Multirate Filters and Wavelets: From Theory to Implementation

Robert D. Turney, Chris Dick[1], and Ali M. Reza[2]

Xilinx Inc.
2100 Logic Dr.
San Jose, CA 95124, USA

ABSTRACT

Multirate signal processing is an enabling technology that brings DSP techniques to applications requiring low-cost and high sample rates. Field programmable gate arrays (FPGAs) provide system-level hardware solutions for signal processing architects demanding both high-performance and flexibility. This workshop provides an introduction to the fundamental theory of multirate filter techniques and provides simple descriptions of polyphase decimators and interpolators. Wavelet theory has development roots in both mathematics and signal processing communities. In this workshop we will explain wavelet theory from both a signal expansion and filter bank viewpoint. Classes of signal processing problems that can benefit from wavelet techniques will be presented with simple examples. Implementation of mulitrate filters and wavelet structures with FPGAs will be explained for real time applications.

INTRODUCTION

Digital signal processing (DSP) systems offer degrees of freedom to the system designer that are unparalleled in the analog signal processing domain. One of the most important of these parameters is the availability of a system sample clock. In the context of digital filters, the sample clock is accessed and exploited using multirate filter techniques. Judicious and creative use of multirate processes in digital signal processing systems allow a designer to realize a hardware efficient solution, that in many instances just would not be feasible using single rate signal processing. This paper provides an introductory level overview of multirate filters. The basic theory of multirate techniques is presented along with an explanation of polyphase interpolators and decimators. The hardware realization of multirate systems using field programmable gate arrays (FPGAs) is also examined.

[1] Robert D. Turney (robert.turney@xilinx.com) and Chris Dick (chris.dick@xilinx.com) are Senior Systems Engineers in the CORE Solutions Group, Xilinx Inc., San Jose CA, USA

[2] Ali M. Reza (reza@uwm.edu) is an Associate Professor in the Department of Electrical Engineering and Computer Science at the University of Wisconsin – Milwaukee, Milwaukee WI, U.S.A

Wavelet theory had been developed independently on several fronts. Different signal processing techniques, developed for signal and image processing applications, had significant contribution in this development [1]. Some of the major contributors to this theory are: multiresolution signal processing [2], used in computer vision; subband coding, developed for speech and image compression; and wavelet series expansion, developed in applied mathematics [3]. The wavelet transform is successfully applied to non-stationary signals and images. Some of the application areas are: nonlinear filtering or denoising, signal and image compression, speech coding, seismic and geological signal processing, medical and biomedical signal and image processing, and communication.

MULTIRATE FILTERS AND REALIZATIONS

Filters, analog or digital, are one of the most widely used signal processing functions. In the analog domain, a filter transfer function is realized by a suitable arrangement of resistors, capacitors, operational-amplifiers (op-amps) and possibly inductors. As with any analog platform, this implementation is exposed to the potential problems that result from temperature dynamics, and component tolerances. The motivation to migrate this type of processing to an all (or mostly) digital implementation involves eliminating these issues, coupled with the system economics, manufacturability and repeatability issues. One might be tempted to start with an analog prototype design and *digitize* this system to produce a DSP implementation. In fact, this is precisely the worst approach to adopt. In making a digital clone of an analog system all of the unique features of the DSP domain are not exploited. These features include adjusting the sample rate at various nodes in a system, together with the creative use of deliberate aliasing. Exploiting these degrees of freedom in a digital design is of paramount importance in order to obtain the maximum performance with a minimum amount of digital hardware. Indeed, it may even be the difference between being able to perform a required function at a specified sample rate at all using a DSP implementation. One of the most important techniques that a DSP designer has at their disposal to exploit the time dimension of a problem is multirate filtering.

In their most basic form, multirate filters are used to decrease (decimation) or increase the sample rate (interpolation) of a stream of samples. Multirate filters are employed in many diverse applications and for many different reasons. One broad motivating factor is to match the sample rate to the signal bandwidth at all, or most, of the nodes in a system or to alter the sample rate so that signals with differing sample rates may be combined in some manner. Ultimately, multirate filters minimize the arithmetic workload required to perform a specified calculation. In turn, this minimizes the number of clock cycles consumed in a software based implementation, the number of gates in an ASSP (application specific standard part) realization, or the number of logic elements consumed in an FPGA design.

Decimation

Decimation can be useful if the sample rate of a signal is considerably greater than twice the signal's bandwidth. The process of *sample rate decimation,* or simply *decimation*,

involves two steps as shown in Figure 1. The first stage is a bandwidth limiting operation (anti-aliasing filter) while the second step performs the down-sampling.

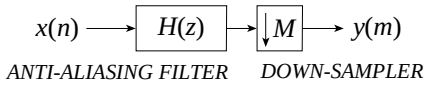

Figure 1: Signal decimation – bandwidth reduction followed by down-sampling.

The anti-aliasing filter is a lowpass design. This filter must reduce the signal bandwidth in accordance with the Nyquist sampling theorem. The filtered signal's bandwidth B must satisfy $B \le f_s^{'}$, $f_s^{'} = f_s / M$, where $f_s^{'}$ is the decimated data stream output sample rate, f_s is the input sample rate and M is the decimation factor. The implicit assumption employed here is that the band of frequencies $B \le f \le f_s / 2$ contains no useful information and may be safely discarded in the application. The bandwidth limiting process is often constrained to preserve phase linearity and so a FIR filter, as shown in Figure 2, is employed.

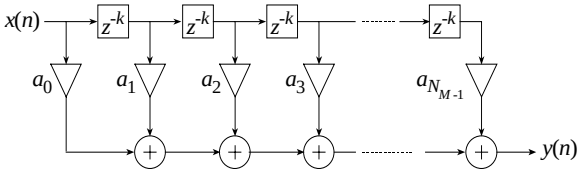

Figure 2: Direct form FIR filter architecture.

The elemental process needed to compute the filter as a *multiply-accumulate (MAC)* operation. The coefficients $a_0, \ldots, a_{N-1}$ control the filter frequency response. The computation workload is a function of the filter *quality*, and the filter quality is reflected by the number of filter taps in the structure.

To establish a context for a discussion on efficient decimating filter stuctures, consider the following design problem.

Example 1

A communication system employs frequency division multiplexing (FDM) to allow multi-user access to the system channel. A decimating filter is to be designed that recovers the baseband channel and adjusts the system sample rate to match the filter bandwidth. The input sample rate is 100 MHz, the passband channel bandwidth is 2 MHz, adjacent channels must be suppressed by at least 50 dB and the maximum allowable passband ripple is 0.5 dB. The decimation factor is 50. The filter must have a linear phase response.

Solution

The signal frequency spectrum together with an overlay of the baseband filter is shown in Figure 3.

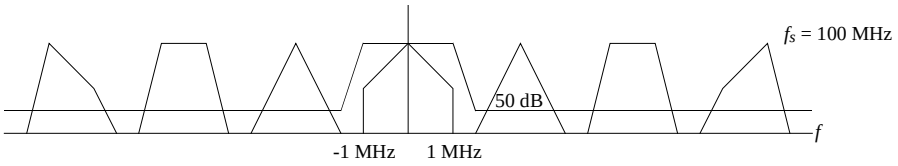

Figure 3: Input signal spectrum and baseband filter specifications.

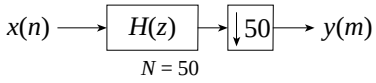

Figure 4: Decimator consisting of an anti-aliasing filter and down-sampler.

Some design iterations in a filter design environment indicate that a 205-tap filter will meet the requirements. Using the *MAC* as the basic unit of work, a simple calculation shows that a real-time data path to support the 205-tap filter must supply 205 Mega-MACs (MMACs) of processing performance. While this is achievable using several hardware approaches, including a DSP processor or FPGA, it is definitely not the most efficient realization. We should examine the simple filter-and-decimate structure with a view to removing any redundancy so that the implementation cost can be minimized. On examining, there is a glaring inefficiency. The filter produces 1 output for every sample presented at its input, and yet only 1 in 50 of these samples survives the down-sampling operation. There must be a method that avoids computing the discarded output samples altogether – this is the function of a *polyphase* realization of the decimator.

