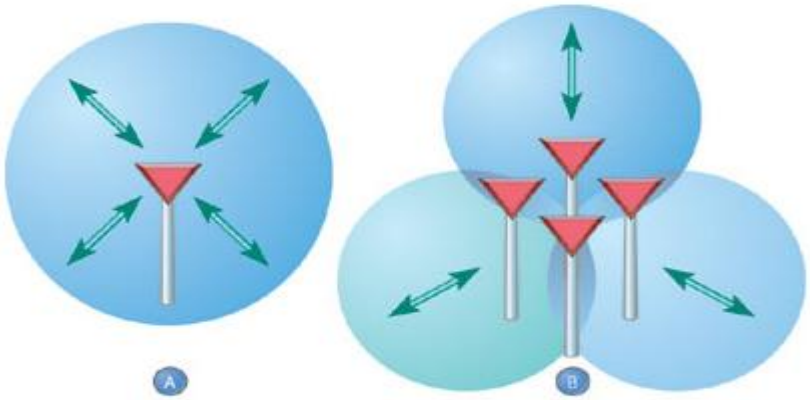
How to create beam-forming smart antennas using FPGAS

- If you could squeeze two or three times more cellular telephone conversations into the same amount of bandwidth, how much would that be worth? To most wireless companies, the answer is, millions of dollars. The art and science of "beam forming" allows normal cellular towers to aim their radio waves at the right user instead of off in all directions. The result is more efficient use of bandwidth and more happy customers. In this article, FPGA design experts explain how beam forming works and how to implement it in standard FPGA chips.
- There are two constants in the cell-phone business: demand for higher data rates and demand for greater user capacity. Both depend on a unique factor known as spectrum efficiency, the ratio of information bits transmitted per amount of spectrum space used (usually expressed in bits/Hertz). Improving that efficiency generally involves tradeoffs between quality of service, power, and coverage.

Non Smart Antennas System



- Traditional omni-directional antennas, as shown in Figure A above act as transducers (that is, they convert electromagnetic energy into electrical energy) and are not an effective way to combat inter-cell and intra-cell interferences.
- One cost-effective solution to this interference challenge is to split up the wireless cell into multiple sectors using sectorized antennas. As Figure B illustrates, sectorized antennas transmit and receive in a limited portion of the cell, typically one-third of the circular area, thereby reducing the overall interference in the system.
- Efficiency can increase still further by using either spatial diversity or by focusing a narrow beam on a single user. The second approach is known as ***beam forming***, and it requires an array of antennas that together perform "smart" transmission and reception of signals, via the implementation of advanced signal processing algorithms.
- Combination of FPGAs, digital signal processing IP, and embedded processors that implement beam-forming applications.
- The methods used to implement such applications and the benefits of improved processing speed, system flexibility, and reduced risk that this approach can deliver.

Smart Antennas

Compared with traditional omni-directional and sectorized antennas, smart-antenna systems can provide:

- Greater coverage area for each cell site
- Better rejection of co-channel interference
- Reduced multipath interference via increased directionality
- Reduced delay spread as fewer scatterers are allowed into the beam
- Increased frequency reuse with fewer base stations
- Higher range in rural areas
- Improved building penetration
- Location information for emergency situations
- Increased data rates and overall system capacity
- Reduction in dropped calls

How Smart antenna is designed?

- A linearly arranged and equally spaced array of antennas forms the basic structure of a beam former.
- In order to form a beam, each user's information signal is multiplied by a set of complex weights (where the number of weights equals the number of antennas) and then transmitted from the array.
- The important point in this transmission is that the signals emitted from different antennas in the array differ in phase (which is determined by the distance between antenna elements) as well as amplitude (determined by the weight associated with that antenna).
- Changing the direction of the beam, therefore, involves changing the weight set as the spacing between the antenna elements is fixed.
- The rest of this Presentation describes two such schemes known as switched and adaptive beam forming.

