Computing the Discrete Fourier Transform on FPGA Based Systolic Arrays

Chris Dick
School of Electronic Engineering
La Trobe University
Melbourne 3083, Australia

Abstract

Reconfigurable logic arrays allow for the creation on the one physical hardware platform many different virtual circuits. A configuration bit-stream loaded into the logic array specifies the virtual circuit implemented. This paper addresses the problem of implementing FFTs using virtual computers based on Xilinx FPGAs. A systolic array processor architecture consisting of processing elements (PEs) employing CORDIC arithmetic is presented. The CORDIC approach removes the requirement for area consuming multipliers in the design. The method is suitable for handling power-of-2 and non power-of-2 transform lengths. The modular nature of the design provides for a highly scalable architecture that provides the system designer with a flexible mechanism for making cost-performance tradeoffs. The array processor and PE architecture are described. Based on simulation results, FPGA device utilization and transform execution time are calculated.

1 Introduction

Reconfigurable logic arrays allow for the creation on the one physical hardware platform many different virtual circuits. A configuration bit-stream loaded into the logic array specifies the virtual circuit implemented. Reconfigurable custom computing machines (CCM) based on field programmable gate arrays (FPGAs), have been constructed that allow the implementation of demanding circuits that cannot be accommodated in a single logic cell array (LCA).

One area in which CCMs have been applied is digital signal processing (DSP). Implementation of digital filters has been discussed by Mintzer in [1] and Dick and harris in [2].

A less explored topic in the DSP arena is the applicability of CCMs for computing discrete Fourier transforms (DFTs). Shirazi et al in [3] describe the implementation of the 2-D DFT on Splash 2. The method uses standard row-column processing. First, 1-D transforms are performed on each row of data, followed by transforms on each column of the row-transformed data. The 1-D fast Fourier transform (FFT) algorithm employed is the decimation-in-time Cooley-Tukey radix-2 algorithm [4]. A custom 18-bit floating point number format is used in the implementation. A single butterfly module and address generators occupy 13 Xilinx XC4010 FPGAs of Splash 2. A 1-D FFT is computed by time division multiplexing the butterfly processor.

In this paper, an FPGA based systolic array processor architecture is described for computing 1-D DFTs. To avoid the requirement for area consuming multipliers, CORDIC (coordinate rotation digital computer) arithmetic is used to implement the systolic array processing elements. To the author's knowledge, this approach to computing DFTs using FPGA custom computing machines has not been reported in the open literature.

The paper is organized as follows. First, the application of CORDIC methods to computing the DFT is described. An overview of the CORDIC computing technique is then given. Next, the FFT algorithm utilized, and the systolic array processor architecture are presented. The architecture and a detailed description of the system PEs is then discussed. Finally, the performance of the systolic array processor is presented.

2 CORDIC Arithmetic Based FFT Processor

The DFT $X(k),\ k = 0, \ldots, N-1$ of an N point vector of numbers $x(n),\ n = 0, \ldots, N-1$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n)\omega^{nk} \qquad k = 0, \ldots, N-1 \tag{1}$$

where $\omega = e^{-j2\pi/N}$ is the Nth complex root of unity. In general, the input data set is complex valued. Conventionally, the procedure for computing a single DFT coefficient is viewed as forming the sum of a sequence of complex products. Usually the complex products are computed using a multiplier and adder that operate on real valued data. Another way of viewing the problem is to consider the calculation of one output value, as the vector sum of rotations of the input complex tuples. The rotation angles being integer multiples of the Nth root of unity. When the computation is viewed from this perspective, a DFT algorithm based on the CORDIC method first described by Volder in [5] can be developed. This approach avoids the requirement for a full multiplier. Multipliers consume a large amount of FPGA logic resources, and so the approach would seem advantageous for implementation on FPGA based CCMs.

The CORDIC based approach for computing the DFT in the context of very-large-scale-integration (VLSI) and wafer-scale-integration (WSI) technologies, has been considered by several authors including Despain in [6] and [7] and Jones in [8] and [9].