Polyphase Decimators

Polyphase filters, both decimators and interpolators, are efficient structures for simultaneously and efficiently filtering and decimating (or interpolating) a signal stream. Some relationships that will be useful in the development of polyphase structures are presented in Figure 5. These figures state certain rules that may be applied to manipulate a dataflow graph. The fundamental set of relationships that are employed to manipulate a multirate signal flowgraph are the *Noble Identities* [5] in Figure 6. These relationships are very powerful tools for performing various types of signal flowgraph transformations. Space requirements do not permit a detailed explanation of the Noble Identities, and the reader is referred to [5] for a more complete treatment.

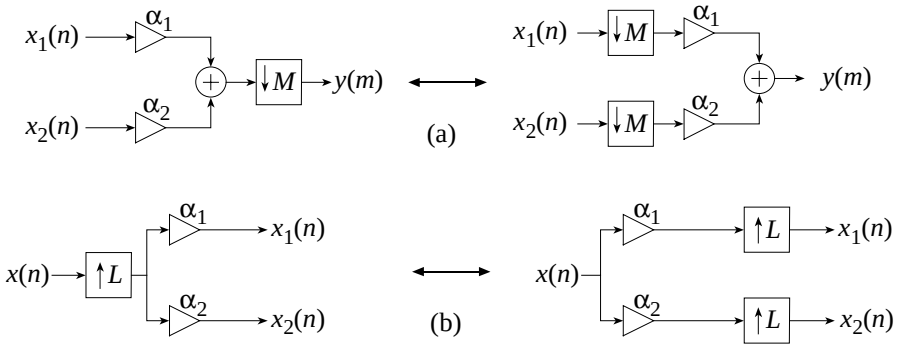

Figure 5: Simple identities for interconnected systems.

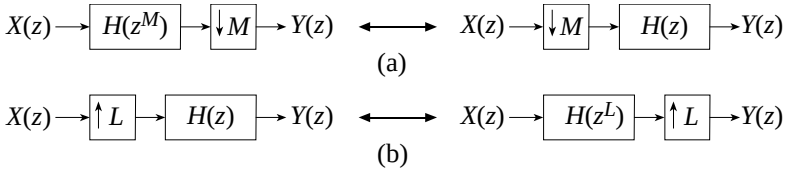

Figure 6: The Noble Identities.

The polyphase decimator structure will be introduced via a simple example. Consider the decimator in Figure 1 with $M = 4$. The transfer function $H(z)$ can be expressed as

$$\begin{aligned} H(z) &= \sum_{n=0}^{N-1} h(n) z^{-n} \\ &= \sum_{n=0}^{(N-1)/4} h(4n) z^{-4n} + \sum_{n=0}^{(N-1)/4} h(4n+1) z^{-(4n+1)} + \sum_{n=0}^{(N-1)/4} h(4n+2) z^{-(4n+2)} + \sum_{n=0}^{(N-1)/4} h(4n+3) z^{-(4n+3)} \\ &= \sum_{n=0}^{(N-1)/4} h(3n) z^{-4n} + z^{-1} \sum_{n=0}^{(N-1)/4} h(4n) z^{-4n} + z^{-2} \sum_{n=0}^{(N-1)/4} h(4n) z^{-4n} + z^{-3} \sum_{n=0}^{(N-1)/4} h(4n) z^{-4n} \\ &= H_0(z) + z^{-1} H_1(z) + z^{-2} H_2(z) + z^{-3} H_3(z) \end{aligned} \tag{1}$$

where $H_0(z) = \sum_{n=0}^{(N-1)/3} h(4n) z^{-4n}$, $H_1(z) = \sum_{n=0}^{(N-1)/4} h(4n+1) z^{-4n}$, $H_2(z) = \sum_{n=0}^{(N-1)/4} h(4n+2) z^{-4n}$ and $H_3(z) = \sum_{n=0}^{(N-1)/4} h(4n+3) z^{-4n}$. The filter structure described by Eq. (1) is shown in Figure 7(a).

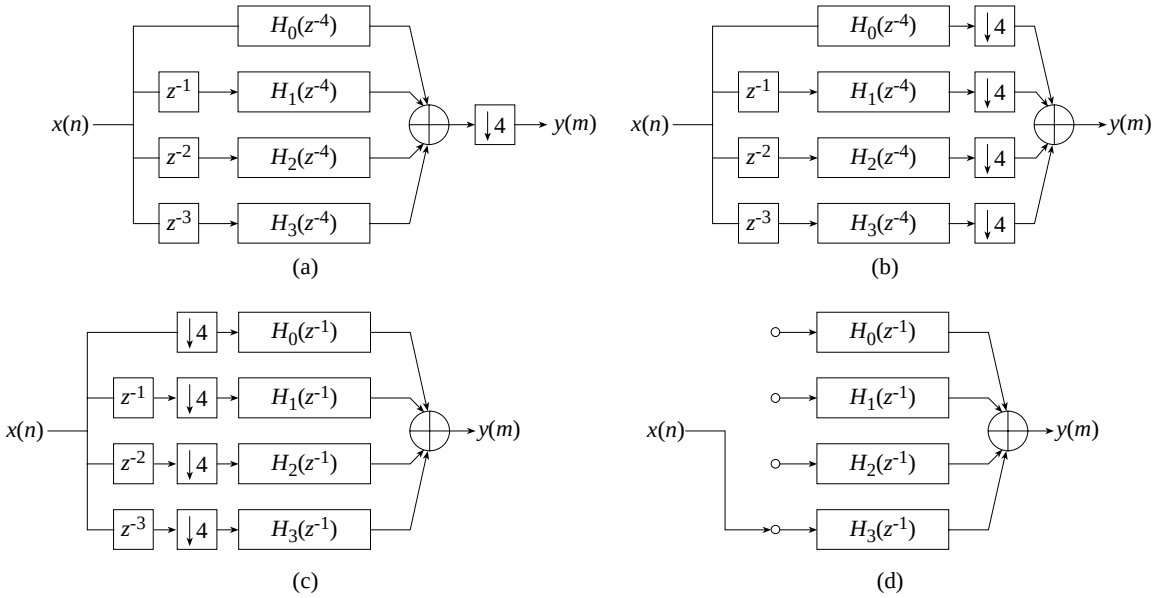

Figure 7: Polyphase decimator, $M = 4$. (a)-(c) illustrate the transformations that produce the input commutator polyphase filter (d).

The filter in Figure 7(a) can be simplified by applying identity (b) of Figure 6 to produce the structure shown in Figure 7(b). Now use Noble Identity (c) to generate the filter in Figure 7(c). Finally, the structure presenting samples to the four polyphase *sub-filters* $H_0(z), H_1(z), H_2(z)$ and $H_3(z)$ is recognized as a simple commutator. This leads to the representation of the polyphase filter shown in Figure 7(d). A key observation is that the sub-filters operate at the output sample rate, in contrast to the filter $H(z)$ in Figure 4 that operates at the high input sample rate.

Decimator Computational Complexity

Analysis of the polyphase structure reveals that we have achieved a significant computation reduction in exchange for a modest increase in algorithm complexity and control. Again, using the MAC as the unit of work, the computation rate in Figure 4 is Nf_s MACs/s. The arithmetic requirements of the polyphase filter is only Nf_s / M. The polyphase structure is more efficient by a factor of M in comparison to the non-polyphase realization. The workload reduction allows very efficient hardware realizations of decimating structures that employ polyphase structures. The computational savings can be exploited in a system in many ways. For example, it may be possible to process more channels in communication system with a given piece of hardware, introduce a higher quality filter, or utilize fewer logic resources in an FPGA implementation.

One interesting point to observe is that the order of the processing stages in the polyphase decimator has been reversed. In Figure 4, the sample rate is reduced by first filtering and then down-sampling. In the polyphase structure the input signal is first decimated and then filtered. At first glance this would appear to violate the Nyquist sampling theorem. However, the aliasing in the polyphase structure has been performed in a controlled manner. M different aliases, each differing in phase angle, have been generated in the

M polyphase sub-filters. Careful analysis of the polyphase architecture reveals that all but one of the aliased spectra are cancelled at the filter output summing node, leaving only a single non-aliased version of the filtered signal at baseband.

