Presented at 2007 Hawaii International Conference on System Sciences: Mobile Computing Hardware Architectures

An FFT for Wireless Protocols

J. Greg Nash
Centar, Los Angeles, California, USA
jgregnash@centar.net, www.centar.net

Abstract

A different approach to parallel FFT implementation is described here based on a new matrix formulation of the discreet Fourier transform (DFT) which decomposes it into structured sets of multiplication-free 4-point DFTs. As a result, (1) implementations are simple, locally connected and structured, thereby allowing lower power and higher performance mappings to modern FPGAs and ASICs; (2) significant added functionality and flexibility accrues from the inherent scalability; and (3) good arithmetic efficiency is retained. These benefits would be best suited to wireless devices where future 4G protocols will utilize the FFT-based digital modulation schemes orthogonal frequency division multiplexing (OFDM) and scalable orthogonal frequency division multiple access (OFDMA).

1. Introduction

The DFT is of central importance to a large variety of signal processing applications: telecommunications, radar (synthetic aperture radar, pulse compression, range-Doppler imaging), antenna arrays (frequency domain beamforming), navigation (GPS), speech processing (speech recognition/synthesis), image processing (digital still/video/cell-phone cameras, high-definition television, video surveillance systems, industrial inspection systems, medical imaging devices), and sonar (LOFARgram) [1][2].

Here the application focus is communications, where the FFT is rapidly gaining acceptance for use in wireless devices via the specification of OFDM and OFDMA modulation in future standards, e.g., 802.11n (next generation wireless LAN), 802.16/e (wireless fixed and mobile metropolitan area networks-WiMax), 802.20 (mobile broadband wireless access), 802.22 (wireless regional area networks), Flash-OFDM (Fast Low-latency Access with Seamless Handoff OFDM), 3GPP LTE (3rd Generation Partnership Project, Long Term Evolution), and HiperMAN (European broadband fixed wireless) [3][4][6][7].

Future OFDM-based protocols require flexibility in choosing the number of sub-carriers. Here a traditional FFT suffers a power-of-two limitation that severely restricts the number of attainable points and leads to a highly non-uniform distribution. More control in the choice of transform sizes can benefit overall system performance as in the recently announced Chinese Digital Multimedia Broadcasting Terrestrial/Handheld standard (DMB-T/H) which uses OFDM based on 3780 sub-carriers rather than the power-of-two value 4096 [5]. Non-power-of-two transform sizes also have been proposed in new wireless protocols mentioned above that use OFDMA as in 802.22 (1024, 2048, 4096, 6144 points) [6] and 3GPP LTE (128, 256, 1024, 1536, 2048 points) [7].

Also, 4G OFDM-based protocols will demand very high throughputs due to requirements for higher bandwidths and multiple data streams associated with multiple-input-multiple-output (MIMO) antennas. Example estimates indicate a need for throughputs of ~10μsec per 1024-point FFT per OFDM stream [8]. With ≥4 streams [9], computation times less than 2.5μsec/1K FFT would then be necessary. High dynamic ranges of 60-100db [10] could be needed as well.

Therefore, to meet the signal processing requirements of future wireless systems a parallel FFT architecture is desired that

- doesn't restrict the DFT size N to be either powers of a radix or factorable into relatively prime numbers
- scales in a simple way to match required system performance
- provides run-time transform size options
- offers high dynamic range for a given word length
- is ideally suited to today's FPGA and ASIC hardware
- provides low latency as well has high throughput
- accommodates 2-D as well as 1-D DFTs
- uses minimal power
- possesses all the locality, regularity and design simplicity of systolic designs

In this paper a novel high-performance, scalable FFT circuit architecture is described which provides this level of generality. Its algorithmic underpinnings are derived from a decimation in time and frequency of the DFT which leads to a much simpler matrix based formulation of the DFT [11]. It combines the performance associated with the use of radix-4 butterflies in traditional FFTs with the generality and design/implementation simplicity of systolic arrays.

