The 8th International Conference on Signal Processing Applications and Technology, Toronto Canada, September 13-16 1998.

Computing Multidimensional DFTs Using Xilinx FPGAs

Chris Dick
chrisd@xilinx.com

Xilinx Inc.
2100 Logic Drive
San Jose
CA 95124

Abstract: This paper reports on a reconfigurable computing architecture that takes advantage of the reduced computational requirements of the polynomial transform method for computing 2-D DFTs. An FPGA architecture is described that is capable of processing 24 512×512-pixel images per second. The proposed system is 46% more area efficient than a row-column DFT processor implemented using the same technology.

1 Introduction

One class of reduced computational complexity algorithms for computing multidimensional FFTs and convolutions are the polynomial transform (PT) algorithms first proposed by Nussbaumer and Quandelle in [1]. These algorithms offer considerable savings in terms of the number of operations required to compute 2-D FFTs. However, they have found little application in practice for several reasons. One factor is probably the higher degree of specialized knowledge required to efficiently code the algorithm in comparison to something like the row-column algorithm. Another factor is due to algorithmic overheads, such as modulo reductions and data re-ordering that are not efficiently implemented by VLSI digital signal processors (DSPs).

Technological advancements in field programmable gate arrays (FPGAs) over the past 10 years have opened new paths for digital signal processing design engineers. The FPGA maintains the advantages of the high specificity of the ASIC while avoiding the high development costs and inability to make design modifications after production. The FPGA also brings design flexibility and adaptability to a signal processing architecture. FPGA-based reconfigurable computers are emerging as a class of computers that can provide near application specific computational performance. Designers can also configure them for a variety of tasks. Such platforms let designers customize specific operations for function and size, and dynamically construct data paths for individual applications. An advantage of this is that the cost of the hardware can be amortized over a wide range of applications.

In this paper a datapath is designed for computing 2-D FFTs based on polynomial transforms in modified rings of polynomials [2]. The design implementation and performance using Xilinx [3] FPGAs is described. The advantages of the implementation in terms of area efficiency and performance in comparison to standard row-column processing are highlighted.

2 Computation of the 2-D Fourier Transform Using Polynomial Transforms

Nussbaumer in [2] describes a technique for computing power-of-2 DFTs based on roots of unity in fields of polynomials. A structurally simpler technique based on polynomial transforms defined in modified rings of polynomials is also described. Although this latter method is not the most computationally efficient of the polynomial transform based methods for computing DFTs, its operation count is lower than that of the conventional row-column approach often used for computing 2-D transforms. The combination of structural simplicity and reduced operations count make the algorithm well suited for FPGA implementation. The algorithm is summarized in Figure 1.

The 2-D transform is computed with N^2 premultiplications by ω^{-n_2} ($\omega = e^{-j\pi/N}$), one polynomial transform of length N and kernel z^2, N reduced DFTs of N terms and output permutations.

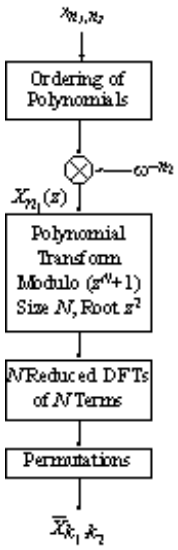

Figure 1: Computing an $N \leftrightarrow N$ DFT using polynomial transforms defined in modified rings of polynomials. $N = 2^t$. $\omega = e^{-j\pi/N}$.

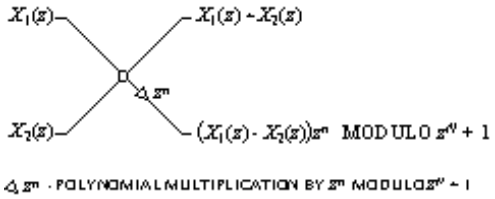

Figure 2: Radix-2 polynomial transform decimation-in-frequency butterfly.

2.1 The Fast Polynomial Transform Algorithm

The polynomial transform is implemented with a minimum number of additions using a radix-2 decimation-in-time (DIT) or decimation-in-frequency (DIF) FFT-type algorithm. This algorithm is referred to as a *fast polynomial transform (FPT)* algorithm. The signal flow graph for a DIF polynomial transform butterfly is shown in Figure 2. Instead of multiplications by powers of roots of unity as used in the radix-2 Cooley-Tukey FFT algorithm, the polynomial transform butterflies use multiplications modulo $z^{N+1}+1$ by powers of z. These amount to simple polynomial rotations followed by sign inversion of the overflow words, and are therefore implemented without any multiplications.