Interpolation

There is a requirement in many systems to increase the sample rate of signal stream. For example, in the audio community three common samples rates are employed – 32 kHz in the broadcast industry, 44.1 kHz for consumer compact disc (CD) media and 48 kHz for digital audio tapes (DAT). If we want to digitally combine (mix) signals from these three environments, a common sample rate for all of the constituent signals must be employed. To preserve audio integrity, the stream at the lower sample rate must have its sample rate increased, that is interpolated, to match the sample rate of the high sample rate signal. Interpolators also find use in digital receivers as part of the timing recovery loop, for constructing sample converters where the input and output rates are incommensurate and in oversampling digital-to-analog converters. Conceptually, the interpolation process is shown in Figure 8.

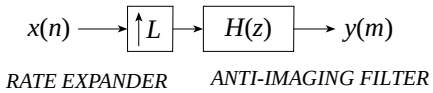

Figure 8: Interpolation circuit showing sample rate expander and interpolation filter.

The L-fold rate expander simply inserts $L-1$ zeros between the available data samples. As shown in Figure 9, the result is the introduction of L spectral images in the interval 0 to 2π radians. The purpose of the filter in Figure 8 is to remove all but the baseband image.

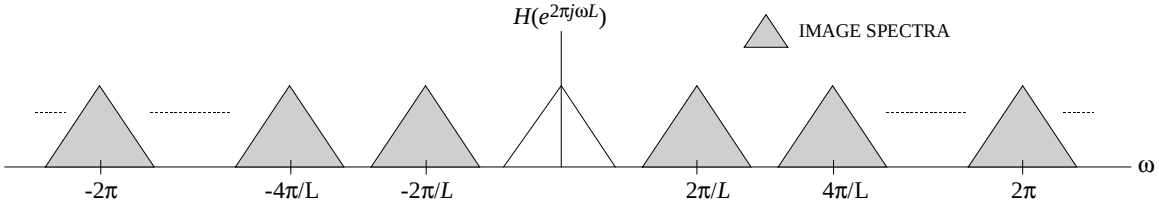

Figure 9: Spectral domain illustration of L-fold rate expansion.

We immediately observe a source of inefficiency in this simple interpolation scheme. Only 1 out of every L input samples presented to the filter represent actual data samples, the other $L-1$ samples are zero, and yet energy is being expended performing arithmetic on these values. A polyphase interpolator is a multirate filter mechanization that exploits this characteristic to produce a computationally efficient method for increasing the sample rate of a signal.

Polyphase Interpolators

Consider the circuit in Figure 10 with $L = 3$. The transfer function $H(z)$ can be expressed as

$$\begin{aligned} H(z) &= \sum_{n=0}^{N-1} h(n) z^{-n} \\ &= \sum_{n=0}^{(N-1)/3} h(3n) z^{-3n} + \sum_{n=0}^{(N-1)/3} h(3n+1) z^{-(3n+1)} + \sum_{n=0}^{(N-1)/3} h(3n+2) z^{-(3n+2)} \\ &= \sum_{n=0}^{(N-1)/3} h(3n) z^{-3n} + z^{-1} \sum_{n=0}^{(N-1)/3} h(3n) z^{-3n} + z^{-2} \sum_{n=0}^{(N-1)/3} h(3n) z^{-3n} \\ &= H_0(z) + z^{-1} H_1(z) + z^{-2} H_2(z) \end{aligned} \tag{2}$$

The filter described by Eq. (2) is shown in Figure 10(a).

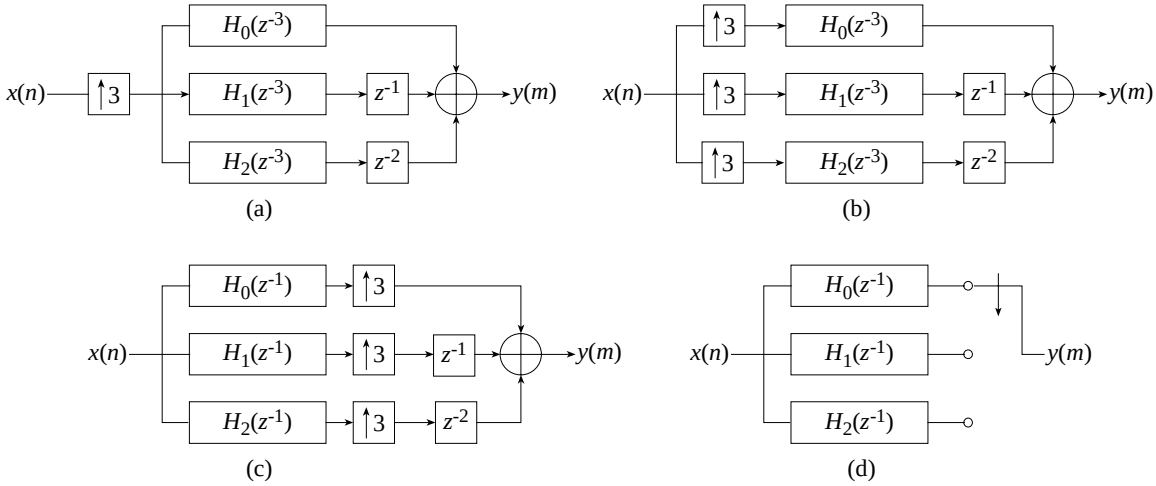

Figure 10: Polyphase interpolator – transformation steps.

Figure 10(b) is obtained simply by applying identity (b) in Figure 5. Making use of one of the Noble Identities allows the rate expanders to migrate through the interpolator sub-filters to the output summing node. In so doing, the order of the sub-filter polynomials collapse to first order. Finally, the sub-filter combining structure is recognized as a commutation operation so producing the polyphase interpolator signal flowgraph of Figure 10(d).

Interpolator Computational Complexity

The polyphase interpolator is computationally more efficient than the simple circuit in Figure 8 by a factor of approximately L. Also note that the sub-filters operate at the input rate in contrast to the higher output rate of the simple approach.

FPGA Decimator/Interpolator Implementation

There is a rich range of FPGAs provided by many semiconductor vendors including Xilinx, Altera, Atmel, AT&T and several others. The architectural approaches are as diverse as there are manufacturers, but some generalizations can be made. Most of the devices are basically organized as an array of logic elements and programmable routing resources used to provide the connectivity between the logic elements, FPGA I/O pins and other resources such as on-chip memory. The structure and complexity of the logic elements, as well as the organization and functionality supported by the interconnection hierarchy, distinguish the devices from each other. Other device features such as block memory and delay locked loop technology are also significant factors that influence the complexity and performance of an algorithm that is implemented using FPGAs.

A logic element usually consists of 1 or more RAM (random access memory) *n*-input look-up tables, where *n* is between 3 and 6, and 1 to several flip-flops. There may also be additional hardware support in each element to enable high-speed arithmetic operations. The architecture of the Xilinx 4th generation Virtex^{TM} [6] device is shown in Figure 11.

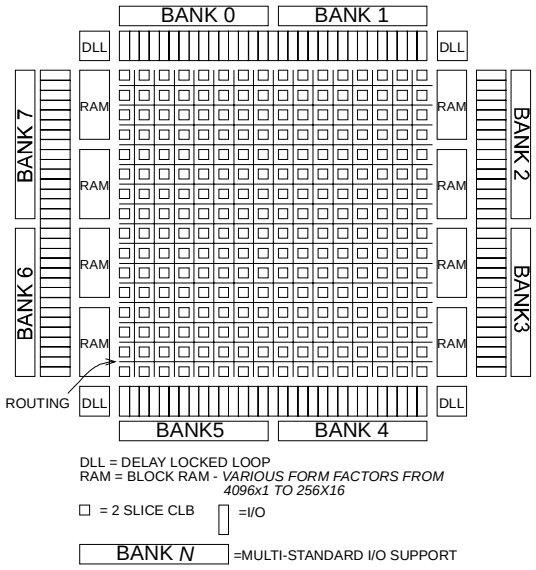

Figure 11: Xilinx Virtex FPGA architecture.