Switched and adaptive beam

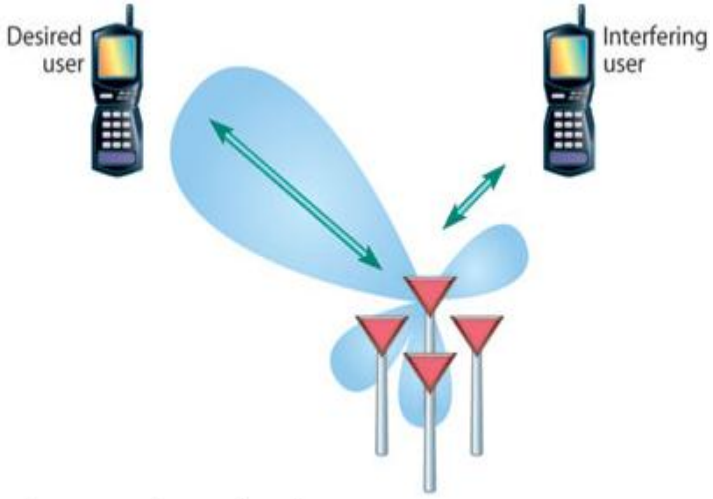


Figure : A beam-forming smart-antennas system

- If the complex weights used are selected from a library of weights that form beams in specific, predetermined directions, the process is called *switched beam forming*.
- In this process, a hand-off between beams is required as users move tangentially to the antenna array.
- If the weights are computed and adaptively updated in real time, the process is known as *adaptive beam forming*.
- The adaptive process permits narrower beams and reduced output in other directions, significantly improving the signal-to-interference-plus-noise ratio (SINR).
- With this technology, each user's signal is transmitted and received by the base station only in the direction of that particular user. This drastically reduces the overall interference in the system.
- A smart-antenna system, as shown in Figure, includes an array of antennas that together direct different transmission/reception beams toward each cellular user in the system

Implementing adaptive beam

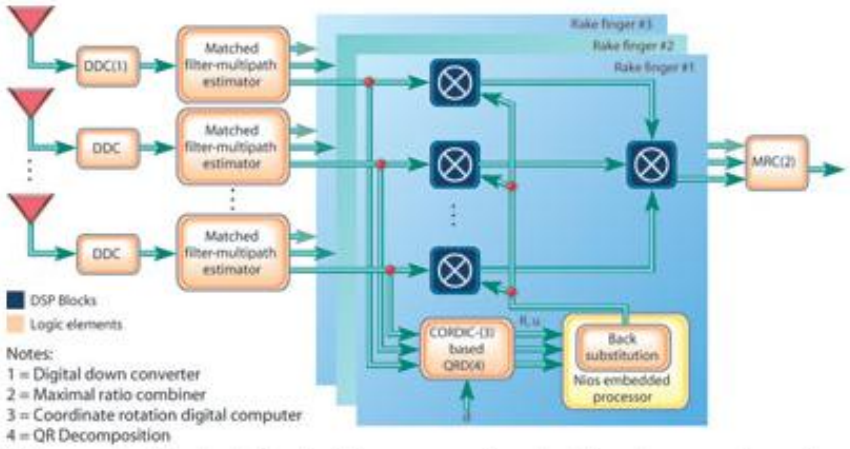


Figure 3: Basic block diagram of adaptive beam forming with FPGA

- Adaptive beam forming can be combined with the well known Rake receiver architectures that are widely used in CDMA-based 3G systems, to provide processing gains in both the temporal and spatial domains.
- This section describes the implementation of a *Rake beam-former structure*, also known as a *two-dimensional Rake*, which performs joint space-time processing. As illustrated in Figure 3, the signal from each receiving antenna is first down-converted to baseband, processed by the matched filter-multipath estimator, and accordingly assigned to different Rake fingers.
- The beam-forming unit on each Rake finger then calculates the corresponding beam-former weights and channel estimate using the pilot symbols that have been transmitted through the dedicated physical control channel (DPCCH).
- The QR-decomposition-(QRD)-based recursive least squares (RLS) algorithm is usually used as the weight-update algorithm for its fast convergence and good numerical properties.

