DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

FPGA BASED SMART ANTENNA IMPLEMENTATION

by
Özgür TAMER

September, 2007
İZMİR

FPGA BASED SMART ANTENNA IMPLEMENTATION

A Thesis Submitted to the
Graduate School of Natural And Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in
Electrical and Electronics Engineering

by
Özgür TAMER

September, 2007
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**FPGA BASED SMART ANTENNA IMPLEMENTATION**" completed by **ÖZGÜR TAMER** under supervision of **ASST. PROF. DR. AHMET ÖZKURT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Asst. Prof. Dr. Ahmet ÖZKURT
Supervisor

Assoc. Prof. Dr. Taner OĞUZER	Assist. Prof. Dr. Mustafa ALTINKAYA
Thesis Committee Member	Thesis Committee Member
Examining Committee Member	Examining Committee Member

Prof. Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

It is a pleasure to thank the many people who made this thesis possible.

I would like to express my deepest gratitude to my supervisor Ahmet ÖZKURT for his guidance and endless patience during the study. He has always encouraged and supported me during the thesis work. It is difficult to overstate my gratitude to my former Ph.D. supervisor Kemal ÖZMEHMET for his invaluable contribution to this thesis. Also I want to thank to my committee members Mustafa ALTINKAYA and Taner OĞUZER who offered guidance and support during the thesis work. I gratefully acknowledge Yeşim YÜKSEL and Gülden KÖKTÜRK as attendants to my thesis progressing meetings and guiding me through the thesis work. I also would like to acknowledge Nalan ÖZKURT for helping me developing the GUI for the testbed of the system.

Thanks to the Dokuz Eylul University and The Scientific and Technological Research Council of Turkey (TÜBİTAK) for providing me with the financial means to complete this project.

I would also acknowledge my colleagues and my workmates, Barış Bozkurt, Adem Çelebi for their advice and their willingness to share their bright thoughts with me and Uğur Torun for his efforts on LaTex writing of this thesis.

My parents who have the highest effort on the satatus I have today, deserve special mention for their inseparable supports. Words fail me to express my appreciation to my wife Sibel whose dedication, love and persistent confidence in me, has taken the load off my shoulder.

Finally I would like to dedicate this thesis to my son Çınar who is 2 months old by I submit this thesis.

Özgür TAMER

FPGA BASED SMART ANTENNA IMPLEMENTATION

ABSTRACT

Adaptive beamformers for sensor arrays, are widely used in RADAR, SONAR and communications applications in order to increase the directivity of the sensor system to the target, while suppressing the interfering signals by adapting the radiation pattern of the antenna array. A signal processing hardware accomplishes the beamforming by adjusting the weights of the sensor array system. Eventhough the hardware for similar applications have been generally preferred as a Digital Signal Processor (DSP), Field Programmable Gate Arrays (FPGA), have become a promising alternative, for various signal processing technics, with increasing logic elements they include. In this manner, logic elements included in the FPGA can be arranged as distinct processors running concurrently, therefore signal processing algorithms running on distinct processors can be implemented by employing FPGA technology.

Systolic arrays are processor arrays, that operate concurrently and pass data between the neighbor processors in order to perform complex functions. Optimal weight extraction based on systolic structures have been the subject of various researches for nearly two decades. In this thesis we propose a rectangular systolic structure for QR decomposition based recursive least squares algorithm for a minimum variance distortionless response beamformer and a folding scheme for this systolic structure. Folding a systolic array reduces the die size of the hardware implementation, accordingly systolic arrays for larger sensor systems will be able to be implemented without employing a larger FPGA chip.

In this thesis, the SystemC library is preferred for developing the necessary software and for the implementation of the algorithm on the hardware. The SystemC is accepted by IEEE as a standard for system development and simulation in 2006. The proposed systolic structure is then implemented on a FPGA based DSP Development board and the results are compared with the conventional systolic array based beamformer hardware. This method can be applied to sensor array beamforming application.

Keywords: Smart Antenna Systems, Systolic Array Architectures, FGPA, Digital Signal Processing

FPGA TABANLI AKILLI ANTEN UYGULAMASI

ÖZ

Uyarlanabilir huzme yönlendiriciler RADAR, SONAR ya da iletişim sistemleri gibi uygulamalarda yaygın olarak kullanılmakta, böylece hedefe doğru olan yönlülük arttırılırken, karışan işaret(ler)in oranı düşürülmektedir. Huzme yönlendirme algılayıcıların işaretlerinin, belli algoritmalar yardımıyla işaret işleme donanımından elde edilen ağırlıklar ile çarpılması ile gerçekleştirilir. Genel olarak algoritmalar sayısal işaret işlemciler (Sİİ - DSP) üzerinde koşan yazılımlar olarak tasarlanmış ve uygulanmıştır, fakat alan programlanabilir kapı dizileri (APKD-FPGA) üzerindeki mantıksal elemanların artması ile sayısal işaret işleme algoritmaları için umut veren donanımlar haline gelmiştir. Her ne kadar APKD sayısal işaret işleme için yeterli elemanı barındırsa da temel olarak algoritmaların bağımsız işlemciler üzerinde eş zamanlı çalışacak şekilde tasarlanması gereklidir.

Sistolik diziler, eşzamanlı çalışan ve komşu işlemciler arasında veri aktarımı yaparak karmaşık işlemleri gerçekleştirmeye yarayan işlemci dizileridir. Sistolik diziler kullanılarak en uygun ağırlıkların elde edilmesi yaklaşık yirmi yıldır üzerinde çalışılan bir konudur. Bu tez çalışması kapsamında en az değişimli bozunumsuz tepki (MVDR) huzme yönlendiricisi için QR ayrıştıması temelli özyineli en küçük kareler (RLS) algoritmasının üzerinde çalışabileceği bir dikdörtgen sistolik dizi yapısı ve bu yapı için bir ayrıştırma ve katlama şeması önerilmektedir. Sistolik dizilerin ayrıştırılıp katlanmasındaki amaç, yapının uygulanabilmesi için gerekli mantık kapısı miktarını azaltarak, APKD'nin daha verimli kullanılmasını sağlamaktır.

Bu çalışmada, sistolik dizinin benzetimi ve uygulaması için C++ programlama dilini temel alan SystemC kütüphanesi tercih edilmiştir. SystemC 2006 yılında IEEE tarafından standart olarak kabul edilmiş ve sistem tasarımları için gelecekte önem kazanması beklenen bir araçtır. Önerilen yapı APKD tabanlı bir işaret işleme kartı üzerinde uygulanmış ve sonuçlar geleneksel sistolik dizi yapısından elde edilen sonuçlarla karşılaştırılarak sunulmaktadır.

Anahtar Sözcükler: Akıllı Anten Sistemleri, Sistolik Dizi Yapıları, APKD, Sayısal İşaret İşleme

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

Mobile communication systems are one of the emerging technologies in recent years. There are now more than 2.2 billion mobile phone subscribers worldwide (ITU 2006), over one over third of the human population. The growth rate of the mobile phone subscribers, which is given in Figure 1.1, is significantly faster than the landline phone service, which now reaches 1.85 billion users after a century after it was invented. This rapid growth of demand on mobile communications, force the service providers to improve their capacity by employing new technologies on their systems.

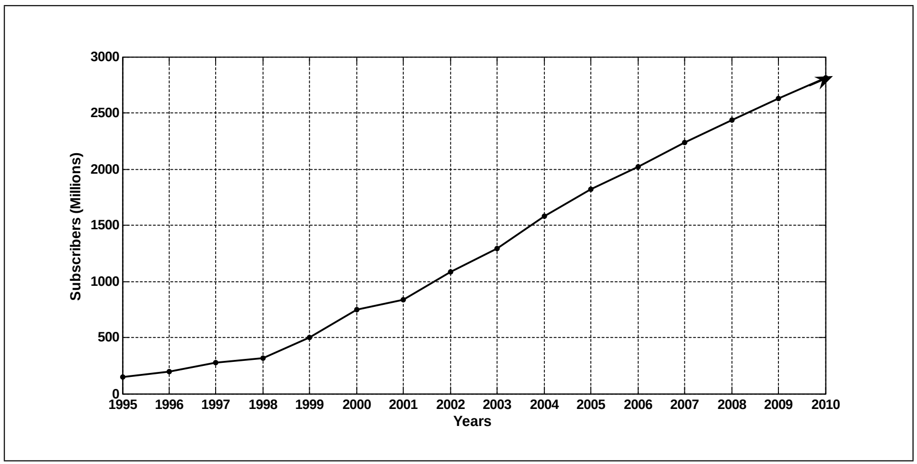


Figure 1.1 Worldwide Cellular Phone Usage Statistics (ITU 2006)

Mobile communication systems are cellular systems that employ base stations to serve its subscribers. Each base station serves a certain region called the cell. The conventional antennas employed in the existing cellular systems are omnidirectional or directional antennas which are presented in Figure 1.2. Omnidirectional antennas are employed in base stations since the early days of the mobile systems. Base stations are located at the center of each cell and the transmitter radiates to every point inside the cell at a specified frequency with an omnidirectional antenna. Directional antennas, also called as sectoral antennas, have been used to overcome the capacity leakage of the cells with omnidirectional antennas. More than one directional antenna can be placed on a base station, each pointing to a different direction so that it is possible to sectorise the cell and use different frequencies in each sector for capacity improvement (Liberti & Rappaport, 1999).

The capacity improvement of the mobile system by using sectoral antennas has become

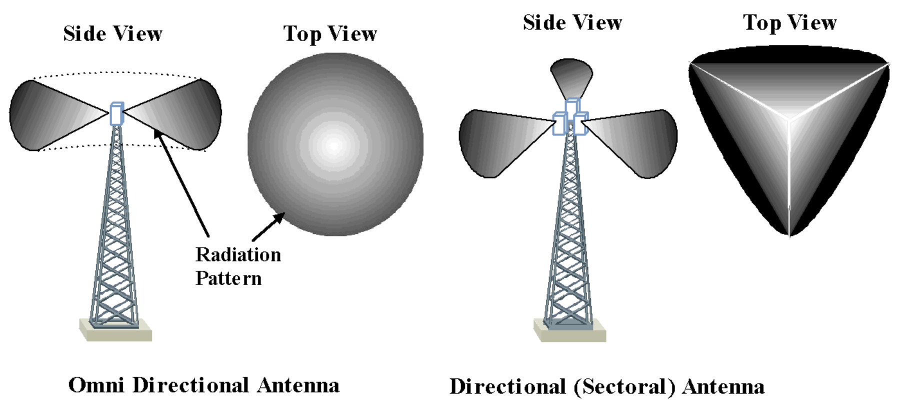


Figure 1.2 Directional and omnidirectional antennas for base stations.

insufficient recently, especially in dense urban areas as the number of subscribers increase, so micro or pico cells are employed inside bigger cells to overcome the problem. Smart antenna systems, however, dynamically sectorize a cell. So that multiple users inside the coverage area of the base station can share the same physical channel without any interference, hence the capacity of the cell can be improved dynamically as can be seen in Figure 1.3 (Zooghby, 2005).