In Section 2 previous work on FFT implementations is summarized. Section 3 derives the proposed "base-4" architecture, Section 4 provides details on an example FPGA-based implementation, followed in Section 5 by a conclusion.

2. Related work

Past systolic array designs that have been proposed for computation of the DFT 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) and the throughput is O(N) arithmetic cycles per DFT. 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 [15]. 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 (complex) multipliers required is N, so that for a 1024-point transform a prohibitive number of multipliers (1024) would be necessary.

Most parallel 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. $O(N)$ delay registers are used to match the outputs and inputs of the different stages. These "pipelined" FFTs are computationally efficient and make 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 used. A good summary of these designs can be found in [16]. 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 [17]; however, there have been no demonstrations of improved performance.

The 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 capability to perform non-power-of-two DFT computations. 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 multipliers (704 18-bit multipliers in Altera's Stratix III EP3SE260) which consume less than 10% of the overall floor-plan area. So rather than minimizing use of multipliers, many times it is a better strategy to use as many as desirable lest they be wasted.

3. Background

3.1. Algorithmic foundation of "base-4" architecture

Here, the derivation of the new matrix equation for the DFT is summarized. (More details can be found in [11] and an alternate approach appears in [18].) The derivations begins with the direct form DFT representation

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

where M is the transform length, $X(n)$ are the time domain input values, $Z(k)$ are the frequency domain outputs and $W_M = e^{-I(2\pi/M)}$. In matrix terms (1) may be represented as

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

where C is a coefficient matrix containing elements W_M^{nk}. If M can be factored as $M = N_1 N_2$, then applying the reindexings $n = n_1 + N_1 n_2$ and $k = k_1 + N_1 k_2$ with $n_1 = 0,1,...,N_1 - 1$, $k_1 = 0,1,...,N_1 - 1$, $n_2 = 0,1,...,N_2 - 1$,

$k_2 = 0,1,\cdots,N_2-1$, it can be shown that if N_1/N_2 is an integer value (1) becomes

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

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

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

$$c_1=\begin{bmatrix}1\\1\\1\\1\end{bmatrix},\ c_2=\begin{bmatrix}1\\-I\\-1\\I\end{bmatrix},\ c_3=\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix},\ c_4=\begin{bmatrix}1\\I\\-1\\-I\end{bmatrix} \quad \text{and}$$

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

where C_B above is the coefficient matrix for a 4-point DFT and also describes a radix-4 decimation in time butterfly. Consequently, in (3) the matrix multiplications by C_{M1} and C_{M2} represent repeated use of a radix-4 butterfly.

The reindexed direct form expression (3) leads to several novel computational features, compared to previous systolic implementations, that are exploited in realizing a base-4 circuit architecture:

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 $M=N_1N_2=mN_2^2=16m$, e.g., transform sizes are only constrained to be integer multiples of 16.
2) In comparing (3) with (2), significant computational advantages of the reindexed form (3) can be seen. In (3) the matrix products $C_{M1}X_b$ and $C_{M2}Y_b^t$ involve only addition/subtraction because the elements of C_{M1} and C_{M2} contain only ± 1 or imaginary numbers $\pm I$, whereas the product CX in (2) requires complex multiplications.
3) The size of the coefficient matrix W_b in (3) is $(M/4)\times(M/4)$ vs. the $M \times M$ size of C in (2), leading to a reduction in the number of overall multiplications by x16 compared to (2).
4) 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).