2.1 CORDIC Arithmetic

Computation of the DFT requires the rotation of complex numbers through integer multiples of the Nth root of unity. Consider the problem of rotating a complex datum $x + jy$ ($j = \sqrt{-1}$) through an angle θ to produce a result q. The direct method of performing the rotation is to compute

$$\begin{aligned} q = (x + jy) \times e^{j\theta} &= x\cos(\theta) - y\sin(\theta) \\ &+ j\left(y\cos(\theta) + x\sin(\theta)\right) \end{aligned} \tag{2}$$

using either 4 real multiplies and 2 real additions, or 3 real multiplies and 3 real additions. Full multipliers are expensive functional units to implement using FPGA technology. An alternative method that avoids the requirement for a full multiplier for performing the rotations is to use CORDIC arithmetic. Based on the description by Despain in [6], the calculation of Eq. (2) can be computed to an accuracy of 1 bit in L bits by using the following iterative procedure. First perform the initialization

$$i = 0 \tag{3}$$
$$z_i = -\theta \tag{4}$$

Then, execute the following procedure L times

$$a_i = \operatorname{sgn}(z_i) \tag{5}$$
$$x_{i+1} = x_i + a_i y_i 2^{-i} \tag{6}$$
$$y_{i+1} = y_i - a_i x_i 2^{-i} \tag{7}$$
$$z_{i+1} = z_i - a_i \tan^{-1}\left(2^{-i}\right) \tag{8}$$
$$i = i + 1$$

where $\operatorname{sgn}(\cdot)$ is the *sign* operator. The initial values of x and y are the real and imaginary components of the complex number that is to be rotated.

The method causes a magnification of the rotated vector's length by a factor K defined as

$$K = \prod_{i=0}^{L-1} \sqrt{1 + 2^{-2i}} \tag{9}$$

For $L > 12$, K is approximately 1.6. What this means in the context of the DFT, is that the output spectrum will be scaled by the constant factor K. In many situations the scaling introduced is not considered a problem. More complex CORDIC iterative procedures can be employed that avoid the magnification, but the resultant hardware is more costly than for the basic *uncompensated* CORDIC procedure described above. The uncompensated algorithm forms the basis of the PEs in the array processor described in section 4 of this paper.

3 Systolic Array FFT Processor

In this section an FPGA computing architecture similar to the FFT array processor described by Jones in [8] is presented.

The FFT algorithm employed is the Cooley-Tukey algorithm [4]. The procedure is to map the 1-D input data set into a 2-D array, and use a pseudo 2-D transform to compute the 1-D DFT. Assume that the transform length N is composite and can be expressed as

$$N = N_1 \times N_2 \tag{10}$$

where N_1 and N_2 are integers. Following the index mappings in [4], an N point transform can be expressed as

$$X(k_1,k_2) = \sum_{n_1}^{N_1-1} \sum_{n_2}^{N_2-1} x(n_1,n_2)\omega_{N_2}^{n_2k_2}\omega_N^{n_2k_1}\omega_{N_1}^{n_1k_1}$$

$$\begin{aligned} k_1 &= 0,\ldots,N_1-1 \\ k_2 &= 0,\ldots,N_2-1 \end{aligned} \qquad (11)$$

The procedure defined by Eq. (11) is to first perform N_1 length N_2 DFTs. These DFTs are the transforms of the rows of the re-ordered input data set and are defined as

$$F(n_1,k_2) = \sum_{n_2=0}^{N_2-1} x(n_1,n_2)\omega_{N_2}^{n_2k_2} \qquad \begin{aligned} n_1 &= 0,\ldots,N_1-1 \\ k_2 &= 0,\ldots,N_2-1 \end{aligned} \qquad (12)$$

Next, each element of the intermediate matrix $F(n_1,k_2)$ $n_1 = 0,\ldots,N_1-1$, $k_2 = 0,\ldots,N_2-1$ is adjusted by the complex factor $\omega_N^{n_2k_1}$, the so-called twiddle factors. Finally, a second set of DFTs is performed on each column of the twiddle factor adjusted matrix. The final 1-D result is obtained by unloading the 2-D matrix $X(k_1,k_2)$ using the appropriate index mapping.

The required computation steps can be performed by the systolic array architecture shown in Figure (1).

