Design and Prototype of Singular Value Decomposition Hardware in IEEE 802.11n MIMO Standards for Software Defined Radio

Yue Wang, Kevin Cunningham, and Prawat Nagvajara
Electrical and Computer Engineering
Drexel University
Philadelphia, PA
yw73@drexel.edu, kac89@drexel.edu, nagvajara@ece.drexel.edu

Jeremy Johnson
Computer Science
Drexel University
Philadelphia, PA
jjohnson@cs.drexel.edu

***Abstract*—This paper presents a custom reconfigurable hardware design based on Software Defined Radio (SDR) for Pre-coding problems of the IEEE 802.11n standard. The pre-coding technique of Multiple-Input-Multiple-Output and Orthogonal Frequency Division Modulation (MIMO-OFDM) systems, such as the IEEE 802 standards, requires to solve a sequence of complex-valued Singular Value Decompositions (SVD) to achieve maximum throughput and strong signal strength. The implementation of SVD developed for 2×2 matrix over complex fixed-point integer and its 4×4 expansion achieves an optimum pipeline rate which equaled the maximum hardware clock rate. The proposed prototype architecture based Field-Programmable Gate Array (FPGA) provides performance gains over standard software libraries, such as the ZGESVD function of Linear Algebra PACKage (LAPACK) library, when running on standard processors.**

***Index Terms*—FPGA, IEEE 802.11n, Jacobi Rotation, MIMO, OFDM, Pipeline, SDR, SVD.**

I. Introduction

A. Background

Software Defined Radio (SDR) requires the implementation of flexible software to meet the requirements of computationally intensive algorithms and power constrained performance. To achieve such flexibility, special purpose hardware, such as Field Programmable Gate Array (FPGA), is used. An example of SDR hardware is the Wireless Open Access Research Platform (WARP) from Rice University [17] in which utilizes FPGA to program its radio transceiver in software.

Multiple Input Multiple Output (MIMO) is a smart antenna technology that uses multiple transmit and receive antennas to increases channel capacity by transmitting multiple data streams over one frequency. The MIMO standards include WLAN IEEE 802.11n, WiMAX IEEE 802.16-2004, WiMAX IEEE 802.16e, 3GPP Release 7, 3GPP Release 8 (LTE) standards [18]. Data shroughput will increase as the number of data streams increases. With Spatial Multiplexing technique, MIMO systems such as the IEEE 802.11n standard can increase the spatial data throughput of the channel. Using Alamouti Space-Time code, MIMO system can also improves signal quality by transmitting redundancy. Spatial multiplexing is suitable for near-field communication and spatial diversity for far-field communication. With higher spectral efficiency and reduced fading, a MIMO system is able to increase link range and data throughput of the communication without additional power and bandwidth [2, 18]. Figure 1 is an illustration of 2×2 MIMO system [18].

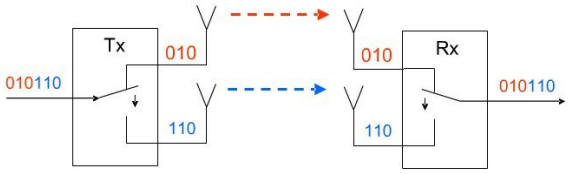

Figure 1. MIMO System [18]

Orthogonal Frequency Division Multiplexing (OFDM) is a Frequency Division Multiplexing (FDM) scheme utilized as a digital multi-carrier modulation method. Multi-carrier modulation divides a broadband channel into narrowband sub-channels. OFDM uses a large number of closely-spaced orthogonal narrowband sub-carriers instead of a single wideband carrier to transport data. The data is divided into several parallel data streams or channels, one for each sub-carrier. In an OFDM system, a single data-stream is transmitted over lower data rate sub-carriers as a coded quantity at each frequency carrier in the same bandwidth. OFDM is very easy and efficient in dealing with multi-path. OFDM is robust against narrowband interference and frequency selective fading. Incorporated with MIMO, OFDM is a promising technology for higher capacity multi-hop networks [4, 18, 7].