The objective of the FPGA signal processing designer is to efficiently map a DSP system to the FPGA hardware, using the logic array, block memory and other special features such as delay locked, in the most efficient and economic way possible.

The logic fabric can be employed to implement various types of arithmetic functional units, for example high-speed (150 MHz) adders, subtracters, multipliers and dividers to name a few. One design option for realization single and multi-rate filter sin FPGAs is to simply schedule a MAC unit to perform the inner-product calculation. In the case of a single rate filter data management and the coefficient addressing is simple. For polyphase decimators and interpolators it is still possible to employ a single MAC data path in conjunction with a slightly more sophisticated control unit to sequence the filter

coefficient set in the correct manner. Performance can easily be purchased by employing a parallel data path based on multi-MAC processing units. The filter coefficients can be stored in block RAM/ROM or in the logic array itself. The same is true of the sample history buffer.

Alternative algorithms are also available to construct the inner-product engine. For example, distributed arithmetic (DA) [4] has been demonstrated to be an extremely useful re-organization of the inner-product equation for FPGA systems. It exploits the look-up table and distributed memory [6] features of the FPGA. An interesting characteristic of the DA approach is that the filter sample throughput is de-coupled from the filter length. The system designer can easily trade FPGA resources for higher performance. Yet another option is based on sigma-delta modulation encoding [8]. Binomial filters have also been shown to be good candidates for certain multi-rate applications [7].

FROM FOURIER TO WAVELET

Wavelet theory is based on analyzing signals to their components by using a set of basis functions. One important characteristic of the wavelet basis functions is that they relate to each other by simple scaling and translation. The original wavelet function, known as mother wavelet, which is generally designed based on some desired characteristics associated to that function, is used to generate all basis functions. For the purpose of multiresolution formulation, there is also a need for a second function, known as scaling function, to allow analysis of the function to finite number of components. In most wavelet transform applications, it is required that the original signal be synthesized from the wavelet coefficients. This condition is referred to as perfect reconstruction. In some cases, however, like pattern recognition type of applications, this requirement can be relaxed. In the case of perfect reconstruction, in order to use same set of wavelets for both analysis and synthesis, and compactly represent the signal, the wavelets should also satisfy orthogonality condition. By choosing two different sets of wavelets, one for analysis and the other for synthesis, the two sets should satisfy the biorthogonality condition to achieve perfect reconstruction.

Fourier Background

The essence of the Fourier transform of a waveform is to decompose or separate the waveform into a sum of sinusoids of different frequencies. In other words, the Fourier transform identifies or distinguishes the different frequency sinusoids, and their respective amplitudes, which combine to form an arbitrary waveform. The Fourier transform is then a frequency domain representation of a function. This transform contains exactly the same information as that of the original function; they differ only in the manner of presentation of the information [1]. Fourier analysis allows one to examine a function from another point of view, the transform domain.

Mathematically this relationship is stated by a pair of equations denoting the forward and inverse transform. In the case of a continuous function the transform pair is known as Fourier Transform (FT) . In the case of continuous periodic functions, the function does not have a finite energy. If $x(t)$ is periodic with a period of T and fundamental frequency of $f_o = 1/T$, $x(t)$ satisfies $x(t) = x(t+T)$ for all t's, and if it has a finite

power, the periodic function can then be expressed as a linear combination of harmonically related sinusoidal functions known as the Fourier Series (FS). This transform converts a continuous periodic function to a sequence of complex numbers. In general, this sequence is infinite. However, in most practical cases, only finite number of coefficients have significant values. The Fourier transform that is applied to discrete sequences and is referred to as discrete time Fourier transform (DTFT).

Calculation of DTFT by computer can only be carried out for finite sequences and for discrete samples of $X\left(e^{j2\pi f}\right)$ in frequency domain. These requirements and constraints result in another formulation of the Fourier transform that is defined for periodic discrete functions. Let $x[n]$ be a periodic sequence with a period of N; i.e., $x[n]=x[n+N]$ for all n's, the pair of the Fourier transform relations, referred to as discrete Fourier transform (DFT), for $x[n]$, is defined by

$$\begin{aligned} &\text{Forward DFT:} \quad X[k]=\sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N} \qquad \text{for } k=0,1,2,\cdots,N-1 \\ &\text{Inverse DFT:} \quad x[n]=\frac{1}{N}\sum_{n=0}^{N-1} X[k]e^{+j2\pi nk/N} \qquad \text{for } n=0,1,2,\cdots,N-1 \end{aligned} \tag{3}$$

In Eq. (3), both discrete functions $x[n]$ and its DFT, $X[k]$, are periodic with the same period N. Although different formulations of the Fourier transform have real application in analyzing signals and systems, the DFT is practically used in real world digital computations. Some of the applications of the DFT in signal processing are spectrum estimation, feature extraction, and frequency domain filtering. Due to advances in fast computation algorithms for the DFT, known as Fast Fourier Transform (FFT) [10], and high-speed hardware implementation, this approach is used for real-time digital signal processing .

One main assumption in using the DFT for calculation of the spectrum of a discrete signal is that the observed signal is stationary during the observation time T_o. In other words, the spectrum of the signal is assumed to remain the same during the observation time. For most practical signals, this assumption is not valid. For example, in speech signals, the spectrum of the signal may vary significantly from one point to another. This depends on the content of the speech and the sampling period. In this case, and other similar cases, the Fourier transform is modified such that a two-dimensional time-frequency representation of the signal is obtained. The modified Fourier transform, referred to as short-time or time-dependent Fourier transform, depends on a window function. For discrete signals, this transformation is referred to as discrete short-time Fourier transform (DSTFT) [11] and is obtained by using a window function, $g[\ell]$. The pair of equations, that define the DSTFT of a discrete sequence, is stated by

$$\begin{aligned} &\text{Forward DSTFT:} \quad X[n,k]=\sum_{\ell=0}^{M-1} x[n+\ell]g[\ell]e^{-j2\pi \ell k/N} \\ &\text{Inverse DSTFT:} \quad x[n+\ell]=\frac{1}{Ng[\ell]}\sum_{k=0}^{N-1} X[n,k]e^{-j2\pi \ell k/N} \end{aligned} \tag{4}$$

In the DSTFT, there is a trade off between the desired resolution in the frequency domain, which is inversely proportional to the actual length of the window in time, and the assumption of short-time stationary. Based on this trade off, the window function is determined. In general, for the DSTFT, after deciding about the window function, the frequency and time resolutions are fixed for all frequencies and all times respectively. This approach does not allow any variation in resolutions in terms of time or frequency.

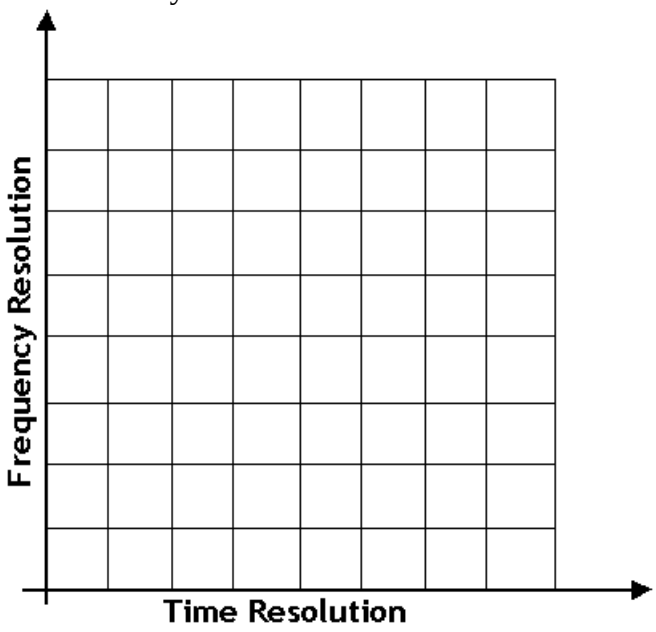

Figure 12: Time-frequency resolution of STFT

Calculation of the DSTFT via a filtering approach requires N complex FIR filters to get N components of $X[n,k]$, for $k = 0,1,2,\cdots,N-1$. As an example, the real parts of three of these complex FIR filters are shown in Figure 13. These functions indicate how different frequencies appear inside a fixed window. The number of samples at all frequencies is the same. This number is determined by the size of the window sequence.

