



LatticeECP/EC FPGAs: A Systolic Array Processor for Software Defined Radio

A Lattice Semiconductor White Paper
April 2005

Lattice Semiconductor
5555 Northeast Moore Ct.
Hillsboro, Oregon 97124 USA
Telephone: (503) 268-8000
www.latticesemi.com

Introduction

FPGAs (field programmable gate arrays) are inherently reconfigurable, providing the scalable, multi-channel and parallel/serial processing required to meet evolving system specifications and standards. Consequently, FPGAs are an ideal platform for Software Defined Radio (SDR). SDRs are radios with a reconfigurable hardware platform that can be used over multiple communications standards. By pushing the receiver's sampling rate close to the antenna, FPGAs allow high-speed adaptation to SDR's full-parallel processing architectures and take advantage of on-chip dedicated high-speed data I/O hardware

A diverse application of SDR technology is found in the United States Department of Defense Joint Tactical Radio Systems (JTRS) program. JTRS is developing hardware that can reconfigure and run multiple channels of various waveforms of different protocols over a broad RF spectrum.^[1] JTRS will provide a family of software programmable radios designed using a Software Communications Architecture (SCA). JTRS provides interoperability through the capability to communicate with current tactical communications near-term. JTRS will also provide integrated information-gathering through multiple modes of different waveforms (modulations) to support joint operations. The radio must therefore be capable of switching among different types of modulated signals, bandwidths, data rates and frame/networking protocols with fast reprogrammability in the field with no hardware changes. Consequently, portability of waveform panels between hardware platforms and operating systems becomes an underlying tenet with JTRS development based on SCA.

Technology trends are being redefined. Ever decreasing silicon geometry is changing the economics of ASICs. The costs of designing complex ASICs threaten to become prohibitive, unless the economics are scaled to address massive markets. Now comes the argument for 'soft silicon' – reprogrammable, reconfigurable devices suited for multiple applications and markets that require low power consumption and the high performance normally associated with ASICs. A range of innovative architectures are emerging, reflecting a departure from long standing assumptions.

The new LatticeECP/EC FPGAs have capabilities that make them promising logic devices for SDR. The devices incorporate the traditional 4-input LUT structure common to FPGAs.

The advantage of the ECP devices is that they are the first low-cost FPGA family with complex DSP block architecture. The PFU (Programmable Function Unit) and PFF (Programmable Fast Function Unit) have four slices, the LSLICE and LFSLICE, respectively. Each slice is individually programmable and the slices, like the PFU/PFF, can be concatenated for wider logic/larger memory. Table 1 shows the modes of the DSP. Here, we have selected the fourth mode of the 18 X 18 configuration. Figure 1 illustrates this configuration.

The LSLICE provides five modes of flexibility:

- Logic
- Ripple
- Single Port RAM
- Dual Port RAM
- ROM

The LFSLICE provides three modes for faster speed and lower cost:

- Logic
- Ripple
- ROM

The LSLICE has distributed memory capability, but the LFSLICE does not. The ratio of number of LSLICES to number of LFSLICES is 1 to 4. The ECP has DSP blocks in addition to the normal FPGA fabric consisting of PFU/PFUs, PLLs, I/O logic and system I/O interfacing. The DSP blocks can provide a dramatic improvement over LUT-based designs. For the ECP 6K, 10K, 15K, 20K and 33K, the numbers of DSP blocks are 4, 5, 6, 7 and 8, respectively.

Table 1. DSP Block Performance

36 X 36 Mode	18 X 18 Mode	9 X 9 Mode
• One Multiplier	• Four Multipliers • Two 52-bit MACs • Two sums of two 18 X 18 multipliers each • One sum of four 18 X 18 multipliers	• Eight Multipliers • Two 34-bit MACs • Four sums of two 9X9 multipliers each • Two sums of four 9 X 9 multipliers each