Singular Value Decomposition (SVD) of the channel characteristic matrix is used in pre-coding, equalization and beamforming for MIMO and OFDM communication systems (e.g., IEEE 802.11n) to efficiently arrange the setup of the data streams. The SVD problems of MIMO and OFDM systems such as the IEEE 802.11n standard are computationally in-

[1]This material is based on research partially sponsored by the IEEE Mini Grant and the National Science Foundation under grant numbers 0854946 and 0923003.

tensive and complex [4]. Custom hardware can be used to improve the computation. The set of sub-carrier matrices of the channel characteristic matrix is sent to the custom hardware by software. The custom hardware returns the corresponding stream of singular values (S or σ) and unitary matrices (U and V) that represents the power and direction of the transmit signals per sub-carriers. SVD hardware provides the opportunity for a significant performance gain over the traditional SVD computed by software. A illustration is shown in Fig. 2.

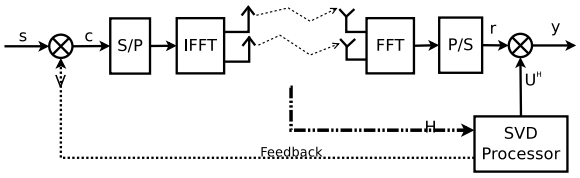

Figure 2. SVD-based Pre-coding Scheme for 2 × 2 MIMO Systems

B. The Requirements of IEEE 802.11n Standard for SVD

IEEE 802.11n is a improved standard over the 802.11a and 802.11g with increment of transmitting throughput. Comparing with previous WLAN standards (802.11a and 802.11g), one of the advantage of IEEE 802.11n standard is MIMO implementation of wide bandwidth options (40 MHz comparing with 20 MHz). For 2 transmit and 2 receive antennas to form 2 data streams (2×2 : 2 configuration), IEEE 802.11n can transmit maximum 300 Mbit/s with 40 Mhz Channel [19]. To accurately predict the strength and direction of the signal for better signal quality and reception, SDR is required to solve a sets 2×2 SVD problems provided by the sub-carriers of the channel. With 4 transmit and 4 receive antennas to form 4 data streams, IEEE 802.11n standard will be able to achieve 600 Mbit/s transmission with 40 MHz Channels and 64 Quadrature Amplitude Modulation (QAM) modulation [19]. Similar to 2×2 : 2 configuration, 4×4 : 4 setup require SDR to solve a set of 4×4 SVD problems.

The transmitter sends the Request-To-Send (RTS) packet containing a training sequence from which the channel characteristic matrices H can be measured at the receiver. The receiver sends (feedback) the Clear-To-Send (CTS) packet containing the SVD of H matrices. The computation time of the IEEE 802.11n standard SVD problems is limited by the Short Inter-Frame Spacing (SIFS), which is the time interval between (RTS) data and CTS (acknowledgement). SIFS is defined 10 us for a 2.4 GHz carrier frequency and 16 us for a 5 GHz carrier frequency [19]. Since SVD computation can start before as soon as the stream of H matrices are available, the computation time for SVD can be estimated more than the defined, hence, 20 us. The time constraint set a maximum data processing time for the SVD computation per transmission. The receiver is required to compute the SVD of the sub-carrier H matrices conservatively within the time constraint.

C. SVD Hardware for MIMO-OFDM in SDR

SDR boards, such as WARP, use an FPGA [17] for reconfigurable radio software. SVD hardware can utilize the unused space of the FBGA. The architecture shown in Fig. 3 supports SVD computation of both 2×2 and 4×4 matrices. The Main Control Unit (MCU) streams the input data from memory to the 2×2 SVD hardware, which returns a stream of the singular values and corresponding unitary 2×2 matrices. When solving 4×4 SVD problems, the MCU iterates a sequence of predefined 2×2 blocks along the iteration and sweep data path until the final result converges.

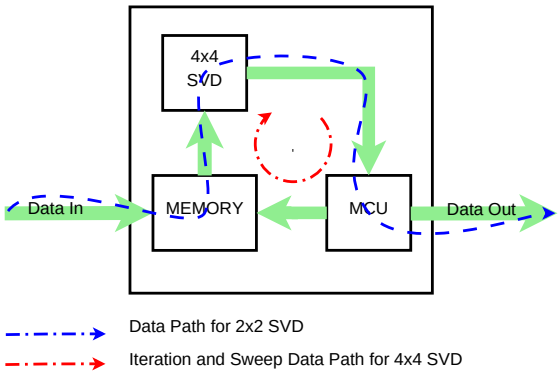

Figure 3. SVD Hardware in Software-Defined Radio Transceiver

II. SVD Theory in MIMO Communication Systems

A. SVD-based MIMO Communication Systems

In SVD-based MIMO-OFDM communication systems, 2×2 (or 4×4) SVD of the sub-carriers of the channel characteristic matrices H are periodically computed. A general channel model of MIMO system can be expressed as,

