

The Fast Fourier Transform on a Reconfigurable Processor

Gregory W. Donohoe

Institute of Advanced Microelectronics
801 University, SE, Suite 206
Albuquerque, NM 87106

John Purviance

3827 Glenhurst St.
Colorado Springs, CO 80906

Pen-Shu Yeh

NASA GSFC Code 564
Greenbelt, MD 20771-0001

Abstract – This paper describes the implementation of the Fast Fourier Transform (FFT) on the Reconfigurable Data Path Processor (RDPP) under development by the Institute of Advanced Microelectronics and NASA Goddard Space Flight Center. The RDPP implements a multi-stage computational pipeline, optimized for systolic signal processing applications. We implement the Goertzel FFT algorithm, which is able to exploit the data path parallelism of the RDPP. The paper introduces the RDPP and the Goertzel algorithm, describes a Fourier Transform Hyperspectral Image data conversion application, and discusses scaling issues.

I. THE RECONFIGURABLE DATA PATH PROCESSOR

The emerging field of Reconfigurable Computing seeks to achieve the performance of dedicated hardware with the flexibility of software through architectures that can be reconfigured at run time to optimize them for different computing tasks [5]. The Reconfigurable Data Path Processor (RDPP) is a reconfigurable multiprocessing chip being developed for high-throughput, data-intensive processing tasks aboard spacecraft. The RDPP contains 16 reconfigurable processing elements, each with a multiplier, an arithmetic/logic unit, data path formatting logic, and data path selection logic, operating on a 24-bit-wide data path. In addition, there are five 24-bit bi-directional I/O ports. The RDPP implements a synchronous data flow computational model, optimized for data-intensive processing tasks such as signal processing [1]. The RDPP operates in two phases: (1) configuration, and (2) execution. In configuration, the processing elements are programmed for specific processing tasks, programmable interconnects are configured to form a

processing pipeline, and a run-time program is loaded. In the execution phase, the RDPP reads and processes data from an input stream and writes to an output stream.

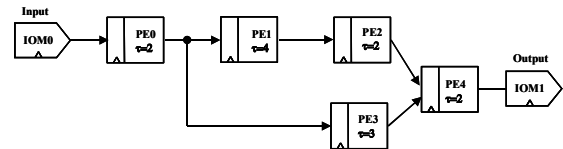


Fig. 1: Example RDPP pipeline.

Fig. 1 illustrates a data pipeline implemented with five processing elements (PEs), an input port, and an output port. τ represents the processing delay through a PE. This example has a processing fork at PE1, and a join at PE4. Both the top and bottom forks execute simultaneously. PE4 performs conditional data path selection on a clock-cycle basis, selecting the output of either PE2 or PE3, as appropriate, to pass on to the output port.

The data flow model is very efficient for many signal processing tasks, such as finite impulse response (FIR) filters, which accept data sequentially, but internally are inherently parallel, and so are naturally suited to a systolic pipeline. The Fast Fourier Transform does not have these properties, and implementing it in a pipelined data path processor presents a special set of challenges.

II. THE FAST FOURIER TRANSFORM

The Discrete Fourier Transform (DFT) is widely used in instrument data processing. The ability to execute the DFT quickly in an embedded system would greatly enhance its usefulness in spacecraft data processing. In its direct form, the Discrete Fourier Transform of a data sequence of length N is computed by:

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn} \quad (1)$$

where n is a time sample, k is a frequency bin, and $W_N = e^{-j2\pi/N}$, $n=0,1,\dots,N-1$, $k=0, 1,\dots,N-1$. $\{x[n]\}$ is the set of input data samples, and $\{X[k]\}$ is the set of complex-valued Fourier coefficient, i.e., the transform. Direct computation of the DFT on a sequence of length N requires $O(N^2)$ complex multiplies and additions. Each complex multiply requires four real multiplies and three additions; a complex multiply-accumulate (MAC) is about four times as costly as a real-valued MAC.

The famous Cooley-Tukey Fast Fourier Transform (FFT) algorithm employs a divide-and-conquer approach to computing the DFT. It minimizes the number of multiply-accumulate operations, enabling the DFT to be computed with $O(N \log N)$ complex MACs. The Cooley-Tukey algorithm computes the FFT on an entire block of data; the number of data points must be a composite number, i.e., $N = 2^m$, where m is an integer. (If the number of samples is not a composite number, it must be “padded”, or extended to a composite number.) This approach dramatically reduces the number of arithmetic operations, and is admirably suited to execution on random-access, sequential machines including Digital Signal Processors (DSPs). It requires complex data shuffling, however, and the entire sequence must be available at once, which makes it difficult to parallelize. Fig. 2 shows a data flow diagram for the Cooley-Tukey algorithm.