3.2. Base-4 architecture

The FFT circuit implementation makes use of two levels of algorithm 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. This approach requires three basic steps:

1) Compute successively N_c column DFTs of length N_r on column inputs X_{ci}, i=1.. N_c, where the computational flow is as shown in Fig. 1. The column results Z_{ci},i=1..N_c, are stored N_c values per processing element (PE) in small right hand side (RHS) PE memories (Fig. 2).
2) Multiply the Z_{ci} by the twiddle factors $W_N^{n,k}$ by moving the Z_{ci} values in systolic fashion from the RHS array through a linear multiplier array to the LHS array. These results are stored N_c values per PE in small left hand side (LHS) memories and become the row inputs X_{ri} for the last step. (Without this step a 2-D DFT is performed.)
3) Compute successively N_r row DFTs of length N_c on row inputs X_{ri}, i=1.. N_r. These are performed logically in the same way as in step 1 using the same (N_r/4) x 4 sized arrays and the FFT output is Z_{ri},i=1..N_r.

In general for the column DFTs, since C_{M1} is $N_1 \times 4$ and X_{ci} is $4 \times N_1$, the matrix product $C_{M1}X_{ci}$ can always be computed on an N_1 x 4 or (N_r/4) x 4 systolic array of PEs (N_r=M=4N_1), each containing nominally two registers and an adder [12]. And since C_{M2} is $4 \times N_1$ and Y_{ci}^t is N_1 x N_1, $C_{M2}Y_{ci}^t$ 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 as shown in Fig. 2. 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_{M1}, C_{M2} and W_b [11]. (This architecture was automatically generated by a special tool [19][20].)

In step 3 above the inputs X_{ri} in the matrix multiply $C_{M1}X_{ri}$ are supplied internally from within the LHS PE memories. Also, because in general $N_c \neq N_r$, this step would require arrays of size (N_c/4) x 4. Therefore, matrix multiplications in step 3 use a different computational flow. Specifically, this to map both the $C_{M1}X_{ri}$ and $C_{M2}Y_{ri}^t$ matrix multiplies to a single physical PE row in the architecture used to do step 1 (Fig. 2). This can be done by projecting the (N_c/4) x 4 2-D matrix multiplies onto a 1-D linear systolic array associated with a physical (step 1) PE row. The number of row DFTs to be performed is N_r, so with N_r/4 physical PE rows available, each PE row will perform 4 row DFTs.

The term "base-4 architecture" refers to the array structure that supports all three steps above. It consists of the array structure shown in Fig. 2 plus a path for RHS to LHS data movement for step 2 and some additional control.

In the column/row factorization it follows that both N_c and N_r must be multiples of 16 as noted in the previous section. Then, since N=N_rN_c, transform lengths are restricted to integer multiples of 256. This restriction is the result of choosing the base b=4. However, if b=2, then a similar analysis would show that a base-2 circuit design could perform any transform that is an integer multiple of 16. Alternatively, if the first level factorization N= N_r N_c is not used and b=4, the direct transform (3) itself can be used so that attainable transform values would also be integer multiples of 16. The same architecture supports any of these implementation approaches, so there are a number of options for matching desired and available transform lengths.

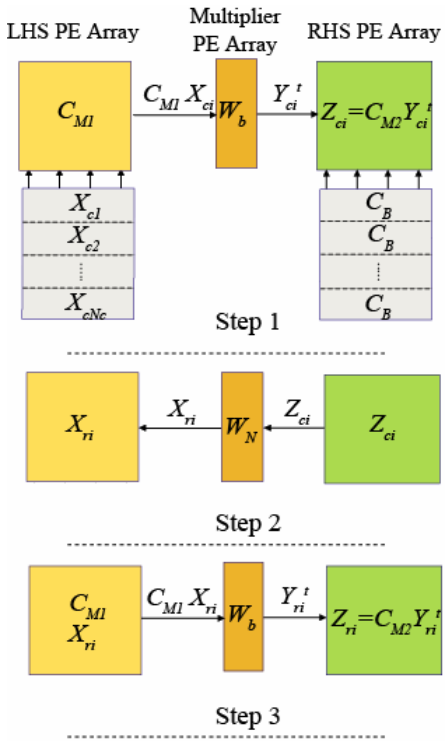

Fig. 1. Functional operation for column/row decomposition

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. Throughputs are also increased because the number of clock cycles per DFT is less than the transform length N (Table 1). (For most pipelined FFTs the throughput is equal to the transform length N in cycles/DFT.)