- The updated beam-former weights are then used for multiplication with the data that has been transmitted through the dedicated physical data channel (DPDCH).
- Maximal ratio combining (MRC) of the signals from all fingers is then performed to yield the final soft estimate of the DPDCH data.

Implementing adaptive beam

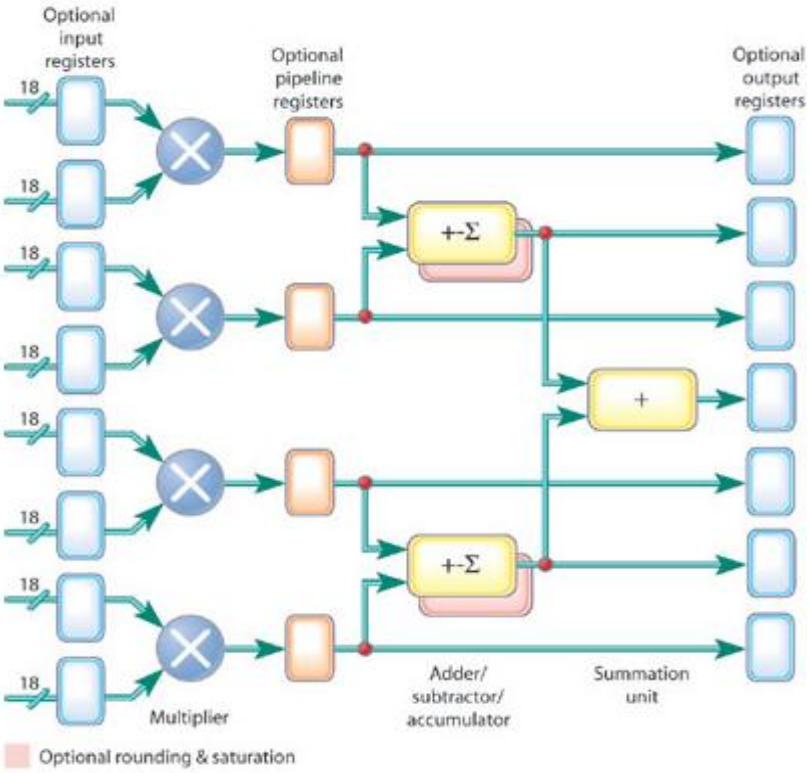


Figure 4: Example DSP block architecture

- Applying complex weights to the signals from different antennas involves complex multiplications that map well onto the embedded DSP blocks available for many FPGAs. The example in Figure 4 shows DSP blocks with a number of multipliers, followed by adder/substractor/accumulators, with registers for pipelining. Such a structure lends itself to complex multiplication and routing required in beam-forming designs.

- Adaptive signal processing algorithms such as least mean squares (LMS), normalized LMS (NLMS), and recursive least squares (RLS) have historically been used in a number of wireless applications such as equalization, beam forming and adaptive filtering. These all involve solving for an over-specified set of equations, as shown below, where $m > N$:

$$
\begin{aligned}
x_1(1)c_0 + x_2(1)c_1 + \ldots\ldots x_N(1)c_N &= y(1) + e(1) \\
x_1(2)c_0 + x_2(2)c_1 + \ldots\ldots x_N(2)c_N &= y(2) + e(2) \\
&\vdots \\
x_1(m)c_0 + x_2(m)c_1 + \ldots\ldots x_N(m)c_N &= y(m) + e(m)
\end{aligned}
$$

- Among the different algorithms, the recursive least squares algorithm is generally preferred for its fast convergence. The least squares approach attempts to find the set of coefficients that minimizes the sum of squares of the errors, in other words:

$$\left\{ \min \sum_m e(m)^2 \right\}$$

- Representing the above set of equations in the matrix form, we have:

$X\mathbf{c} = \mathbf{y} + \mathbf{e}$ ..(1)