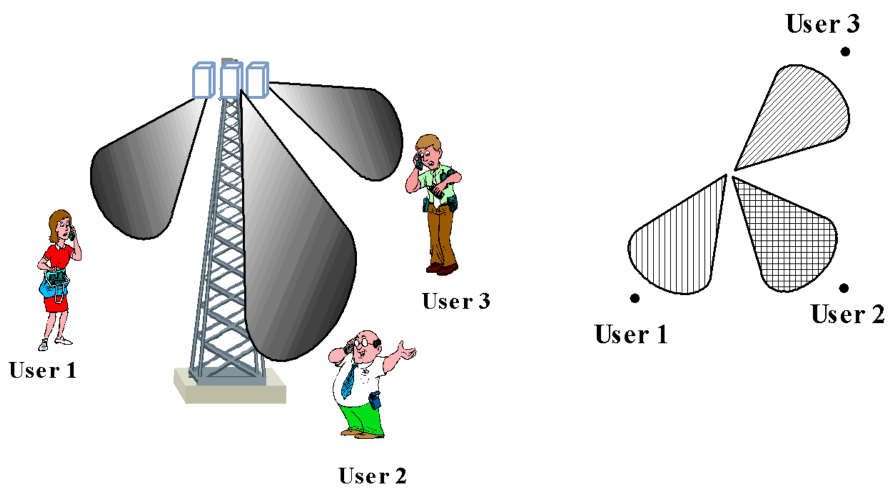


Figure 1.3 Base station with a smart antenna

As it can be observed in Figure 1.4, a smart antenna system is an antenna array equipped with a digital signal processor. The digital signal processor adjusts the complex weights of the antennas so that the radiation pattern of the antenna array is adjusted according to the locations of the users (Liberti & Rappaport, 1999).

Even if the smart antenna technology is a promising one for improving the capacity of the

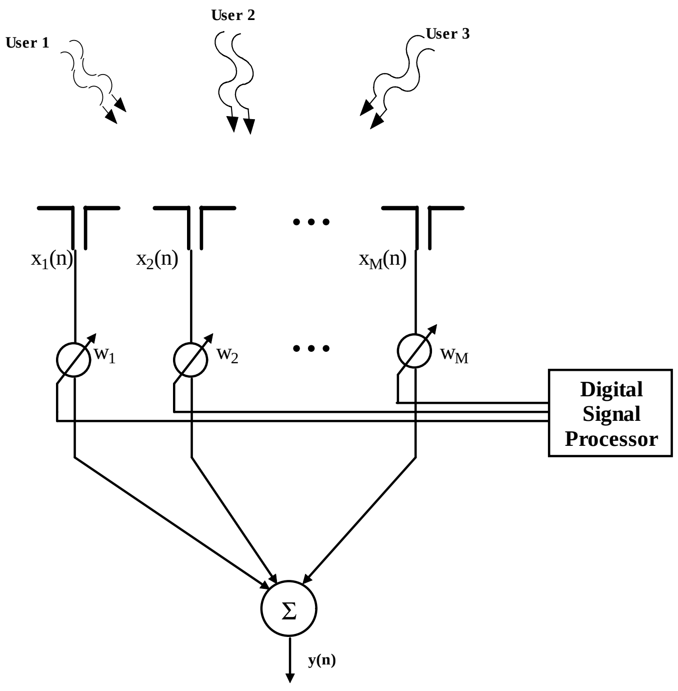


Figure 1.4 Block diagram of a smart antenna

communication system, signal processing algorithms with high operation count may affect the performance of the whole system and cause a degraded performance. Although more powerful signal processing hardware emerge each day, research on improving the performance of the algorithms are also carried out for smart antenna systems (Boukalov & Haggman, 2000).

Parallel algorithms are promising candidates for high performance signal processing applications like smart antenna systems. Systolic array is an architecture on which parallel algorithms can run. A systolic array is an arrangement of processors in an array where data flows synchronously between neighbors across the array, usually with different data flowing in different directions. Processors perform a sequence of operations on data that flows between them and operate concurrently. Each processor at each step takes in data from one or more neighbors, processes it and, in the next step, outputs results in the opposite direction(s) (Quinton & Robert., 1991). Two basic types of systolic arrays; linear and rectangular are demonstrated in Figure 1.5. H. T. Kung and Charles Leiserson were the first to publish a paper on systolic arrays in 1978 (Kung & Leiserson, 1978) , and coined the name. Systolic array based recursive least squares algorithm was first proposed by Gentleman & Kung (1981) and many beamforming algorithms based on systolic arrays have been proposed since then.

A Field Programmable Gate Array (FPGA) is an adequate hardware for implementing a

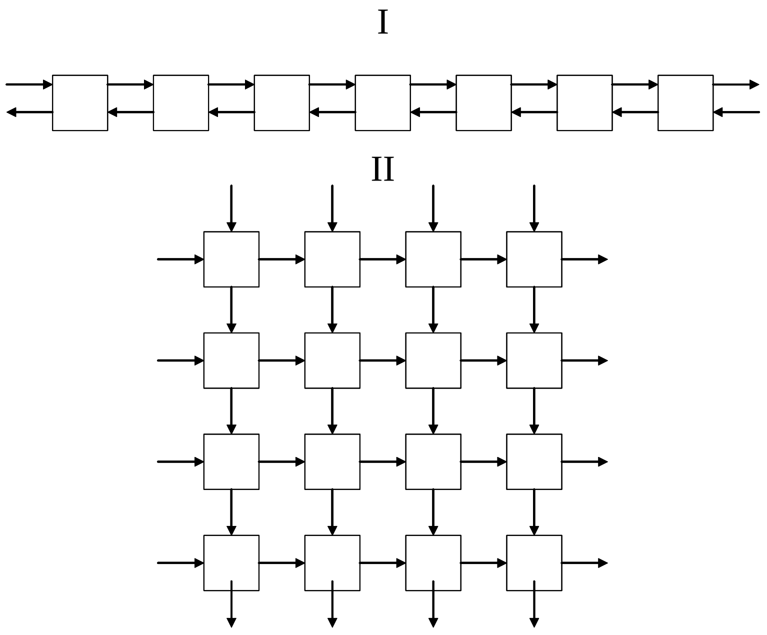


Figure 1.5 Some simple examples of systolic array models.I linear systolic array, II square systolic array

parallel processing structure such as a systolic array. A gate array is an IC chip on which gates are placed in matrix form without connections among the gates. FPGAs are programmable gate arrays, so that, users can easily and inexpensively realize their own logic networks in hardware. FPGAs are composed of repeating units, where units consist of programmable logic devices (PLD), logic gates, random-access memory, and often other types of components (Muroga, 2000). The internal structure of a bus based FPGA is presented in Figure 1.6. The user describes the interconnections and functions of these units by using hardware description languages, to function as distinct processors that will run inside the FPGA concurrently. Thus we can conclude that they are suitable for a systolic array implementation.

1.1 Aim of the Thesis

The aim of this thesis is to develop and implement a smart antenna system based on FPGAs. The FPGAs have the advantage that the logic blocks can be programmed as distinct processors for specialized functions and can run concurrently as denoted in the previous section. The concurrent operation of the processors will improve the performance of the signal processing algorithm and the throughput rate of the weights form the hardware.

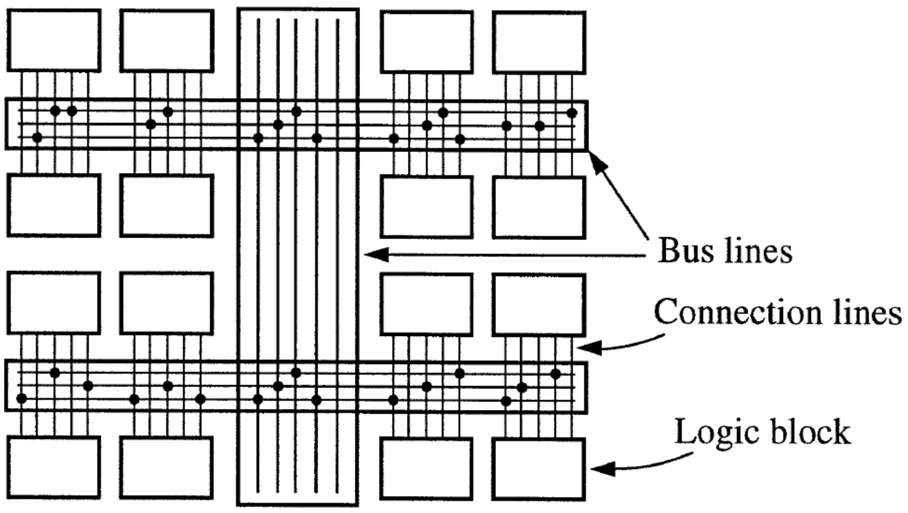


Figure 1.6 Bus-based array type of FPGA

We preferred the QR decomposition based recursive least squares algorithm (QRD RLS) as the beamforming algorithm for the smart antenna which will run on a systolic array. Eventhough parallel operation of the algorithms on systolic arrays improve the performance and reduce the operation count, mathematical operations such as division and square root demand large number of logic elements and the FPGA chip may lack of resources for most of the applications. Our aim is to reduce the number of logic elements for the implementation by partitioning and folding the systolic array structure which is also proposed in this thesis.

For the development and implementation of the systolic array the SystemC library for the C++ programming language is preferred since it reduces the efforts for developing signal processing algorithms with large operation count.

1.2 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter Two presents smart antenna systems and their application to mobile communications in detail. Chapter Three introduces the systolic array based QRD RLS algorithm for beamforming of the smart antenna system while the proposed folded systolic array based MVDR beamformer is described in Chapter Four. Hardware and software implementation of the conventional and proposed beamformer systems is given in detail in Chapter Five whereas results of the implementation are presented in Chapter Six. The results of the thesis are concluded and discussed in Chapter Seven.

CHAPTER TWO
SMART ANTENNAS FOR MOBILE COMMUNICATIONS

A smart antenna extracts a desired signal $d(n)$, out of a linear combination of signals incident on the antenna array, by artificially forming a beam into the direction of the desired user and minimizing the influence of the signals from the interfering users by multiplying the vector of incident signals on M antennas, $x(n)$ by a weight vector w as;

$$d(n) = w^H x(n) \tag{2.0.1}$$

The weights of the array elements are calculated by adaptive algorithms running on a signal processing hardware. Figure 2.1 shows the radiation pattern of a smart antenna, which is directing its main beam to the desired user, and a null of the array factor to the interfering user.