Table 1. Base-4 transform length vs. throughput.

Points	N_r	N_c	Throughput (cycles/DFT)
256	16	16	220
512	32	16	284
768	32	24	460
1024	32	32	668
1536	48	32	796
2048	64	32	924

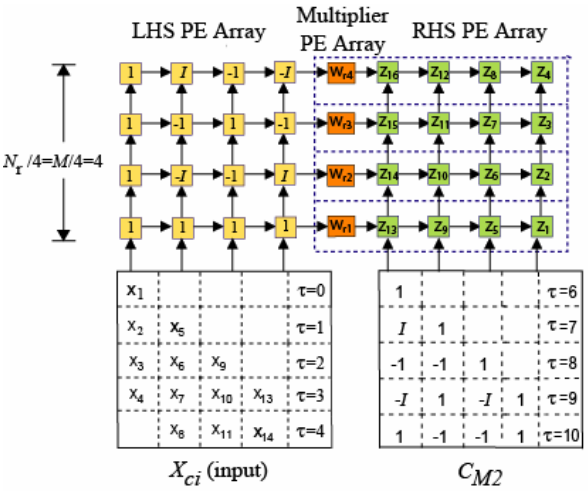

Fig. 2. Array design for $N_r=M=16$ showing PEs and data flow during step 1. Here W_{ri} represents row i of matrix W_b. (Subscripts "b" not shown for z.)

3.3. Dynamic range extension

For OFDM-based applications high dynamic ranges are required for a given word length because high peak-to-average signals are generated. For this reason and to avoid the considerable design complexity and additional logic associated with fixed word length circuits, a unique type of block floating point (BFP) circuitry has been added and is briefly described here.

With the important processing confined locally to a row, it is natural to provide separate BFP hardware for each PE row as shown in the dashed boxes in Fig. 2. Therefore, the base-4 circuit implementation provides BFP operations as follows (steps below correspond to those in Section 3.2):

Step 1: Each RHS PE stores an exponent associated with an element of Z_{ci}.
Step 2: During twiddle multiplication, inputs associated with the same row DFT are normalized to the same exponent.
Step 3: Each RHS PE stores an exponent associated with an element of Z_{ri}.

On output the step 3 exponents are combined with the step 2 normalized exponents to produce a single exponent associated with each FFT output value. Thus, a BFP operation is performed on row DFT inputs and a floating point (FP) operation on each row DFT output. Note that for larger transform sizes the number of BFP/FP regions increases because the number of physical rows is increased.

3.4. 2-D DFTs and long length 1-D DFTs

Larger FFT implementations, specifically 2-D arrays and transform sizes from ~8196-points and higher, are of less relevance to the wireless communications market. However, many other signal processing applications make use of larger transform sizes. One approach to doing this is to use a "third level" of factorization. That is, when either N_r or N_c is equal to or greater than 256, it becomes possible to process that column or row as if were a separate 1-D FFT. For example, if $N_r = N_c = 256$ (64K-point transform), then each column i can factored again as $N_{ci} = n_r\, n_c$, where n_r and n_c are the new row and column parameters associated with a single column. This third level factorization preserves the small array size and also reduces the relative overall number of multiplications and keeps the precision good.