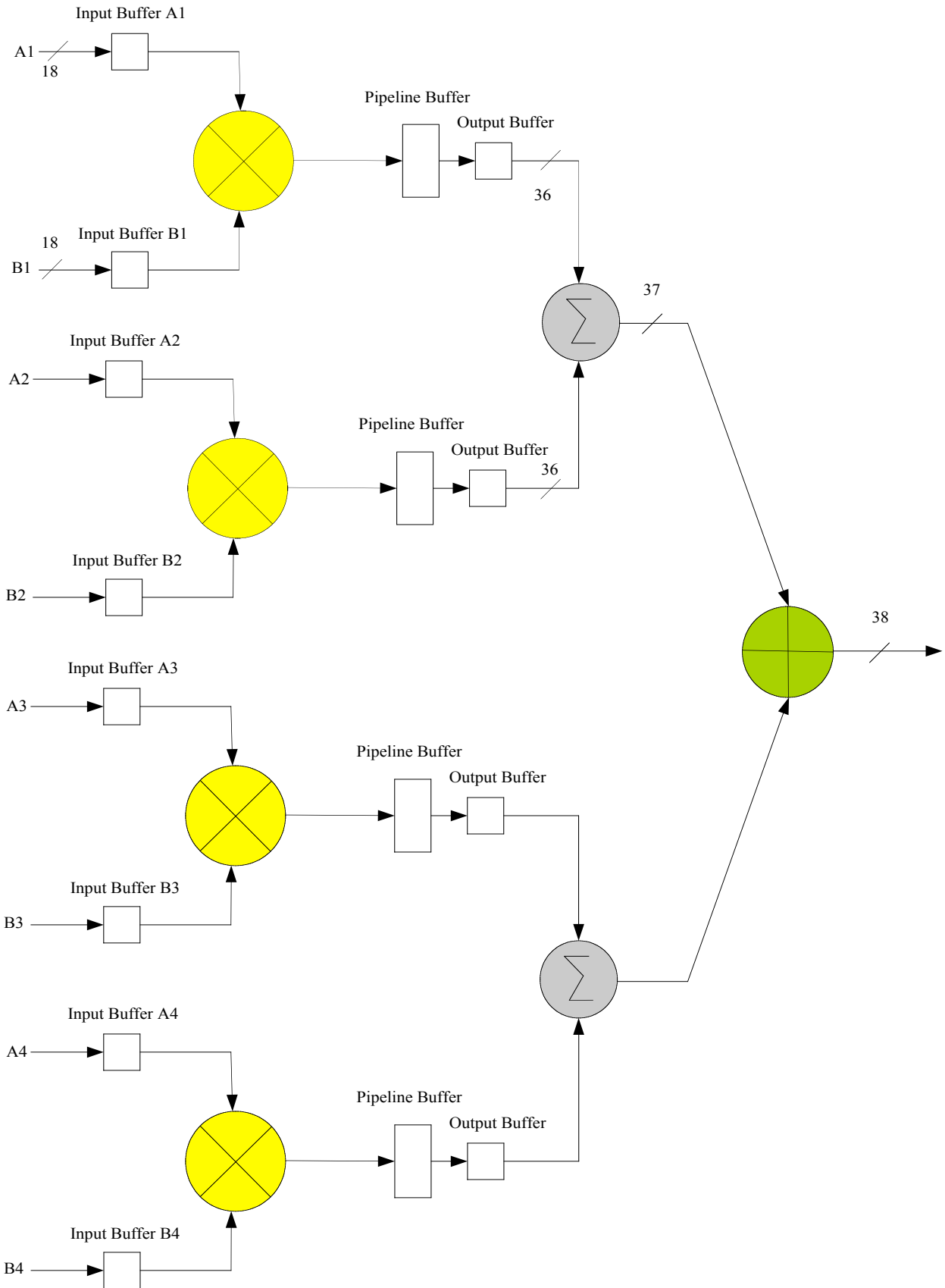


Figure 1 -- DSP Block Mode 4 – 18x18 MULT (One sum of four products)

As a basic example of incorporating DSP blocks into the FPGA fabric, the conventional general purpose DSP chips typically contain one to four multiply and accumulate (MAC) units with fixed data-width multipliers; this leads to limited parallelism and limited throughput. Their throughput is increased by higher clock speeds. The LatticeECP-DSP has many DSP blocks that support different data-widths. DSP blocks give an approximate increase in speed of 3x compared to LUT-based multipliers of comparable FPGA devices. For the ECP20 there are 28 dedicated 18-bit x 18-bit multipliers, greatly enhancing such basic functions as FIR, IIR FFT, correlators, and turbo convolutional encoders. If complex multiply is required, Mode 3 of the 18 X 18 will give seven and Mode 3 of the 9 X 9 will give fourteen. The parallelism capability allows the designer to use highly parallel implementations of DSP functions. The designer can optimize the DSP performance vs. area by choosing appropriate levels of parallelism.

Software Defined Radio

Software defined radio may be on track to deliver on its promise as the future of wireless communications. The hardware section is still the hard entity. SDR has three common barriers: a low-cost, low-power, frequency-agile front end; portable software that can be implemented on any platform; and a low-power processor on which to perform that processing. The last two requirements are a continuing focus of FPGAs. From the RFI of the SDR Forum, the following is quoted from Berkeley Wireless Research Center, Dept. of Electrical Engineering, Berkeley University:

“Direct-mapping processing elements on FPGAs has been shown to be more energy efficient than implementing algorithms on sequential processor architectures, due to the exploitation of spatial resources and parallel processing. Instruction-based processors are also notoriously difficult to program under real-time constraints, with caches and interrupts preventing deterministic execution time in most cases. The primary challenge in using FPGAs as the processing fabric for higher layer processing is in the design environment and programming model.”

For years, military SDR thinking has emphasized digital down conversion (DDC) and baseband demodulation. The goal was, and remains, to accommodate baseband multi-modulation schemes, bandwidths, data rates and multiple protocols for surveillance, and

linking of communication networks ^[1]. This goal can now be accomplished with key FPGAs providing lots of adaptive parallel computation. A good example that we will discuss here is that of adaptive beamforming. Beamforming is not only a military entity found in Naval sonar and airborne surveillance, but also is found in 3G cellular, medical ultrasound and phased-array radar.