$$r = \mathrm{H}s + n, \tag{1}$$

where s is the sent signal, r is the received signal, H is the channel attenuation and n is the noise in the channel. These signals are vectors whose components are the sent and received modulation symbols, a component for each antenna. These symbols are complex numbers when using modulation modes such as QAM. H is a matrix whose elements are complex numbers, attenuations of both magnitude and phase. A time-invariant channel model is achieved by estimating H periodically using a pre-defined training sequence transmitted between the transmitter and the receiver. The received signal at the receiver antenna is expressed in terms of the components of H, e.g., in a 2×2 system, $r_1 = h_{11}s_1 + h_{12}s_2 + n_1$ and $r_2 = h_{21}s_1 + h_{22}s_2 + n_2$. Separation of the components of s into the corresponding components of r is done by pre-coding and equalization techniques based on SVD.

For systems with N_t transmitter antennas and N_r receiver antennas, the singular value decomposition of a matrix H $(N_r \times N_t) \in C^{m \times n}$ is

$$\mathrm{H} = U \Sigma V^H, \tag{2}$$

where $U \in C^{m \times m}$ and $V \in C^{n \times n}$ are unitary ($U^H U = I$ and $V^H V = I$). $\Sigma \in R^{m \times n}$ is the diagonal matrix with the singular values of $\mathrm{H}^H\mathrm{H}$, σ_i, arranged in decreasing order. Most telecommunication literature uses the superscript ^{H} to denote

conjugate transpose of a matrix (Hermitian) even though H also denotes the channel characteristic matrix.

SVD-based pre-coding techniques make use of the fact that a column of V is an Eigenvector of $\mathrm{H}^H\mathrm{H}$, which corresponds to an Eigenmode of the communication channel. For instance, singular value σ_i defines the quality of the i^{th} Eigenmode. The pre-coding technique transmits the matrix-vector product, $p = V \cdot s$, which is called preprocessing. From Eq. 1, the received signal becomes

$$r = \mathrm{H}p + n = \mathrm{H}Vs + n \tag{3}$$

The classical SVD-based equalization technique to separate r into Eigenmodes is the multiplication of U^H with r

$$y = U^H r = \Sigma(s) + U^H n. \tag{4}$$

In other words,

$$y_i = \sigma_i s_i + n_i;\ i = 1, \ldots, N_r, \tag{5}$$

where s_i denotes the i^{th} component of signal s. The multiplication of U and V (pre-coding and equalization) does not change the bit-error rate (BER) calculation because they are unitary. Therefore, the performance analysis is similar to the parallel SISO communication systems. Figure 2 shows the SVD-based pre-coding scheme. By periodically transmitting training sequences, the system obtains the H matrices of the sub-carriers and computes the singular values, U and V. The receiver transmits back the V matrices and stores the U matrices used for equalizing the pre-coded received signal synchronized to the corresponding V.

Other SVD-based equalization techniques include zero-forcing and Minimum-Mean Square Error (MMSE) equalization [4].

B. SVD Algorithm

Previous SVD works has focused on general $n \times n$ matrices using the Brent-Luk-Van Loan systolic array, where the matrix size, n, tends to be a large number. None of the previous proposed hardware has provided a pipelined architecture for computing complex value 2×2 singular values and the unitary matrices, U and V. The Golub-Kahan-Reinsch SVD algorithm implemented in LINPACK/LAPACK and used in MATLAB, is the most commonly used algorithm for the software computation of SVD in a uniprocessor system [9]. The algorithm first uses Householder bi-diagonalization to bi-diagonalize the original matrix. The result is then iteratively diagonalized. The time complexity of Golub-Kahan-Reinsch SVD algorithm is $O(mn^2)$ [9].

The SVD algorithm used in the MIMO system is modified from the two-sided Jacobi algorithm used by Hemkumar [10, 11]. The algorithm developed from the cyclic Jacobin algorithm originally proposed by Forsythe and Henrici in the 1960s [8]. Performing SVD computation with CORDIC (Coordinate Rotation Digital Computer) was proposed by Cavallaro and Luk [5]. In 1983, the BLV systolic array was proposed for general $n \times n$ matrices with a time complexity of O(m + nlog(n)) and hardware complexity of O(n^2) processors [3]. The two-sided Jacobi algorithm has also been used to solve SVDs of a quaternion matrix [14]. Researchers have recently been interested in solving SVD problems in MIMO and OFDM systems [22, 4]. The algorithm used in this paper was from Hemkumar [11].