Assume that the transform length can be expressed as an even power of 2 so that $N_1 = N_2 = n$. The 1-D input sequence is first mapped into a 2-D array of dimension $n \times n$ and placed in the input buffer memory. The systolic array architecture consists of two n-by-n arrays of processing elements. The first n-by-n processor array is referred to as the Row-Processor (RP). The second PE array is the Column-Processor (CP). Each row in the input data buffer is directed into the corresponding row of PEs in the RP by a 2 wire complex serial bus. Each PE in the RP computes a single DFT coefficient of the row transforms defined by Eq. (12). The row busses in Figure (1) broadcast the same data to all PEs in the same row. Each PE is a bit-serial processor, so each bus connecting all PEs in the same row carries a serial bit stream of complex data samples. Each PE, in parallel with all other PEs in the RP, computes one DFT value of the row transforms. The next step is to apply the twiddle factors to the partial transform result. Rather than explicitly performing this operation, the rotations performed by the twiddle factors are combined with the column transforms that are computed by the CP array. Thus, no separate hardware resource or time is required to perform the adjustment by the twiddle factors. The Column-Processor array does not perform a direct DFT on the complex data entering each column of processors in the CP, but instead computes a *modified* transform that is the result of combining the adjustment by the twiddle-factors, and the column transforms into a single operation. The CP array operates on the matrix $F(\cdot)$ to produce the final output result according to

$$X(k_1,k_2) = \sum_{n_1=0}^{N_1-1} F(n_1,k_1)e^{-j2\pi n_1\left(\frac{k_1}{N_1}+\frac{k_2}{N}\right)}$$

$$\begin{aligned} n_1 &= 0,\ldots,N_1-1 \\ k_2 &= 0,\ldots,N_2-1 \end{aligned} \qquad (13)$$

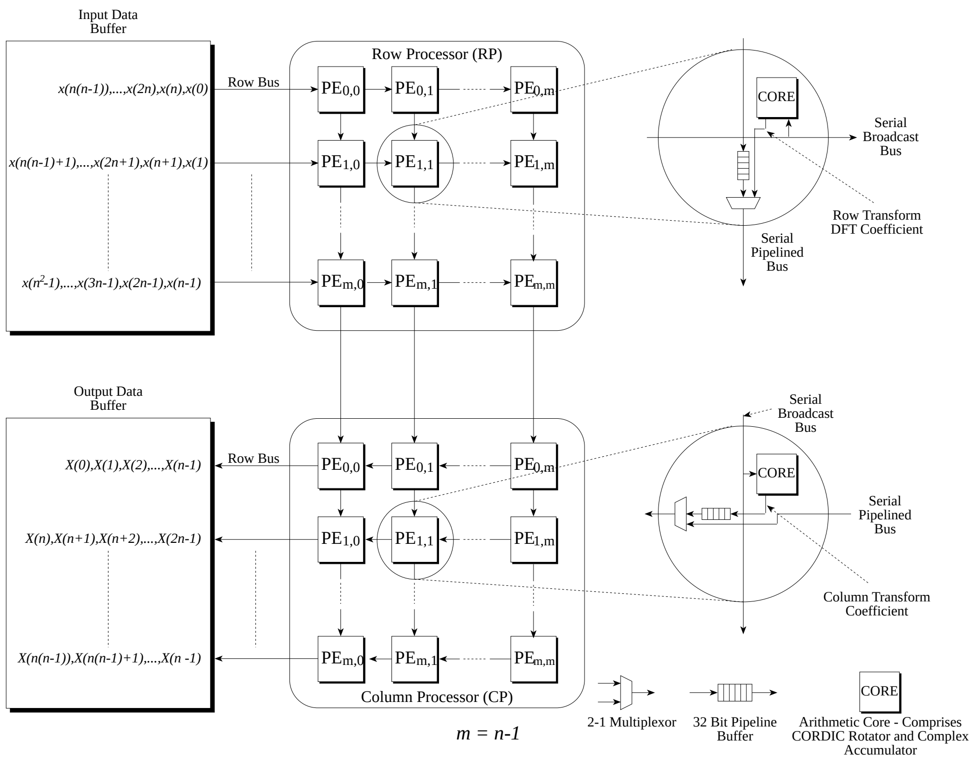
A pipelined complex serial bus interconnects each column of PEs in the RP array. The pipe stages are implemented using configurable logic block (CLB) [10] memory. Two CLBs are required for each stage of the pipeline. The output of each column of PEs in the row-processor broadcasts data to a column of PEs in the CP array. Each column-aligned linear array of PEs in the Column-Processor computes one of the modified transforms defined in Eq. (13). Each PE computes a single value of Eq. (13). A horizontal pipelined serial bus connects each row of processors in the CP array. This bus passes the DFT coefficients from a PE, through other PEs in the row, and finally through to the output buffer memory.

