

Parallel Adaptive Equalizer Employing Sub-Convolution: VLSI Architecture Realized in a Field Programmable Gate Array

Andrew A. Grary, PhD.
Jet Propulsion Laboratory

Abstract-This paper provides an overview of a parallel adaptive equalizer architecture and a high-rate FPGA realization. The very large scale integration (VLSI) architecture for implementing a frequency domain least-mean squares (LMS) complex equalizer incorporates a simple *sub-convolution* method, digital vector processing, specialized FFT-IFFT hardware architectures, and the discrete Fourier transform-inverse discrete Fourier transform (DFT-IDFT) overlap and save filter method. The architecture is based on recent work aimed at reducing complexity of parallel processing/filtering. A key property of the architecture is that the equalizer tap length may be chosen completely independently of the FFT-IFFT lengths and input data block lengths. Theoretically unlimited tap lengths are possible with short FFT-IFFT pairs. It will be demonstrated that the new parallel architecture is very well suited for processing multi-Gbps digital communication data rates with relatively low speed CMOS implementations. The VLSI equalizer architecture and FPGA realization presented processes complex demodulated symbols at 1/4th the modulation symbol rate, has 32 coefficients, is decision directed, and will process data modulated with quadrature phase-shift keying (QPSK) and 16 quadrature amplitude modulation (QAM). The characteristics and performance of the FPGA implementation of the equalizer are presented.

I. Introduction

Advances in GaAs have made processing rates of several Giga-hertz (GHz) possible. However, the widespread use of high-speed GaAs components is costly in terms of both nonrecurring engineering costs and reproduction costs. It is difficult, if not impossible, to implement all of the functionality of all-digital receivers for modern satellite communications on a single GaAs application specific integrated circuit (ASIC). CMOS has much higher transistor density and typically lower nonrecurring engineering and reproduction costs than GaAs. These factors have led NASA and JPL to develop *the all-digital parallel receiver* for processing near-maximum Nyquist data rates [1,2,3,4,5]. A block diagram of the receiver is given in Figure 1.

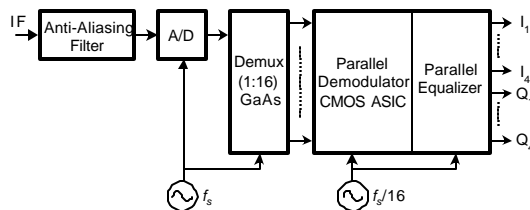


Figure 1. Parallel Digital Receiver

The objective of the parallel algorithms and architectures is to remove the limitation caused by the processing rate difference of GaAs and CMOS as illustrated in Figure 1, with the CMOS demodulator ASIC operating at 1/16th the sample rate. The equalizer developed here requires 1/4th rate parallel processing since it operates on one sample per symbol; the demodulator processes 4 symbols per demodulator clock cycle, or 16 samples per clock since 4 samples per symbol are assumed.

II. LMS Equalizer and Parallel Processing

Figure 2 illustrates a 1/4th rate frequency domain fast LMS algorithm. This is a specific manifestation of the algorithm given by Haykin [6] and Shynk [7]. Here $x(n)$ is the sequence of the complex demodulated baseband QPSK or 16-QAM symbols, $y(n)$ is the equalizer output and $d(n)$ is the desired response created from $y(n)$ (decision directed equalizer [8,9]).

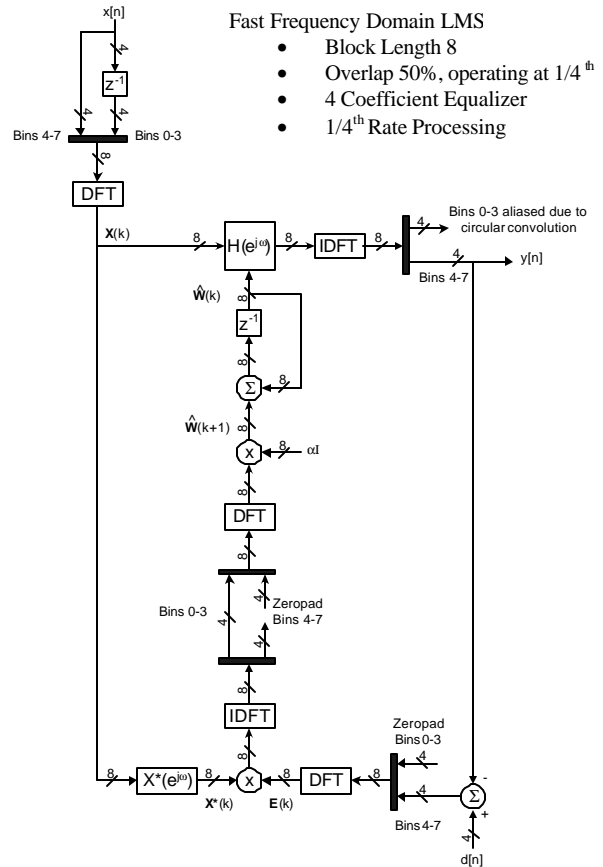


Figure 2. Frequency Domain Fast LMS [6,7]