Systolic Array Processor for Beamforming

Array signal processing, or multi-channel data processing, has a growing number of important applications ranging from radar, sonar, geo-/astrophysical exploration, and biomedical signal processing to wireless communications. In all these cases, the systems are required to operate reliably in the presence of strong interferers. Many technologies have similar processing architectures that may be implemented via a systolic array. Some examples are Multi-User Detection, Channel Estimation and Adaptive Beamforming.

The spatial filtering problem is of interest. A number of independent sensors (antenna elements) are placed at different points in space (linearly, cylindrically, spherically and almost randomly). In effect, the sensors provide a means of sampling the received signal in space. Spatial filtering, known as beamforming, is used to distinguish between the signal and interference/noise.

In many cases the spatial filtering is required in a time-varying environment (e.g., mobile sources); i.e., the beamformer must be able to adapt its steering characteristics in real time. In situations in which the data are processed on a sample-by-sample basis, the method of choice for fast and numerically robust adaptive beamforming is the QR Decomposition Recursive Least Squares (QRD-RLS) algorithm, based on a sequence of rotations called Givens rotation. The resulting QRD-RLS algorithm requires $O(M^2)$ operations per sample, where M is the number of sensors. Unfortunately, this time complexity does not comply with the capabilities of many current systems based on general-purpose, programmable DSPs. With the tremendous advances in microelectronics, the idea of parallel processing using systolic/wavefront arrays has become practically feasible. Gentleman & Kung ^[2] first explored a parallel and pipelined triangular processor array realization of the QRD-RLS algorithm based on Givens rotations, and later refined by various researchers. It will be

discussed later how the CORDIC algorithm can replace the complex Givens rotation and can be implemented easily with FPGAs when there is a lack of multipliers or when a higher speed FPGA without multipliers, such as the LatticeEC, is used. Figure 2 illustrates the relationship of SDR, systolic array processor, Givens rotations and CORDIC (if used).

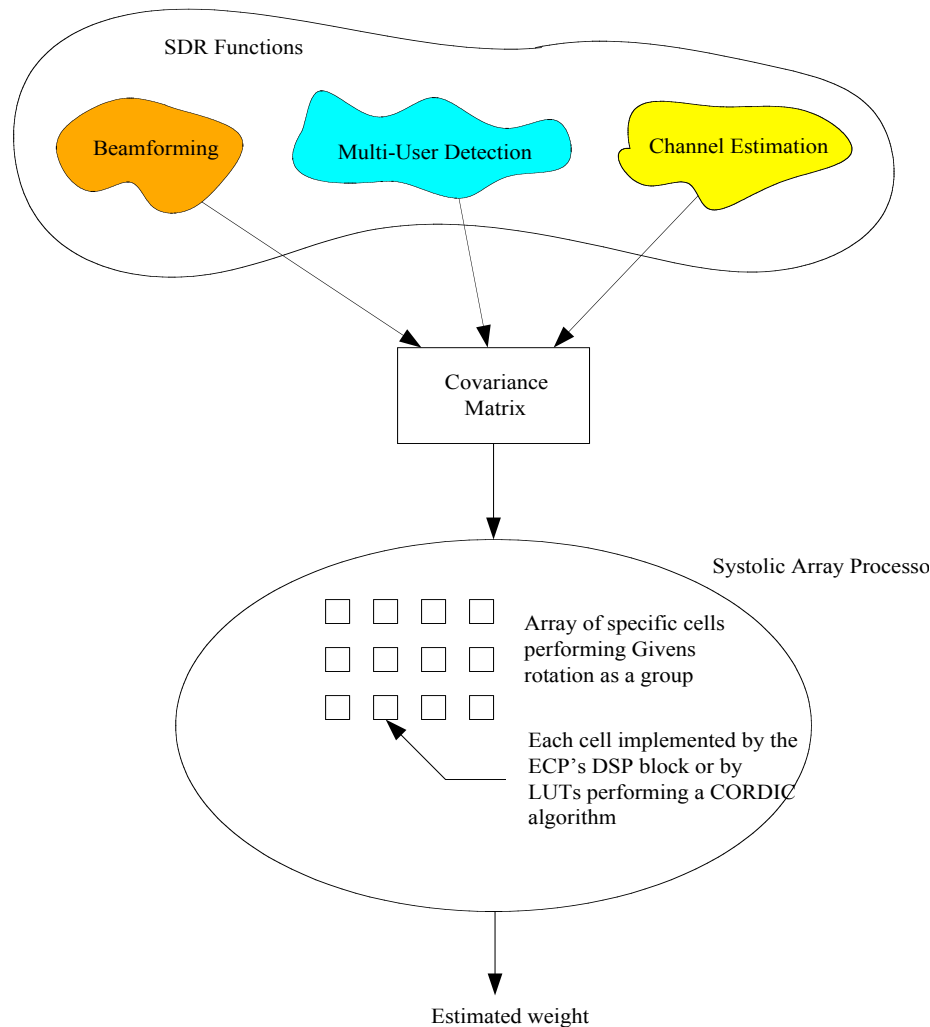


Figure 2 -- SDR-to-Systolic Array-to-Processing Link

With the advance of configurable computing devices (i.e., FPGAs), it becomes possible to design a configurable beamformer which is not only adaptive to time-varying input data but is also structurally adaptive to changes of the parameters. Consider the wireless communications environment. Although it is generally accepted that beamforming concepts will be used in future mobile communications systems, it is not at all clear today how, precisely, the beamforming will be integrated. Different standards and different estimation/detection algorithms within these standards make it impossible to define the exact structural parameters of the beamformer.