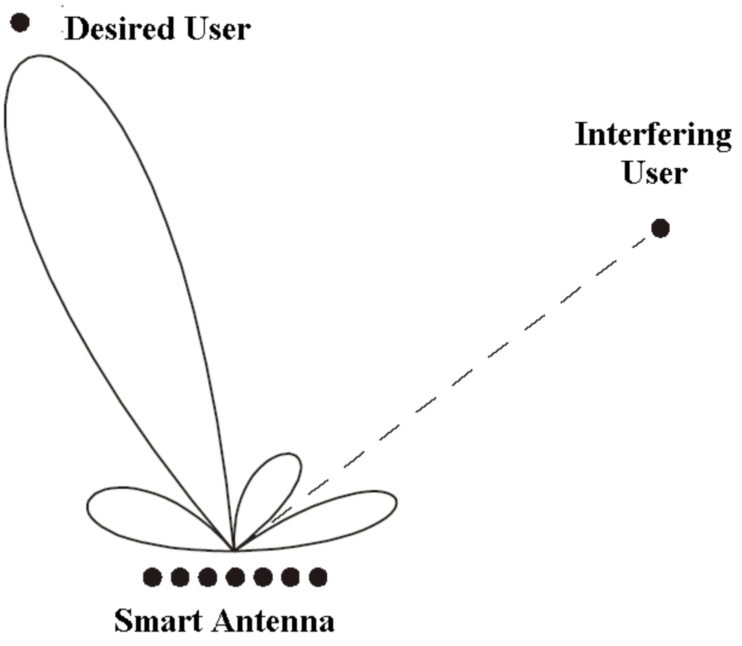


Figure 2.1 Radiation pattern of a smart antenna

2.1 Types of Smart Antennas

Smart antennas can be examined in two basic types; the switched beam antennas and the adaptive beamforming antennas.

2.1.1 Switched Beam Antennas

The simplest implementation of a smart antenna is the switched beam system, in which a single transceiver is connected to the RF beamforming unit. The RF beamforming unit, switches to one of the predefined set of beams, according to the received signal power or minimum bit error ratio. The maxima and nulls of the radiation pattern can not be adjusted for the directions of users, so this technique has limited capabilities as a smart antenna.

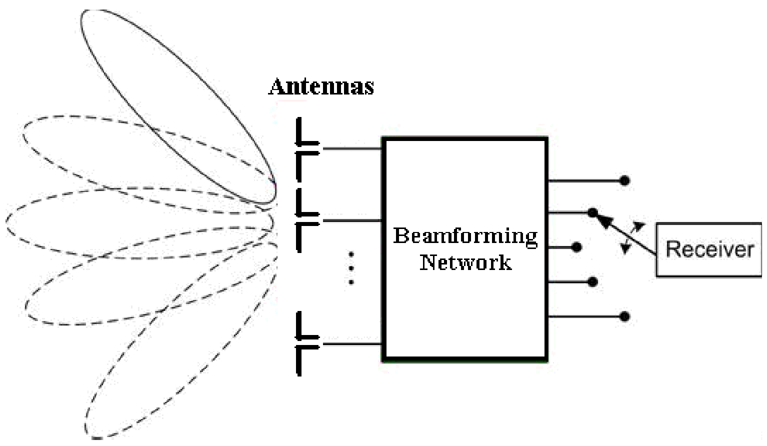


Figure 2.2 Block diagram of a switched beam antenna

2.1.2 Adaptive Beamforming Antennas

The adaptive beamforming antennas use antenna arrays which are equipped with strong signal processing capability hardware to automatically change the beam pattern in accordance with the changing signal environment. It not only directs maximum radiation in the direction of the desired mobile user, but also introduces nulls at the interfering directions, and tracks the desired mobile user at the same time. Figure 2.3 presents the receiver block of an adaptive beamforming antenna. The adaptation is achieved by multiplying the incoming signal with complex weights and then summing them together to obtain the desired radiation pattern. These weights are computed adaptively by the signal processor to adapt the pattern to the changes in the signal environment. The complex weight computation based on different criteria is incorporated in the signal processor in the form of software algorithms.

The transmitter part of a smart antenna has a similar structure with the receiver part. The

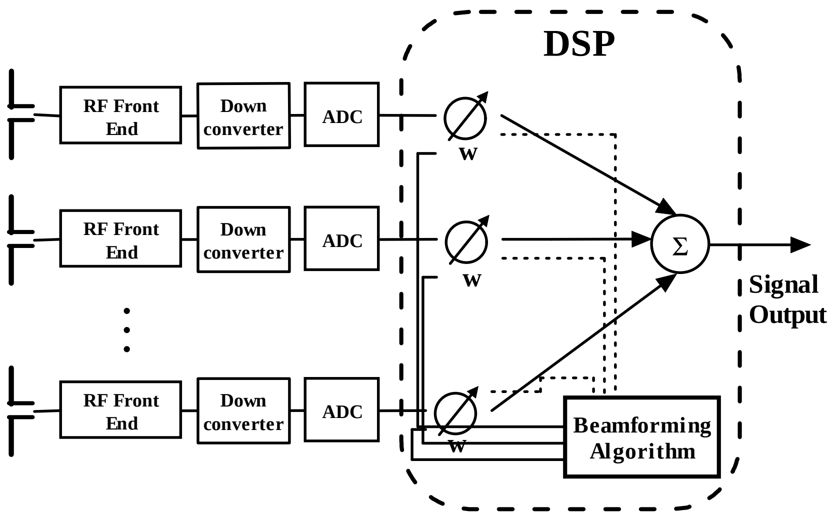


Figure 2.3 Receiver block of an adaptive beamformer

block diagram of the transmitter part is given in Figure 2.4. Here the weights derived from the received signals are also used for the transmitter block. If the uplink and downlink frequencies are close enough, same weights will be adequate for both of them.

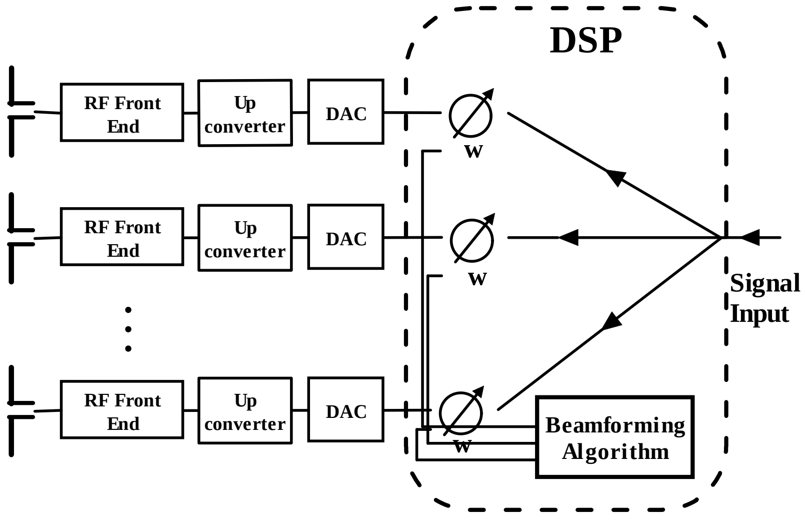


Figure 2.4 Transmitter block of an adaptive beamformer

Forming the beam of an antenna adaptively by assigning weights to the antennas can be realized by employing different techniques which are explained in detail below.

2.1.2.1 Beamforming Techniques

If, L signals with arrival angles $\{\theta_1, \theta_2, ..., \theta_L\}$ impinge on a uniform linear array with M elements, the received signal at the m^{th} element at time instant t can be written as;

$$x_m(t) = \sum_{i=1}^{L} s_i(t) e^{jkd\sin\theta_i} + n(t) \tag{2.1.1}$$

where k denotes the wave number and $n(t)$ denotes the noise at that time instant. Same equation can be represented in vector form as;

$$X(t) = \mathbf{A}\mathbf{S}(t) + \mathbf{N}(t) \tag{2.1.2}$$

for

$$\begin{aligned} \mathbf{X}(t) &= [\ x_1(t) \quad x_2(t) \quad ... \quad x_M(t)\ ]^T \\ \mathbf{S}(t) &= [\ s_1(t) \quad s_2(t) \quad ... \quad s_M(t)\ ]^T \\ \mathbf{N}(t) &= [\ n_1(t) \quad n_2(t) \quad ... \quad n_M(t)\ ]^T \end{aligned}$$

where $S(t)$ denotes the signals impinging on the array elements, $N(t)$ denotes the noise received by the elements and A is the steering matrix, whose columns are the array response vectors for a signal impinging from θ_i direction, given by

$$\mathbf{A} = [\ a(\theta_1) \quad a(\theta_2) \quad ... \quad a(\theta_L)\ ]^T$$

where

$$a(\theta_L) = e^{jkd\sin\theta_L}$$

In adaptive arrays, complex weights are applied to the element outputs given by;

$$\mathbf{W} = [\ w_1 \quad w_2 \quad ... \quad w_M\ ]^T \tag{2.1.3}$$

Then the array output can be written as;

$$y(t) = \sum_{l=1}^{M} x_l(t) w_l^* = W^H X(t) \tag{2.1.4}$$

2.1.2.1.1 Beam Steering In the beam steering approach, phase angles of the weights are selected to steer the main beam of the array in a particular direction, while magnitude of the weights are unchanged. In other words, the array main beam is steered toward the DOA of the desired source. This technique can be used with spatial reference algorithms since it needs the DOA information.

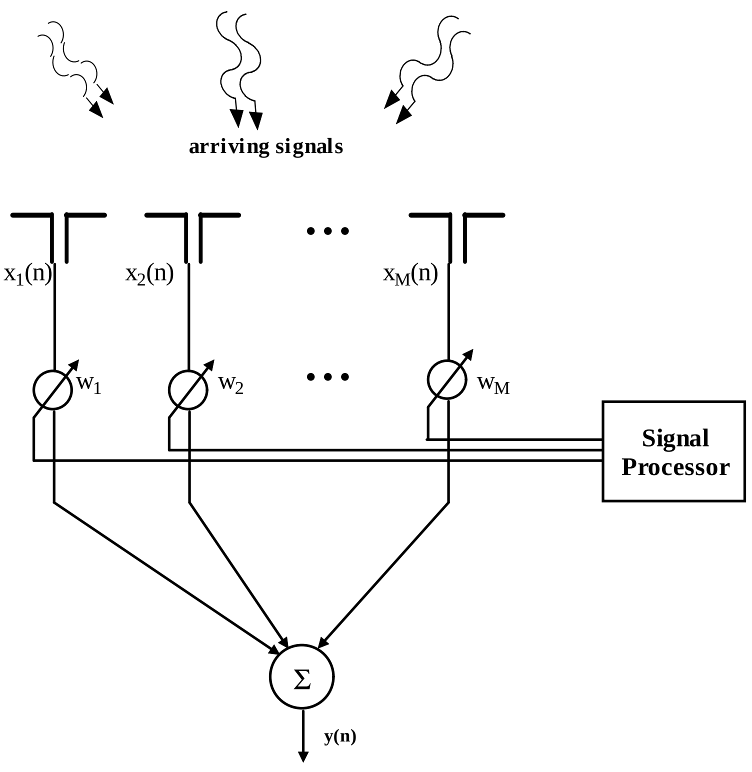


Figure 2.5 Block diagram of a beamformer

2.1.2.1.2 Minimum Mean Square Error (MMSE) In the MMSE criterion, the error between the desired signal $d(t)$ and the output of the beamformer is minimized. The mean square error is given by;

$$E[e(t)] = E\{[d(t) - W^H X(t)][d(t) - W^H X(t)]^H\} \tag{2.1.5}$$

The gradient of the MMSE with respect to W can be written as

$$\nabla E[e(t)] = E\{[-2d(t)X^H(t) + 2X(t)X^H(t)W\} \tag{2.1.6}$$

The optimum weight can be evaluated by setting this gradient to zero;

$$W_{opt} = R^{-1}r \tag{2.1.7}$$

where R is the array correlation matrix and r is given by (Zooghby, 2005);

$$r = E[d^H(t)X(t)] \tag{2.1.8}$$

2.1.2.1.3 Minimum Variance Distortionless Response Beamformer (MVDR) The aim of the MVDR system is to minimize the output residual of the total interference and noise which can be expressed as;

$$\min W^H RW \text{ subject to } W^H A_d = c \tag{2.1.9}$$

where A_d is the steering matrix for the direction of the desired signals and c is the constraint vector. If all elements of c is 1 then the technique is called the MVDR. In order to minimize the variance of the output power, we have to minimize $W^H R_N W$. If we take the gradient of the equation with respect to W and set it to zero we get;

$$\bigtriangledown[W^H R_N W + \lambda(1 - W^H A_d)] = R_N W - \lambda A_d = 0 \tag{2.1.10}$$

thus the optimum weight vector can be evaluated as;

$$W_{opt} = \lambda R_N^{-1} A_d \tag{2.1.11}$$

where λ is given by;

$$\lambda = \frac{1}{A_d^H R_N^{-1} A_d} \tag{2.1.12}$$

For an M-element array with M degrees of freedom, the number of interferers must be less than or equal to M – 2, since one has been used by the constraint vector in the look direction. Some *apriori* knowledge of the desired signal is required by the MVDR beamformer. Since in the MVDR approach the weight vector that minimizes the output power is a function of the spatial correlation matrix, some degree of coherency between the uplink and downlink is needed to provide an estimate of the correlation matrix for transmission. One advantage of the MVDR beamformer is that it does not require any knowledge of the directions of the interference, rather only those of the desired signal(s) (Veen & Buckley, 1999).