We have chosen the FFT-IFFT lengths such that we may achieve the desired rate reduction for hardware implementation. However, the maximum number of coefficients is 5 or less. Obviously larger FFT-IFFT pairs may be used to achieve larger tap length; however, this leads to a more complex design and more transistors in a concurrent hardware implementation. However, if a very high rate hardware implementation of the algorithm is the goal, computational efficiency is not necessarily the primary design consideration. The architecture may be tremendously computationally efficient, but it may require so many transistors for concurrent implementation it may not be possible to implement. Alternatively, if hardware reuse is used – an extreme example of this is a pure software implementation – with maximum clock rate of clk_{Max} , the architecture is not capable of processing as much data as the full parallel or concurrent architecture operating at clk_{Max} . A hardware design employing partial concurrent or parallel operation and partial hardware reuse is often highly specialized to a specific algorithm and is most often tedious to design and results in a design difficult to debug and test.

The new equalizer architecture developed here allows trades to be made between computational efficiency, and hardware complexity and processing rate. The trade between computational efficiency and the rate and complexity of hardware is made by eliminating the lower bound on FFT-IFFT lengths of Figure 2 created by the equalizer tap length. Through the methods developed here using *sub-convolution*, this lower bound is completely eliminated through simple and generic digital signal processing methods. The FFT-IFFT lengths may be chosen completely independently of the equalizer tap length, and in fact will be determined by the processing rate reduction desired.

The goal is to create an equalizer architecture that operates in parallel without hardware reuse for the system illustrated in Figure 1. The FFT-IFFT lengths are chosen to give the desired decrease in processing rate, in this case 4 resulting in FFT-IFFT length of 8. The system of Figure 1 has 4 samples per symbol, the demodulator operates at $1/16^{\text{th}}$ the A/D rate, $1/4^{\text{th}}$ the symbol rate, and the non-fractionally spaced equalizer operates at $1/4^{\text{th}}$ the symbol rate. The coefficient length of the equalizer is chosen to be 32 to meet system requirements.

III. Parallel Sub-Convolution Filter Banks

Figure 3 illustrates a parallel DFT-IDFT filtering architecture for frequency domain filtering or correlation using the overlap and save method. The DFT-IDFT length is $L+1$ (L is odd), and M , the downsample rate, is the number of samples the input window “slides”. The architecture in Figure 3 has 50% input vector overlap, that is the downsample rate, is equal to half the input vector length, $M = (L+1)/2$. With such a architecture a $M+1$ tap filter may be implemented in the frequency domain.

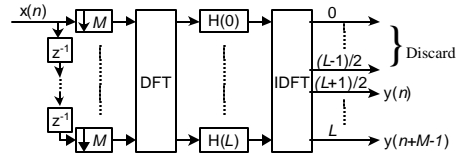


Figure 3. Overlap-and-Save FIR Filter

The filter, $h(n)$, is zero-padded to length $L+1$ and then transformed to the discrete frequency domain via the DFT, to obtain the frequency domain coefficients, $H(k)=\text{DFT}\{h(n)\}$ [10]. It is obvious that any FIR filter with an order M or less can be used with this same architecture. Similar derivations for FIR filters have been developed for implementation in software and hardware. The limitation to all of these methods in a concurrent VLSI implementation is that the DFT-IDFT, or fast Fourier transform-inverse fasts Fourier transform (FFT-IFFT) lengths are increased to increase the order of the FIR filter to be implemented.

Consider the simple convolution sum of equation 1. The convolution may be broken into numerous sub-convolutions, each time shifted input convolved with a *sub-filter*, as indicated.

$$y(n) = \sum_{k=0}^{L+1} x(n-k)h(k) = \sum_{k=0}^{j_1} x(n-k)h(k) + \sum_{k=j_1+1}^{j_2} x(n-k)h(k) + \dots + \sum_{k=j_R}^{L+1} x(n-k)h(k) \quad (1)$$