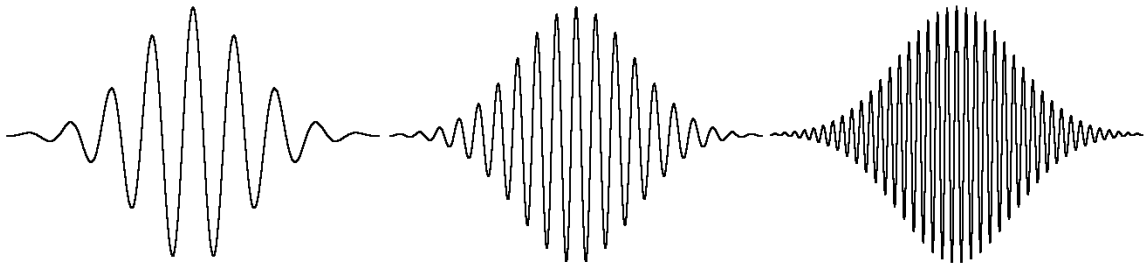

Figure 13: Real parts of three different filters associated to STFT

Wavelet Background

A *wave* is usually referred to an oscillating function of time or space, such as sinusoid. Fourier analysis is a wave analysis, which expands signals in terms of sinusoids or equivalently complex exponentials. Wave transformation of signals has proven to be extremely valuable in mathematics, science, and engineering, especially for periodic, time-invariant, or stationary phenomenon. A *wavelet* is a small wave with finite energy, which has its energy concentrated in time or space to give a tool for the analysis of transient, nonstationary, or time-varying phenomenon. The wavelet still has the oscillating wavelike characteristics, but also has the ability to allow simultaneous time, or space, and frequency analysis with a flexible mathematical foundation. This is illustrated by an example in Figure 14.

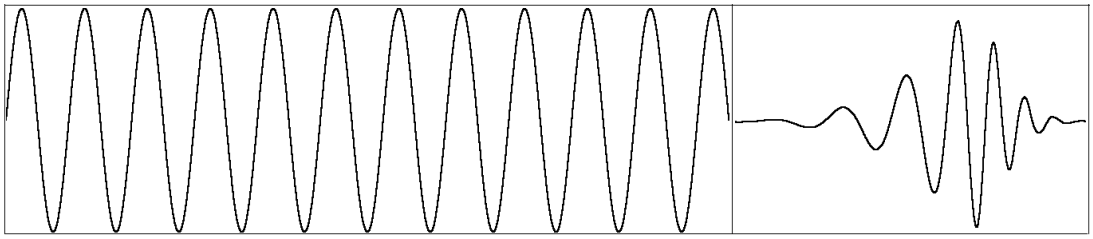

Figure 14: Comparison of a Wave and a Wavelet: Left graph is a Sine Wave with infinite energy and the right graph is a Wavelet with finite energy.

Wavelets are used to analyze signals in much the same way as complex exponentials (sine and cosine functions) used in Fourier analysis of signals. The compactness and finite energy characteristic of wavelet functions differentiate wavelet decompositions from other Fourier like analysis in their applicability to different circumstances. Wavelet functions not only can be used to analyze stationary signals but also it can be used to decompose nonstationary, time-varying or transient signals. In this discussion, for simplicity and uniform representation, the most common dyadic and discrete formulation are discussed.

The Wavelet transform can be defined for different class of functions. The intention in this transformation is to address some of the shortcomings of the STFT. Instead of fixing the time and the frequency resolutions Δt and Δf, one can let both resolutions vary in time-frequency plane in order to obtain a multiresolution analysis. This variation can be carried out without violating the Heisenberg inequality [11]. In this case, the time resolution must increase as frequency increases and the frequency resolution must increase as frequency decreases. This can be obtained by fixing the ratio of Δf over f to be equal to a constant c. In terms of the filter bank terminology, the analysis filter bank consists of band-pass filters with constant relative bandwidth (so-called *constant-Q* analysis). The way that the time-frequency plane is resolved in this approach is as shown in Figure 15. In this case, the frequency responses of the analysis filters in the filter bank are regularly spaced in a logarithmic scale. These filters are naturally distributed into octaves. With this approach, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies.

Two very close short bursts can eventually be separated if one goes to higher analysis frequencies in order to increase time resolution. The wavelet analysis, as explained, works best if the signal is composed of high frequency components of short duration plus low frequency components of long duration. The concept of changing resolution at different frequencies can be obtained by introducing what is referred to as wavelet packets [1]. Depending on the signal, arbitrary time-frequency resolutions, within the uncertainty bound [11], can be chosen. As an example, three of these functions for three different frequencies are shown in Figure 16.

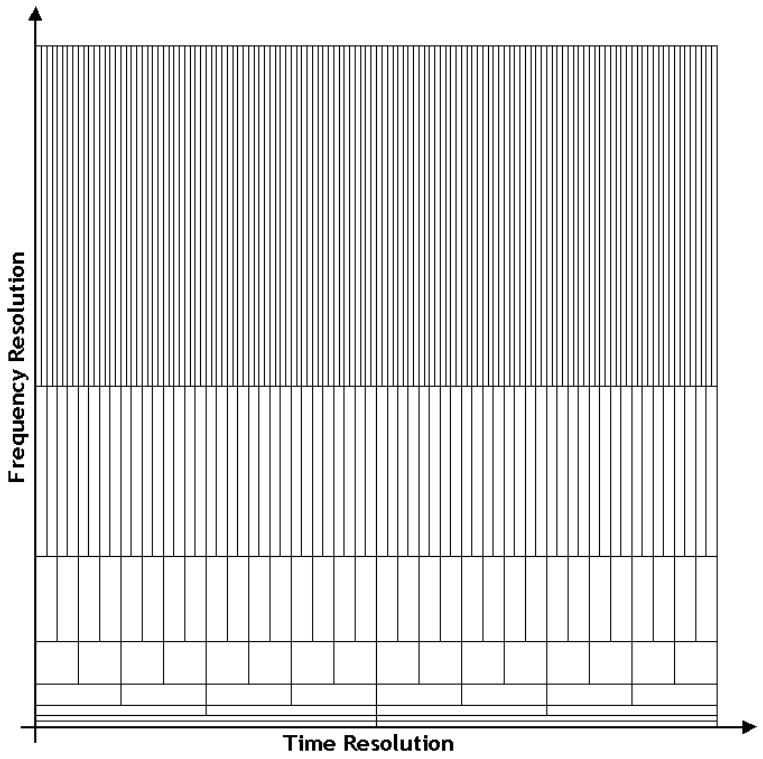

Figure 15: Time-frequency resolution for constant Q filter bank.

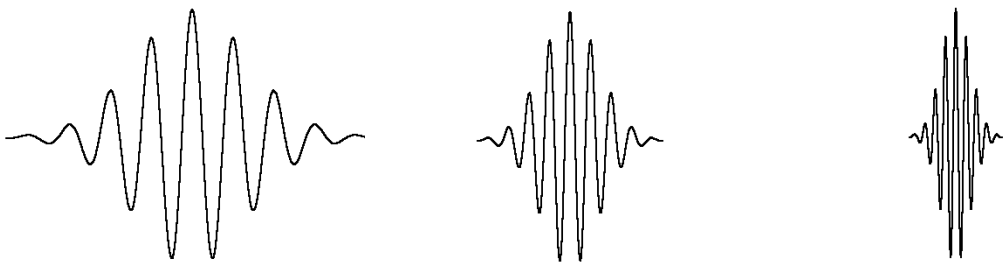

Figure 16: Three different wavelets with different frequencies and time duration's.