Each polynomial in the FPT calculation consists of N terms. Each polynomial coefficient is complex valued because of the complex input premultiplications. The calculation in each arm of the butterfly therefore requires $2N$ additions. There are $N/2$ butterflies in each of $\log_2 N$ processing ranks giving the total number of additions as $2N\log_2 N$.

2.2 Input Premultiplications

Assuming real-valued input data, for example an image frame, each premultiplication is the product of a real-number with a complex number and requires 2 real multiplications and no additions. Fast multipliers, such as array multipliers are expensive functional units to implement using FPGAs. The design used here is a radix-2 Booth recoded serial-parallel multiplier [4]. For a B-bit multiplicand and multiplier, B clock cycles are required to form a two's complement product. Only the most significant B-bits of the $2B$-bit result are retained. A $16 \leftrightarrow 16$-bit multiplier occupies 40 4000 series configurable logic blocks (CLBs) [3], while an $18 \leftrightarrow 18$-bit unit occupies 43 CLBs. Not all of the occupied CLB flip-flops are used. When the multiplier is incorporated into a larger design, data registers can be merged into the unused locations to give better device utilisation. The 2 real multiplications needed for each premultiplication are implemented with one time-division multiplexed multiplier. The N^2 premultiplications are executed in

$$T_{PM} = \frac{B2N^2}{f}$$

seconds, where f is the system clock frequency.

2.3 Polynomial Transform

The architecture of a polynomial transform butterfly is shown in Figure 3.

The design is pipelined so that data fetches, stores and calculations are overlapped. A sum or difference of the complex input data is computed in 2 clock cycles - the real and imaginary parts being handled separately.

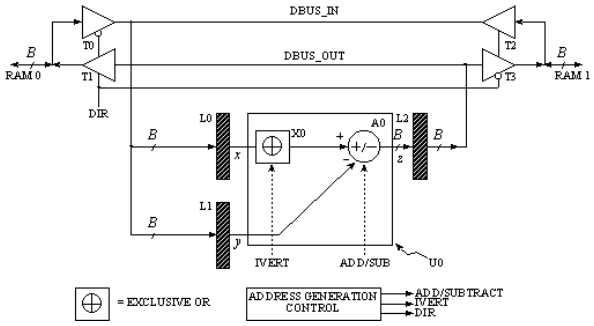

Figure 3: Radix-2 polynomial transform butterfly.

The execution time T_{PT} for the polynomial transform is

$$T_{PT} = \frac{2N^2 \log_2 N}{f} \quad \text{seconds}$$

The polynomial transform arithmetic core occupies 27 4000 series CLBs. In addition to the processing core a complete polynomial transform engine requires a memory address generator which occupies another 40 CLBs.

2.4 FFT Processor

The final stage of the calculation requires *N N*-point reduced DFTs. These DFTs can be calculated using a radix-2 Cooley-Tukey FFT partitioning with modified phase factors. The algorithm used is based on the DIT radix-2 butterfly shown in the bottom right-hand corner of Figure 4. The multiplication by the complex phase factor, ω_N^k, is implemented using 4 real multiplications and 2 real additions. Two complex additions are required to form the final butterfly output. Each complex addition is implemented with 2 real additions. A schematic of the butterfly circuit implemented using a Xilinx FPGA is shown in Figure 4. The real and imaginary components of a complex datum are each kept to a precision of *B*-bits. The butterfly processor reads a complex datum in one clock cycle, i.e., 2*B*-bit words are read from memory on each read access. An