- where X is a matrix (mxN, with $m>N$) of noisy observations, **y** is a known training sequence, and **c** is the coefficient vector to be computed such that the error vector **e** is minimized

Adaptive algorithms

$X\mathbf{c} = \mathbf{y} + \mathbf{e}$ …(1)

- Direct computation of the coefficient vector **c** involves matrix inversion, which is generally undesirable for hardware implementation due to numerical instability issues.
- Matrix decomposition based on least squares schemes, such as Cholesky, LU, SVD, and QR-decompositions, avoid explicit matrix inversions and are hence more robust and well suited for hardware implementation.
- Such schemes are being increasingly considered for high-sample-rate applications such as digital predistortion, beam forming, and MIMO signal processing. FPGAs are the preferred hardware for such applications because of their ability to deliver enormous signal-processing bandwidth.
- FPGAs provide the right implementation platform for such computationally demanding applications with their inherent parallel-processing benefits (as opposed to serial processing in DSPs) along with the presence of embedded multipliers that provide throughputs that are an order of magnitude greater than the current generation of DSPs.
- The presence of embedded soft processor cores within FPGAs gives designers the flexibility and portability of high-level software design while maintaining the performance benefits of parallel hardware operations in FPGAs.

QRD-RLS algorithm

$X\mathbf{c} = \mathbf{y} + \mathbf{e}$ …(1)

The least squares algorithm attempts to solve for the coefficient vector **c** from *X* and *y*. To realize this, the QR-decomposition algorithm is first used to transform the matrix *X* into an upper triangular matrix *R* (*N* x *N* matrix) and the vector **y** into another vector **u** such that $R\mathbf{c}=\mathbf{u}$. The coefficients vector **c** is then computed using a procedure called *back substitution*, which involves solving these equations:

$$c_N = \frac{u_N}{R_{NN}} \quad (2)$$

$$c_i = \frac{1}{R_{ii}}\left(u_i - \sum_{j=i+1}^{N} R_{ij} C_j\right) \text{ for } i = N-1, \ldots 1 \quad (3)$$

The QRD-RLS algorithm flow is depicted in Figure 5.