The systolic architecture described allows for full overlap of the row transforms with the column transforms. The time complexity of the algorithm is n time-steps, where a time-step is the amount of time required to compute a CORDIC rotation.

4 Processing Element Architecture

The architecture of the systolic array PE is shown in Figure (2).

The major functional unit of a PE is the CORDIC vector rotation unit. To minimize the hardware requirements, the CORDIC method selected for this work was the conventional uncompensated CORDIC algorithm defined above. The range of convergence for this algorithm is -1.7433 to 1.7433 radians [8]. The DFT requires rotations of the input data through angles in the range $-\pi$ to $+\pi$. The algorithm can be modified to extend the range of convergence of the basic method to the range of angles required in the DFT calculation, however, an extra 2 iterations of the CORDIC algorithm are required [8]. With these additional iterations the magnification factor increases to a value of approximately 3.2 per rotation [8]. This increase in magnification factor could potentially result



Input Data Buffer
x(n(n-1)),...,x(2n),x(n),x(0)
x(n(n-1)+1),...,x(2n+1),x(n+1),x(1)
x(n²-1),...,x(3n-1),x(2n-1),x(n-1)
Row Bus
Row Processor (RP)
PE0,0
PE0,1
PE0,m
PE1,0
PE1,1
PE1,m
PEm,0
PEm,1
PEm,m
CORE
Serial Broadcast Bus
Row Transform DFT Coefficient
Serial Pipelined Bus
Output Data Buffer
X(0),X(1),X(2),...,X(n-1)
X(n),X(n+1),X(n+2),...,X(2n-1)
X(n(n-1)),X(n(n-1)+1),...,X(n -1)
Row Bus
Column Processor (CP)
Serial Broadcast Bus
Serial Pipelined Bus
Column Transform Coefficient
2-1 Multiplexor
32 Bit Pipeline Buffer
Arithmetic Core - Comprises CORDIC Rotator and Complex Accumulator
m = n-1

Figure 1: Systolic array architecture for computing the DFT. The external CORDIC rotator control memory is not shown in this figure.

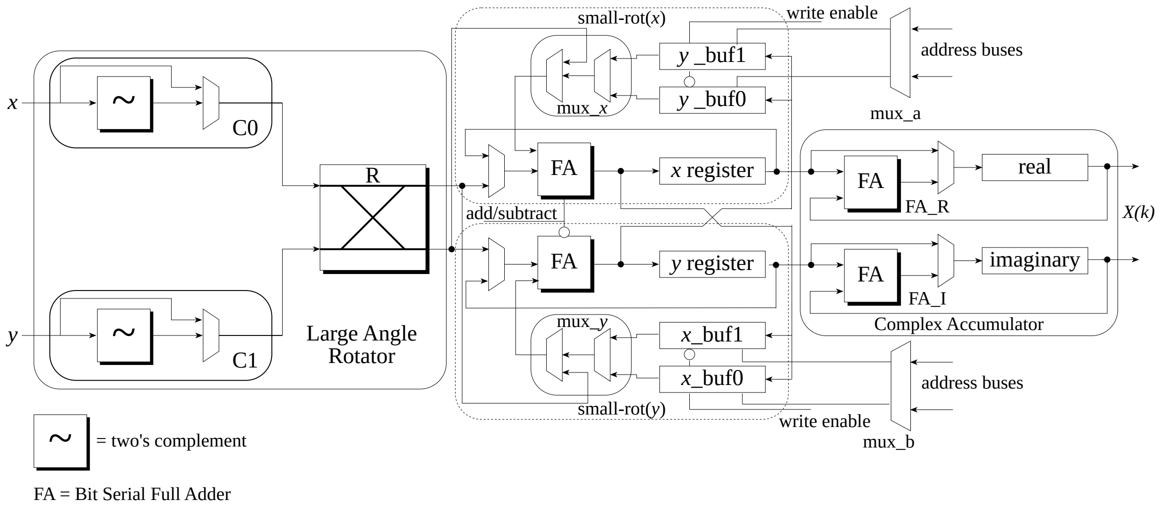


Figure 2: Architecture of the processing element arithmetic core.

in more stringent scaling strategies being necessary for a fixed-point implementation than would be required by the uncompensated CORDIC algorithm.