III. The Goertzel Algorithm

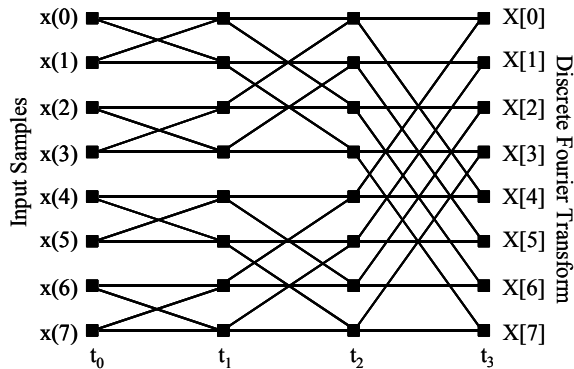


Fig. 2: Flow diagram for Cooley-Tukey FFT. t_i are time steps.

Another approach to computing the FFT is the Goertzel algorithm, which offers some advantages for computation on a pipelined machine. The Goertzel algorithm formulates the DFT as a recursive filter, computing the Fourier coefficient in the individual frequency bins independently. It does not require

simultaneous access to the entire signal array, but can accept the input signal samples sequentially.

Table 1 summarizes the important differences between the two approaches.

Cooley-Tukey	Goertzel
$O(N \log N)$ complex multiply-accumulates.	$O(N^2)$ real multiply-accumulates.
Block computation.	Point computation.
N must be a composite number ($N=2^m$).	N is arbitrary.
Butterfly network.	Small recursive loop.
Complex, multi-step data dependencies.	Coefficients computed independently.
Difficult to parallelize.	Easy to parallelize.

To develop the Goertzel algorithm, we observe that $W_N^{-kN} = 1$. With this, we can rewrite (1) as

$$X[k] = \sum_{\ell=0}^{N-1} x[\ell]W_N^{-k(N-\ell)} \quad (2)$$

This has the form of a convolution of the input sequence with the sequence W_N^{-kn} , $n \geq 0$, which defines a digital filter with the transfer function:

$$H_k(z) = \frac{1}{1 - W_N^{-k} z^{-1}} \quad (3)$$

which is easily implemented as a one-pole recursive filter with complex coefficients. This requires N complex multiply-accumulates, or $4N$ real MACs, for each point in the transform that we compute. To compute the complete the transform, we need N^2 complex multiply-accumulates.

We can replace the complex MACs with real-valued operations by replacing the complex-valued one-pole filter with a two-pole recursive filter with real coefficients, one of which is -1 . We define an intermediate variable $v_k[n]$:

$$v_k[n] = x[n] + 2 \cos(2\pi k / n) v_k[n-1] - v_k[n-2] \quad (4)$$

for $0 \leq n \leq N-1$. If we require a complex-valued result, then we compute the final result with a single complex multiply and a subtraction after running the real-valued filter for N iterations:

$$X[k] = v_k[N] - W_N^k v_k[N-1] \quad (5)$$

If, on the other hand, we only need the magnitude of the transform, then we merely stop after N iterations of the filter, and $|X[k]| = v_k[N] - v_k[N-1]$. Fig. 3 below

shows the data flow diagram for the Goertzel algorithm. A more detailed development is found in [2].

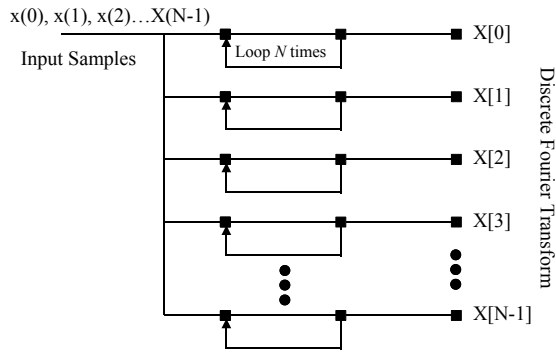


Fig. 3: Goertzel DFT data flow.

IV. THE FOURIER TRANSFORM HYPERSPECTRAL IMAGER EXAMPLE

We now examine the Goertzel algorithm running the RDPP for a Fourier Transform Hyperspectral Imager (FTHSI). The system used in this study is the Hypercam airborne imager developed by Kestrel Corporation. The specifications are essentially identical to those flown in the FTHSI aboard the Air Force Mighty Sat II.1 satellite [3,4]. This is a push-broom system which images a single line through a Sagnac interferometer. The interferometer converts a one-dimensional line of data to a two-dimensional image: the horizontal dimension represents the spatial extent of the image (pixels), and the vertical dimension is the Fourier transform of the spectral intensity. The image is captured by a focal plane array and digitized. In order to form a spectral image, we must take the magnitude of the Inverse Discrete Fourier Transform (IDFT) of each column. (The computation of Inverse DFT is identical to the forward transform, except that W_N^k in (1) is replaced by W_N^{-k}).

The system records a single interferogram in 512 pixels. Before taking the IDFT, the interferogram is appodized and cropped to 256 pixels. By a well-known property of the IDFT of a real-valued sequence, the resulting spectrum is symmetrical, with the right half a mirror image of the left half. Thus, only 128 pixels of actual transform data are required.