Consider a 2×2 matrix A with complex elements in rectangular representation for the sub-carriers of a 2×2 MIMO system.

$$A = \begin{bmatrix} (a_r, a_i) & (b_r, b_i) \\ (c_r, c_i) & (d_r, d_i) \end{bmatrix} \tag{6}$$

Transform A into polar representation,

$$A = \begin{bmatrix} A\mathrm{e}^{i\theta_a} & B\mathrm{e}^{i\theta_b} \\ C\mathrm{e}^{i\theta_c} & D\mathrm{e}^{i\theta_d} \end{bmatrix} \tag{7}$$

Perform unitary $\Re$ transformation to transform the elements of the second row to real numbers,

$$\begin{bmatrix} \mathrm{e}^{i\theta_\alpha} & 0 \\ 0 & \mathrm{e}^{i\theta_\beta} \end{bmatrix} \begin{bmatrix} A\mathrm{e}^{i\theta_a} & B\mathrm{e}^{i\theta_b} \\ C\mathrm{e}^{i\theta_c} & D\mathrm{e}^{i\theta_d} \end{bmatrix} \begin{bmatrix} \mathrm{e}^{i\theta_\gamma} & 0 \\ 0 & \mathrm{e}^{i\theta_\delta} \end{bmatrix} = \begin{bmatrix} A\mathrm{e}^{i\theta_{a'}} & B\mathrm{e}^{i\theta_{b'}} \\ C & D \end{bmatrix} \tag{8}$$

$$\text{where } \theta_\alpha = \theta_\beta = -\frac{\theta_d + \theta_c}{2}$$

$$\theta_\gamma = -\theta_\delta = \frac{\theta_d - \theta_c}{2}.$$

Perform the two-sided Jacobi rotations, where c and s denote cosine and sine, to annihilate A(2,1),

$$\begin{bmatrix} \mathrm{c}_{\theta_\phi} & -\mathrm{s}_{\theta_\phi} \\ \mathrm{s}_{\theta_\phi} & \mathrm{c}_{\theta_\phi} \end{bmatrix} \begin{bmatrix} A\mathrm{e}^{i\theta_{a'}} & B\mathrm{e}^{i\theta_{b'}} \\ C & D \end{bmatrix} \begin{bmatrix} \mathrm{c}_{\theta_\psi} & \mathrm{s}_{\theta_\psi} \\ -\mathrm{s}_{\theta_\psi} & \mathrm{c}_{\theta_\psi} \end{bmatrix} = \begin{bmatrix} W\mathrm{e}^{i\theta_w} & X\mathrm{e}^{i\theta_x} \\ 0 & Z \end{bmatrix} \tag{9}$$

where $\theta_\phi = 0$, and $\theta_\psi = tan^{-1}(\frac{C}{D})(-\pi \leq \theta_\psi \leq \pi)$.

Perform unitary $\Re$ transformation again to transform the elements of the first row of the above result to real values,

$$\begin{bmatrix} \mathrm{e}^{i\theta_\xi} & 0 \\ 0 & \mathrm{e}^{i\theta_\eta} \end{bmatrix} \begin{bmatrix} W\mathrm{e}^{i\theta_w} & X\mathrm{e}^{i\theta_x} \\ 0 & Z \end{bmatrix} \begin{bmatrix} \mathrm{e}^{i\theta_\zeta} & 0 \\ 0 & \mathrm{e}^{i\theta_\omega} \end{bmatrix} = \begin{bmatrix} W & X \\ 0 & Z \end{bmatrix}$$

where

$$\theta_\xi = -\frac{(\theta_x + \theta_w)}{2},\ \theta_\omega = \frac{\theta_w - \theta_x}{2},\ \theta_\eta = \theta_\zeta = \frac{\theta_x + \theta_w}{2}. \tag{10}$$