To achieve the rotations through the set of angles required by the DFT, a two stage rotation process consisting of a large angle rotator followed by a CORDIC based small angle rotator was adopted. The large angle rotator handles rotations through angles in the set $\Theta = \{0, \frac{\pi}{2}, -\pi, \frac{-\pi}{2}\}$. Whilst the small angle rotator implements rotations through angles in the range $\frac{-\pi}{2}$ to $\frac{\pi}{2}$. A rotation through an angle θ is achieved by expressing θ as the sum of a large angle Φ and a small angle ϕ

$$\theta = \Phi + \phi \tag{14}$$

where $\Phi \in \Theta$ and $\frac{-\pi}{2} \leq \phi \leq \frac{\pi}{2}$. The large angle rotation is easily achieved by the appropriate combination of negation and transposing of the input data. Referring to Figure (2), the large angle rotator consists of the two processing units labeled $C0$ and $C1$, which either pass the input data, or the two's complement of the input data to their outputs. The data router R passes its input data, or the exchange of the data to its outputs. Each two's complementer is implemented in 1 CLB. The data router is also implemented in 1 CLB. Three control signals are required to configure the large angle rotator. Each control signal is generated by a ROM control store implemented by configuring a CLB as memory. The total CLB count for the large angle rotator is 6.

There are two small angle rotators, one for computing each of the iterations defined by Eq. (6) and Eq. (7). These units are labeled small-rot(x) and small-rot(y) in Figure (2). Consider performing the iteration Eq. (6). The register x, implemented with CLB RAM, currently holds the value x_i. The update of x is performed in a bit-serial fashion. The register pair y_buf0 and y_buf1 form a *ping-pong* buffer. One of the registers in the ping-pong buffer is used for the update of x, whilst the second register is simultaneously being updated with the new value of y being computed by the functional unit small-rot(y). The bit-shifting operation required in the update procedure is implemented by appropriate addressing of the ping-pong buffer register. For example, to generate the value $2^{-i}y_i$ used in the ith CORDIC iteration, the ping-pong buffer address sequencer starts its access at bit address i of the ping-pong buffer register. The value $2^{-i}y_i$ is sign-extended to the correct word length by continuously addressing the most significant bit of y_i for the appropriate number of cycles.

In each iteration of the algorithm $2^{-i}y_i$ is either added to, or subtracted from x_i to form x_{i+1}. This process is controlled by Eq. (8). There is no need for the on-line calculation of the term $\tan^{-1}(2^{-i})$ in Eq. (8). The $z-$ values are computed off-line and stored in system RAM. Each PE computes one DFT coefficient. In computing one coefficient of a $N = 2^p$ transform, where p is even, $\sqrt{N}$ angles are used by each PE. Each CORDIC rotation is performed in L cycles. Therefore, $\sqrt{N} \times L$ bits of information are required to control the small-angle rotators. For a value of $N = 1024$ and $L = 16$, this would mean $32 \times 16 = 512$ bits of control information for the small angle rotators of each PE. Whilst this control information could be stored internal to the LCA using CLBs configured as memory, the number of CLBs required is considered to be too large. For the example just cited, 16 CLBs would be required to store the 512 bits of small-angle rotator control information. The solution is to store this information in memory external to the LCA. This does not present a problem in terms of external memory bandwidth. Using current FPGA technology like the Xilinx XC4010PG191-4, 10 PEs can be supported by the one device. Therefore only a $512 \times 8 = 4096$ bit memory is required to store all the control information for all of the PEs implemented in the one LCA. In addition, only a single connection from memory need go to each PE, so any potential LCA routing problems are avoided.

Address generation for the external control store is performed by a 9 bit counter resident in the same LCA as the PEs. The address generator occupies 5 CLBs.

The hardware for the update of the imaginary component, y, of the PE input data is essentially a duplicate of that used for updating the real component of the input data. However, only one large angle rotator and one external control store is required by each PE.

The CLB count for all of the components in the CORDIC rotator is shown in Table (1).