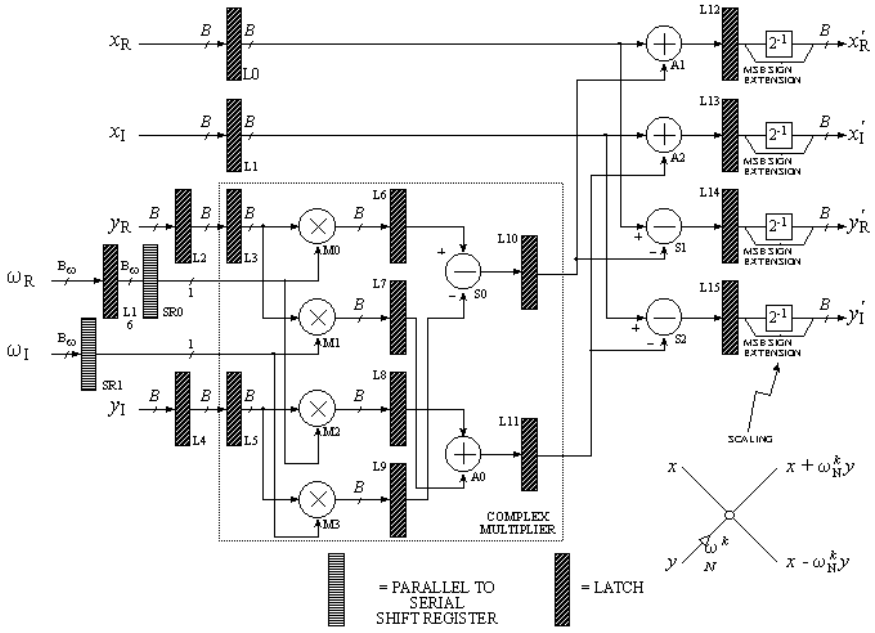

Figure 4: Block diagram of the Radix-2 DIT butterfly used in the FPGA FFT processor.

18-bit fractional fixed-point representation is used

for each of the real and imaginary components of a complex number. This requires a 36-bit wide datapath to memory. The phase factors, ω_N^k, are kept to a precision of 16-bits ($B_\omega = 16$). The complex product engine for computing $y\omega_N^k$ is based on the Booth recoded multiplier outlined earlier. Four multipliers, M0,M1,M2 and M3, operating in parallel are used for the complex multiplication. The subtractor S0 and the adder A0 are used to combine the multiplier outputs to form the final product. Each multiplier produces a result in 16 clock cycles. The butterfly design is pipelined to allow uninterrupted processing. Data fetches, stores, the complex multiplication, as well as the sum $(x + y\omega_N^k)$ and difference $(x - y\omega_N^k)$ calculations are overlapped. One complete butterfly is completed every 16 clock cycles. The pipelined processing offsets the time-penalty of the slow serial-parallel multipliers used in the design. The time to perform 1 DFT is $B\omega N/2\log_2 N / f$ seconds. The execution time T_{FFT} for all N FFTs is

$$T_{FFT} = B_\omega \frac{N^2 \log_2 N}{2f}$$

CLBs. As highlighted earlier, there are unused flip-flops in the multiplier layout. A dense placement was achieved by merging butterfly pipeline registers into these locations. For example, the registers L2 and L3 in Figure 4 are merged into the multiplier M0. The basic multiplier design has a latched output, so register L6 is already included in the 43 CLB figure quoted earlier for implementing an 18 × 18-bit multiplier. So even though a multiplier by itself occupies 43 CLBs, registers L2, L3, the multiplier M0 and the latch L6 can all be accommodated in 43 CLBs if the layout is carefully controlled.

In addition to the butterfly processor, data input, output and a phase factor memory address generators are required. In the current system a change of transform size requires re-loading the FPGA. This was done to minimize the overheads associated with a fully programmable address generator. The area that the address units occupy vary with the transform size. For a 512-point transform, 134 CLBs are required for the address generators. So a complete 512-point FFT processor occupies 428 4000 series CLBs.

3 System Architecture

An architecture that exploits parallelism of the polynomial transform DFT algorithm will maximize the system throughput. The parallel processing

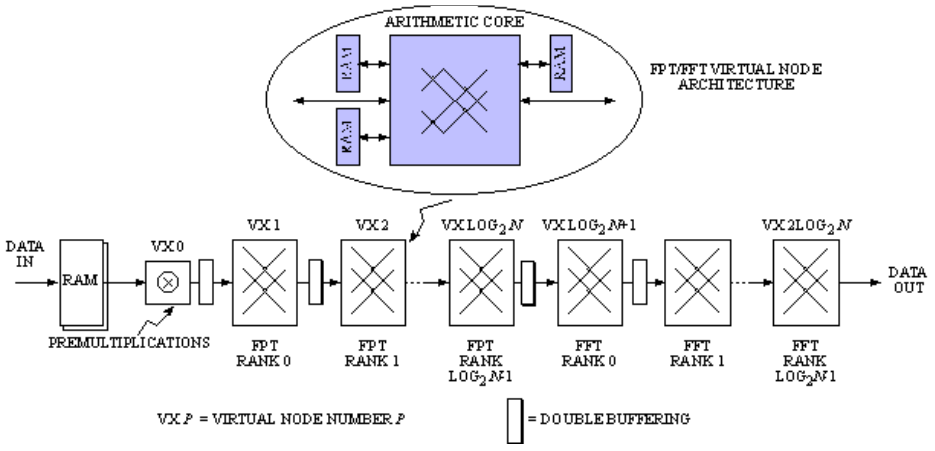

Figure 5: Multi-FPGA architecture for computing 2-D DFTs based on polynomial transforms. An FPGA virtual processor is used for each processing stage of the FPT and FFT. The input premultiplications are also allocated to a separate virtual processor.