2.2 Spatial Channel Model

Conventional propagation models, focus on the power delay profile without taking into account the angular distribution of the arriving signals to the receivers. Channel models that characterize the arrival angles of multipath components are known as the spatial channel models. The directional channel impulse response can be written as;

$$h(t, \tau, \theta, \phi) = \sum_{l=0}^{L} A_l(t) e^{j\varphi l(t)} a(\theta_l(t), \phi_l(t)) \delta(t - \tau_l(t)) \tag{2.2.1}$$

where $A_l(t)$ is the amplitude, $\varphi_l(t)$ is the phase, and $\tau_l(t)$ is the time delay of the signal component and $a(\theta,\phi)$ is the response of the array to a signal arriving from the direction (θ,ϕ) given by (Zooghby, 2005);

$$a(\theta,\phi) = [\ 1 \quad e^{-j\psi} \quad \ldots \quad e^{-j(P-1)\psi}\ ]^T \tag{2.2.2}$$

for a uniform linear array (ULA) with M elements given in Figure 2.6 the array response becomes;

$$a(\theta) = [\ 1 \quad e^{jkd\cos\theta} \quad \ldots \quad e^{j(M-1)kd\cos\theta}\ ]^T \tag{2.2.3}$$

Since the ULA elements are placed on the z axis, the array response does not change with the colatitude angle (ϕ).

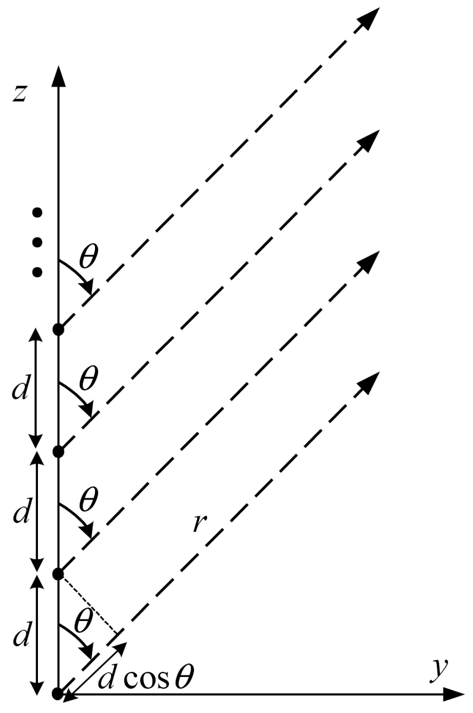


Figure 2.6 Uniform Linear Array (ULA)

2.3 Adaptive Beamforming Algorithms

Beamforming algorithms are used to calculate the complex weights w for the individual antenna elements. They can be investigated in three main types; temporal reference (TR) algorithms, spatial reference (SR) algorithms and blind algorithms. A branching chart of these algorithms are given in Figure 2.7. According to a previous work by Fuhl & Bonek (1998) temporal reference algorithms and spatial reference algorithms perform equally well. However temporal reference algorithms are the most robust against interferers close to the base station with wide angular spread.

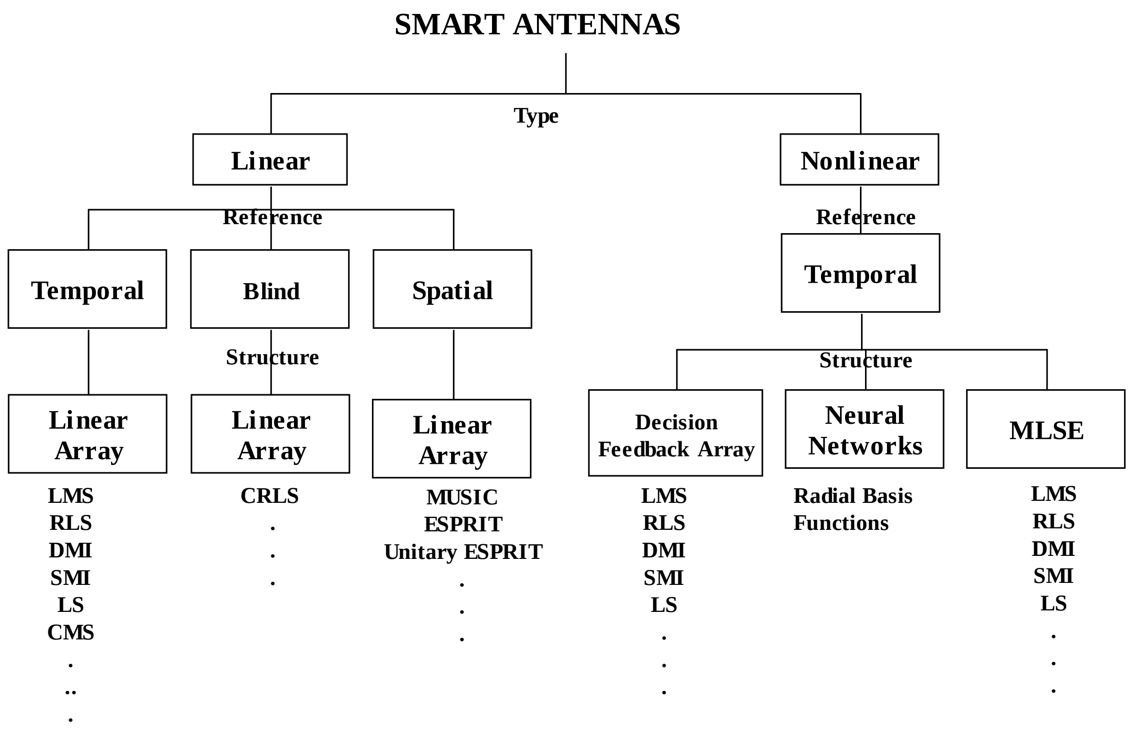


Figure 2.7 Adaptation algorithms for smart antennas (Fuhl & Bonek, 1998)

2.3.1 Temporal Reference Algorithms

TR algorithms are based on the knowledge of some temporal properties of the received signals, such as the known training sequence within the burst, or the constant envelope of the signal. The receiver adjusts the complex weights in such a way that the difference between the combined signal and the known training sequence is minimized.

Most of the downlink beamforming approaches are based on the knowledge of the DOAs. This information is not directly determined by TR algorithms, therefore an additional SR algorithm would have to be included to estimate the DOAs which increases the computational complexity of the signal processing unit.

The problem in a TR algorithms is to minimize the mean square error between the array output and the reference signal d by choosing the right weight vector w. The solution to this problem is given by;

$$w = R^{-1}\varrho \tag{2.3.1}$$

where $\mathbf{R}$ is the $M \times M$ covariance matrix of the antenna output $x(n)$ and it is given by;

$$\mathbf{R} = E[x(n)x^H(n)] \tag{2.3.2}$$

and $\boldsymbol{\varrho}$ is the $M \times 1$ cross-correlation vector between the antenna output $x(n)$ and the desired response d;

$$\boldsymbol{\varrho} = E[x(n)r^*(n)] \tag{2.3.3}$$

where M is the number of antennas.

If we replace the covariance matrix $\mathbf{R}$ and the cross-correlation vector $\boldsymbol{\varrho}$ by their estimates from a finite length sequence $\widehat{R}$ and $\widehat{\varrho}$, respectively, we speak of the Direct Matrix Inversion (DMI) or Sampled Matrix Inversion (SMI) solution.

The direct inversion of the estimated covariance matrix requires large number of operations as stated in Fuhl & Bonek (1998). As a solution to this problem algorithms determining the weight vector w iteratively have been developed. Least Squares (LS), Least Mean Square (LMS) and Recursive Least Squares (RLS) algorithms are examples to the iterative algorithms.

2.3.1.1 The Least Mean Squares Algorithm

As it is the case with all temporal reference algorithms, a reference signal ($d(n)$) is used to update the weights at each iteration as follows;

$$w(n+1) = w(n) - \mu \bigtriangledown_w \text{(MSE)} \tag{2.3.4}$$

where $\bigtriangledown_w$ is the gradient of the mean square error (MSE) between the reference signal $d(n)$ and the array output $x(n)$ given by

$$\begin{aligned} \text{MSE}(w(n)) &= E[\left|d(n+1) - w^H(n)x(n+1)\right|^2] \\ &= E[|d(n+1)|^2] + w^H(n)Rw(n) - 2w^H(n)E[x(n+1)d(n+1)] \end{aligned} \tag{2.3.5}$$

where R is the array correlation matrix.

In the LMS algorithm main aim is to find the optimum weight that would minimize the MSE. Modifying the weights in the negative direction of the MSE, should minimize the error surface because of the quadratic structure of the MSE (Zooghby, 2005). The gradient of the MSE can be calculated as;

$$\begin{aligned}\nabla_w \text{MSE}(\mathbf{w}(n)) &= 2\mathbf{R}\mathbf{w}(n) - 2E[\mathbf{x}(n+1)\mathbf{d}(n+1)] \\ &= 2\mathbf{x}(n+1)\mathbf{x}^H(n+1)\mathbf{w}(n) - 2\mathbf{x}(n+1)\mathbf{d}(n+1) \\ &= 2\mathbf{x}(n+1)\boldsymbol{\varepsilon}^*\end{aligned} \tag{2.3.6}$$

where ε is the error given by;

$$\boldsymbol{\varepsilon} = \mathbf{w}^H(n)\mathbf{x}(n+1) - \mathbf{r}(n+1) \tag{2.3.7}$$

and the weight vector can be updated as;

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\mathbf{x}(n+1)\boldsymbol{\varepsilon}^* \tag{2.3.8}$$

The constant μ, also called the step size, determines how close the weights approach the optimum value after each iteration and it controls the convergence speed of the algorithm. Typical values for the step size are $0 < \mu < Trace(R)$ (Zooghby, 2005).