In addition to the CORDIC rotation unit, a complex accumulator is required to form the sum of the rotated input data, so forming the result for one DFT coefficient. The real and imaginary parts of the accumulator are each kept to a precision of 32 bits. The accumulator update is performed using two bit-serial full adders (FAs) labeled FA_R and FA_I in Figure (2). FA_R updates the real part of the running sum, whilst FA_I updates the imaginary part of the DFT value. Each FA is implemented in 1 CLB. The complex accumulator occupies 4 CLBs.

The total CLB count for one PE is given by summing the number of CLBs required for the CORDIC unit, which is 38, to the 4 CLBs required for the complex accumulator, plus an additional 2 CLBs for the

Component	CLB Count
large angle rotator	6
x register	1
y_buf0	1
y_buf1	1
mux_x	2
mux_a	2.5
mux_b	2.5
full adder x	1
y register	1
x_buf0	1
x_buf1	1
mux_y	2
full adder y	1
Control/Address Generators	15
Total:	38

Table 1: CLB breakdown for each hardware component of the CORDIC rotator.

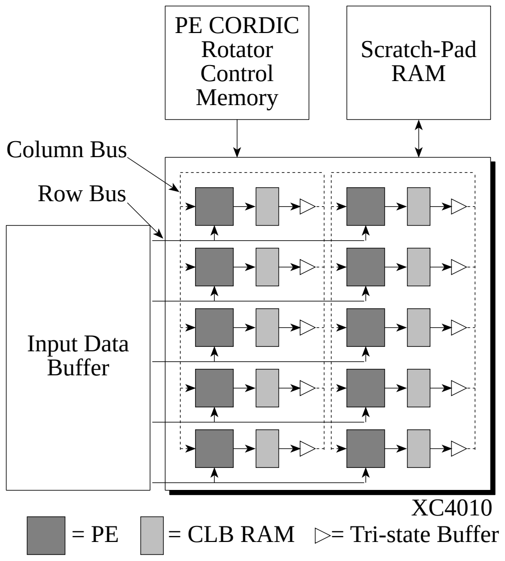


Figure 3: System architecture for computing large transforms using a 2×5 PE array implemented in one XC4010 device.

pipelined serial bus in each RP and CP. This gives a total of 44 CLBs. The control/address generators for each PE can be shared. If this is done, then each PE can be implemented in 29 Xilinx 4000 series CLBs.

5 Performance

The DFT method outlined above describes a general technique for computing transforms. The Cooley-Tukey decomposition strategy can be applied iteratively to the row and column processor arrays. In addition, not all of the PEs in Figure (1) need be physical processors. At the expense of execution time, physical PEs can be time-division-multiplexed to compute the required transforms. Consider the problem of computing a 1000 point transform. First factorize the transform length as $1000 = 10 \times 100$. With this factorization a 1000 point transform can be computed by performing 10 100-point DFTs followed by 100 10-point DFTs. The phase-factor adjustment that is applied between the row and column transforms is not explicitly mentioned in this discussion because it is incorporated as part of the column processing procedure. The 100 point transforms are done as 20 10-point DFTs. Each 10 point DFT is computed as 5 2-point transforms followed by 2 5-point DFTs. A 2×5 PE array can be implemented in one 4010 device as shown in Figure (3). The 10-point transforms are implemented by first computing 5 2-point transforms on the PE array in Figure (3). Five serial streams of data are read from the input buffer into the 5 row buses of the PE array. These transforms are computed in parallel. The results from this calculation are stored in CLB RAM. Next, 2 5-point transforms are computed on the results from the just completed 2-point transforms. Data is input to the PEs by sequentially enabling the tri-state buffers connected to the Column Buses in the processing array of Figure (3). The 2 5-point transforms are computed in parallel, and the results written to the memory labeled "Scratch-Pad RAM". The one 2×5 PE array is effectively being time-division-multiplexed between the RP and CP shown in Figure (1). The Scratch-Pad RAM is used for temporary storage of intermediate results in the calculation.