Perform the two-sided Jacobi rotations on the above to

annihilate A(1, 2),

$$\begin{bmatrix} c_{\theta_\lambda} & -s_{\theta_\lambda} \\ s_{\theta_\lambda} & c_{\theta_\lambda} \end{bmatrix} \begin{bmatrix} W & X \\ 0 & Z \end{bmatrix} \begin{bmatrix} c_{\theta_\rho} & s_{\theta_\rho} \\ -s_{\theta_\rho} & c_{\theta_\rho} \end{bmatrix} = \begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix} = \Sigma \quad (11)$$

$$\text{where } \tan(\theta_\lambda + \theta_\rho) = -(\frac{X}{Z-W})$$

$$\tan(\theta_\lambda - \theta_\rho) = -(\frac{X}{Z+W})$$

$$U^H = \begin{bmatrix} c_\lambda & -s_\lambda \\ s_\lambda & c_\lambda \end{bmatrix} \begin{bmatrix} e^{i\theta_\xi} & 0 \\ 0 & e^{i\theta_\eta} \end{bmatrix} \begin{bmatrix} e^{i\theta_\alpha} & 0 \\ 0 & e^{i\theta_\beta} \end{bmatrix} \quad (12)$$

$$V = \begin{bmatrix} e^{i\theta_\gamma} & 0 \\ 0 & e^{i\theta_\delta} \end{bmatrix} \begin{bmatrix} c_\psi & s_\psi \\ -s_\psi & c_\psi \end{bmatrix} \begin{bmatrix} e^{i\theta_\zeta} & 0 \\ 0 & e^{i\theta_\omega} \end{bmatrix} \begin{bmatrix} c_\rho & s_\rho \\ -s_\rho & c_\rho \end{bmatrix} \quad (13)$$

In other words,

$$U^H A V = \Sigma = \begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix} \quad (14)$$

For the singular value matrix, Σ, it is not guaranteed that P is greater than Q. The normalized SVD (NSVD) algorithm used by Brent et. al. can be used to exchange P and Q if needed [3]. Permutation matrices can also be used to permute P and Q. The third technique to exchange P and Q is obtained by using $(\theta_\lambda + \frac{\pi}{2})$ and $(\theta_\rho + \frac{\pi}{2})$ instead of θ_λ and θ_ρ in Eq. 11.

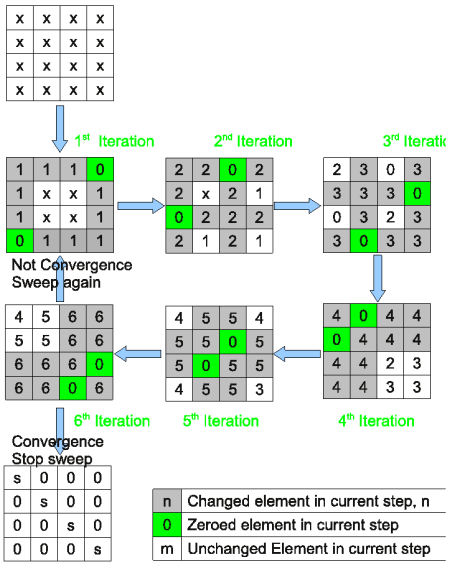

Figure 4. Iteration and Sweep Processes of 4×4 SVD

As shown in Fig. 4, the SVD of a 4×4 complex matrix can be divided into six back-to-back 2×2 SVDs. Each 2×2 SVD is called one iteration. The purpose of each iteration is to annihilate the off-diagonal elements A(i,j) and A(j, i). After six iterations all the off-diagonal elements have been annihilated once. With each iteration, however, the elements in i^{th} and j^{th} columns and rows of A will be affected by the transformation, namely, pre and post multiplication by 2×2 U^H and V of (A(i,i), A(i,j), A(j,i), A(j,j)) sub-matrix. The six 2×2 SVDs constitute a single sweep. Multiple sweeps have to be performed to converge to the diagonal matrix Σ. For 4×4 SVD, 3 sweeps are sufficient enough for an accurate result for MIMO system.

III. Reconfigurable SVD Pipeline Hardware for IEEE 802.11n

This section presents the custom design SVD hardware for IEEE 802.11n MIMO system of SDR prototyped on a FPGA. The implementation of the two-sided Jacobi algorithm was based on fixed-point signed fraction arithmetic (QN format). Xilinx CORDIC V4.0 Coregen was used under Xilinx ISE Project Navigator environment for the Givens Rotation and conversion between polar and rectangular representation. Because of the well-defined dynamic range of the 1Q15 data for the MIMO application, the use of fixed-point computations is acceptable without a loss of accuracy or precision. The CORDIC cores produce verified correct results with an implementation efficient in both size and power. The translation cores were used to convert rectangular to polar representation whereas the rotation cores were used to convert polar to rectangular representation.