2.3.1.2 The Recursive Least Squares Algorithm

An alternative to LMS is the exponentially weighted recursive least squares (RLS) algorithm. At the nth time instant , w(n) is chosen to minimize a weighted sum of past squared errors;

$$\min_{w(n)} \sum_{t=0}^{n} \lambda^{n-t} \left|\mathbf{d}(n) - \mathbf{w}^H(n)\mathbf{x}(t)\right|^2 \tag{2.3.9}$$

where λ is a positive constant less than one which determines how quickly previous data are forgotten.

The algorithm is initialized by first setting

$$\mathbf{R}^{-1}(0) = \frac{1}{\delta}\mathbf{I}, \quad \delta > 0 \tag{2.3.10}$$

$$\mathbf{w}(0) = 0 \tag{2.3.11}$$

where $\mathbf{I}$ is the identity matrix and δ is a very small number.

The weights are then updated as;

$$\mathbf{w}(n) = \mathbf{w}(n-1) - \mathbf{k}(n)\boldsymbol{\alpha}^*(n) \tag{2.3.12}$$

where $\mathbf{k}(n)$ and $\boldsymbol{\alpha}(n)$ are given by (Veen & Buckley, 1999);

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{R}(n-1)\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^{H}(n)\mathbf{R}(n-1)\mathbf{x}(n)} \tag{2.3.13}$$

$$\boldsymbol{\alpha}(n) = \mathbf{d}(n) - w^{H}(n-1)\mathbf{x}(n) \tag{2.3.14}$$

2.3.2 Spatial Reference Algorithms

Spatial reference (SR) algorithms rely on the information regarding the direction of arrival of the desired signal and its multipath components. If the received signal is a narrowband signal, the propagation delay between two neighboring antenna elements corresponds to a certain phase shift. SR algorithms exploit this fact to estimate the direction of arrivals (DOA) and, form the desired antenna patterns using this information as stated by Zooghby (2005). These patterns usually have main beams towards the direction of the desired user and nulls at the interferers' directions as stated in previous sections.

Beamforming by using spatial algorithms is achieved as follows. First some DOA estimation technique is applied to the received signal to determine the signal's DOAs and their associated amplitudes. The DOA with the maximum amplitude θ_{max}, which would indicate the strongest path, is selected and its array response vector a (θ_{max}) is chosen as the downlink beamforming weight (Zooghby, 2005).

The signals transmitted by each user will arrive at the BS from different directions due to multipath propagation. The SR algorithm will resolve these paths separately so that an optimum combining scheme can be used. The estimated DOAs can be directly used for downlink beamforming.

The array factor has to be known exactly to estimate the DOA's, all deviations from the ideal case will reduce the performance of the estimation process. Also the number of resolvable signals depends on both the number of antenna elements and the coherence of the incident signals which limits the performance of the algorithms.

If the signals arrive from angles which are too close to be resolved, the algorithm will not be able to perform its task anymore and one user must be assigned to another physical channel.

Practically the vertical angular spread can be neglected in most practical cases so that a linear array is satisfactory in most applications.

2.3.2.1 Multiple Signal Classification (MUSIC)

The approach of the MUSIC algorithm is to estimate the dominant subspace of the observations, and then find the elements of the array manifold that are closest to this subspace.

The array output may be written as;

$$\mathbf{x}(t) = A\mathbf{s}(t) + \mathbf{n}(t) \tag{2.3.15}$$

where $s(t)$ is the amplitude and phase of the signals and $n(t)$ is the additive noise at time instant t. If no noise were present, determining the direction of arrival's (DOA) was finding the unit elements of array manifold. In the presence of noise a different method is necessary since the observations are full rank.

The subspace estimation step is achieved by performing an eigendecomposition on the covaricance matrix R of the received data. If the noise and signals are uncorrelated, and the noise is spatially white the covariance matrix can be evaluated by;

$$\mathbf{R} = E[\mathbf{x}(t)\mathbf{x}^*(t)] = \mathbf{ASA}^* + \sigma^2\mathbf{I} \tag{2.3.16}$$

where $\mathbf{S}$ is the covariance matrix of the emitter signals, $\mathbf{A}$ is the array response vector and σ^2 is the noise power in each channel. $\mathbf{S}$ must be full rank to apply the MUSIC algorithm. The eigendecomposition of $\mathbf{R}$ has the following form as mentioned by Swindlehurst & Kailath (1992) ;

$$\mathbf{R} = \sum_{i=1}^{d} \lambda_i e_i e_i^* = \mathbf{E}_s\mathbf{\Lambda}_s\mathbf{E}_s + \sigma^2\mathbf{E}_n\mathbf{E}_n^* \tag{2.3.17}$$

where $\mathbf{E}_s$ is composed of n eigenvectors corresponding to the first d eigenvalues, where d is the rank of the matrix $\mathbf{A}$. The remaining (d-m) eigenvectors compose the E_n matrix. The span of the E_s defines the signal subspace and the $\mathbf{E}_n$ defines the noise subspace (Tamer & Kokturk, 2004). So that if R is available, we can evaluate the DOA's by finding the vectors on the array manifold that have zero projection in the noise subspace which are the zeros of the function (Swindlehurst & Kailath, 1992);

$$f(\theta) = \frac{\mathbf{a}^*(\theta)\mathbf{E}_n\mathbf{E}_n^*\mathbf{a}(\theta)}{\mathbf{a}^*(\theta)\mathbf{a}(\theta)} \tag{2.3.18}$$

However, due to various sources of error, only an estimate $\widehat{E}_n$ of the noise subspace eigenvectors is available (Swindlehurst & Kailath, 1992). So by using the MUSIC algorithm we estimate the DOA's as those values of θ that minimize the;

$$f(\theta) = \frac{\mathbf{a}^*(\theta)\widehat{\mathbf{E}}_n\widehat{\mathbf{E}}_n^*\mathbf{a}(\theta)}{\mathbf{a}^*(\theta)\mathbf{a}(\theta)} \tag{2.3.19}$$

2.3.2.2 Estimation of Signal Parameters via Rotational. Invariant Techniques (ESPRIT)

ESPRIT is similar to MUSIC in that it exploits the underlying data model and generates estimates that are asymptotically unbiased and efficient. In addition, the algorithm does not require knowledge of the array geometry and element characteristics; thus array calibration is not required, eliminating the need for the associated storage of the array manifold. It is also computationally less complex since it does not need a search procedure (Roy et al. (1986)).

Given the sensor array system defined by equations 2.3.15 and 2.3.16. If we define the array response vector as;

$$A = [A^T, (A\Phi)^T] \tag{2.3.20}$$

where the matrix Φ is a diagonal $k \times k$ matrix of the phase delays between the doublet sensors for the d wavefronts;

$$\Phi = diag[e^{j\phi_1}, ..., e^{j\phi_1}], \phi_k = \omega_0 \Delta \sin\theta_k / c \tag{2.3.21}$$

The $M \times k$ matrix A is the direction matrix whose columns are the signal direction vectors for the k wavefronts. Defining C_{zz} as;

$$C_{zz} = R_{zz} - \sigma^2 I = \begin{bmatrix} C_{xx} & C_{xy} \\ C_{yx} & C_{yy} \end{bmatrix} = \begin{bmatrix} ASA^* & AS\Phi^*A^* \\ A\Phi SA^* & A\Phi S\Phi^*A^* \end{bmatrix} \tag{2.3.22}$$

The basic concept upon which ESPRIT, is based is the elements of Φ can be obtained without knowledge of A. They are the non-zero generalized eigenvalues associated with the matrix pencil {C$_{xx}$,C$_{xy}$}. Furthermore, the associated generalized eigenvectors can be shown to be proportional to optimal signal copy vectors (Roy et al., 1987).

2.3.3 Blind Algorithms

Blind algorithms try to extract the unknown channel impulse response and the unknown transmitted data from the received signal by the antenna elements. Eventhough they do not know the actual bits, blind algorithms use additional knowledge about the transmitted signal like the constant envelope or the fixed symbol rate of the signal or the finite alphabet property of the modulation scheme.

Blind algorithms with training sequences are called the semi-blind algorithms which show better performance than temporal reference algorithms or blind algorithms alone (Laurila, 2000). The computational complexity of pseudo-blind algorithms is reduced in comparison to blind ones, which is a very important fact for their use in practical systems (Robert, 1998).

2.3.3.1 Constrained RLS Algorithm

A constrained algorithm is a temporal reference algorithm which does not require a desired signal, instead it only requires the look direction information, which is the probable direction of arrival of the signal from the desired user. In the adaptive process, the algorithm progressively learns statistics of noise and interference arriving from directions other than the look direction (Frost, 1972).

The constrained algorithms are able to maintain a chosen frequency response in the look direction, while minimizing the output noise power because of a simple relation between the look direction frequency response and the weights of the array elements. If we choose the look direction 0^o (perpendicular to the antenna array), then the wavefront received by the sensor at the center being the reference with 0 phase angle, each sensor at both sides will have equal phase delays. However waveforms arriving other than the look direction will not produce similar voltage components on any of the antennas. The interfering waveforms are canceled by using an adaptive algorithm.

For a specified look direction θ_i the constraint vector is given by

$$\underline{c}^{i^h} = [\ 1 \quad e^{j(2\pi/\lambda)d\sin\theta_i} \quad e^{j(2\pi/\lambda)2d\sin\theta_i} \quad \ldots \quad e^{j(2\pi/\lambda)(N-1)d\sin\theta_i}\ ] \tag{2.3.23}$$

and the array response is denoted as

$$\underline{r}^i = \underline{c}^{i^h} \underline{w} \tag{2.3.24}$$

So the optimized weight vector of the MVDR beamformer can be evaluated as (Huo & Lung, 1998)

$$\underline{w}^i_{opt}(n) = \frac{\underline{r}^i \underline{R}^{-1}_{xx}(n)\underline{c}^i}{\underline{c}^{iH} \underline{R}^{-1}_{xx}(n)\underline{c}^i} \tag{2.3.25}$$

2.4 Advantages of Smart Antenna Systems

Regardless of differetn techniques and algorithms mentioned in the previous section, smart antennas systems brings out important advantages to the communication systems to which they are applied.

The most important advantage of smart antenna employment is improving the network capacity. By using a smart antenna a single base station can serve more users sharing the same physical channel. This technique is called the Spatial Division Multiple Access (SDMA). If there exist two co- channel users as presented in Figure 2.8, then one of the adaptive arrays directs its main beam to the user 1 and a null of the array pattern to the user 2 and the other arrays directs its main beam to the user 2 while placing a null of the pattern in the direction of user 1. So, user 1 and user 2 can communicate with the base station on the same physical channel. More co-channel users can exist if we use more adaptive antennas in a smart antenna system.

