High-Performance Scalable Base-4 Fast Fourier Transform Mapping

Dr. J. Greg Nash, Centar
(jgregnash@centar.net, www.centar.net)

A novel, scalable parallel FFT architecture mapping is described here that supports transform lengths which aren't powers of two or four, that provides low latency as well as high throughput, that can do both 1-D and 2-D discreet Fourier transforms (DFTs), that is ideally suited to today's complex FPGA architectures, that possesses all the regularity and design simplicity of systolic arrays and that is naturally suited to a parameterized HDL form. Its algorithmic underpinnings are based on an observation that with suitable permutations, the DFT coefficient matrix can be partitioned into regular blocks of smaller "base-4" matrices (equivalent to a decimation in time *and* frequency) [1]. From this new base-4 matrix DFT description we have derived a new latency and throughput optimal base-4 FFT architecture. It combines the performance of traditional radix-4 "pipelined FFTs" with the design and implementation simplicity of systolic arrays, and yet is versatile.

An N point DFT is defined by

$$Z[k] = \sum_{n=1}^{N} X[n]\, e^{-j(2\pi/N)(k-1)(n-1)} \quad k = 1, 2 \ldots N \quad \text{or} \quad Z = CX \tag{1}$$

where $X[n]$ are the time domain input values, $Z[k]$ are the frequency domain outputs and C is a coefficient matrix containing elements $W_N^{kn} = e^{-2j\pi(n-1)(k-1)/N}$. In order to transform C into the desired base-b ($b=4$) format it is necessary to find a permutation matrix P that reorders X and Z according to

$$\begin{aligned} X_b &= P\ [X_1\, X_2\, X_3\, X_4\, X_5 \ldots X_{N-3}\, X_{N-2}\, X_{N-1}\, X_N]^t \\ &= [X_1\, X_{1+N/4}\, X_{1+N/2}\, X_{1+3N/4}\, X_2 \ldots X_{N/4}\, X_{N/2}\, X_{3N/4}\, X_N]^t . \end{aligned} \tag{2}$$

and $Z_b = P\ Z$. With this value of P, C can be transformed into $C_b = PCP^t$, so that $Z_b = C_b X_b$. This transformation allows C_b to be written as an $(N/b)\times(N/b)$ array of bxb blocks, each block $C_b[i,j]$ specified by

$C_b[i,j] = W_M[i,j] * c_{D((j-1) \bmod (b))+1} C_{((i-1) \bmod (b))+1}$ where $c_{Di} = c_i^t I$ with c_i a b-element vector, C_i is a bxb matrix, each row being c_i^t, and $W_M[i,j]$ is an element in the $(N/b)\times(N/b)$ matrix,

$$W_M = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1 & W^1 & W^2 & W^3 & \cdots \\ 1 & W^2 & W^4 & W^6 & \cdots \\ 1 & W^3 & W^6 & W^9 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{3}$$

With this block reformulation it is possible to factor (1) into

$$\begin{aligned} Y &= W_M \bullet C_{M1} X \\ Z &= C_{M2} Y^t \end{aligned} \tag{4}$$

where "$\bullet$" in (4) corresponds to an element by element multiply [2]. In (4) C_{M1} and C_{M2} contain N/b^2 submatrices $C_B = [c_1 \mid c_2 \mid \ldots \mid c_b]^t$ with the form $C_{M1} = \left[C_B^t \mid C_B^t \mid \ldots \right]^t$ and $C_{M2} = [C_B \mid C_B \mid \ldots]$, and Z, X have been redefined as follows:

$$Z = \begin{bmatrix} Z_1 & & Z_{N/4} \\ Z_{1+N/4} & \cdots & Z_{N/2} \\ Z_{1+N/2} & & Z_{3N/4} \\ Z_{1+3N/4} & & Z_N \end{bmatrix}, X = \begin{bmatrix} X_1 & & X_{N/4} \\ X_{1+N/4} & \cdots & X_{N/2} \\ X_{1+N/2} & & X_{3N/4} \\ X_{1+3N/4} & & X_N \end{bmatrix}. \tag{5}$$

For base-4 designs (b=4), $c_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$, $c_2 = \begin{bmatrix} 1 \\ -j \\ -1 \\ j \end{bmatrix}$, $c_3 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$, $c_4 = \begin{bmatrix} 1 \\ j \\ -1 \\ -j \end{bmatrix}$

and C_B describes a radix-4 decimation in time butterfly.