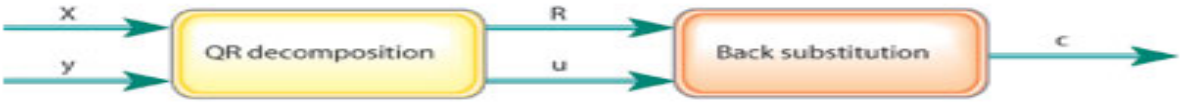


Figure 5: QR-decomposition-based least squares

QRD-RLS algorithm

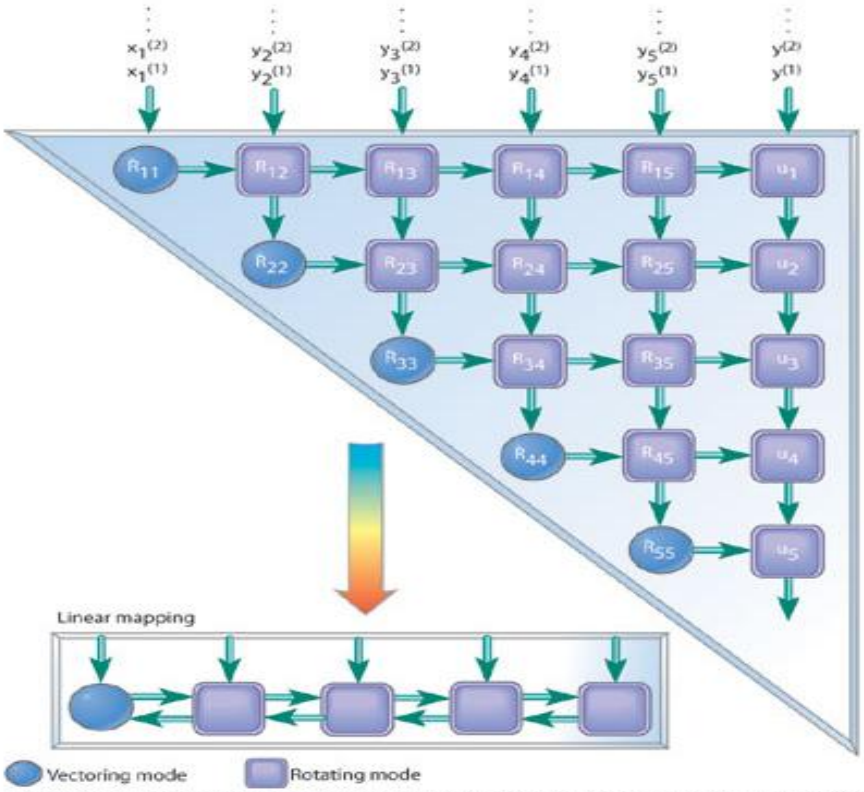


Figure 6: Triangular systolic array example for CORDIC-based QRD-RLS

- The QR-decomposition of the input matrix *X* can be performed, as illustrated in Figure 6, using the well-known systolic array architecture.
- The rows of matrix *X* are fed as inputs to the array from the top along with the corresponding element of the vector **y**. The **R** and **u** values held in each of the cells once all the inputs have been passed through the matrix are the outputs from QR-decomposition. These values are subsequently used to derive the coefficients using back substitution technique.
- Each of the cells in the array can be implemented as a coordinate rotation digital computer (CORDIC) block. CORDIC describes a method of performing a number of functions, including trigonometric, hyperbolic,

and logarithmic functions.2 The algorithm is iterative and uses only add, subtract, and shift operations, making it attractive for hardware implementations.

- The number of iterations depends on the input and output precision, with more iterations being needed for more bits.
- For complex inputs, only one CORDIC block is required per cell. Many applications involve complex inputs and outputs to the algorithm, for which three CORDIC blocks are required per cell. In such cases, a single CORDIC block can be efficiently timeshared to perform the complex operations

Weights and measures

- The beam-former weights vector **c** is related to the **R** and **u** outputs of the triangular array as **Rc=u**. **R** being an upper triangular matrix, **c** can be solved using a procedure called back substitution.
- As outlined in Haykin and Zhong Mingqian et al., the back-substitution procedure operates on the outputs of the QR-decomposition and involves mostly multiply and divide operations that can be efficiently executed in FPGAs with embedded soft processors.
- Some FPGA-resident processors can be configured with a 16x16 -> 32-bit integer hardware multipliers.
- The software can then complete the multiply operation in a single clock cycle. Since hardware dividers generally are not available, the divide operation can be implemented as custom logic block that may or may not become part of the FPGA-resident microprocessor. Between the multiply and divide accelerators, back-substitution becomes easy and efficient.

FPGA advantages

- Smart-antenna technology requires a lot of processing bandwidth, in the neighborhood of several billion multiply-and-accumulate (MAC) operations per second.
- Such computationally demanding applications can quickly exhaust the processing capabilities of many DSPs. Some FPGA chips with embedded DSP blocks, on the other hand, provide throughput in excess of 50 GMAC/sec, offering a high-performance alternative for beam-forming applications.
- There are a number of beam-forming architectures and adaptive algorithms that provide good performance under different scenarios, such as transmit/receive adaptive beam forming and transmit/receive switched beam forming. FPGAs with embedded processors are flexible by nature, providing options for various adaptive signal-processing algorithms.
- The standards for next-generation networks are continually evolving and this creates an element of risk for beam-forming ASIC implementations.
- Transmit beam forming, for example, utilizes the feedback from the mobile terminals. The number of bits provided for feedback in the mobile standards can determine the beam-forming algorithm that is used at the base station. Moreover, future base stations are likely to support transmit diversity, including space/time coding and multiple-input, multiple-output (MIMO) technology. Since FPGAs are remotely upgradeable, they reduce the risk of depending on evolving industry standards while providing an option for gradual deployment of additional transmit diversity schemes.