Adaptive beamforming when applied to large arrays has several hurdles to overcome. First is the obvious computational load associated with beamforming arrays with many elements, in which the beams are formed to cover all space. A second hurdle is that if the beamformer is to respond quickly to a changing environment, then only a limited time is available to estimate the correlation (or data) matrix. This results in a poor estimate of the correlation matrix that degrades the beamformer white noise suppression capability and therefore array gain.

Suitability of FPGAs to Systolic Arrays

Next generation beamforming systems must deliver ever-increasing performance. This means higher data rates, better spectrum efficiency and, inevitably, more algorithmic complexity. And, of course, these systems must all be delivered at lower cost, reduced time-to-market and be compatible with the latest evolving standards.

These conflicting requirements can be satisfied by exploiting the benefits of the latest FPGA devices incorporating DSP. These devices provide a highly parallel DSP capability without the expense and design time of an ASIC. Further, FPGAs can be reprogrammed to change their functionality as easily as a conventional programmable DSP processor. The following are reasons why an FPGA is suited for systolic array applications:

- FPGA fabric is generic; therefore the architecture is not being defined for the user.
- FPGA has many LUTs and multipliers; therefore it is better suited for parallel operations.
- FPGA is high performance. It easily allows time division multiplexing for slower data rates, and so can accommodate multiple channels running simultaneously.
- FPGA is lower power than a DSP solution because parallel operations allow for lower operating frequency and lower EMI.
- FPGA footprint is compatible over several densities.
- FPGA cost is coming down fast - \$1.50 per 1K LUTs in high volume (greater than 250,000 pieces), rivaling the cost of ASIC or DSP processor solutions.

The MVDR Problem Summarized

The Minimum Variance Distortionless Response (MVDR) beamforming problem amounts to minimizing, in the least squares sense, the combined output from an antenna array subject

to K independent linear equality constraints each of which corresponds to a chosen “look direction.” It was shown ^[3] how a (p+1) x (p+1) triangular systolic array could be applied to the problem of RLS minimization, subject to linear constraints. In effect, the top rows (one for each simultaneous constraint) of the triangular array are used to implement constraint-preprocessing operations. The remainder of the triangular array is used to perform the QR decomposition on the transformed data matrix produced by the “constraint processor.”

The MVDR is a constrained least squares minimization problem. Its solution is given by the formula,

$$\mathbf{w}^{(k)}(n) = \frac{\mu^{(k)} \mathbf{M}^{-1}(n) \mathbf{c}^{(k)*}}{\mathbf{c}^{(k)T} \mathbf{M}^{-1}(n) \mathbf{c}^{(k)*}} \quad (6)$$

where $\mathbf{M}(n)$ is the (weighted) covariance matrix defined by

$$\mathbf{M}(n) = \mathbf{X}^H(n) \mathbf{X}(n).$$

$\mathbf{X}(n) = \mathbf{B}(n)[\mathbf{x}(t_1), \dots, \mathbf{x}(t_n)]$ where $\mathbf{x}(t_n)$ is the p-element vector of (complex) signal samples received by the array at time t_n and $\mathbf{B}(n)$ is a diagonal matrix of ‘forgetting’ factors. (7)

$\mathbf{c}^{(k)}$ is a constraint vector for the k -th look direction imposed upon the weight vectors and $\mathbf{X}(n)$ is an $n \times p$ sample data matrix.

Using QR decomposition on the data matrix $\mathbf{X}(n)$ we have

$$\mathbf{Q}(n) \mathbf{X}(n) = \begin{pmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{pmatrix}, \quad (8)$$

where $\mathbf{R}(n)$ is a $p \times p$ upper triangular matrix, and $\mathbf{Q}(n)$ is an orthogonal transformation.

Then,

$$\mathbf{M}(n) = \mathbf{R}^H(n) \mathbf{R}(n)$$

And so $\mathbf{R}(n)$ is the Cholesky square root factor of the covariance matrix

$\mathbf{M}(n)$ implementation of the systolic array for MVDR beamforming illustrated in Figure 3 ^[4].

Four types of cells are implemented in the systolic array whose functions are detailed in Figure 4. The functions are those of the square-root-free Givens rotation algorithm.