First we observe that each sample vector input to the DFT of Figure 3, and therefore the frequency domain vector, is a time delay of M samples from the next sample vector input. From (1), it is obvious that each of the sums are themselves a convolution with a block of the filter or *sub-filter*, we call these *sub-convolutions* with sub-filters, the sum of their outputs is equal to the convolution of the input, $x(n)$, with the filter $h(n)$. Each of these sub-convolutions may be implemented in the frequency domain using the technique illustrated in Figure 3, then the results summed to yield the convolution output. To break a convolution up into R equal length sub-convolutions, each $(L+1)$ in length, using this method would require R DFTs, R IDFTs, and R sub-filters. Assuming 50% overlap, the DFT-IDFT pairs would each be in $(L+1)$ length, however simplifications requiring only one DFT-IDFT pair are possible with one additional constraint. We can derive the constraint simply by realizing that each input vector to the DFT of Figure 3 is a shift in time of M samples, therefore each frequency domain vector is separated in time from the previous or next vector by M sample periods. From (1), if $j_i + M = j_{i+1} \forall i$, that is the time delay between each sub-filter is equal to the time delay between time-consecutive input vectors (figure 3), then the convolution of (1) may be calculated in the frequency domain by simply delaying the frequency domain vectors and multiplying by the appropriate

frequency domain sub-filter. These sub-filters are generated as follows.

$$H_k(i) = DFT\{h_k(n)\} \quad i=0,\dots,L, \quad k=1,\dots,R \quad (2)$$

$$n=0,\dots,L$$

and $h_k(n)$ is the k^{th} zero padded sub-filter given by:

$$h_k(n) = h(n+(k-1)M) \quad n=0,\dots,\frac{L-1}{2}, \quad k=1,\dots,R \quad (3)$$

$$= 0 \quad n = \frac{L+1}{2}, \dots, L, \quad k=1,\dots,R$$

Using simple properties of linearity only one DFT-IDFT pair of this length is required as all of the frequency domain sub-convolutions may be calculated then summed in the frequency domain then transformed back into the time domain. The resulting architecture is illustrated in Figure 4. This system performs convolution at a rate of $1/M$ that of the sample rate of $x(n)$. It is clear that the length of the DFT-IDFT pairs may be chosen with rate reduction as the principal design criterion independent of FIR filter length. This simple architecture then allows relatively short DFT-IDFT lengths to be used to reduce the processing rate of arbitrarily high order FIR filtering or correlation operations, yielding overall simple designs.

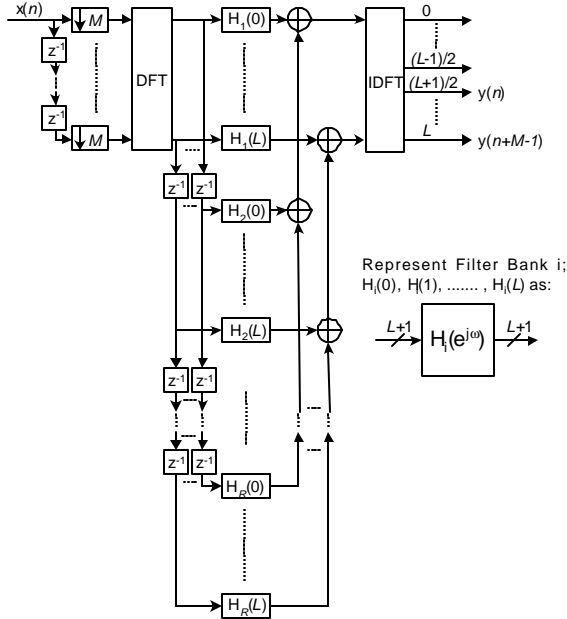


Figure 4. Parallel Sub-convolution Filter Bank Architecture

IV. Frequency Domain Fast LMS Architecture Employing Sub-Convolution and Sub-Correlation

Figure 5 illustrates a new frequency domain LMS architecture employing sub-convolution and sub-correlation. The architecture is similar to a traditional frequency domain LMS with block length 32 and 16 input sample overlap implementing 16 equalizer coefficients. However, in this

architecture only 8-point FFT-IFFT pairs are required and it is extendable to 32 (or larger) coefficient length.

Assuming constant \mathbf{a} for all frequency bins, the architecture of Figure 5 performs the same as the traditional fast frequency domain LMS with block length 32 (32-point FFT-IFFTs) with 50% input block overlap, and 16 coefficients. Obviously in the 32-point FFT-IFFT case, if a frequency dependent step size \mathbf{a} is desired, there is more frequency resolution possible. All the mathematical operations of the 32-point fast LMS equalizer employing 32-Point FFT-IFFTs; as in Figure 2 with modifications such as 32 sample block inputs sliding 16 samples per clock and 32-point FFT-IFFTs, are performed in the new architecture illustrated in Figures 5 and 6.