The RDPP requires two processing elements (PEs) to compute one Fourier magnitude coefficient using the Goertzel algorithm. Since the RDPP has 16 PEs, we can compute up to 8 Fourier coefficients in parallel, on one pass through the data. All of the PE pairs require the same data input, so only one of the RDPP's 5 bi-directional I/O ports is required for input data.

The procedure compute the Goertzel FFT with a single RDPP is:

1. *Configuration:* configure 8 recursive filters. Load each of the filters with the filter coefficient, $2\cos(\pi k/64)$, corresponding to this frequency bin, k .
2. *Execution:* Feed the input sequence into the RDPP, running the RDPP 128 times.
3. *Termination.* Halt the RDPP and read out each of the 8 results.

We repeat this process 15 more times, each time loading a different set of filter coefficients for the new values of k . The processing speed of an RDPP PE is not yet known. However, if we assume that one iteration through the filter requires 50 ns, then we can compute 8 FFT samples in 6.4 μ sec, plus configuration and read-out time, or an entire sequence of 128 samples in about 200 milliseconds. If high speed is essential, we can assemble a bank of up to 16 RDPP chips in parallel, and compute the entire 128-sample FFT in tens of microseconds.

Recall that the RDPP implements a 24-bit-wide, fixed-point data path, with data-path-formatting elements incorporated into each processing element. The Kestrel FTHSI digitizes the signal to only 12 bits, but for this analysis we will assume the more challenging case of 16 bit data. To compute each Discrete Fourier component, we need to run the filter 128 times. We scale the filter coefficients, $2\cos(2\pi k/N)$, to be less than 1; this guarantees the stability of our filter. We choose a format so that there is 1 sign bit and 16 fraction bits, leaving 7 integer bits, which are all zero.

Each 24-bit multiply produces a 48-bit product, which is then scaled back to 24 bits. Each addition has the potential to generate a carry out of the fraction field, and add 1 to the integer field. In the worst case, after processing 128 samples, the integer field can hold a value of up to 127, which fits in the 7 bit integer field without overflow. Fig. 4 shows the structure of the filter loop for the FTHSI application.

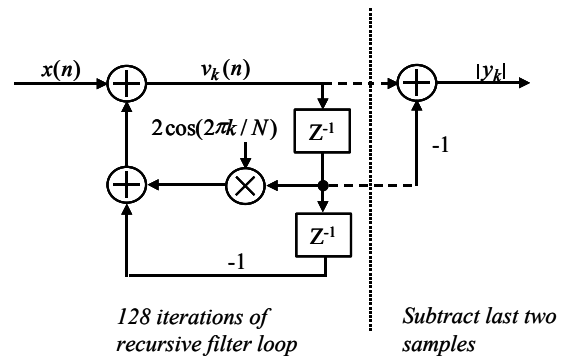


Fig. 4: Filter structure for the FTHSI.

V. CONCLUSION

Reconfigurable computing based on reconfigurable systolic pipelines can significantly speed up many signal processing applications. The familiar Cooley-Tukey form of the FFT requires only $O(N \log N)$ complex multiply-accumulate operations, but because of its block-structured nature and intricate interconnect patterns, does not map well onto a systolic array. The Goertzel FFT algorithm can be formulated as a bank of recursive filters, which map neatly onto a data flow architecture, requiring $O(N^2)$ real-valued multiply-accumulates.

Implemented on the 24-bit Reconfigurable Data Path Processor, Goertzel algorithm will enable Fourier Transform Hyperspectral Imager data conversion on an instrument such as the Kestrel Hypercam, processing a 16-bit data stream without numerical error due to overflow. By ganging 16 RDPP processor chips in parallel, the FTHSI data conversion can be completed in a few tens of microseconds.

REFERENCES

- [1] G.W. Donohoe and P.-S. Yeh, "A low power reconfigurable processor", *Proc. IEEE Aerospace Conference*, Big Sky, MT, March 9-16, 2002.
- [2] [MITRA98] Mitra, Sanjit K., *Digital Signal Processing: a Computer-based Approach*, McGraw-Hill, 1998.
- [3] Otten, L. John III, Eugene W. Butler, J. Bruce Rafert, R. Glenn Sellar, "The design of an airborne Fourier transform visible hyperspectral imaging system for light aircraft environmental remote sensing", *Imaging Spectrometry*, SPIE Vol. 2480, April 1995.
- [4] Otten, L. John III, Andrew D. Meigs, R. Glenn Sellar, "Calibration and performance of the airborne Fourier transform visible hyperspectral imager (FTHSI)", *Proc. Second International Airborne Remote Sensing Conference and Exhibition*, San Francisco, CA, 24-27 June, 1996.
- [5] J. Villaseñor and W.H. Mangione-Smith, "Configurable Computing", *Scientific American*, June, 1997.