3.5. Partitioning

The scalability of the architecture is also reflected in the ease with which FFTs can be partitioned to run on fixed hardware so that OFDMA run-time transform length options are possible. This possibility can be seen if the DFT expression in (3) is rewritten as $Y_b = W_b \bullet \left[C_B^t \,|\, C_B^t \,|\, ... \right]^t X_b$, where X_b is a row or column input. This shows that C_B is applied multiple times to X_b, computing the same result each time. Therefore, it is possible for a single set of four rows to compute all necessary elements of Y_b and then Z_b. As an example, consider a 1024-point transform (N_r=N_c=32). This would nominally use LHR and RHS 8 x 4 PE arrays. However, if only four PE rows are used, as shown in Fig. 2., then the computation could be partitioned, first by calculating from (3)

$$\begin{bmatrix} y_{b11} & \cdots & y_{b18} \\ \vdots & \vdots & \vdots \\ y_{b41} & \cdots & y_{b48} \\ \hline 0 & & \end{bmatrix} = W_b \bullet \begin{bmatrix} C_B \\ 0 \end{bmatrix} X_b$$

$$\begin{bmatrix} z_{b11} & \cdots & z_{b14} \\ \vdots & \vdots & \vdots \\ z_{b41} & \cdots & z_{b44} \end{bmatrix} \,|\, 0 = \begin{bmatrix} C_B & C_B \end{bmatrix} \begin{bmatrix} y_{b11} & \cdots & y_{b41} \\ \vdots & \vdots & \vdots \\ y_{b18} & \cdots & y_{b48} \end{bmatrix} \,|\, 0$$

which provides half the answer, and then doing the

same calculation with $C_{M1} = \left[0 \mid C_B^t\right]^t$. Operationally, this could be done by doing step 1 (Section 3.2) twice, i.e., stream the input X_b through the four rows two separate times, using different sets of coefficients W_{ri} each time. Then step 2 would take twice as long because the array uses 4 rather than the 8 nominal PE rows. Finally, step 3 can be done as before since all the row DFT processing is still confined to a single PE row. The main difference is that now each physical PE row would do eight rather than four row transforms. In this way a simple partitioning scheme is possible, whereby any size FFT can be performed on any set or sets of four PE rows. Other finer grain partitioning schemes are possible as well. Also, various options and associated tradeoffs with respect to memory usage exist as to how to order steps 1-3 during the computations.

An estimate of the approximate number of clock cycles per transform can be obtained from multiplying the values in Table I by the reduction in hardware used. For example, from Table I the base-4 1024-point transform in nominal array form (8 x 4 LHS and RHS arrays) takes 668 clock cycles. Using a single four PE row slice, as described above, the processing time would be ~2 times slower, or ~2x668=1336 cycles.

3.6. Computational performance

3.6.1. 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 transitions between steps (Section 3.2). Since the combined processing requires N_c column DFTs of length N_r and N_r row DFTS of length N_c, the overall throughput *Thrpt* in cycles per transform can be shown to be

$$Thrpt = \underbrace{N_c(N_r/4)}_{column\ DFTs} + \underbrace{4(N_c+1)}_{twiddle\ \ multiplication} + \underbrace{N_c^2/4}_{row\ DFTs} + delay\ .$$

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 in step 3 where $N_c \neq N_r$ [11]. The delay in switching between to and from the twiddle step 2 is twice the time to traverse the right-left direction of the array or $6b = 24$, where the multiplier PE is assumed to contain b delay stages. Therefore, the approximate throughput becomes

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

The throughput for a variety of FFT calculations using this formula are shown in Table 1.

3.6.2. 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. 2 it can be seen that the maximum length data path in the array is the time to travel the length (of the array ($N_1/4 = N_r$ cycles), so that

$$L = N_r + Thrpt\ .$$

4. 256-point FFT circuit example

4.1. Circuit description

To demonstrate this base-4 architecture a 256 point 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. The design has circuit pins for real and imaginary inputs/outputs, *Z/X*, a single global reset, and two clocks. The circuit architecture in terms of PEs and multipliers is shown in Fig. 2.

To achieve a high dynamic range for OFDM applications a 16-bit word length was chosen. A set of 62 256-point full-scale 16-bit "single tone" transform inputs (random phase and random frequency with no noise added) showed that the mean signal-to-quantization-noise ratio was 89.0 db and the mean maximum dynamic range (signal power to maximum noise value ratio) was 96.3db.