It is shown in [12] that any continuous function can be represented by the following expansion, defined in terms of a given scaling function and its wavelet derivatives:

$$x(t) = \sum_{k=-\infty}^{\infty} c_{j_o}(k)\varphi_{j_o,k}(t) + \sum_{j=j_o}^{\infty}\sum_{k=-\infty}^{\infty} d_j(k)\psi_{j,k}(t) \tag{5}$$

In this expansion, the first summation gives a function that is a low resolution or coarse approximation of $x(t)$ at scale j_o. For each increasing j in the second summation, a higher or finer resolution function is added, which adds increasing details. The choice of j_o sets the coarsest scale whose space is spanned by $\varphi_{j_o,k}(t)$. The rest of the function is spanned by the wavelets providing the high-resolution details of the function. The set of coefficients in the wavelet expansion represented by Eq. (5) is called the *discrete wavelet transform* (DWT) of the function $x(t)$. These wavelet coefficients, under certain conditions, can completely describe the original function, and in a way similar to Fourier series coefficients, can be used for analysis, description, approximation, and filtering. If the scaling function is well behaved, then at a high scale, samples of the signal are very close to the scaling coefficients.

In order to work directly with the wavelet transform coefficients, we should present the relationship between the expansion coefficients at a given scale in terms of those at one scale higher. This relationship is especially practical by noting the fact that the original signal is usually unknown and only a sampled version of the signal at a given resolution is available. As mentioned before, for well- behaved scaling or wavelet functions, the samples of a discrete signal can approximate the highest achievable scaling coefficients.
It can be shown [12][13] that the scaling and wavelet coefficients at scale j are related to the scaling coefficients at scale $j+1$ by the following two relations.

$$c_j(k) = \sum_m h(m-2k)c_{j+1}(m) \tag{6}$$

$$d_j(k) = \sum_m h'(m-2k)c_{j+1}(m) \tag{7}$$

Equations (6) and (7) state that scaling coefficients at higher scale, along with the wavelet and scaling filters, $h(k)$ and $h'(k)$ respectively, can be used to calculate the wavelet and scaling coefficients or *discrete wavelet transform* coefficients, at lower scales. In practice, a discrete signal, at its original resolution is assumed the corresponding scaling coefficients. For a given wavelet system, with known wavelet filters $h(k)$ and $h'(k)$, it is possible to use (6) and (7), in a recursive fashion, to calculate the discrete wavelet transform coefficients at all desired lower scales. By using multirate signal processing, it is possible to calculate the two summations by using two FIR filters. Outputs of these filters are calculated for only even indices and the filters are used with their indices being negated. These differences can be incorporated into the filtering operation by decimation of the output of the filter [5] and by reversing the order of the filter coefficients. These calculations are continued until $c_{j_o}(k)$ and $d_{j_o}(k)$ are calculated. The collection of these coefficients, namely $\left\{d_{J-1}(k), d_{J-2}(k), \cdots, d_{j_o+1}(k), d_{j_o}(k), c_{j_o}(k)\right\}$, is called DWT of the original signal $x(k)$. As an example a three-stage analysis operation is shown in Figure

17. Closer look at the number of coefficients obtained for DWT reveals that this number equals to the original number of points in the discrete signal. This is due to the decimation by two that operates in each stage of the process.

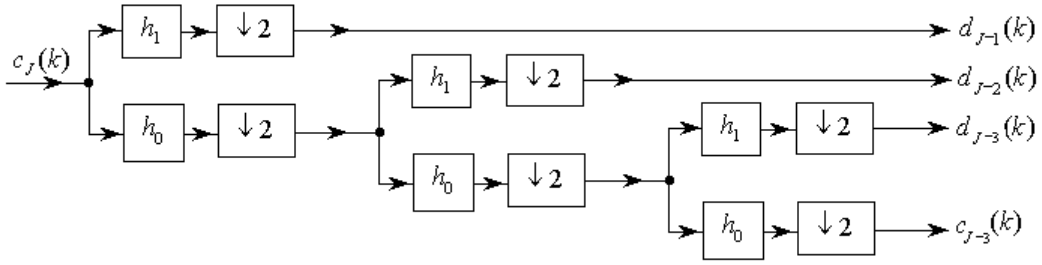

Figure 17: Three-stage wavelet decomposition, DWT analysis, tree.

Since the pair of filters used in the calculation of DWT is complementing lowpass and highpass filters; the final result provides sequences that are coming from different frequency bands of the original signal. The DWT divides the original signal bandwidth in a logarithmic fashion. These filters are referred to as analysis filters in filter bank as well as wavelet literatures.

With reference to Eq. (5), in which the original continuous signal is written in its wavelet expansion form, mathematical relations for synthesis filters can be derived. It has been shown [12][13] that the higher resolution scaling coefficients are related to the lower resolution scaling and wavelet coefficients by the following relationship.

$$c_{j+1}(k) = \sum_m c_j(m)h(k-2m) + \sum_m d_j(k)h'(k-2m) \tag{8}$$

This equation indicates how the DWT sequences at resolution j_o can be used, in an iterative fashion, to reconstruct the scaling coefficients at the highest achievable resolution, J. The three-stage synthesis operation is better explained in Figure 18. Recall that the analysis filter bank efficiently calculates the DWT using banks of decimating digital filters. Similarly, the synthesis filter bank efficiently calculates the inverse DWT by reconstructing the original discrete signal by using interpolating digital filters.

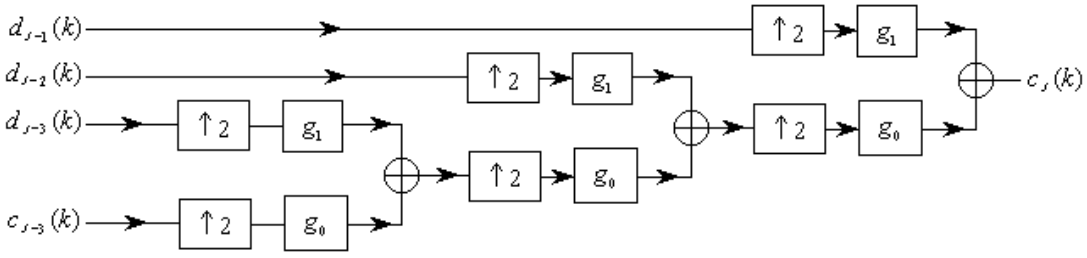

Figure 18: Three-stage wavelet reconstruction, DWT synthesis tree.

In summary the forward and inverse wavelet transform for discrete function $x[k]$ is obtained through the following relations:

$$\text{Forward DWT:}\quad \begin{cases} c_J(k) = x(k) \\ \left[\begin{array}{l} c_{j-1}(k) = \sum_m h_o(m-2k)c_j(m) \\ d_{j-1}(k) = \sum_m h_1(m-2k)c_j(m) \\ \text{for } j = J, J-1, \cdots, j_o+1 \end{array}\right. \\ \left\{d_{J-1}(k), d_{J-2}(k), \cdots, d_{j_o+1}(k), d_{j_o}(k), c_{j_o}(k)\right\} \end{cases} \tag{9}$$

$$\text{Inverse DWT:}\quad \begin{cases} c_{j+1}(k) = \sum_m c_j(m)g_o(k-2m) + \sum_m d_j(k)g_1(k-2m) \\ \text{for } j = j_o, j_o+1, \cdots, J-1 \\ x(k) = c_J(k) \end{cases}$$

WAVELET REALIZATIONS AND IMPLEMENTATIONS

In this section, first we discuss how to realize different wavelet filters needed for calculation of forward and inverse DWT. Then, the issue of full integration of these building blocks is considered. In the realizations presented here, we only use low level computation or operation blocks. The most common block, in this case, are addition, multiplication, decimation by two or up-sampling and delay units. The filter bank realizations of the discrete wavelet transform and its inverse are based on two basic building blocks, shown in Figure 19.