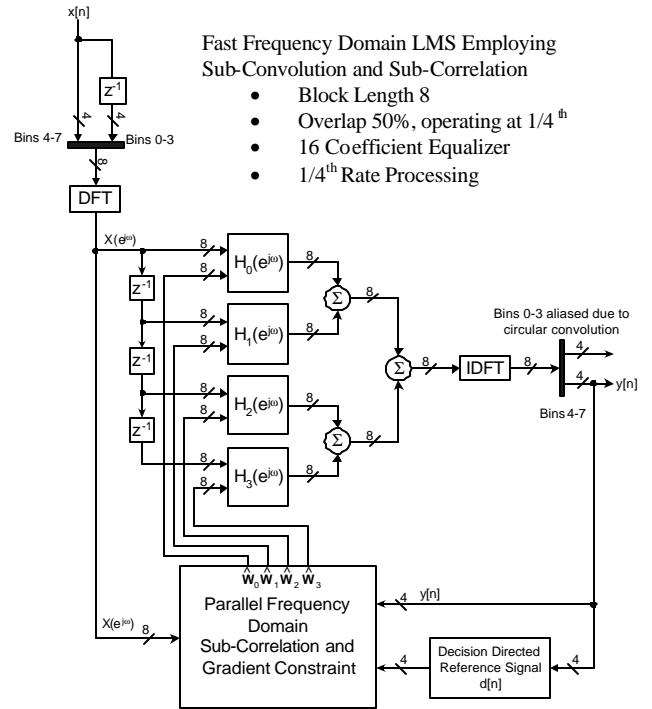


Figure 5. Parallel Frequency Domain Fast LMS Architecture Employing Sub-Convolution and Sub-Correlation

The convolution and correlation operations of the fast LMS have been replaced by sub-convolution and sub-correlation. There are other minor modifications that follow directly from linear and multirate systems theory. Figure 7 illustrates the function of the vector downsampler used throughout the architectures in Figures 5 and 6. The architecture of Figures 5 and 6 may be extended to arbitrarily long tap lengths, while still using 8-point FFT-IFFT pairs for the $1/4^{\text{th}}$ processing rate reduction.

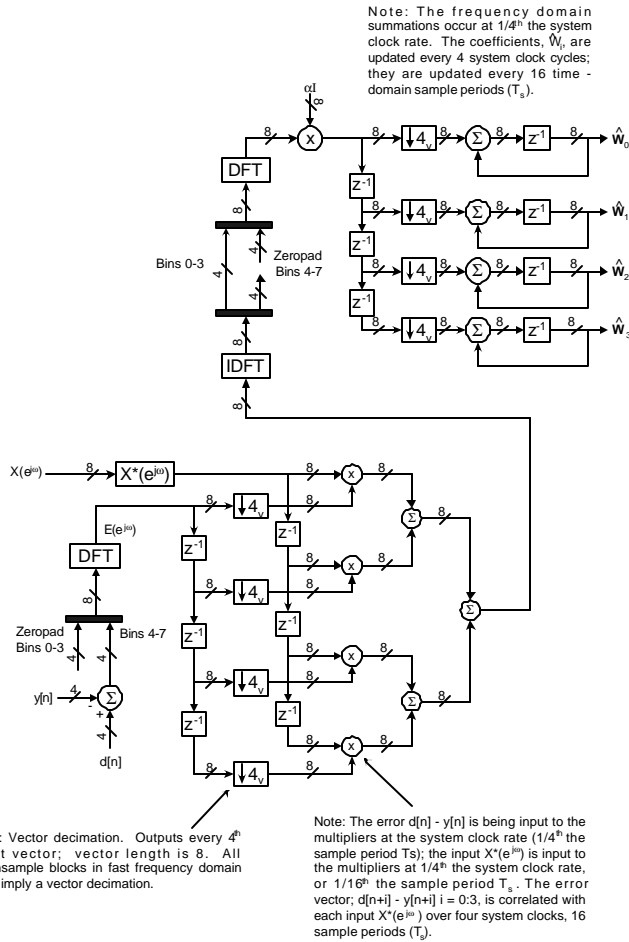


Figure 6. Frequency Domain Sub-Correlation and Gradient Constraint

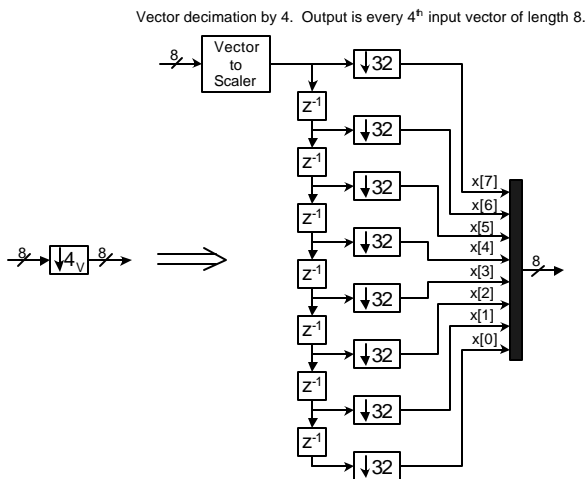


Figure 7. Length-8 Vector Downsample by 4

V. FPGA Implementation

The design process of the 11-tap parallel equalizer is illustrated in figure 8. The design tools used in the

equalizer design are as follows: Cadence Signal Processing Worksystem (SPW), Synopsys VSS VHDL compiler/simulator, Synplicity Simplify Pro, Xilinx ISE Foundation 4.1, and Cadence NC-Verilog. The floating-point and fixed point parallel equalizer hardware architecture was developed and verified in SPW. Once the fixed point architecture met the desired precision, the fixed point architecture was converted into VHDL using the Xilinx CoreGen tool to generate Xilinx optimized hardware adders, subtractors, multipliers, counters, comparators and registers. Note that the VHDL code was hand written. No code generation by SPW was used. This decision was made to make debugging the VHDL design easier and allowing more control over the VHDL code.