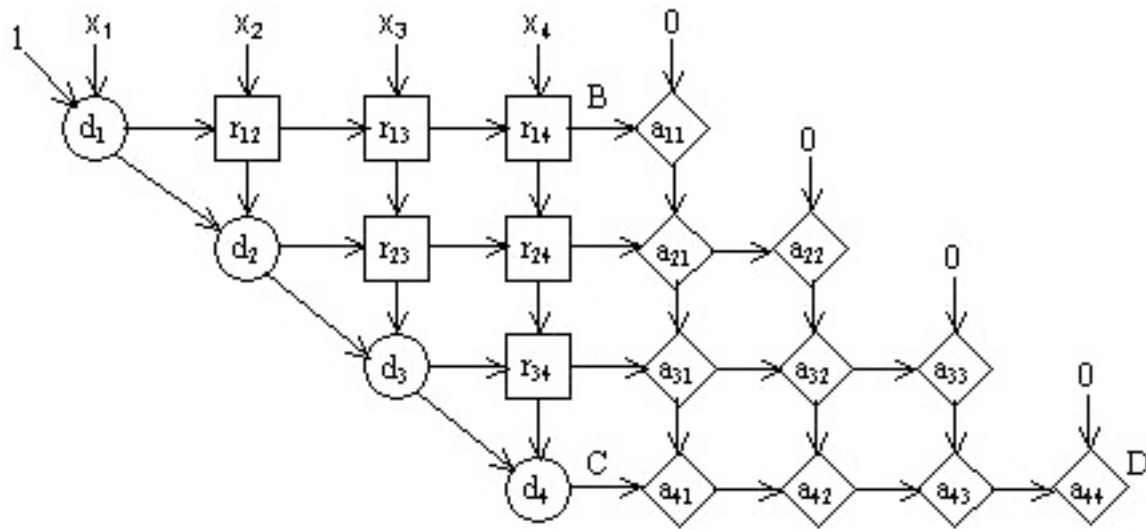


Figure 3 -- Systolic array for MVDR beamforming

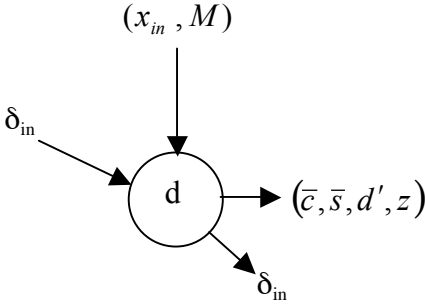
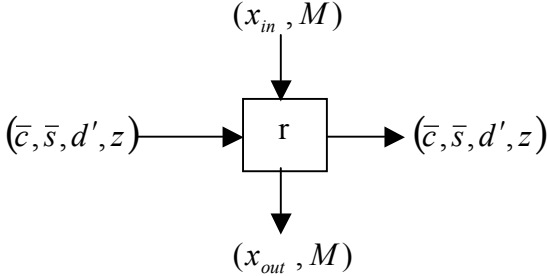
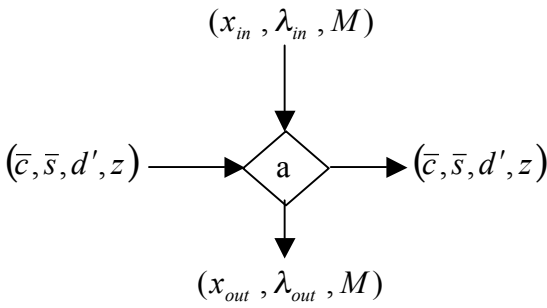
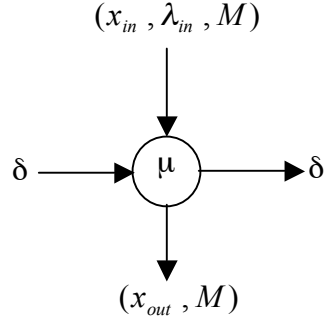
Cell	Mode 1 (if $M = 1$)	Mode 2 (if $M = 0$)
a Boundary 	$d' \leftarrow \beta^2 d + \delta_{in} x_{in} ^2$ If $x_{in} = 0$ or $\delta_{in} = 0$, then $(\bar{c} \leftarrow 1; \bar{s} \leftarrow 0)$ otherwise $(\bar{c} \leftarrow \beta^2 d / d'; \bar{s} \leftarrow \delta_{in} x_{in} / d')$ $d \leftarrow d'; \delta_{out} \leftarrow \bar{c} \delta_{in}; z \leftarrow x_{in}$ $\bar{c} = \cos \theta, \bar{s} = \sin \theta$	$\bar{c} \leftarrow 1; \bar{s} \leftarrow 0$ $d' \leftarrow d; \delta_{out} \leftarrow \delta_{in}$ $z \leftarrow x_{in}$
b Internal 	$x_{out} \leftarrow x_{in} - zr$ $r \leftarrow \bar{s} * x_{in} + \bar{c} r$	$x_{out} \leftarrow x_{in} - zr$
c Constraint 	$x_{out} \leftarrow x_{in} - za$ $a \leftarrow (\bar{s} * x_{in} + \bar{c} a) / \beta^2$ $\lambda_{out} \leftarrow \lambda_{in} + d' a ^2$	$x_{out} \leftarrow x_{in}; \lambda_{out} \leftarrow \lambda_{in}$ If $x_{in} = 1$, then $(a \leftarrow z * d')$
d Final 	If $\lambda_{in} = 0$, then $(x_{out} \leftarrow 0)$ otherwise $(x_{out} \leftarrow -\delta x_{in} \mu / \beta^2 \lambda_{in})$	$x_{out} \leftarrow x_{in}$