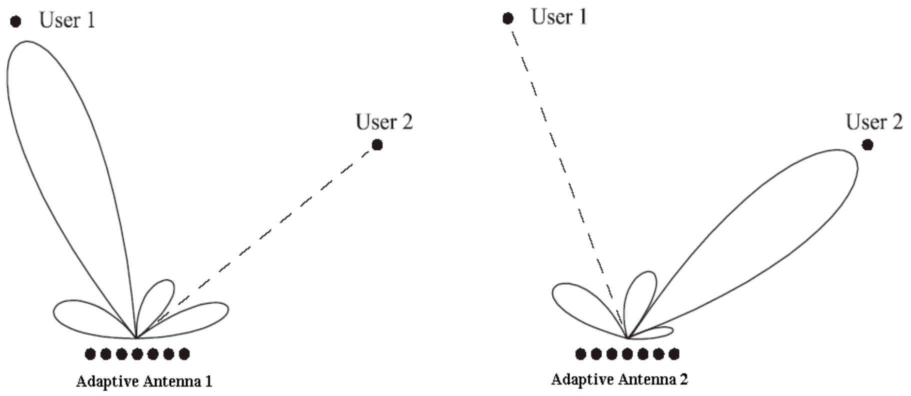


Figure 2.8 SDMA for mobile communications

Interference reduction on the downlink and on the uplink are also important advantages of smart antenna systems. Interference can be rejected using directional beams and/or by forming nulls in the base station receive antenna pattern in the direction of interfering co-channel users.

A base station equipped with smart antennas is less likely to interfere with nearby co-channel base stations than if it used an omnidirectional antenna.

Interfering users can be blocked by, forming base station antenna array radiation pattern nulls in the directions of interfering signals that come from co-channel mobiles. Interference rejection can also be fulfilled by steering maximas of the base station receiving antenna towards mobiles within a cell, just like the interference reduction approach on the downlink (Dietrich, 2000).

In rarely populated areas, extending coverage is often more important than increasing the capacity of the system. In such cases, the gain provided by adaptive antennas can extend the range of a cell to cover a larger area and more users, than would be possible with omnidirectional or sector antennas. The high gain of the smart antenna systems, also increases the transmission quality and the Quality of Service of the communication system and enable usage of higher data rates.

The transmitting power for the uplink and downlink are automatically adjusted in cellular communication systems. Thus increasing the transmitted power with a more directive beam, by using a smart antenna system, increases the power received by the user. So that the auto gain control system at the users mobile device reduces the transmitting gain and also systems power consumption.

2.5 Employing Smart Antennas in Mobile Communication Systems

The application of adaptive antennas is not feasible and practical in the mobile side of current mobile communication systems because of the limited physical dimensions for a large enough antenna array and also the signal processors (Robert, 1998) . So, throughout this thesis we consider adaptive antennas at the base station side only.

In mobile communications the signal from the user comes from an angle range around the nominal DOA due to local scattering. Since this angular spread strongly influences spatial separability, it is an important parameter for setting up applications using adaptive antennas. As the angular spread increases especially for urban environments, it becomes harder to employ a smart antenna system. Thus the best channel assignment strategy for the system is to mutually maximize the angle between the DOAs of the different users.

In existing systems such as GSM, changes in the system's architecture are necessary to exploit the benefits of smart antennas. The base station (BS) has to be equipped with the adaptive antenna facilities, and therefore the hardware and software of the BS have to be upgraded. But, because of the additional spatial information, employing SDMA also has an impact on the protocols used, thus they have to be updated too.

During call setup, if the user tries to establish a connection, the mobile station request a signalling channel for setting up the call. This call request signal is also used for calculating the weights for forming the beam of the smart antenna system. Since this request contains no training sequence, only algorithms that do not need this information (spatial reference or blind algorithms) can be used at the base station for calculating the weights.

If the base station requests a connection to the mobile station, the call request message has to be transmitted into the whole cell area, since the location of the user is not known by the base station initially. When the mobile station answers the call request the beamforming hardware calculates the weights of the antennas and forms the beam of the smart antenna system.

In conclusion, there are no principle restrictions preventing the utilization of SDMA in GSM and smart antenna systems can replace the conventional antennas in cellular communication systems., but much care has to be taken concerning protocol aspects (Robert, 1998). The cellular system has to control the smart antenna beamforming hardware to direct the maxima of the array to the desired user and null of the array to the interfering users.

CHAPTER THREE
SYSTOLIC ARRAYS FOR BEAMFORMING ALGORITHMS

Eventhough the smart antenna systems improve the capacity and communication quality of the communication systems, they require much processing power and may cause delays or performance degrading of the system.

Parallel algorithms improve the performance of the system by splitting the operations to many parts and processing them concurrently on distinct processors. As a result, many researchers have focused on parallel processing algorithms for signal processing applications. Systolic array is an architecture on which parallel algorithms can run. Systolic arrays are arrays of processors which are connected to the nearest neighbors in a mesh-like topology. Processors perform a sequence of operations on data that flows between them and operate concurrently. Each processor at each step takes in data from one or more neighbors, processes it and, in the next step, outputs results in the opposite direction(s).

3.1 Systolic Arrays

Systolic array, takes their names from the analogy with the regular pumping of blood by the heart. It is similar with the blood pumping of the heart since data flows synchronously between processing elements of the array with each trigger signal for example a clock signal.

A systolic array is an arrangement of processors in an array where data flows synchronously between neighbors across the array, usually with different data flowing in different directions. Each processor at each step takes in data from one or more neighbors, processes it and, in the next step, outputs results in the opposite direction(s) and the result of the operation can be evaluated from one of the cells one by one. Two basic types of systolic arrays; linear and rectangular were presented in Figure 1.5 in Chapter One. "H. T. Kung and Charles Leiserson were the first to publish a paper on systolic arrays in 1978 (Kung & Leiserson, 1978) , and coined the name" as mentioned in Chapter One.

Matrix multiplication is a good example for a systolic algorithm application. As presented in Figure 3.1 one matrix is fed in a row at a time from the top of the systolic array and is passed downwards, while the other matrix is fed in a column at a time from the left hand side of the systolic array and passes from left to right. Dummy values occurs in the cells until each processor has seen one whole row and one whole column of the input data. At this point, the result of the multiplication is stored in the array and can now be output a row or a column at a time (Kung, 1985).

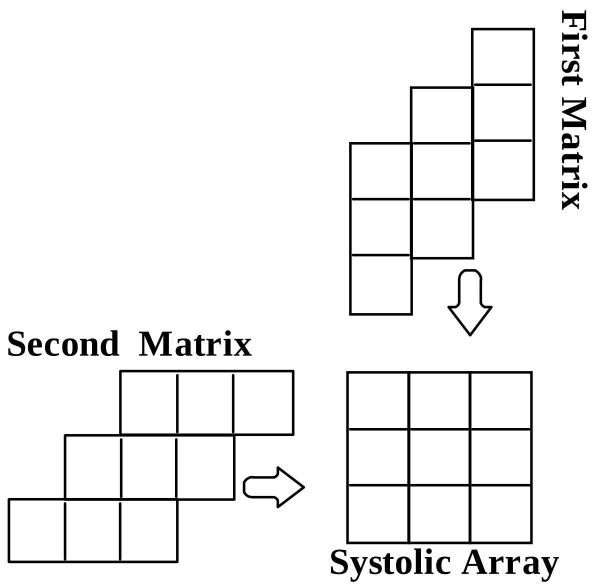


Figure 3.1 Matrix multiplication using a systolic array

Systolic array-based recursive least squares algorithm has first been proposed by (Gentleman & Kung, 1981). Their algorithm was based on the QR decomposition technique and their structure was pipelined on a triangular array. However, their pipelined structure consisted of two separate steps of QR-updates and backward substitution which made it very hard to realize.

A fixed parallelogram structure for the parallel weight extraction was proposed by McWhirter (1990). His approach was based on first adapting the array to the input data and then freezing the parameters and evaluating the weights. Updating and evaluating the weights is not efficient, as the systolic array needs to be frozen at each update period.

McWhirter & Shepherd (1989) suggested a systolic array for a minimum variance distortionless response (MVDR) beamformer based on first forming the upper triangular matrix by using Given's rotations and then freezing the array and evaluating the inverse covariance matrix and store it in the processing elements. The procedure is then run in the first mode and the weights are obtained at the outputs consequently.

A systolic array architecture for parallel weight extraction without the need for forward and backward substitution is presented by Tang et al. (1994). Their algorithm is mainly based on the systolic array presented by McWhirter & Shepherd (1989) and function recursively to update the instantaneous optimal weight vector after the initialization process. Their systolic architecture is also a modular and expandable one that makes it suitable for VLSI hardware implementation.

3.2 The QRD RLS Algorithm

Weights ($\underline{w}(n)$) of the sensors of the RLS algorithm at time t_n can be found as

$$\underline{\mathbf{R}}_{xx}(n)\underline{\mathbf{w}}(n) + \underline{\boldsymbol{\varrho}}(n) = \underline{0} \tag{3.2.1}$$

Here $\underline{R}_{xx}$ is the $(M-1) \times (M-1)$ data covariance matrix $\underline{X}$, where M is the number of sensors, and $\underline{\varrho}(n)$ is the $(M-1)$ element cross correlation vector McWhirter (1990) which is given by

$$\begin{aligned} \underline{\mathbf{R}}_{xx}(n) &= \underline{\mathbf{X}}^H(n)\underline{\mathbf{X}}(n) \\ \underline{\boldsymbol{\varrho}}(n) &= \underline{\mathbf{X}}^H(n)\underline{\mathbf{d}}(n) \end{aligned} \tag{3.2.2}$$

We can use the QR decomposition technique, used in solving matrices, for evaluating equation 3.2.1. The QR decomposition of a matrix is the decomposition of the matrix into an orthogonal and a triangular matrix. This matrix decomposition can be used to solve linear systems of equations like the linear least squares problem.

The QR decomposition can be applied to the least squares problem given above as;

$$\underline{\mathbf{Q}}(n)\underline{\mathbf{X}}(n) = \begin{bmatrix} \underline{\mathbf{R}}(n) \\ \underline{0} \end{bmatrix} \tag{3.2.3}$$

where $\underline{Q}(n)$ and $\underline{R}(n)$ denote $(M-1) \times (M-1)$ orthogonal matrix and $(M-1) \times (M-1)$ upper triangular matrix respectively. Since $\underline{Q}(n)$ is an orthogonal matrix the residue vector $\underline{e}(n)$ can be evaluated as;

$$\|\underline{\mathbf{e}}(n)\| = \left\| \begin{bmatrix} \underline{\mathbf{R}}(n) \\ \underline{0} \end{bmatrix} \underline{\mathbf{w}}(n) + \begin{bmatrix} \underline{\mathbf{u}}(n) \\ \underline{\mathbf{v}}(n) \end{bmatrix} \right\| \tag{3.2.4}$$

where

$$\begin{bmatrix}\underline{\mathbf{u}}(n)\\ \underline{\mathbf{v}}(n)\end{bmatrix} = \underline{\mathbf{Q}}(n)\underline{\mathbf{y}}(n) \tag{3.2.5}$$

and the $\underline{d}(n)$ denotes the reference data of the systolic array (McWhirter, 1990). The RLS weight vector that minimizes $\|\underline{e}(n)\|$ can be computed by

$$\underline{\mathbf{R}}(n)\underline{\mathbf{w}}(n) + \underline{\mathbf{u}}(n) = \underline{0} \tag{3.2.6}$$

Since our system is a real time one we need to refresh $\underline{R}(n)$ and $\underline{u}(n)$ as

$$\underbrace{\underline{\mathbf{Q}}^T \cdot \begin{bmatrix}\underline{\mathbf{R}}\\ \underline{\mathbf{x}}^T\end{bmatrix}}_{\begin{bmatrix}\underline{R}\\ 0\end{bmatrix}} \cdot \underline{\mathbf{w}} = \underbrace{\underline{\mathbf{Q}}^T \cdot \begin{bmatrix}\underline{\mathbf{y}}\\ \underline{\boldsymbol{\beta}}^T\end{bmatrix}}_{\begin{bmatrix}\underline{\mathbf{u}}\\ \zeta\end{bmatrix}} \tag{3.2.7}$$

Thus for each new data vector we first need a triangular update step and and then backsolve the previous equation (Manolakis et al., 2005).