The design was targeted to an Altera Stratix II EP2S15 FPGA (90nm technology). Altera Quartus II tools (v5.1) were used to design and evaluate the FFT circuit. 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. The circuit required 4496 adaptive logic modules and used 48.9K memory bits.

The maximum clock speed was 361MHz, which corresponds to a throughput of 0.66μsec/FFT. The number of clock cycles/FFT was 240 rather than the 220 in Table 1 because a simpler control scheme, whereby a complete transform was completed before starting a new one, was used. By overlapping the start/finish of different transforms, a throughput closer to 220 cycles/FFT could be expected.

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. The I/O clock then runs faster by a ratio of 256/240 or 385 MHz. Timing analysis shows that this clock can run at speeds up to 399MHz.

4.2. Precision

The base-4 circuit is based on a matrix expression that is not as efficient in a "complexity analysis" sense as the traditional FFT. However, the base-4 matrix expression (3) uses a form of "strength reduction" that trades off multiplications for additions. Since essentially all additions are done to full precision, the round-off errors occur primarily in the multiplications, which are already reduced considerably in number compared to the usual DFT matrix expression $Z=CX$ in (2). For example, the total number of base-4 multiplications performed for a 1024-point FFT is approximately the same as for a traditional 1024-point radix-2 FFT. The actual measured precision for a 1024-point transform, shown in Table 2, based on a large number of random (real and complex) input data sets, shows that the 16-bit base-4 FFT has a factor of ~4 better precision than a 16-bit Altera streaming BFP FFT circuit and is within a factor of ~2 of that for a 20-bit Altera FFT circuit. In this case bit-accurate Altera FFT Matlab circuit models were used that are generated from Altera's Megacore (v2.2.0) utility.

Table 2. Measured precision for 1024-point FFTs.

Error ($(\bar{x}-x)/\bar{x}$)	Altera 16-bit	Altera 20-bit	Base-4 16-bit
Mean	0.00038	-0.000039	0.000097
Standard Deviation	0.00118	0.000217	0.000412
Minimum Error	-0.0292	-0.01008	-0.01556
Maximum Error	0.0192	0.00426	0.0283

4.3. Scaling

For DSP applications in general, and wireless applications in particular, an important system issue is that of matching required DSP system throughput to available hardware resources, because hardware translates directly to cost and power. The proposed FFT architecture described here is fundamentally scalable in that it is based on a matrix representation of the DFT (3), where larger DFT matrices correspond directly to larger circuit arrays as described in Section 3.2. One way of achieving different resource-speed tradeoffs to meet such throughput challenges is to simply change N_r and N_c keeping N the same. For example, a 1024 point FFT could be computed using three different sets of values a shown in Table 3. Here, the transform time can be varied by a factor of ~4 in this simple way.

Table 3. Example of estimated performance for scaling options obtained by varying N_r and N_c, keeping N the same. (The number of real multipliers is N_r.)

N_r (multipliers)	N_c	Transform Size	Throughput (cycles/DFT)
32	32	1024	688
16	64	1024	1576
64	16	1024	424

4.4. Power dissipation

Power dissipation is a critical parameter for mobile wireless systems. The base-4 architecture already achieves low power by

1. Use of many small memories (one per PE), so that they are both low power and fast. (Only 14% of the total circuit power dissipation comes from the memories.)
2. Reuse of data flowing through registers (systolic processing) so that unnecessary memory reads and writes are avoided.
3. Localized interconnects to minimize wiring overhead. (Total interconnect dynamic power is only 46% of the total power for the 256-point circuit.)

Power dissipation was 2.4W for the 256-point FFT, corresponding to 1611nJ/FFT. By gating unused circuitry it is estimated that power dissipation could be further reduced by 10-15%.