Figure 4 -- Processing cells required for MVDR systolic array (square-root-free Givens rotation algorithm)

The systolic approach offers a fast rate of convergence that is inherent in least-squares estimation. The QR decomposition can be implemented with either reflection or rotational operators for obtaining sparse covariance invariant data. Householder transforms is a reflection method and a rotational operator called Givens (or, sometimes, Jacobi) transformations are two important procedures. The basis underlying Householder transforms is that they map any given data set into another sparse data set that consists mostly of zeros, but which is “equivalent” to the original data set. The sparse data has the singular advantage that, when compared to the original data matrix, the sparse data matrix is computationally more efficient ^[5]. Similarly, Givens rotations can be used to zero any given component of a vector while retaining covariance invariance. A succession of Givens operators can be used to zero multiple elements, acting much like a (single) Householder transform. Reflections lead to fewer computations than rotations. However, rotations allow a higher degree of parallelism, since they are employed on smaller data sets ^[6]. So, the Givens rotations technique is the tool of choice in the systolic array processor for MVDR beamforming. A wordlength bound ^[7] has been given as: $\text{wordlength}\{\text{invert } \mathbf{M}\} \geq \log_2 \lambda_{\max} / \lambda_{\min}$ where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues in M respectively. Figure 5 ^[7] illustrates finite word lengths effects (loss) of various least-squares algorithms averaged over 10 runs. Direct inversion of $\mathbf{M}(n)$ by means of the Cholesky method exhibits a defined threshold of around 25 bits. Householder has a much lower threshold of 13 bits and Givens near 16 bits. The Givens is advocated in most beamforming processing.

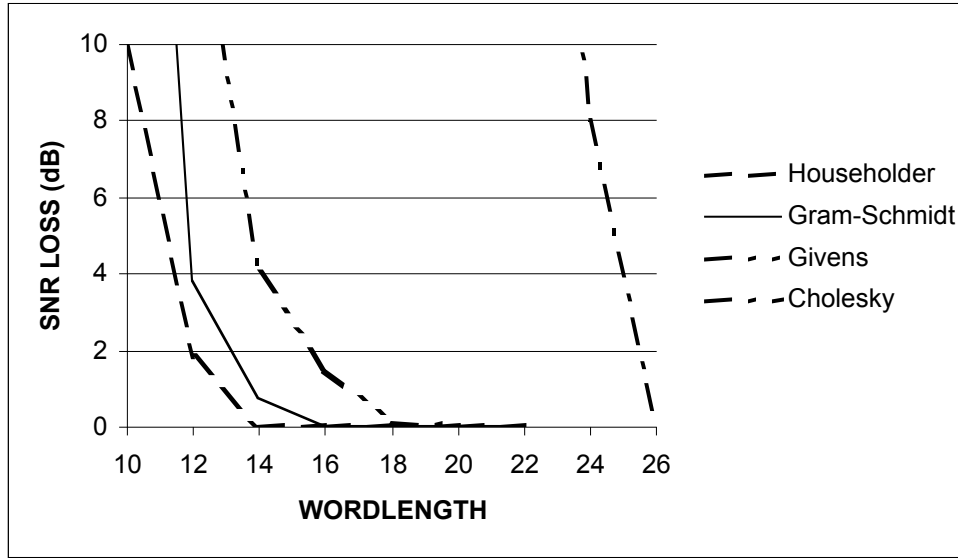


Figure 5 -- Finite wordlength effects of various least-squares algorithms.

The QR decomposition of $\mathbf{X}(n)$ may be completed using a recurring update of the form

$$\hat{\mathbf{Q}}(n) \begin{bmatrix} \beta R(n-1) \\ \bar{x}^T(n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \end{bmatrix} \quad (9)$$

it can be shown that an inverse update formula is obtained

$$\hat{\mathbf{Q}}(n) \begin{bmatrix} \beta^{-1} R^{-H}(n-1) \\ 0 \end{bmatrix} = \begin{bmatrix} R^{-H}(n) \\ u^T(n) \end{bmatrix}. \quad (10)$$

The signal flow graph in Figure 3 illustrates this combined processor for performing the recursive QR decomposition^[8] and inverse updating. The upper triangular array on the left side is a conventional Gentleman-Kung QR decomposition. The lower triangular array on the right stores and updates the matrix $\mathbf{R}^{-H}(n-1)$ according to the equation (10). The Givens rotation parameters, computed by the left hand boundary cells, pass to the right across the entire array. They are used within the Gentleman-Kung processor to perform the update in equation (9) and also within the right hand triangular array to perform the inverse update equation (10).

In the case of the MVDR, the inverse update technique is not applied directly to the inverse conjugate matrix, but to a constant projection^[5] whereby an additional column of internal cells are appended to the Gentleman-Kung processor input.

Figure 6 illustrates a block diagram implementing Lattice building blocks of the Internal Cell of Figure 4.