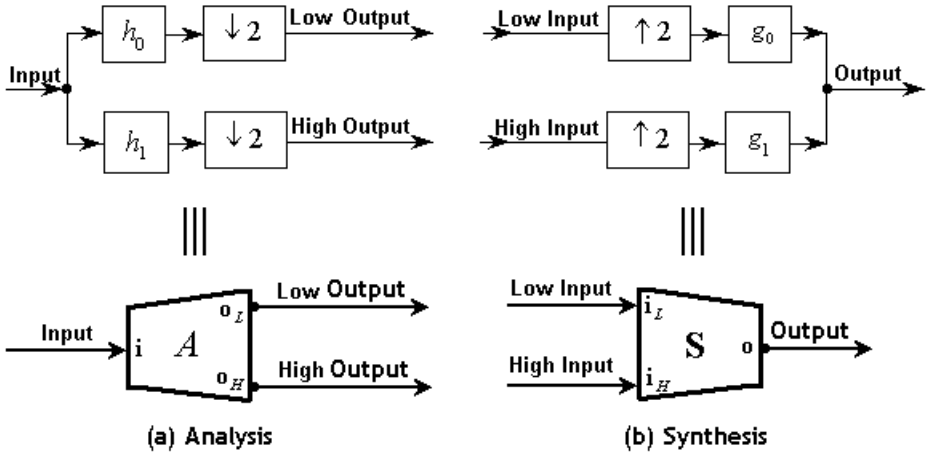

Figure 19: Basic Blocks for (a) analysis and (b) synthesis filter banks.

In the forward transform, the two decimating FIR filters are $h_0(k)$ and $h_1(k)$, and in the inverse transform, the two interpolating filters are $g_0(k)$ and $g_1(k)$. Perfect

reconstruction requires that the output of the synthesis block be the same as a delayed and scaled version of the input of the analysis block. Based on this desired condition, the analysis and synthesis filters should always satisfy the following two[13]:

$$\begin{cases} h_1(k) = (-1)^k g_0(k) \\ g_1(k) = -(-1)^k h_0(k) \end{cases} \tag{10}$$

In the orthogonal wavelet systems, knowledge of the scaling filter is sufficient for design of the analysis and synthesis filters [13]. For a given even size, K, FIR scaling filter $h(k)$, we have

$$\begin{cases} g_0(k) = h(k) \\ h_0(k) = g_0(K-1-k) \\ g_1(k) = -(-1)^k h_0(k) \\ h_1(k) = (-1)^k g_0(k) \end{cases} \tag{11}$$

In the signal processing literatures, similar conditions are used for two-band filter banks for perfect reconstruction, without the imposed orthogonality [5][13]. For that situation, which is referred to as *quadrature mirror filter* (QMF) or *conjugate mirror filter* (CMF), based on frequency response symmetry, the following conditions are established:

$$\begin{cases} g_0(k) = h(k) \\ h_0(k) = g_0(k) \\ g_1(k) = -(-1)^k h_0(k) \\ h_1(k) = (-1)^k g_0(k) \end{cases} \tag{12}$$

Direct realization of the two building blocks shown in Figure 19 require that all computations be carried out at the higher input rate for the analysis filters and at the higher output rate for the synthesis filters. Realization of the analysis and synthesis filters can be accomplished at the lower rate by taking advantage of polyphase representation of these filters.

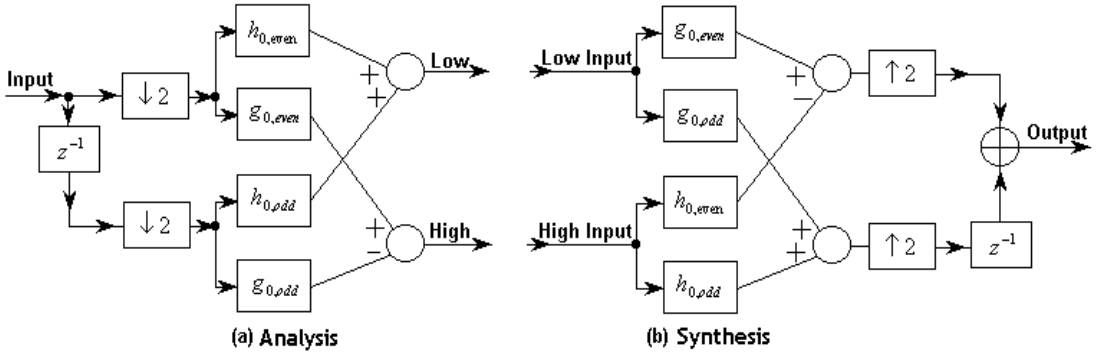

Figure 20: Polyphase realization of the basic (a) analysis and (b) synthesis blocks.

In this section, we discuss how resources can be shared between different filters used in any one- dimensional analysis or synthesis wavelet tree. To this end, and for better explanation of implementation issues, delay units, adders and multipliers are put together as blocks. This block diagram representation help to better discuss the issue of proper time-sharing and data scheduling. Any of the filter realization that uses polyphase structure discussed in the previous section can be simplified as shown in Figure 20.

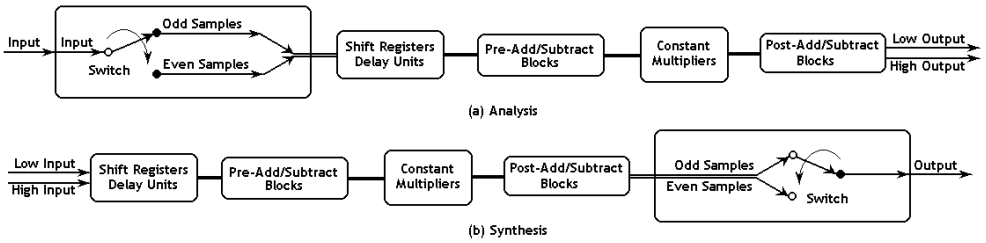

Figure 21: (a) Analysis and (b) synthesis shown in blocks of operation tasks.

Block representation of individual tasks helps to decide which part of the filtering operation can be shared between different branches of any wavelet tree. The purpose is to optimize the resource allocation within a given FPGA by specifically sharing the multiplication block, which is very demanding in resources. For example, in a multi-stage DWT, shown in Figure 22, one set of multipliers is sufficient for all stages. In the case of wavelet packet, for any stage one set of multipliers is enough to carry out the task. Alternatively, for faster multipliers, one set of multipliers can be shared between two consecutive wavelet packet stages.

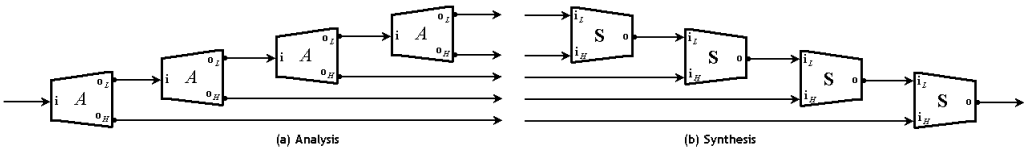

Figure 22: Four stage DWT tree for (a) analysis and (b) synthesis.

In Figure 23, one method for implementation of five stage DWT is shown. In this case, the constant multipliers and post add/subtract blocks are shared between all stages. The pre-add/subtract block can also be shared between different stages. Since constant multipliers use most of resources, they should be minimized as much as the algorithm allows. Add/subtract blocks can be reduced to the minimum level if that does not cause any increase in other resources. For example, the complexity of the logic's used for control purposes and extra shift registers needed for proper pipelining should be compared by the savings that will be resulted from sharing the add/subtract blocks.