The butterfly unit occupies 294 Xilinx 4000 series

butterfly functional units. Unique FPT and FFT butterfly processors are used for each rank of the FPT and FFT processes respectively. The 2-D transform architecture is shown in Figure 5.

The algorithm is partitioned and mapped onto a set of *virtual processors (VP)*. There does not have to be a one-to-one mapping between virtual processors and physical processors. Multiple virtual processors can be accommodated in a single FPGA or a single virtual processor can be partitioned across several physical processors. For the partitioning shown in Figure 5 virtual processor VX 0 performs the input premultiplications, while virtual processors VX 1 to VX $\log_2 N$ perform the FPT butterfly ranks 0,…,$\log_1 N$-1 respectively. Each of these VP's implement one polynomial butterfly that is time division multiplexed to compute the $N/2$ polynomial butterflies in a processing rank of the FPT. Similarly, virtual processors VX $\log_2 N + 1$ to VX $2\log_1 N$ each compute one of the $\log_2 N$ butterfly ranks of the FFT algorithm. Each of these virtual processors is a radix-2 FFT butterfly that is time division multiplexed over the $N/2$ butterflies in a single processing stage of the FFT. Closer observation reveals that a physical FPT butterfly processor can be time division multiplexed over multiple virtual FPT butterfly processors. Three physical FPT butterfly units are sufficient for this design.

The CLB count is $C_{\mathrm{PTFFT}} = C_{\mathrm{PM}} + 3C_{PT} + C_{FFT}\log_2 N$, where C_{PM}, C_{PT} and C_{FFT} are the CLB counts for the premultiplication stage multiplier, PT butterfly and FFT butterfly respectively. For a parallelized row-column technique to achieve the same processing throughput as the parallel PT approach, separate row and column processors are required. Each of these would consist of an FFT butterfly module for each of the $2\log_2 N$ processing ranks. The CLB count is $C_{\mathrm{RC}} = 2C_{\mathrm{FFT}}\log_2 N$. For $N = 512$ $C_{\mathrm{PTFFT}} = 4123$ and $C_{\mathrm{RC}} = 7704$. The two approaches produce the same transform throughput but the parallel PT architecture uses 54% of the logic resources of the parallel row-column implementation. In terms of the number of FPGAs, the logic requirements for the PT architecture is equivalent to approximately four Xilinx 25,000 gate 4025 devices. The system computation rate is quite high. With some layout optimizations and using faster 2 ns FPGAs, the clock frequency is estimated to be 50 MHz. With this frequency and $N = 512$, a new 2-D transform is generated every 42 milliseconds. This corresponds to a frame rate of 24 frames per second. The arithmetic performance is 405 million operations per second.

4 Conclusion

FPGA based machines offer the DSP system designer a new means of implementing data paths that are highly optimized for a particular algorithm. This approach maintains the high-performance advantages of dedicated hardware based solutions, but with the flexibility of software.

There are several ways to use such machines. The FPGA machine may be the final target hardware, be used as a rapid prototyping system, or in an environment where a single piece of hardware is required to perform different processing tasks at different times. By combining knowledge of both fast algorithms and computer arithmetic, these machines can exploit the reduced computational requirements of procedures whose computational advantage may not be able to be exploited in software solutions. The polynomial transform method is a case in-point. Implemented on a DSP processor, the computational advantages are sacrificed due to increased control overhead in implementing the modulo arithmetic and the data permutations. These aspects are not an issue when the algorithm is implemented using FPGAs.

Finally, it is useful to place the proposed FPGA PT architecture in context with a DSP processor like the Texas Instruments TMS320C30 (C30). The rating of one functional unit for a 40 MHz device is 20 MFLOPS. To achieve the same transform execution time as the multi-FPGA architecture, $10N^2\log_2 N$ real operations need to be performed every 42ms for $N = 512$. Even without any concurrent inefficiencies, a parallel VLSI DSP solution would require approximately 30 C30 processors.

[1] H. J. Nussbaumer and P. Quandelle, "Computation of Convolutions and Discrete Fourier Transforms by Polynomial Transforms", *IBM J. Res. Develop.*, vol. 22 no. 2, pp. 134-144, Mar. 1978.

[2] H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*, Springer-Verlag, New York, 1981.

[3] Xilinx Inc., *The Programmable Logic Data Book*, 1998.

[4] Kai Hwang, *Computer Arithmetic Principles Architecture and Design*, John Wiley & Sons.,New York, 1979.