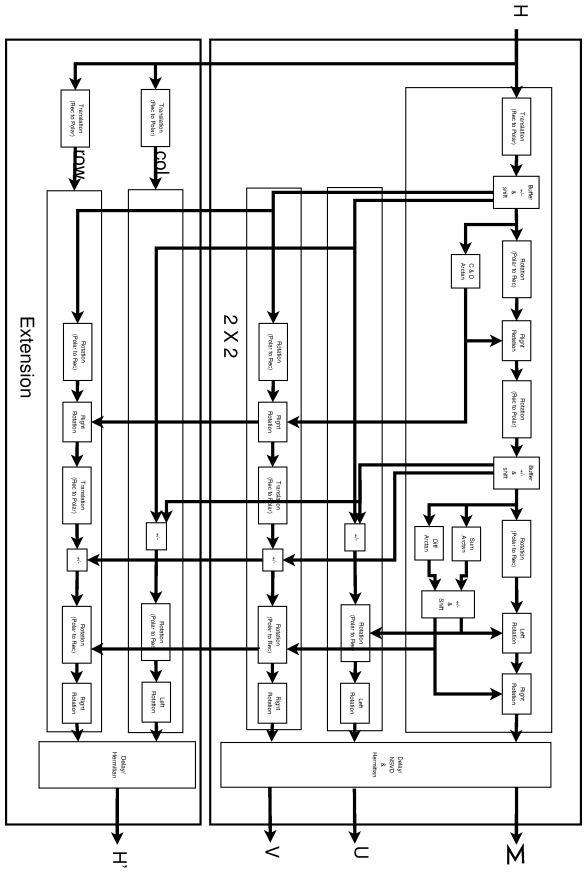

Figure 5. The RTL Diagram of the 4×4 SVD Pipeline Hardware

Fig. 5 shows a block diagram of the pipeline hardware comprising rotation, translation and arctangent CORDIC cores and, custom cores (+/- denotes add, subtract and multiplexer). The diagram is the graphic representation VHDL code for the mathematic algorithm listed in Sec II. The VHDL coding of the SVD algorithm has used the IEEE 1164 standard logic, arithmetic and unsigned number library. The top-row block of the 2×2 is only compute Σ for the 2×2 MIMO systems. The bottom two rows of the 2×2 block (pipelines) compute U^H and V respectively. The extension block is required to compute for the 4×4 MIMO systems. Table I lists the

	CORDIC cores			Custom cores
	Rotation	**Translation**	**Arctan**	**+/- shifter**
Σ	20	8	3	15
U	6	0	0	2
V	12	6	0	4
Row	8	4	0	8
Col	12	12	0	8

Table I
SPACE UTILIZATION OF 4×4 SVD PROCESSOR

space complexity for the proposed SVD hardware measured by the number of cores used according to Fig. 5, where +/- and shift denotes add/subtract and bit-shift to conform to the 1QN or 2QN format as magnitude or phase angle inputs to CORDIC cores. The design included guard bits and rounding bits as precision provisions. The prototype implemented on Virtex 4 xc4vlx200 used approximately 40% of the slices. The CORDIC stages and the $\Re$ transformation resulted in a 173 clock cycle latency for 16-bit data synthesized on the ML605 board and a 138 clock cycle latency for 12-bit data synthesized on the XUP5 board according to simulation results from Modelsim.

IV. VERIFICATION AND BENCHMARK RESULTS

One hundred sets of IEEE 802.11n channel matrices H were provided by the SDR team of Drexel University. The data were in double floating-point format. Each of the channel matrix consisted of fifty-two 2×2 decoupled blocks. The data were converted into MQ(N-1) fixed-point signed fraction format. The set of data were streamed to the SVD processor. A total of 5200 2×2 SVDs were verified. The same set of data were also modified to be tested for solving the 4×4 SVD.

The hardware model of the two-sided Jacobi algorithm was first programmed in MATLAB m-code. This custom m-coded SVD function was verified against the MATLAB SVD function, [U, Σ, V] = svd(H), for correctness. It was verified that Σ was the same for both the m-code function and MATLAB SVD function. Since the MATLAB SVD function uses the Golub-Kahan-Reinsch SVD algorithm [9], U and V were different from the hardware model. The correctness of U and V was verified by comparing the original H and UΣV^H. The discrepancy was at most two percent.

There were also mismatches between the MATLAB hardware model and the experimental data from Modelsim and the FPGA board due to the fact that MATLAB data are double precision floating-point and the experimental data were 2Q17 fixed-point. The maximum percentage difference was less than two percent, more than sufficient for radio applications.