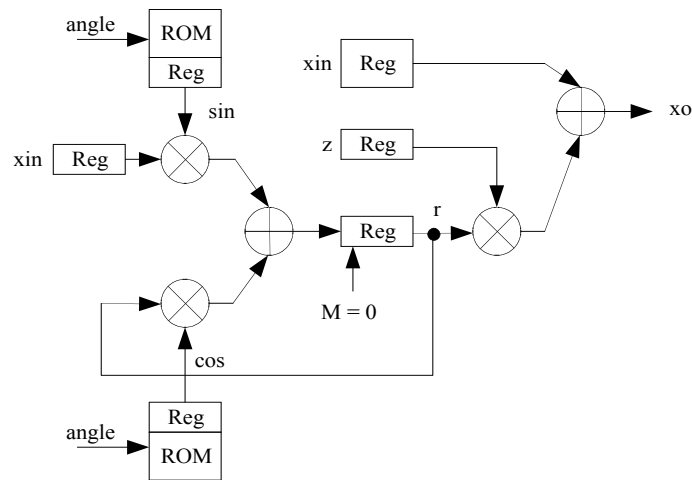


Figure 6 -- DSP Implementation of Internal Cell for Givens Rotation

When attempting to implement the Givens rotations in an FPGA, one problem has to be tackled: the computation uses floating-point operations, which may be costly to implement in a gate array fabric. One possible solution is to transform the input and internal data in such a way that fixed-point computations can be used. Then, the well-known CORDIC algorithm can be used. FPGAs that do not have DSP embedded have in the past gained airplay to use the CORDIC, and many users are drawing an unqualified conclusion that they must use it for square roots, divide, etc. There are also some misconceptions about square root and divides. The simple reckoning is that if you want to do Givens rotation type operations on devices, then implement square roots and divides directly when you have DSP capability. A serial square rooter is simpler, faster, and more regular than a CORDIC. Same for divide. Another problem is the scaling factor introduced for $\sqrt{x^2 + y^2}$ type operations by CORDIC. The bottom line is that unless it comes free, why perform QR or other type operations using the CORDIC? It's more expensive than direct arithmetic, and not guaranteed to converge in a given number of iterations. Therefore, timing/latency calculations are always worst case.

CORDIC Algorithm and Implementation

We should still address the CORDIC. When no or too few multipliers are available, as in the case of the EC FPGA, the CORDIC can be implemented with LUTs. The CORDIC algorithm^[9] is used as a processor and is ideally suited as basic arithmetic units^[10] because many algorithms in digital signal processing and numerical linear algebra (vector/matrix operations) may be cast into a framework mainly involving rotations^[11].

The basic idea underlying the CORDIC scheme is to carry out vector (“macro”-) rotations by an arbitrary rotation angle θ via a series of “micro-rotations” using a fixed-set of elementary angles. With proper choice of elementary angles, all computations can be implemented efficiently in FPGAs and VLSI using a sequence of shift and add/subtract operations. Algorithms such as CORDIC are used to replace the complex Givens rotations that contain trigonometric functions, which are not applicable in hardware implementation. Two modes are defined for the CORDIC: the “vectoring” mode and the “rotation” mode. The vectoring mode determines the phase and magnitude of one vector. The rotation mode actually performs the rotation of the input vector. To execute the Givens rotations, the vectoring mode is first applied to determine the angle to be rotated. Rotation mode follows that and rotates the vector through a set of sub-rotations for a predetermined number of loops. Both vectoring mode and rotation mode include an initialization part and recursive part.

Determination of the rotation angle and the rotation itself uses a similar number of clock cycles. This permits a systolic array processor that is driven by a global clock^[12]. The CORDIC has been exploited to realize every small cell in a systolic array as shown in Figures 3 and 4. There are three kinds of cells differing in shape whereby each cell is always defined as the CORDIC Processor Element (CPE). The round cell performs the “vectoring” and the square performs the rotation mode for an overall QR decomposition. Not shown is the cell that performs the constraint post processing. For complex data, three CPEs are used^[12]. Figure 7 illustrates the structure that can be implemented in an FPGA.