3.3 Implementation of the QRD RLS Algorithm using Givens Rotations

3.3.1 Givens Rotations

A Givens rotation can be represented by a matrix as;

$$G(i,k,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \tag{3.3.1}$$

where c represent the cosine term and s represent the sine term given by $c = cos(\theta)$, $s = sin(\theta)$. The c and s terms appear at the intersection points of the i^{th} and k^{th} rows and columns as given in equation

$$\begin{aligned} g_{ii} &= g_{kk} = c \\ g_{ik} &= -g_{ki} = s \end{aligned} \tag{3.3.2}$$

The result of the product of a vector x with the matrix $G(i, k, \theta)$ is the x vector rotated θ radians in the i, k plane. The main usage of the Givens Rotations is to introduce zeros in vectors or matrices.

If the matrix $G(i, k, \theta)$is multiplied with matrix A from the left, $G \cdot A$ as given by;

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \tag{3.3.3}$$

If we know the values of a and b, we can calculate the c and s parameters of the rotation matrix, with the resultant matrix as;

$$\begin{aligned} r &= \sqrt{a^2 + b^2} \\ c &= \frac{a}{r} \\ s &= \frac{b}{r} \end{aligned} \tag{3.3.4}$$

3.3.2 Applying Givens Rotations to the QR RLS problem

The orthogonal triangularization of the input data matrix can be carried out using a sequence of Givens rotations (Gentleman & Kung, 1981) which is explained briefly in the previous subsection. Givens rotations are used to eliminate the leading nonzero element x_i of one vector by rotating it with the corresponding term r_i of the other, c and s represent the cosine and sine of the rotation angle which are given in what follows.

$$\begin{aligned} r_i^{'} &= \sqrt{r_i^2 + x_i^2} \\ c &= \frac{r_i}{r_i^{'}} \\ s &= \frac{x_i}{r_i^{'}} \end{aligned} \tag{3.3.5}$$

where $r_i^{\prime}$ denotes the updated r$_i$ parameter.

3.4 Systolic Array Based Implementation of the QRD RLS Algorithm

Givens Rotations can be applied in a systolic array structure proposed by (Gentleman & Kung, 1981) as presented in Figure 3.2. Each processing element in this systolic structure is called the "cell". There exists three types of cells for this structure; boundary, internal cells and the "last cell" which are demonstrated in detail in Figure 3.3. Inputs denoted by "x" from the upper side of the systolic array are the signals received by the antennas, while the input denoted as "d" is the reference signal received by the reference antenna.

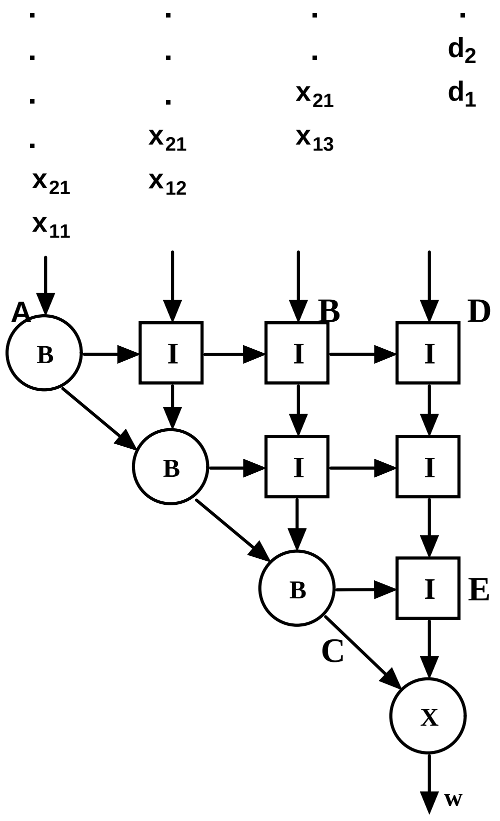


Figure 3.2 Systolic array structure for the QRD RLS algorithm

The boundary cells have the duty of calculating and updating the Givens Rotation parameters c and s according to the input data value and pass these values to the internal cells at the same row to apply the Givens Rotation to the whole row. The internal cells get the c and s values from the neighbor cells and apply them to the data they receive from the upper neighbor cells. They also feed their bottom neighbor with the new data. The last cell multiplies the two inputs and gives the result at the output.

Table 3.1 depicts the functions of the cells for adaptive and frozen mode operations of the systolic array. The parameters with a comma over them denote the updated parameter of the same parameter depicted with the same letter. As can be seen in Figure 3.2 the systolic array is composed of two main parts, a triangular part marked with letters ABC and the column on the most right marked with letters DE. In the adaptive mode of operation, the triangular part of the

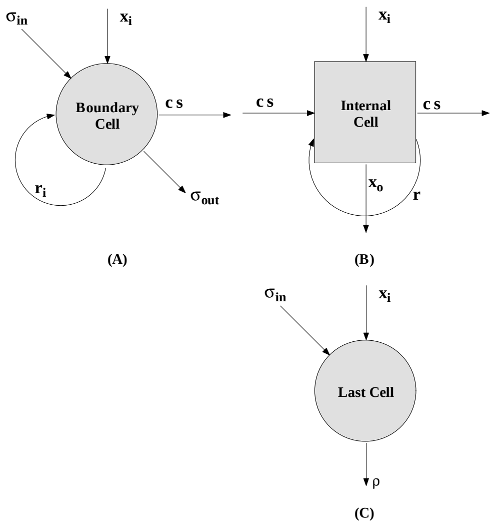


Figure 3.3 Cell types of the systolic array for the QRD RLS algorithm

systolic array performs the QR decomposition of the incoming data matrix. Elements of the evolving triangular matrix are stored in the triangular part of the systolic array and as the data vector passes downwards, it is completely eliminated by rotating one element to zero at each row. As a result of eliminating the data vector x(n) the stored triangular matrix is updated. The right hand column of cells marked as DE applies the Givens Rotations parameters calculated by the boundary cells to the reference signal $\mathbf{d}(n)$ which is also rotated to produce $\mathbf{u}(n)$.

When the systolic array is switched to frozen mode it operates like a fixed linear combiner with the constant weight vector $\mathbf{w}(\mathbf{n})$. If a p dimensional unit matrix is input to the main triangular array, while a vector of zeros is input to the DE column the corresponding (M-1) element output vector will be the required weight vector. The operation of the systolic array can be seen in Figure 3.4 (McWhirter, 1990).

The algorithm briefly explained above uses the Givens rotation as a building block and it can be implemented by a fully parallel and pipelined triangular systolic array. But unfortunately after the adaptive mode of operation we have to back substitute to evaluate the weights which makes this approach inapplicable for a pipelined processor (Proudler et al., 1996).

Table 3.1 Functions of cells of the QRD RLS algorithm

Cell Type	Adaptive Mode	Frozen Mode
Boundary	$r' = \sqrt{r^2 + \lvert x \rvert^2}$ if x=0 $c = 1, s = 0$ otherwise $c = r/r', s = x/r'$ $r = r', \delta_{out} = c\delta_{in}$	$c = 1$ $s = x/r$ $\delta_{out} = \delta_{in}$
Internal	$x' = cx - sr$ $r = sx + cr'$	$x' = x - sr$

3.5 The QRD Based Constrained RLS MVDR Beamformer

A fully parallel and pipelined systolic array for the MVDR adaptive weight extraction system without the need for forward or backward substitution is proposed by Tang et al. (1994).

The upper triangular matrix R(n) is given by;

$$R(n) = Q(n)X(n)$$

where $\mathbf{X(n)}$ is the n snapshot data matrix of n sensors, therefore with size $n \times n$ and $\mathbf{Q(n)}$ is a $n \times n$ unitary matrix. The initial parameter vector is given by;

$$s^i(n) = R^{-H}(n)c^i \tag{3.5.1}$$

where c^i is the constraint vector, denoting the response of the array to a specific angle. The QR decomposition can be used to update the optimal weights recursively as;

$$Q(n)X\,(n) = \widehat{Q}(n) \begin{bmatrix} \beta R(n-1) \\ 0 \\ \vdots \\ x(t_n) \end{bmatrix} = \begin{bmatrix} R(n) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.5.2}$$

and updating the parameter vector s^i is given by;

$$\widehat{Q}\,(n) \begin{bmatrix} \frac{1}{\beta}s^i(n-1) \\ \sharp \\ 0 \end{bmatrix} = \begin{bmatrix} s^i(n) \\ \sharp \\ \sharp \end{bmatrix} \tag{3.5.3}$$

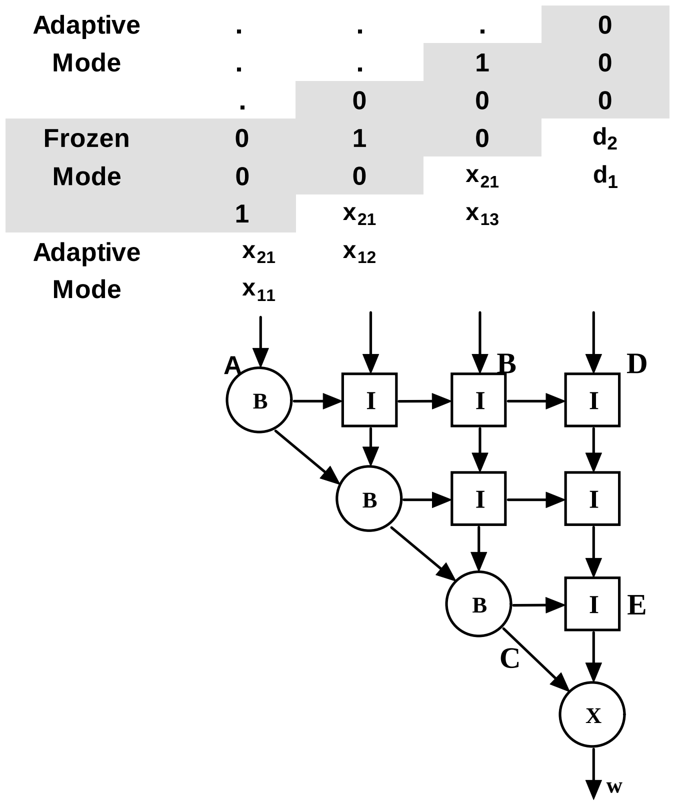


Figure 3.4 Operation of the systolic array for QRD RLS algorithm

where the sign $\sharp$ denotes the existing parameters of the corresponding matrix/vector. The same unitary matrix used to update the upper triangular matrix, $R(n-1)$, can be used to update the lower triangular matrix $R^{-H}(n-1)$

$$\widehat{Q}(n)\begin{bmatrix}\frac{1}{\beta}R^{-H}(n-1)\\ \sharp\\ 0\end{bmatrix}=\begin{bmatrix}R^{-H}(n)\\ \sharp\\ \sharp\end{bmatrix} \tag{3.5.4}$$

The combined updating equation of the system is given by;

$$\widehat{Q}(n)\begin{bmatrix}\beta R(n-1) & \vdots & \frac{1}{\beta}s^i(n-1) & \vdots & \frac{1}{\beta}R^{-H}(n-1)\\ 0 & \vdots & \sharp & \vdots & \sharp\\ x(t_n) & \vdots & 0 & \vdots & 0\end{bmatrix}=\begin{bmatrix}R(n) & \vdots & s^i(n) & \vdots & R^{-H}(n)\\ 0 & \vdots & \sharp & \vdots & \sharp\\ 0 & \vdots & \sharp & \vdots & \sharp\end{bmatrix} \tag{3.5.5}$$

The weight vector is given by;

$$\widehat{w}^{i^T}(n)=s^{i^T}(n)R^{-H*}(n)$$

Substituting equations for updating the weights with the MVDR weight equation given in

Chapter Two we get the MVDR weight vector for the QRD CRLS algorithm as;

$$w^{i^T}(n) = \frac{r^i}{|s^i(n)|^2}\widehat{w}^{i^T}(n) \tag{3.5.6}$$

The QRD CRLS algorithm can run on a systolic array given in Figure 3.5. The input signals from the antennas are acquired by the cells at the top of the columns. The weights of the system can be extracted from the cells stated by "C-5" that are at the bottom of the columns on the right.