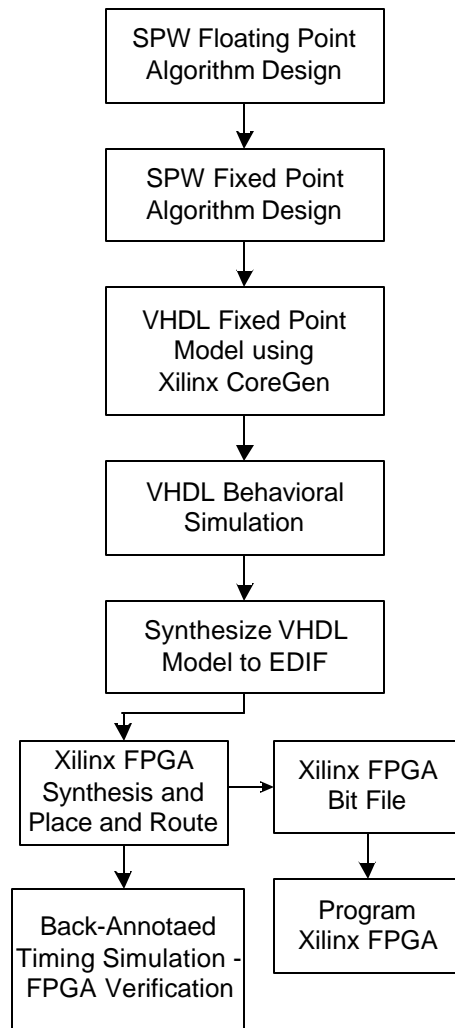


Figure 8. FPGA Design Flow

The resulting VHDL code was simulated and compared against the SPW fixed-point model to verify the bit accuracy of the VHDL model. Once the VHDL model is verified, the VHDL code is then synthesized to the target FPGA, in this

case the Xilinx Virtex 2 8-million gate FPGA, xc2v8000-5-ff1152. After synthesis to the target FPGA, the timing model of the synthesized design is simulated to verify the operation of the design and to make sure no timing errors exist. Once this simulation is verified, the resulting bit file of the synthesized design can be loaded into the target FPGA.

The hardware system drawing of the equalizer implemented in the Xilinx Virtex 2 8-million gate FPGA is illustrated in Figure 9. The signal paths that have $2x \times L \times B$ indicate a complex vector signal, $2x$ (real vector and imaginary vector), with a vector length L , and each element B bits wide. Signal paths that have $L \times B$ indicate a vector of length L with each elements B bits wide. The internal signals marked CPU Control, indicate signals that can be made programmable, but for the scope of this design they are not programmable and are synthesized as fixed constants. The scope the FPGA implementation will focus on the synthesis of the DSP algorithms that create the parallel equalizer, the DSP core. The synthesis results given are only for the DSP core. The FPGA does have remaining resources that will allow a CPU interface. Any results showing the design hit of adding a CPU interface is beyond the scope of this paper.

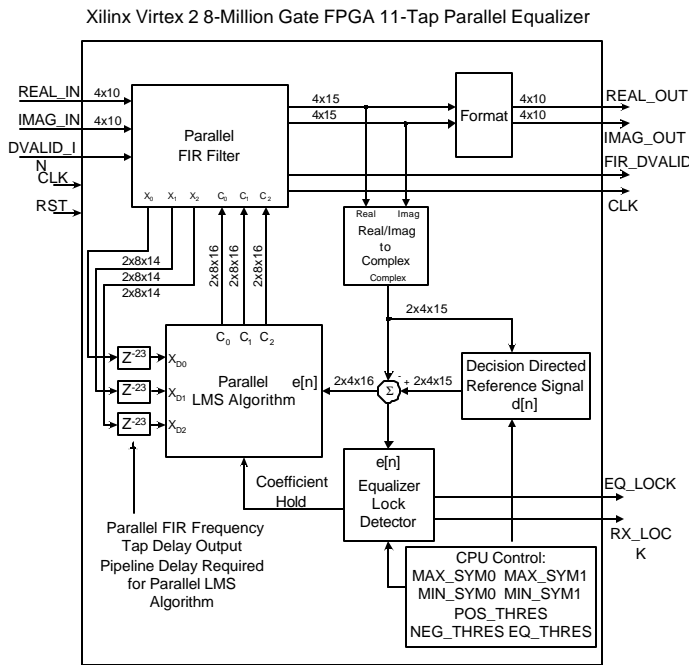


Figure 9. Xilinx Virtex 2 8-Million Gate FPGA 11-Tap Parallel Equalizer.