By comparing (4) with (1), the computational advantages of the manipulation leading to the FFT algorithm form (4) for base-4 designs are readily evident. In (4) the matrix products $C_{M1}X$ and $C_{M2}Y^t$ involve only exchanges of real and imaginary parts plus additions because the elements of C_{M1} and C_{M2} contain only ± 1 or $\pm j$, whereas the product CX in (1) requires complex multiplications. Also, the size of the coefficient matrix W_M in (4) is $(N/b)\times(N/b)$ vs. the $N\times N$ size of C in (1); consequently the number of overall direct multiplications in (4) is reduced by a factor of x16 compared to the direct form (1) on which past systolic FFT implementations are based. Note that distribution of the elements in C_{M1} and C_{M2} does not impose significant bandwidth requirements because full complex numbers are not used.

The overall processing for an M point FFT is done using the factorization $M=N_1N_2$, followed by a series of "row/column" 1-D FFTs on N_1 and N_2. Each of these 1-D FFTs is performed by a secondary factorization into 2-D FFTs according to (4) and again using a "row/column" approach. The first equation in (4) is equivalent to performing a "column" FFT and the second equation is equivalent to performing a "row" FFT. In between there is a "twiddle factor" multiplication by the W^p in W_M (3). Because both the row and column FFTs in the secondary factorization (4) are broken down entirely into sets of 4-point DFTs, they can be done without complex multiplications. Also, the usual matrix transpose in between column and row DFTs is not necessary.

A systolic array architecture mapping was performed using the mathematical formulation (4) as input to the mapping tool SPADE [2]. Behavioral simulations of this architecture using a register transfer level simulator verify its operation. Performance estimates of the FFT computation times are shown in Table 1 for a variety of transform sizes.

Size (points)	T (cycles/DFT)	T (μsec/DFT)	Multipliers	Adders
256	210	1.0	4	32
512	274	1.3	8	64
1024	658	3.1	8	64
2048	914	4.3	16	128
4096	2322	10.8	16	128
8192	3346	15.6	32	256

Table 1. Performance estimates and arithmetic requirements for various transform sizes (16-bits fixed point) based on a partially populated Altera Stratix EP1S60 "medium speed grade" FPGA chip. In this table "T" is the throughput. (Computational latency for each transform size above is approximately equal to the inverse of the throughput time/DFT).

Although Table 1 only shows transforms that are powers of two, the base-4 FFT lengths are not limited to powers of two or four. For example, the base-4 FFT is capable of 29 transform lengths from 256 to 65,536 *vs*. only 5 possible lengths for a radix-4 pipelined FFT.

The single biggest drawback to past use of systolic arrays has been the substantial arithmetic hardware that is normally required because systolic approaches use a number of complex multipliers equal to the size of the transform. Thus, a 1024-point DFT would require 1024 complex multipliers, compared to the 8 multipliers shown in Table 1 for the base-4 FFT.

Traditional "pipelined" FFTs, although computationally efficient, are difficult to map into VLSI because in general each butterfly, delay/commutator, and twiddle factor ROM has a different circuit design and/or its operation varies from stage to stage. Also, the butterflies do not usually work with 100% resource efficiency, the designs are limited to transform lengths that are powers of two or four, 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 large granularity. Finally, the latency (time to do the first DFT in a series) is low because the pipeline has to be "filled" first. Alternatively, the base-4 FFT architecture is comprised of simple, identical, small processing elements (PEs), arranged in regular arrays with each PE operating at near 100% efficiency. Performance figures in Table 1 compare very well to larger custom ASIC designs and recent FPGA implementations.

[1] C. C. W. Hui, T. J. Ding, J. V. McCanny, and R. F. Woods, "A New FFT Architecture and Chip Design for Motion Compensation based on Phase Correlation," Proc. Int. Conf. on Application Specific Systems, Architectures and Processors (ASAP 96), pp. 83-92.

[2] J. Greg Nash, "Hardware Efficient Base-4 Systolic Architecture for Computing the Discrete Fourier Transform," Proc. 2002 IEEE Workshop on Signal Processing Systems (SIPS'02), pp. 87-92.