5. Summary

The base-4 FFT architecture is intended to strike a balance between the flexibility of direct systolic designs and the computational efficiency of a pipelined designs. In this way fast, regular and scalable implementations are possible that have the necessary functionality to support OFDMA applications. In particular the ability to partition FFTs on to a fixed array size (chosen to meet system throughputs) allows different transform sizes to execute dynamically on the same hardware. Also, non-power-of-two computations are possible. A 256 point FFT circuit implementation example was described that provides a level

performance higher than other 90nm traditional pipelined FFTs of which we are aware.

6. References

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

[2] E. Oran Brigham, Fast Fourier Transform and Its Applications, Prentice-Hall, 1988.

[3] Hui Liu, Guoqing Li, OFDM-Based Broadband Wireless Networks: Design and Optimization, Cambridge University Press, 2005.

[4] http://www.ieee802.org/20/

[5] 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.

[6] A PHY/MAC Proposal IEEE 802.22 WRAN Systems (http://www.ieee802.org/22/Meeting_documents/2006_Mar/22-06-0005-05-0000_ETRI-FT-I2R-Motorola-Philips-Samsung-Thomson_Proposal.ppt).

[7] Physical layer aspects for evolved Universal Terrestrial Radio Access, 3GPP TR 25.814 V7.1.0, Oct. 2, 2006. (http://www.3gpp.org/ftp/Specs/archive/25_series/25.814/)

[8] Angela Doufexi and Simon Armour, "Design considerations and physical layer performance results for a 4G OFDMA system employing dynamic subcarrier allocation," Proc. IEEE 16th Int. Symp. On Personal, Indoor and Mobile Radio Communications, pp. 357-361, 2005.

[9] Ludwig Schwoerer and Ernst Zielinski, "Optimized FFT Architecture for MIMO Applications," Proc. 13th European Signal Processing Conference (EUSIPCO2005). (http://www.arehna.di.uoa.gr/Eusipco2005/defevent/papers/cr1361.pdf.)

[10] Bosco Leung and Behzad Razavi, RF Microelectronics. Prentice Hall, 2004.

[11] J. Greg Nash, "Computationally efficient systolic architecture for computing the discrete Fourier transform", IEEE Trans. Signal Processing, Vol. 53, pp. 4640-4651, Dec. 2005.

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

[13] Chih Kuo, Ching-Hua Wen, Chih-Hsiu Lin, and An-Yeu Wu, "VLSI design of a variable-length FFT/IFFT processor for OFDM-Based communication systems," EURASIP Journal on Applied Signal Processing 2003:13, pp. 1306–1316.

[14] Zhong Hu and Honghui Wan, "A novel generic fast Fourier transform pruning technique and complexity analysis," IEEE Trans Signal Proc., Vol. 53, NO. 1, Jan. 2005, pp. 274-282.

[15] S. He and M. Torkelson, "A Systolic array implementation of common factor algorithm to compute DFT," Proc. Int. Symp. Parallel Architectures, Algorithms and Networks, Kanazawa, Japan, pp. 374-381, 1994.

[16] S. He and M. Torkelson, "A new approach to pipeline FFT processors," Proc. Int. Conf. Parallel Processing (IPPS 96), pp. 766-770.

[17] V. Boriakoff, "FFT computation with systolic arrays, a new architecture", IEEE Trans. Circuits Systems II, Vol. 41, pp.278-284, Apr. 1994.

[18] J. Greg Nash, "Hardware efficient base-4 systolic architecture for computing the discrete Fourier transform," Proc. IEEE Workshop on Signal Processing Systems, pp.87-92, 2002.

[19] J. Greg Nash, "Automatic generation of systolic array designs for reconfigurable computing," Proc. Int. Conf. Engineering of Reconfigurable Systems and Algorithms (ERSA 02), pp.176-182, CSREA Press, 2002.

[20] J. Greg Nash, "Constraint directed CAD tool for automatic latency-optimal implementation of 1-D and 2-D Fourier transforms ", Proc. SPIE ITCom, Reconfigurable Technology: FPGAs and Reconfigurable Processors for Computing and Communications IV, Vol. 4867, pp. 8-19, July 2002.