The inputs and outputs of the equalizer FPGA assume a parallel complex input and output with a vector length of 4 elements. The inputs to the FPGA are REAL_IN, IMAG_IN. The inputs REAL_IN and IMAG_IN are the

signed vector complex inputs to the parallel FIR filter with a vector length of 4. The bit resolution of each element of REAL_IN and IMAG_IN are 10-bits. DVALID_IN is a flag that marks each complex vector of 4 elements as being valid or not valid. The outputs of the FPGA are REAL_OUT, IMAG_OUT. The outputs REAL_OUT and IMAG_OUT are the signed vector complex outputs of the parallel FIR filter with a vector length of 4. The bit resolution of each element of REAL_OUT and IMAG_OUT are 10-bits. Other single bit inputs are the FPGA clock (CLK), reset (RST), and DVALID_IN. One last single bit output is FIR_DVALID and EQ_LOCK. The signal CLK and RST is the FPGA global clock and reset. The reset is assumed to be asynchronous to the global clock. An internal logic circuit will synchronize the reset to the global clock to assure that the reset is applied synchronous to the global clock. The signals DVALID_IN and FIR_DVALID are flags that mark each complex input and output vector of 4 elements as being valid data or invalid data. DVALID_IN is the flag for the data going into the parallel FIR filter and FIR_DVALID is the flag for the data coming out of the parallel FIR filter. FIR_DVALID is a delayed version of DVALID_IN due to internally progressing through the parallel FIR filter pipeline. These flags allow the equalizer to always run at the fastest clock frequency even though the data may be at a slower symbol-rate. If the data is running as fast as the FPGA clock, then the DVALID_IN and FIR_DVALID will always be high, flagging the data as valid. This allows the data to be read from an external FIFO. The input to the parallel equalizer requires that the complex data, real and imaginary, to be synchronous to the FPGA global clock. The EQ_LOCK bit specifies if the equalizer has converged or not by having an equalizer lock detection circuit monitoring the equalizer error signal. For this design, the input and output consists of a 41-bit data bus plus a clock, reset and lock detection status.

Internal resolutions of the parallel equalizer allow for an LMS error signal of 16-bits. The resulting coefficients from the LMS are 16-bits. The point in the design that has the largest resolution is in the coefficient accumulators since they utilize 24 bits. To minimize the resolution in both the parallel FIR filter and LMS algorithm the $1/N$ scaling that is required for each FFT-IFFT pair is performed inside the FFT. This helps to minimize the bit width of the parallel frequency domain tap delay line buses in the FIR filter and LMS algorithm. Applying $1/N$, where N is the length of the FFT, scaling in the FFT will cause αI , rate of convergence coefficient, to be scaled by a scale factor of $1/N^2$. To undo this scaling affect due to placing the $1/N$ scale factor in the FFT is to scale αI by N^2 . Assuming a no coefficient pipeline delay, scaling αI by N^2 causes the equalizer to consistently converge.

However, the parallel equalizer is designed with a pipelined architecture. The parallel equalizer approximately a 52-clock

cycle delay from the first valid sample entering the parallel FIR filter, before the first new coefficient values are calculated from the parallel LMS algorithm. It has been shown through simulation only, that scaling $\mathbf{a}l$ by N^2 with a pipeline delay of 52-clock cycles cause the equalizer to become unstable and diverge. Again, using simulation results, it is best that $\mathbf{a}l$ not be scaled by N^2 . Not scaling αl with a known coefficient pipeline delay will result in the equalizer to consistently converge. It is beyond the scope of this paper to mathematically show the effects of the coefficient delay due to a pipeline architecture. The reader should consult [6] on determining the optimal value of $\mathbf{a}l$.

The hardware implementation of the $\mathbf{a}l$ multiplier was chosen to require that $\mathbf{a}l$ is a power of two. This multiplier is implemented as a left or right shift register instead of as a real or complex multiplier.

A Xilinx optimization made to the design, was to implement the pipeline delay between the parallel FIR filter frequency domain tap delay line values and LMS algorithm sub-correlator in the Xilinx Virtex 2 embedded LUT's. Since this is a large pipeline delay, delaying 48 14-bit values 23-clock cycles, using registers that used Xilinx slices would not allow this design to fit in the Xilinx 8-million gate FPGA. This 23-clock cycle pipeline delay was implemented using a shift register CoreGen core that was set to use the on-chip memory LUT's. These values that are delayed from the parallel FIR filter are delayed so they can be multiplied by their equivalent frequency domain error value inside the LMS sub-correlator.