The SVD processor design was prototyped on the Digital Reconfigurable Computing (DRC) board (RPU110-L200). The software running on the host PC streamed the test data via the Hyper Transport bus to the input FIFOs in the FPGA (Virtex 4). When the input FIFOs were full, the SVD core was enabled to process the data in a stream fashion. The results (Σ, U and V) were streamed into the output FIFOs. The results were written into DDRII DRAMs through a DDR2 controller, to the CPU, and to an output file for verification. The artitecture of the prototype is shown in Fig. 6.

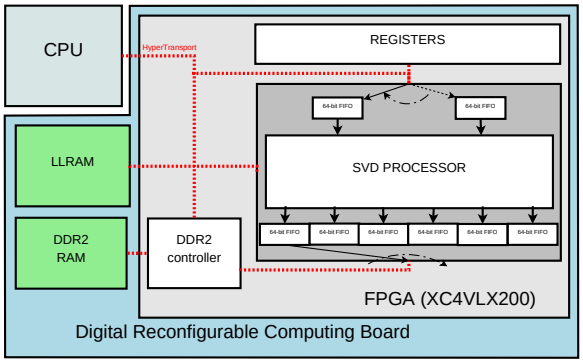

Figure 6. The RTL Diagram of the 4×4 SVD Pipeline Hardware

The performance, in terms of data rate, was compared to the ZGESVD function of Linear Algebra PACKage (LAPACK). The ZGESVD function is based on the Golub-Kahan-Reinsch SVD algorithm. Intel's Math Kernel Library (MKL) version 10.2.5.035 includes LAPACK 3.2. The ZGESVD function processes double precision floating-point data.

The test data from above were used for benchmarking. The benchmarking code was based on Intel's MKL LAPACK examples [6]. The input data required approximately 400 KB cache. With -O3 gcc compiling switch, the computation time of the code on Intel Duo T9300 CPU with clock frequency 2.50 GHz was 0.041832 s. Therefore the data rate was 12.43 MHz for a single core. With -fopenmp and -O3 option for OpenMP C code, the data rate was 20.98 MHz. The average data rate on a Intel Quad Q9300 CPU with clock frequency 2.50GHz was approximately 12 MHz. With OpenMP C code, the data rate is projected to be 40 MHz.

Table II summarizes the data rate attained by software and hardware. It also lists FPGA LUT (Lookup Table) slices for DRC (Virtex 4), Virtex 5 and Virtex 6. For SVD hardware, the frequencies were obtained from the Xilinx ISE synthesis reports. The clock frequency from the post mapping report was slightly different. The benchmark showed a six-fold speedup for Virtex 6 over the OpenMP Quad core. The benchmark results give a convincing argument for using SVD hardware e.g., implemented on an FPGA, in a radio transceiver. In the 802.11n Standard, the required data rate is less than 100 MHz for radio transceivers (see Section II-A). Since the SVD hardware achieves rates greater than the required data rate, it

may be possible to reduce the space of the design by reusing hardware and operating at a lower rate.

Platform	Data Width	Data Rate (Mhz)	Slices
Intel Duo T9300	Double Float	12.43	N/A
Intel T9300 OpenMP	Double Float	20.98	N/A
Intel Quad Q9300	Double Float	12	N/A
Intel Q9300 OpenMP	Double Float	40	N/A
DRC Virtex 4	16	202.47	72669
Virtex 5	12	233.15	54072
Virtex 6	16	323.82	74925

Table II
HARDWARE COMPLEXITY

V. CONCLUSION

The custom designed SVD core described provides high-throughput computation for complex 2×2 pre-coding and equalization schemes from the IEEE 802.11n MIMO standard. The MIMO standards that benefit from the core include IEEE 802.16-2004, IEEE 802.16e, 3GPP release 7, and 3GPP Release 8 (Long Term Evolution). The core attained an optimal pipeline rate, that is, the hardware clock frequency was equal to the data throughput rate. The core computes Σ, U, and V, whereas, previously reported hardware only computes Σ, which is not sufficient for MIMO and OFDM systems.

The developed pipeline SVD hardware attained a higher data rate than the ZGESVD function from LAPACK. A prototype was implemented on the Digital Reconfigurable Computing RPU110-L200. The prototype consisted of 55 CORDIC cores. It achieved an optimum pipeline rate with 173 clock-cycle latency for 16-bit data design (1Q15 fixed-point input format). When implemented on the xc6vlx240t FPGA, the hardware attained six times the rate of ZGESVD running on an Intel Quad Q9300 using OpenMP.