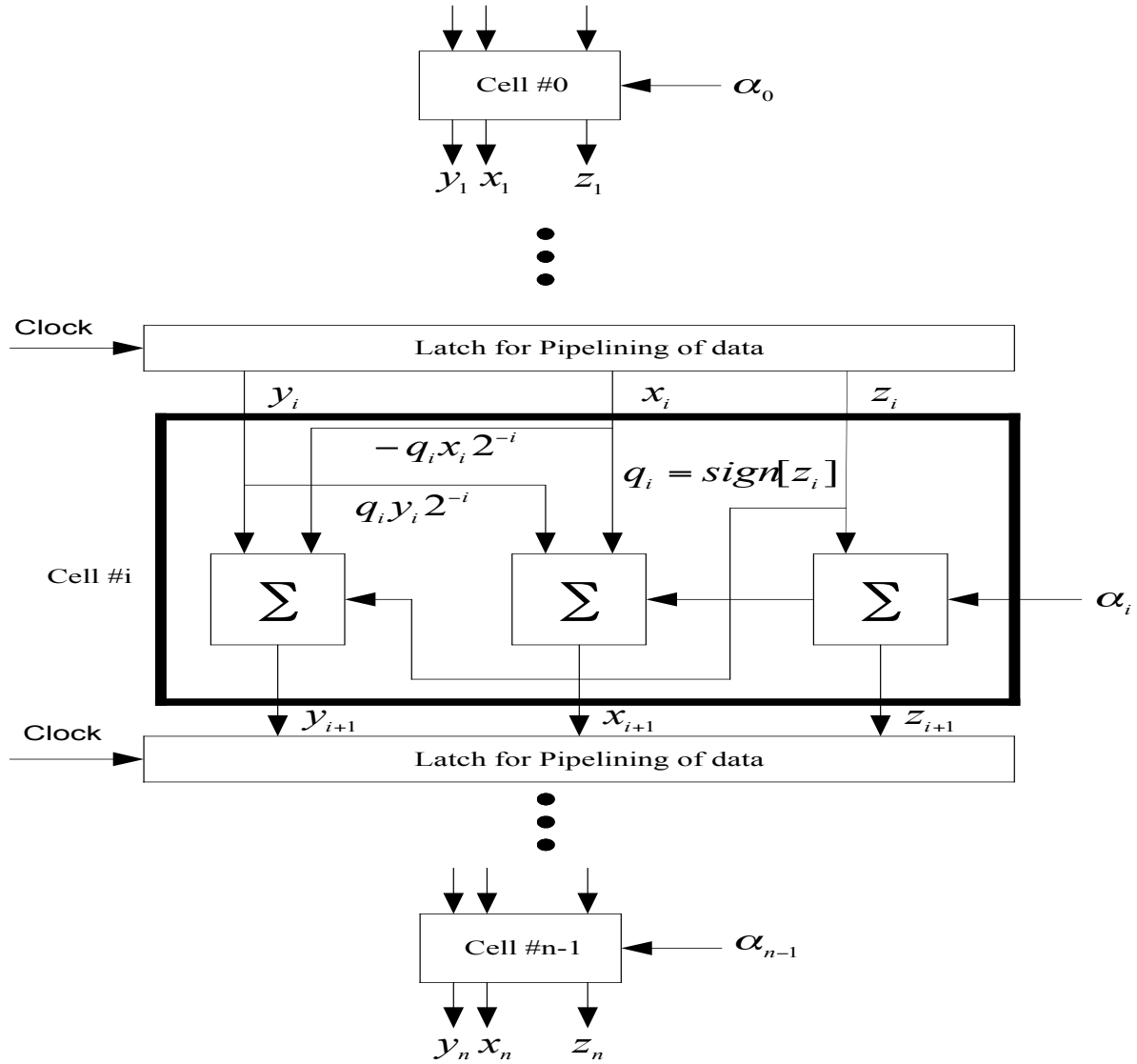


Figure 7 -- CORDIC block diagram for parallel word configuration

Conclusion

Reconfiguration for Software Defined Radios is the goal of JTRS and the SDR Forum. FPGAs are now powerful devices to implement this reconfiguration in real-time. This paper has presented a special technology, i.e., beamforming, to illustrate the intensity of the processing necessary, and so the need for a low-cost, low-power device capable of parallel processing and reconfiguration that delivers a clear set of benefits over DSP processor solutions.

References:

- [1] <http://jtrs.army.mil/>
- [2] Gentleman, W.M. and H.T. Kung, "Matrix Triangularization by Systolic Arrays," Proc. SPIE, vol. 298, Real-Time Signal Processing IV, 1981.
- [3] McWhirter, J.G., "Recursive Least-Squares Minimization Using a Systolic Array," Proc. of SPIE, vol. 431, Aug. 23-25, 1983, San Diego, CA.
- [4] McWhirter, J.G. and T.J. Sheppard, "Systolic Array Processor for MVDR Beamforming," IEEE Proc., Vol.136, Pt. F, No.2, April 1989.
- [5] Steinhardt, A.D., "Householders Transforms in Signal Processing," IEEE ASSP Magazine, July 1988.
- [6] Golub, G. and C. Van Loan, "Matrix Computations", Johns Hopkins Press, 1983.
- [7] Rader, Charles, M., and Allan D. Steinhardt, "Hyperbolic Householder Transformation," IEEE Trans. ASSP, vol.34, No.6, Dec. 1986.
- [8] Shepard, T.J. and T.G. McWhirter, "Modified Givens Rotation for Inverse Updating in QR Decomposition," SPIE vol. 2027.
- [9] Volder, Jock, E., "The CORDIC Trigonometric Computing Technique," IRE Trans. On Elec. Comp., vol.8, no.3, Sept 1959.
- [10] Hu, Yu Hen, "CORDIC-Based VLSI Architectures for Digital Signal Processing," IEEE signal Processing Magazine, July 1992.
- [11] Hall, B., M. Streiff, U. Fleisch, and R. Zimmermann, "Hardware Implementation of a Systolic Antennae Array Signal Processing Based On CORDIC Arithmetic," IEEE ICASSP 1997, vol. 5, pp 4141.
- [12] Rader, C.M., "VLSI Systolic Arrays for Adaptive Nulling," IEEE SP Mag., July 1996.