 1024-point) that were designed to accept a continuous 16-bit input stream *X(n)*, while generating a continuous output stream *Z(n)* at the same rate ("streaming") because this mode is common to many signal processing applications. The design has circuit pins for real and imaginary inputs/outputs, *Z/X,* a single global reset, and two clocks. Because the base-4 designs compute 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.

For fair comparisons the base-4 and Altera designs were targeted to the same underlying hardware, specifically Altera Stratix II EP2S15 and EP2S60 devices (speed grade -3, 90nm technology) for the 256 and 1024-point designs. Altera Quartus II tools (v5.1) were used to design and evaluate both FFT circuits. The base-4 circuit operation was verified by comparing the Quartus simulator result with a bit-accurate simulation model. The Quartus II timing analyzer finds the critical path that determines the maximum clock frequencies.

The base-4 and Altera results are summarized in Table 2. Because the base-4 design provides higher dynamic range for a given bit-length, 20-bit Altera FFTs were used in the comparison. For the Altera design the clock speed listed in Table 2 is also the complex sample rate. For the base-4 design the complex sample rate is larger than the clock speed in the table by a ratio of the transform size to number of clock cycles to compute a transform. For example, for the 256 point circuit the sample rate would be 256*361/240 or 385 MHz. However, the timing simulations show that the output data clock could run at 391 MHz. In the table the number of "adaptive logic modules" (ALMs) is included because this value is roughly comparable to a Xilinx "slice".

Table 2. Comparison of Altera and base-4 FFT circuits.

Category	**Altera (256)**	**Base-4 (256)**	**Altera (1024)**	**Base-4 (1024)**
Throughput (cycles/DFT)	256	240	1024	688
Clock speed (MHz)	302	361	297	356
Throughput (μsec)	0.85	0.66	3.45	1.93
ALMs	4192	4496	5102	8490
Memory Bits	48640	48880	194560	202992
18-bit multipliers	12	16	12	32

As can be seen from Table 2, both the base-4 circuits provide higher throughput because the transform calculation uses fewer clock cycles and the clock rate is higher. Furthermore, the increased performance is obtained without a more than proportional usage of memory and logic. To show this Table 2 calculates a figure of merit (FOM) that is the product of throughput (cycles/DFT), number of ALMs, memory (Kbits) divided by the clock frequency. The two very different FFT implementations track each other closely in this regard. The base-4 FFT does use more multipliers; however, the Altera complex multipliers are completely hardwired so consume little chip space and large numbers are available (704 18-bit multipliers in Altera's Stratix III EP3SE260).

In a non-streaming application where all FFT input data is available simultaneously, the base-4 latency would be approximately the same as the throughput values shown in Table 2 because the pipeline depths are so small.

Table 3. Comparison of equivalent Altera and base-4 FFT circuits

Points	**Altera FOM**	**Base-4 FOM**
256	0.17	0.15
512	0.71	0.70
1024	3.4	3.3
2048	22.8	17.9*
4096	97.8	96.1*

(*estimate)

7. CONCLUSION

A high dynamic range, high-performance systolic FFT has been 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 DFTs. It is scalable so that any implementation can do any size FFT and hardware can be added in identical blocks to increase throughput. It is inherently fast because it uses fewer clock cycles per transform than most pipelined FFTs and runs at high clock rates due to localized interconnects and a simple underlying architecture. 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, an adder and memory. Implementation examples on modern FPGAs show that the high performance is possible and is achievable with reasonable hardware costs. Better performance can be expected with an optimized base-4 design.

8. REFERENCES

[1] Ronald N. Bracewell, The Fourier Transform and Its Applications, 3rd Edition, McGraw-Hill, 1999.

[2] Zhi-Xing Yang, Yu-Peng Hu, Chang-Yong Pan, and Lin Yang, "Design of a 3780-point IFFT processor for TDS-OFDM,", IEEE Trans. Broadcasting, Vol. 48, pp.57-61, Mar. 2002.

[3] J. G. Nash, "Computationally efficient systolic architecture for computing the discrete Fourier transform," IEEE Transactions on Signal Processing, Volume 53, Issue 12, Dec. 2005, pp. 4640-4651.

[4] S. M. Currie, et. al, "Implementation of a single chip, pipelined, complex, one-dimensional fast Fourier transform in 0.25um bulk CMOS", Proc. Application-Specific Systems, Architectures and Processors, 2002, pp. 335 – 343.

[5] Shousheng He and M. Torkelson "Designing pipeline FFT processor for OFDM (de)modulation", Proc. Int. Symp. Signals, Systems, and Electronics, 1998, pp. 257 - 262.

[6] Se Ho Park, et. al., "Sequential Design of a 8192 Complex Point FFT in OFDM Receiver," Proc. The First IEEE Asia Pacific Conference on ASIC 1999, pp. 262 – 265.

[7] Yu-Wei Lin, Hsuan-Yu Liu, and Chen-Yi Lee, "A Dynamic Scaling FFT Processor for DVB-T Applications," IEEE J. Solid-State Circuits, Vol. 39, No.11, Nov. 2004, pp.2005-2013.

[8] S. Y. Kung, VLSI Array Processors, Prentice Hall, 1988, pp. 138-139.