

Preface

Most real-world spectrum analysis problems involve the computation of the real-data discrete Fourier transform (DFT), a *unitary* transform that maps elements of the linear space of real-valued N -tuples, \mathbf{R}^N , to elements of its complex-valued counterpart, \mathbf{C}^N , and when carried out in hardware it is conventionally achieved via a *real-from-complex* strategy using a complex-data version of the fast Fourier transform (FFT), the generic name given to the class of fast algorithms used for the efficient computation of the DFT. Such algorithms are typically derived by exploiting the property of *symmetry*, whether it exists just in the transform kernel or, in certain circumstances, in the input data and/or output data as well. In order to make effective use of a complex-data FFT, however, via the chosen real-from-complex strategy, the input data to the DFT must first be converted from elements of \mathbf{R}^N to elements of \mathbf{C}^N .

The reason for choosing the computational domain of real-data problems such as this to be \mathbf{C}^N , rather than \mathbf{R}^N , is due in part to the fact that computing equipment manufacturers have invested so heavily in producing digital signal processing (DSP) devices built around the design of the complex-data fast multiplier and accumulator (MAC), an arithmetic unit ideally suited to the implementation of the complex-data radix-2 *butterfly*, the computational unit used by the familiar class of recursive radix-2 FFT algorithms. *The net result is that the problem of the real-data DFT is effectively being modified so as to match an existing complex-data solution rather than a solution being sought that matches the actual problem.* The increasingly powerful field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC) technologies are now giving DSP design engineers far greater control, however, over the type of algorithm that may be used in the building of high-performance DSP systems, so that more appropriate algorithmically-specialized hardware solutions to the real-data DFT may be actively sought and exploited to some advantage with these technologies.

The first part of this monograph thus concerns itself with the design of a new and highly-*parallel* formulation of the fast Hartley transform (FHT) which is to be used, in turn, for the efficient computation of the DFT. The FHT is the generic name given to the class of fast algorithms used for the efficient computation of the discrete Hartley transform (DHT) – a unitary (and, in fact, *orthogonal*) transform and close relative of the DFT possessing many of the same properties – which,

for the processing of real-valued data, has attractions over the complex-data FFT in terms of reduced arithmetic complexity and reduced memory requirement. It's *bilateral* or reversal property also means that it may be straightforwardly applied to the transformation from *Hartley space* to *data space* as well as from *data space* to *Hartley space*, making it thus equally applicable to the computation of both the DFT and its inverse. A drawback, however, of conventional FHT algorithms lies in the loss of *regularity* (as relates to the algorithm structure) arising from the need for two sizes – and thus two separate designs – of butterfly for efficient fixed-radix formulations, where the regularity equates to the amount of *repetition* and *symmetry* present in the design. A generic version of the double butterfly, referred to as the “GD-BFLY” for economy of words, is therefore developed for the radix-4 FHT that overcomes the problem in an elegant fashion. The resulting single-design solution, referred to as the *regularized radix-4 FHT* and abbreviated to “ R_4^2 FHT”, lends itself naturally to *parallelization* and to mapping onto a regular computational structure for implementation with parallel computing technology.

A *partitioned-memory architecture* for the parallel computation of the GD-BFLY and the resulting R_4^2 FHT is next developed and discussed in some detail, this exploiting a single *locally-pipelined* high-performance processing element (PE) that yields an attractive solution, particularly when implemented with parallel computing technology, that is both *area-efficient* and *scalable* in terms of transform length. High performance is achieved by having the PE able to process the input/output data sets to the GD-BFLY in parallel, this in turn implying the need to be able to access simultaneously, and without conflict, both multiple data and multiple twiddle factors, or trigonometric coefficients, from their respective memories.

A number of pipelined versions of the PE are described using both fast fixed-point multipliers and phase rotators – where the phase rotation operation is carried out in optimal fashion with hardware-efficient Co-Ordinate Rotation DIgital Computer (CORDIC) arithmetic – which enable arithmetic complexity to be traded off against memory requirement. The result is a set of scalable designs based upon the partitioned-memory single-PE computing architecture which each yield a hardware-efficient solution with universal application, such that each new application necessitates minimal re-design cost, as well as solutions amenable to efficient implementation with the silicon-based technologies. The resulting area-efficient and scalable single-PE architecture is shown to yield solutions to the real-data radix-4 FFT that are capable of achieving the *computational density* – that is, the throughput per unit area of silicon – of the most advanced commercially-available complex-data solutions for just a fraction of the silicon resources.

Consideration is given to the fact that when producing electronic equipment, whether for commercial or military use, great emphasis is inevitably placed upon minimizing the unit cost so that one is seldom blessed with the option of using the latest state-of-the-art device technology. The most common situation encountered is one where the expectation is to use the smallest (and thus the least expensive) device that is capable of yielding solutions able to meet the performance objectives, which often means using devices that are one, two or even three generations behind the latest specification. As a result, there are situations where there would be great merit

in having designs that are not totally reliant on the availability of the increasingly large quantities of expensive embedded resources, such as fast multipliers and fast memory, as provided by the manufacturers of the latest silicon-based devices, but are sufficiently flexible to lend themselves to implementation in silicon even when constrained by the limited availability of embedded resources.

The designs are thus required to be able to cater for a range of resource-constrained environments where the particular resources being *consumed* and traded off, one against another, include the *programmable logic*, the *power* and the *time* (update time or latency), as well as the *embedded resources* already discussed. The choice of which particular FPGA device to use throughout the monograph for comparative analysis of the various designs is not considered to be of relevance to the results obtained as the intention is that the attractions of the solutions developed should be valid regardless of the specific device onto which they are mapped – that is, a “good” design should be *device-independent*. The author is well aware, however, that the intellectual investment made in achieving such a design may seem to fly in the face of current *wisdom* whereby the need for good engineering design and practice is avoided through the adoption of ever more powerful (and power consuming) computing devices – no apologies offered.

The monograph, which is based on the fruits of 3 years of applied industrial research in the U.K., is aimed at both practicing DSP engineers with an interest in the efficient hardware implementation of the real-data FFT and academics /researchers/students from engineering, computer science and mathematics backgrounds with an interest in the design and implementation of sequential and parallel FFT algorithms. It is intended to provide the reader with the tools necessary to both understand the new formulation and to implement simple design variations that offer clear implementational advantages, both theoretical and practical, over more conventional complex-data solutions to the problem. The highly-parallel formulation of the real-data FFT described in the monograph will be shown to lead to scalable and device-independent solutions to the latency-constrained version of the problem which are able to optimize the use of the available silicon resources, and thus to maximize the achievable computational density, thereby making the solution a genuine advance in the design and implementation of high-performance parallel FFT algorithms.

L-3 Communications TRL Technology,
Shannon Way, Ashchurch, Tewkesbury,
Gloucestershire, GL20 8ND, U.K.

Dr. Keith Jones

The Regularized Fast Hartley Transform
Optimal Formulation of Real-Data Fast Fourier
Transform for Silicon-Based Implementation in
Resource-Constrained Environments

Jones, K.J.

2010, XVII, 200 p. With online files/update., Hardcover

ISBN: 978-90-481-3916-3