The SVD core is suitable for Software Defined Radio (SDR) applications, whereas high performance CPUs, such as Intel Quad Q9300, or GPUs are unlikely to meet the power and space constraints.

REFERENCES

[1] A. Ahmedsaid, A. Amira, and A. Bouridane. Improved SVD systolic array and implementation on FPGA. In *Field-Programmable Technology (FPT), 2003. Proceedings. 2003 IEEE International Conference on*, pages 35 – 42, 15-17 2003.

[2] Emre Telat Ar and I. Emre Telatar. Capacity of multi-antenna gaussian channels. *European Transactions on Telecommunications*, 10:585–595, 1999.

[3] R. P. Brent, F. T. Luk, and C. Van Loan. Computation of the singular value decomposition using mesh-connected processors. Technical report, Cornell University, Ithaca, NY, USA, 1983.

[4] H. Busche, A. Vanaev, and H. Rohling. SVD-based MIMO precoding and equalization schemes for realistic channel knowledge: Design criteria and performance evaluation. *Wirel. Pers. Commun.*, 48(3):347–359, 2009.

[5] J. R. Cavallaro and F. T. Luk. CORDIC arithmetic for an SVD processor. *Journal of Parallel and Distributed Computing*, 5(3):271 – 290, 1988.

[6] Intel Corporation. Intel(R) Math Kernel Library LAPACK Examples. http://software.intel.com/sites/products/documentation/hpc/mkl/lapack/mkl_lapack_examples/index.htm, 2010.

[7] Mondin, M. Orthogonal Frequency Division Multiplexing Modulation: OFDM. 2008 http://www.infotech.polito.it/it/personale/scheda/(nominativo)/marina.mondin

[8] G. E. Forsythe and P. Henrici. The cyclic Jacobi method for computing the principal values of a complex matrix. *Transactions of the American Mathematical Society*, 94(1):1–23, 1960.

[9] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[10] N.D. Hemkumar. A systolic VLSI architecture for complex SVD. *Rice University Master of Science Thesis*, May 1991.

[11] N.D. Hemkumar and J.R. Cavallaro. A systolic VLSI architecture for complex SVD. In *Circuits and Systems, 1992. ISCAS '92. Proceedings., 1992 IEEE International Symposium on*, volume 3, pages 1061 –1064 vol.3, 10-13 1992.

[12] S. Hsiao and J. M. Delosme. Parallel singular value decomposition of complex matrices using multidimensional CORDIC algorithms. volume 44, pages 685 –697, mar 1996.

[13] Xilinx Inc. Cordic v4.0 datasheet. http://www.xilinx.com/support/documentation/ip_documentation/cordic_ds249.pdf, 2009.

[14] N. Le Bihan and S. J. Sangwine. Jacobi method for quaternion matrix singular value decomposition. *Applied Mathematics and Computation*, 187(2):1265 – 1271, 2007.

[15] M. E. Ma, W.and Kaye, D. M. Luke, and R. Doraiswami. An FPGA-Based singular value decomposition processor. In *Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on*, pages 1047 –1050, may 2006.

[16] , Ove Edfors Magnus and et al. OFDM Channel Estimation By Singular Value Decomposition. IEEE Trans. Commun. 1996. Vol.46 p931-939

[17] P. Murphy. Wireless open-access research platform. http://warp.rice.edu/, 2010.

[18] Schindler. S. Introduction to MIMO systems - Application Note 1MA102. Technical report, Rohde and Schwarz, http://www.rohde-schwarz.de/ps/rus/tools/show_8446_document/MIMO_Eng.pdf, 2006.

[19] IEEE 802.11n: STANDARD for Information Technology - Telecommunications and information exchange between systems - Local and Metropolitan networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 5: Enhancements for Higher Throughput, (October 2009)

[20] V. Strumpen, H. Hoffmann, and A. Agarwal. A stream algorithm for the SVD. Technical report, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2003.

[21] B. Yang and J. F. Böhme. Reducing the computations of the singular value decomposition array given by Brent and Luk. *SIAM J. Matrix Anal. Appl.*, 12(4):713–725, 1991.

[22] C. Zhan, K. Jheng, Y. Chen, T Jheng, and A Wu. High-convergence-speed low-computation-complexity SVD algorithm for MIMO-OFDM systems. In *VLSI Design, Automation and Test, 2009. VLSI-DAT '09. International Symposium on*, pages 195 –198, 28-30 2009.