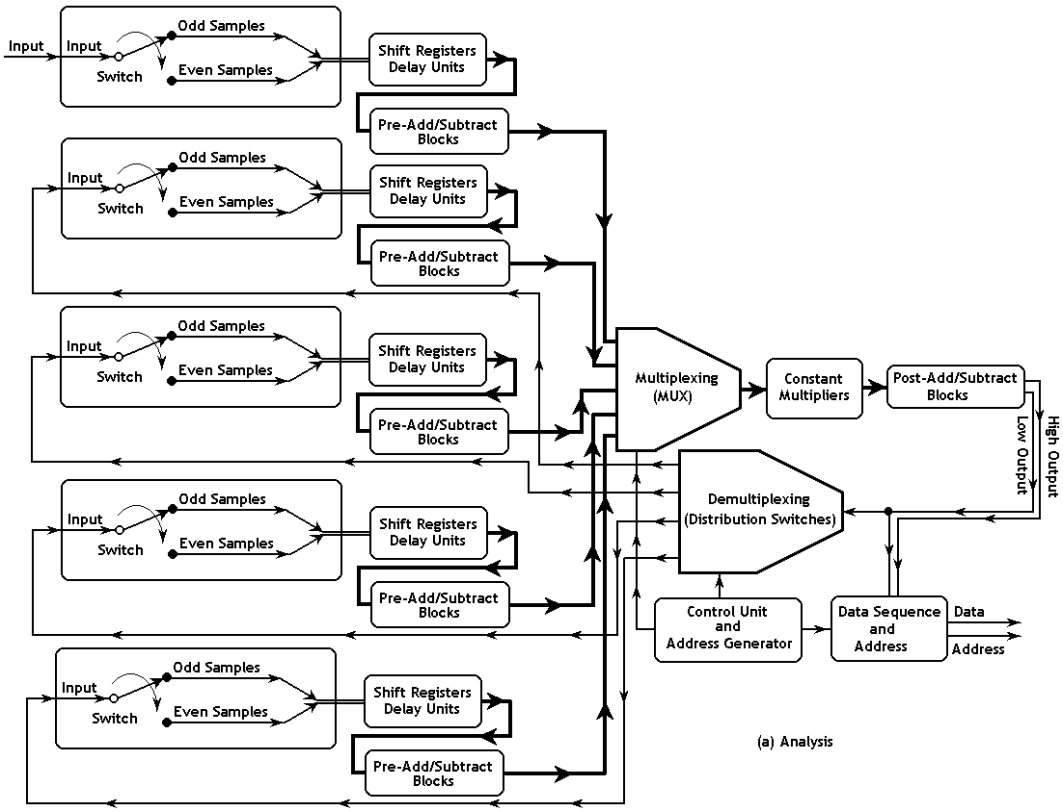

Figure 23a: Implementation of five stage DWT analysis.

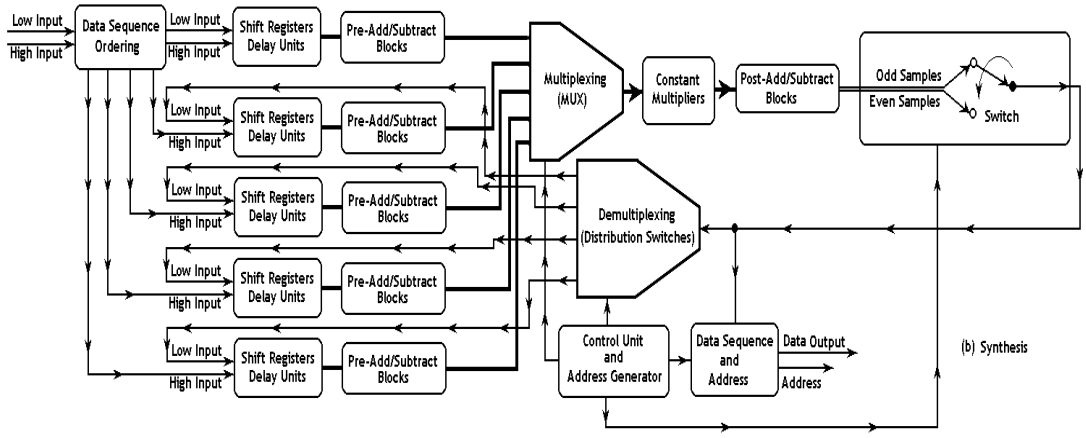

Figure 23b: Implementation of five stage DWT synthesis.

The reason that the DWT requires only one set of constant multipliers, working at the highest signal rate is as follows. In the first stage, the multipliers are working at half speed therefore they are free half time to do other computations. For the second stage, the multipliers are needed for one-fourth of the original clock rate. This means that half of the remaining time from the first stage can be dedicated to the second stage. In that case, the multipliers will be free for one-forth of the original clock rate. Using this extra time for the third stage, which only needs one-eighth of the original clock rate, is still giving us enough time to spend on the following stages. No matter how many stages for the DWT is required, always there is enough time to allocate a single bank of constant coefficient multipliers to all of them. This architecture behaves like the limit of the geometric series ($\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\cdots=1$) and always has an open cycle for a finite frame.

In the case of inverse DWT, synthesis, the situation is the same, in this case, however, the multipliers start at lower rates and as data becomes available they go to higher rates. Maximum rate of use of multipliers is the same in both analysis and synthesis. The situation for customized tree structure will be different. Depending on the decomposition tree, proper resource allocation should be planned. A general approach for customized tree structure cannot be formulated unless the decomposition tree is known before hand. One general rule of thumb, which is not based on an optimum use of resources, is to allocate one set of multipliers to the first stage plus any other following stages that look like a DWT stage. For all other stages, use the structure from wavelet packet. Another ad-hoc approach would be to decompose the customized tree to several DWT-like trees and allocate one set of multiplier to each one of those DWT-like trees. In this way, it may be possible to use one set of multipliers for several slower DWT-like trees.

CONCLUSION

This white paper has provided an overview of the multirate filter fundamentals. Some options for the implementing this type of data path using programmable logic were highlighted. Although the focus of this work was based on polyphase filter concepts, many other alternative options are available. These include Lagrange interpolation, and Farrow filters [14] and cascaded-integrator-comb filters [9] to name a few.

In general, the goal of most modern wavelet research is to create a mother wavelet function that will give an informative, efficient, and useful description of the signal of interest. It is not easy to design a uniform procedure for developing the best mother wavelet or wavelet transform for a given class of signals. However, based on several general characteristics of the wavelet functions, it is possible to determine which wavelet is more suitable for a given application. Once that mother wavelet has been determined designing the wavelet tree can proceed. The realization and implementation of the forward and inverse DWT in the appropriate technology (DSP processor or FPGA) can

then take place. Wavelet signal processing in the transform domain in the form of Denoising and Compression algorithms can now be explored with the assurances that the Wavelet transform can be computed in real time.

REFERENCES

[1] Riol, O., and Vetterli, M., "Wavelets and Signal Processing", *IEEE Signal Processing Magazine*, Vol. 8, No. 4, pp. 14-38, October 1991.

[2] Mallat, S. G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", *IEEE Transactions on Pattern Recognition and Machine Intelligence*. Vol. 11, No. 7, pp. 674-693, July 1989.

[3] Daubechies, I., "The Wavelet Transform, Time-Frequency Localization and Signal Analysis", *IEEE Transactions on Information Theory*. Vol. 36, No. 5, pp. 961-1005, September 1990.

[4] S. A. White, "Applications of Distributed Arithmetic to Digital Signal Processing", *IEEE ASSP Magazine*, Vol. 6(3), pp. 4-19, July 1989.

[5] P.P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[6] Xilinx Inc., *The Programmable Logic Data Book*, 1999.

[7] C. H. Dick and f. j. harris, "FPGA Multirate Filters: A Case Study Using Virtex", *ICSPAT'99*, Orlando, Florida, Oct. 1999.

[8] C. H. Dick and f. j. harris, "High-Performance FPGA Filters Using Sigma-Delta Modulation Encoding", *The International Conferences on Acoustics Speech and Signal Processing - (ICASSP'99)*, Phoenix Arizona, March 15-19 1999.

[9] E. B. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation", *IEEE. Trans. Acoust., Speech Signal Processing*, Vol. 29, No. 2, pp. 155-162, April 1981.

[10] Brigham, E. O., *The Fast Fourier Transform*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1974.

[11] Allen, J. B. and Rabiner, L. R., "A Unified Approach to Short-Time Fourier Analysis and Synthesis", *Proceedings of IEEE*. Vol. 65, No. 11, pp. 1558-1564, 1977

[12] Burrus, C. S.; Gopinath, R. A.; and Guo, H., *Introduction to Wavelets and Wavelet Transforms: A Primer*, Prentice Hall Inc., Upper Saddle River, New Jersey, 1998.

[13] Strang, G. and Nguyen, T., Wavelets and Filter Banks. Wellesley-Cambridge Press, Wellesley, Massachusetts, 1996.

[14] T. I. Laakso and V. Valimaki, "Splitting the Unit Delay", *IEEE Signal Processing Magazine*, pp. 30-60, Jan. 1996.