The portion of the parallel equalizer design that is the most sensitive to timing is the vector decimation clock. For the parallel equalizer, the decimation clock is determined by the following equation:

$$\downarrow d_v = \frac{2(M+1)}{N} \quad \text{Eq 1.}$$

Where M is the tap length of the equalizer and N is the length of the FFT-IFFT pairs. For an 11-tap equalizer, using Eq. 1, the decimation clock is the system clock divided by 3. This requires a divide by 3 clock to perform the vector decimation in the LMS sub-correlator and in the coefficient accumulators. One other design issue added to this clock generation is the data valid signal. The decimation clock has to be generated when the data valid signal is high, i.e. the current complex vector is valid data. The design of the decimation clock does not use a digital clock manager (DCM). The clock circuit uses a counter, comparator, and both the rising edge and falling clock edges of the system clock. This circuit generates a clock pulse that has the width of one period of the system clock

with a period of the system clock divided by 3. This pulse is generated using the falling clock edge of the system clock. Another words this is a divide by 3 clock, on the falling edge of the system clock, with a 33% duty cycle if data valid is always high. When data valid is not high, the circuit qualifies the data valid with the operation of the counter and comparator. The data valid signal effectively controls the duty cycle of the decimation clock since when the complex vector is not valid the duty cycle of the decimation clock shrinks. This circuit is designed this way to make this circuit design scale as the number of taps of the equalizer increase or decrease. The only logic components that will change as the tap length changes are the counter and the comparator. As the tap length of the equalizer increases, the duty cycle of the decimation clock will decrease but will always pulse high for one period of the system clock on the system clock's falling edge. This decimation clock architecture allows for a variable tap length equalizer to be design.

The equalizer is designed to interface to a digital receiver that output 4 10-bit symbols per clock. The equalizer is designed to handle demodulated I and Q for QPSK and 16-QAM signals. The part of the design that limits it to processing QPSK or 16-QAM signals is in the design architecture of the decision directed reference signal, $d[n]$. The I and Q symbol values that are set in the modulator can be loaded into the decision directed reference signal state machine. The symbol vales used to construct $d[n]$ are 15 bits. A state machine des threshold comparisons to the received I and Q data symbols to generate the reference symbols of what should have been received by the receiver since the equalizer knows what symbols are in modulator. This design allows the equalizer to not have to use a training sequence. The $d[n]$ state machine can be easily modified to extend the parallel equalizer's capability to process BPSK, 8-PSK, 16-PSK, 32-QAM, 64-QAM or any other type of known signal.

The parallel equalizer has a lock detector circuit that constantly monitors an average error signal over a set period of time. This is used to determine if the equalizer has converged and operating on actual data. The equalizer is designed to run independently of a receiver. If the receiver has not acquired and started to produce the demodulated I and Q symbol streams, the equalizer will be averaging a large error signal that will be greater than the set threshold. When this occurs, the equalizer will reset itself after every integration period of the average error signal until the average error is less than the determined threshold. Once the error is less than the threshold, this indicates that the equalizer has converged and is tracking the signal characteristics of its input signal. The coefficients at this point should have converged to the same will be to adaptively change based on the input I and Q symbols. The lock detector will also check against false lock or loss of lock by consecutively verifying the average error each integration

period a set number of iterations after the equalizer goes into lock or out of lock. The lock detector has the ability to lock the coefficients out of the LMS algorithm after a defined number of consecutive lock periods. When the coefficients are locked, if the equalizer goes out of lock, the adaptive coefficients are no longer held.

The parallel equalizer design was synthesized on a Sun Ultra 60 workstation with 2GB of RAM. It took the Xilinx FPGA synthesis tools approximately 12 hours to completely synthesize the design from translate to place and route to generating the back annotated timing simulation model. The Xilinx FPGA synthesizer was configured for high effort with a timing constraint set on the global system clock of 18ns, approximately 55.5MHz. No constraints were set for the two decimation clocks. It was left up to the place and route tool to determine if the decimation clocks where to be routed as global clock or not. The decimation clocks will be running a factor 3 slower than the global system clock for this design. Figure X shows the FPGA real estate usage for the Xilinx Virtex 2 8-million gate FPGA. Table X shows the synthesized parallel equalizer's time specifications.

Xilinx FPGA Logic Element's	Total	% Usage
Slices	23,310 out of 46,592	50%
Slice Registers	33,024 out of 93,184	35%
4-input LUT's	22,202 out of 93,184	23%
IOB's	426 out of 824	51%
18x18 Multipliers	144 out of 168	85%
Global Clocks	2 out of 16	12%
Gate Count	1,142,483	

Table 1. Xilinx Virtex 2 8-million gate FGPA device usage (xc2v8000-5-ff1152).

Constraint	Requested	Actual	Logic Levels
System Clock	18 ns	17.654 ns	3
LMS sub-correlator Decimation clock	18 ns	6.174 ns	2
LMS Coefficient Decimation clock	18 ns	7.038 ns	13

Table 2. Place and Route Time Constraints.