Let T_N be the execution time for an N-point DFT. The execution time T_{10} for a 10-point transform is

$$T_{10} = T_2 + 5t_{\mathrm{r}} \tag{15}$$

where t_{r} is the time required to perform a CORDIC rotation. For a 100-point transform

$$T_{100} = 20T_{10} \tag{16}$$

The total time to compute a 1000-point transform is

$$\begin{aligned} T_{1000} &= 10T_{100} + 100T_{10} \\ &= 300T_{10} \\ &= 300(T_2 + 5t_{\mathrm{r}}) \end{aligned} \tag{17}$$

The time to compute a CORDIC rotation is $t_r = BL/f$ where B is the number of bits of precision used in the CORDIC rotator, L is the number of iterations used for each CORDIC rotation, and f is the system clock frequency. The 2-point transforms are trivial, and computed without the application of the CORDIC rotator. The execution time for the 2-point transforms is $T_2 = 2B/f$, which is simply the time to perform a bit-serial addition of 2 B-bit numbers.

Using the Xilinx FPGA timing analyzer *xdelay*, the system clock rate using an XC4010PG191-4 FPGA was determined to be 15.3 MHz. Values of $B = 32$, and $L = 16$ were used in the design. The transform execution time for $N = 1000$ as a function of number of LCAs is presented in Table (2).

Execution Time (ms)	Number of XC4010 LCAs
51.45	1
25.73	2
10.29	5
5.15	10

Table 2: Execution time for a 1000-point transform as a function of the number of LCAs.

6 Conclusion

Systolic array architectures have the critical advantages of modularity, regularity, local interconnection, and highly pipelined multiprocessing. These properties give rise to highly scalable architectures. DFT processors using a either a small number or large number of FPGAs can be constructed using a systolic approach. This gives the system architect an easy method for selecting an operating-point in cost-performance space. In addition, the approach easily lends itself to the implementation of power-of-2 or non power-of-2 DFTs. The implementation of a 1000-point DFT was described in the paper. On one Xilinx 4010 FPGA this transform is executed in 51.45 ms. On a 10 FPGA system the execution time reduces to 5.15 ms. The DFT example in the paper is for a non power-of-2 transform length. Following a similar factoring strategy as was used for the 1000-point transform, power-of-2 transforms can be computed. Implementing the PEs using CORDIC arithmetic removed the requirements for area consuming multipliers in the design.

References

[1] L. Mintzer, "FIR Filters with Field Programmable Gate Arrays", *Journal of VLSI Signal Processing*, No. 6, pp. 120-127, 1993.

[2] C. H. Dick, and f. harris, "FPGA Implementation of High Order FIR Filters by Re-quantizing the Input Data Stream", *Proc. of SPIE Conf. Photonics East '95, Field Programmable Gate Arrays (FPGAs) for Fast Board Development and Reconfigurable Computing*, Philadelphia Pennsylvania, USA, pp. 10-21, Oct. 25-26, 1995.

[3] N. Shirazi, P. M. Athanas, and A. L. Abott, "Implementation of a 2-D Fast Fourier Transform on a FPGA-Based Custom Computing Machine", To appear in *Proceedings of the Fifth International Workshop on Field Programmable Logic and Applications*, University of Oxford, UK, Aug. 1995.

[4] J. W. Cooley, and J. W. Tukey, "An Algorithm for the Machine Computation of Complex Fourier Series", *Mathematics of Computation*, vol. 19, pp. 297-301, Apr. 1965.

[5] J. E. Volder, "The CORDIC Trigonometric Computing Technique", *IRE Trans. on Electronic Computers*, vol. 8 no. 3, pp. 330-334, 1959.

[6] A. M. Despain, "Fourier Transform Computers Using CORDIC Iterations", *IEEE Trans. on Computers*, vol. c-23, no. 10, pp. 993-1001, Oct. 1993.

[7] A. M. Despain, "Very Fast Fourier Transform Algorithms Hardware for Implementation", *IEEE Trans. on Computers*, vol. c-28, no. 5, May 1979.

[8] K. J. Jones, "Bit-Serial CORDIC DFT Computation with Multidimensional Systolic Processor Arrays", *IEEE Journal of Oceanic Engineering*, vol. 18 no. 4, Oct. 1993.

[9] K. J. Jones, "Parallel DFT Computation on Bit-Serial Systolic Processor Arrays", *IEE Proc.*, vol. 140, no. 1, pp. 10-18, Jan. 1993.

[10] Xilinx, Inc., *The Programmable Logic Data Book*, 1994.