By examining Tables 1 and 2, the parallel equalizer has met the desired timing requirements. Note that the Xilinx synthesis tool places the LMS coefficient decimation clock on a global clock along with the main global system clock. The LMS sub-correlator decimation clock has not been placed on a global clock buffer. Table 2 shows number of logic levels associated with each clock. With a system clock of 55 MHz, the 11-tap parallel equalizer is capable if processing a symbol rate of 220Mega-Symbols per

second. For QPSK, the maximum bit rate is 440Mbps and for 16-QAM, the maximum bit rate is 880Mbps.

This FPGA effort illustrates that the high end Xilinx FPGA's are capable of performing on the performance level of ASIC's. Note that the target speed of 75 MHz with 15-taps was desired but could not be attained in the high end Xilinx Virtex 2 due to a shortage of multipliers. This design was performed before the Xilinx Virtex 2 pro was released. Since the Virtex 2 pro has high speed I/O and more embedded on-chip multipliers (300+), a 15-tap parallel equalizer could live in a high end Virtex 2 pro. The problem with the Virtex 2 pro is that you plenty of on-chip multipliers, but you may risk running out of slices. As you approach the maximum number of slices less routing resources may be available to route the internal parallel busses. By examining the equalizer architecture, the number of routed parallel busses is staggering and the length of these buses increases as the tap length increases.

VI. Conclusion

We have presented a recently developed fast LMS frequency domain equalizer architecture employing sub-convolution and the FPGA realization. The new architecture employs simple design techniques for designing parallel filter bank architectures based on the concept of separating a convolution into what we call sub-convolutions. These techniques allow arbitrarily long convolution or correlation to be performed using the overlap-and-save method with virtually any FFT-IFFT length; the FFT-IFFT length may be chosen based solely upon the rate reduction desired. We have shown that the lower bound on FFT-IFFT lengths in the fast LMS algorithms created by tap length is removed using sub-convolution and sub-correlation techniques.

The major design considerations of the VLSI architecture realization in a Xilinx ??? FPGA were presented. The gate count, and speed of the Xilinx FPGA were 300K gates (??? CLBs), and 75 Mhz. The non-fractionally spaced equalizer architecture presented operates at 1/4th the input symbol rate, uses 8-point FFT-IFFTs, has 16 coefficients and is extendable to higher tap lengths. Given the 75Mhz clock rate the equalizer can process and output in real time 300 Mega-symbols per second; this results in 600 Mbps assuming QPSK modulation (and variations such GMSK) and 1.2 Gbps assuming 16-QAM. A complexity comparison between this equalizer architecture and the traditional frequency domain fast LMS equalizer is given in [11].

References

- [1] W. Schober, F. Lansing, K. Wilson: JPL, E. Webb: NASA/GSFC, "NASA High Rate Instrument Study," National Aeronautics and Space Administration, Jet Propulsion Laboratory, JPL Publication 99-4, Jan., 1999

- [2] R. Sadr, P.P. Vaidyanathan, D. Raphaeli, S. Hinedi, "*Parallel Digital Modem Using Multirate Filter Banks*," JPL Publication 94-20, Jet Propulsion Laboratory, Pasadena, CA, August 1994.
- [3] Andrew A. Gray, "*Very Large Scale Integration Architectures for Nyquist-Rate Digital Communication Receivers*", PhD Dissertation, University of Southern California, Los Angeles, CA, May 2000
- [4] Gerald Grebowski, Andrew A. Gray, Meera Srinivasan, "Method and Apparatus for High Data Rate Demodulation", US Patent 6,177,835, Jan. 23, 2001
- [5] Andrew A. Gray, Meera Srinivasan, Marvin Simon, Tsun-Yee Yan, "*Flexible All-Digital Receiver for Bandwidth Efficient Modulation*", Int. Telemetry Conference, Nov. 1999
- [6] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Upper Saddle River, New Jersey 07458
- [7] John J. Shynk, "*Frequency-Domain and Multirate Adaptive Filtering*," IEEE Signal Processing Magazine, January 1992.
- [8] John Proakis, *Digital Communications*, McGraw-Hill, Inc. New York, 1995.
- [9] Theodore Rappaport, *Wireless Communications*, PTR Prentice Hall, Englewood Cliffs, NJ, 1996.
- [10] A.V. Oppenheim, R. W. Schaffer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey 1989.
- [11] Andrew Gray, Scott Hoy, Parminder Ghuman, "Parallel VLSI Equalizer Architectures for Multi-Gbps Satellite Communications", IEEE GlobeComm, Nov. 2001.
- [12] Andrew Gray, Parminder Ghuman, Scott Hoy, "Multi-Gbps 16-QAM All-Digital Receiver", International Telemetry Conference, Oct. 2001.