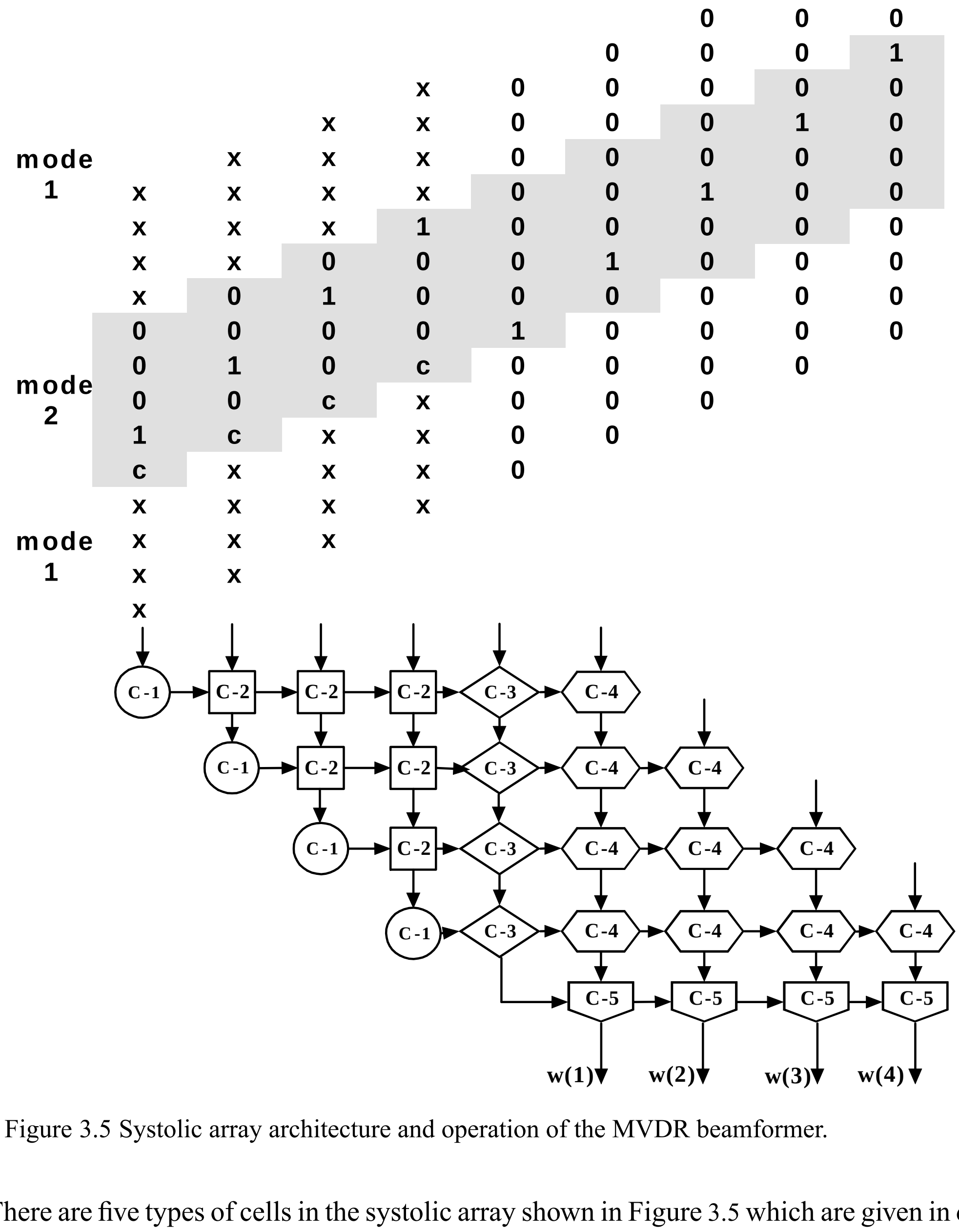


Figure 3.5 Systolic array architecture and operation of the MVDR beamformer.

There are five types of cells in the systolic array shown in Figure 3.5 which are given in detail with input and output diagrams in Figure 3.6. The C-1 type cell calculates the Givens Rotations parameters and passes them to the other cells on the same line to apply the QR decomposition to the whole data matrix. The C-2 type cell gets the Givens Rotations parameters, calculated by the neighboring C-1 type cell and applies them to the data acquired from its upper neighbor or the sensor. So, the upper triangularization of the data matrix is accomplished. The C-3 and C-4 types of cells perform the multiplication and accumulation operations to compute the optimal weight vector and the C-5 type of cell normalizes the optimal weight vector.

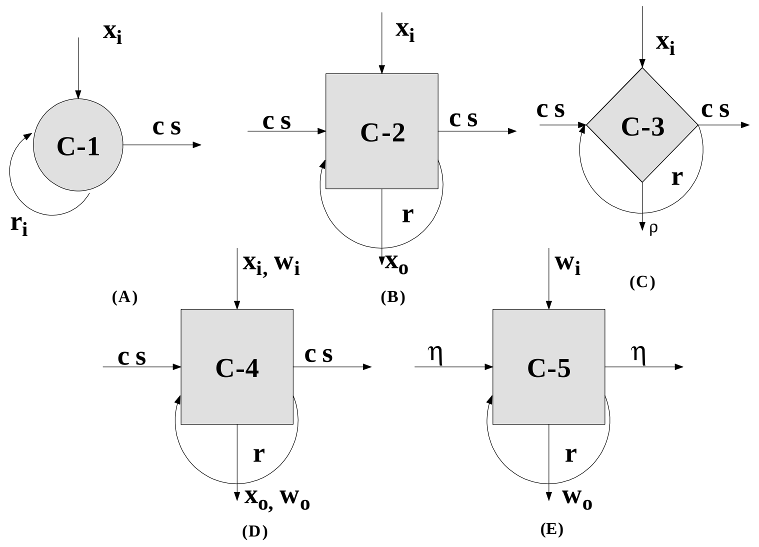


Figure 3.6 Cells of the systolic array for MVDR beamforming

Operation of the systolic array can be observed in the upper part of Figure 3.5. The system is first run in the adaptive mode for "n" cycles, where n is the number of antennas, to form and store the upper triangular matrix in cells C-1 and C-2. The adaptive mode and the frozen mode operations of the cells is presented in Table 3.2. During this phase, the input to the first row of the C-1 and C-2 type cells, is the acquired data from the antenna array, while input of the other cells are zero.

Within the second phase of operation, the cells operate in the frozen mode for n+1 cycles. The input to the left part of the systolic array is the constraint vector followed by an identity matrix. While an identity matrix is the input for the right hand part of the systolic array.

The cells of the systolic array operate in adaptive mode at the recursive updating phase of operation, the inputs of the first row of the C-1 and C-2 cells are the acquired data matrix, while the input to the other cells are zero. The optimal weights of the beamformer can be acquired at the outputs of the C-5 type cells consequently. The adaptive and frozen mode functions of the cells are denoted in Table 3.2 (Tang et al., 1994).

Table 3.2 Functions of cells of the QRD RLS based MVDR beamformer

Cell Type	Adaptive Mode	Frozen Mode
C-1	$r' = \sqrt{\beta^2 + \|x\|^2}$ $c = \frac{\beta \times r'}{r}$ $s = \frac{x}{r}$ $r = r'$	$s = \frac{x}{r}$ $c = 1$
C-2	$x' = -s\beta r + cx$ $r' = c\beta r + s^* x$	$x' = cx - sr$
C-3	$x' = \frac{-sr}{\beta} + cx$ $r' = \frac{cr}{\beta} + s^* x$ $y = r'$ $\eta' = \|r\|^2 + \eta$	$x' = x$ if $x = 1$ then $r = s^*$
C-4	$x' = \frac{-sr}{\beta} + cx$ $r' = \frac{cr}{\beta} + s^* x$ $y = r'$ $w' = yr^* + w$	$x' = x$ if $x = 1$ then $r = s^*$
C-5	$w' = \frac{rw}{\beta^{2\eta}}$	$w' = w$

CHAPTER FOUR
FOLDED SYSTOLIC ARRAY BASED QRD RLS ALGORITHM

Advantages and usages of systolic arrays have been mentioned in the previous chapter. However computational complexity of the systolic arrays can exceed the physical limitations of a fixed VLSI structure like the included logic elements or the number of pins. These limitations could be overcome by dividing the problem into smaller problems and remapping them for the VLSI structure. The division process of the main systolic array to smaller structures is called the partitioning of the systolic array.

Partitioning operation must satisfy some conditions in order to be considered as applicable. These conditions are listed by Moldovan & Fortes (1986) as;

- Accuracy of the algorithm must not be affected from the partitioning process.
- No additional time delays must be inserted by the partitioning process.
- Partitioning must not cause a raise to the resources used by the systolic structure.
- Numerical stability of the algorithm must not be affected from the partitioning scheme.

4.1 Partitioning and Folding a Systolic Array

The purpose of partitioning is to solve the given problem by, utilizing the limited resources regardless of the actual size of the problem as mentioned above. The systolic array structure can be partitioned properly to be folded in to a smaller array structure. There exists two basic partitioning schemes; locally sequential globally parallel (LSGP) and locally parallel globally sequential (LPGS) (Jainandunsing, 1986). Both partitioning schemes for a 6x6 systolic array can be seen in Figure 4.1.

In LSGP, the main structure is partitioned into tiles which contain cells of the systolic array. The tiles execute in parallel while the cells inside the tiles operate in serial. The size of the array is equal to the total number of tiles and the period of the array is lower bounded by the block size. The schedule of operations in the LSGP scheme, must not assign more than

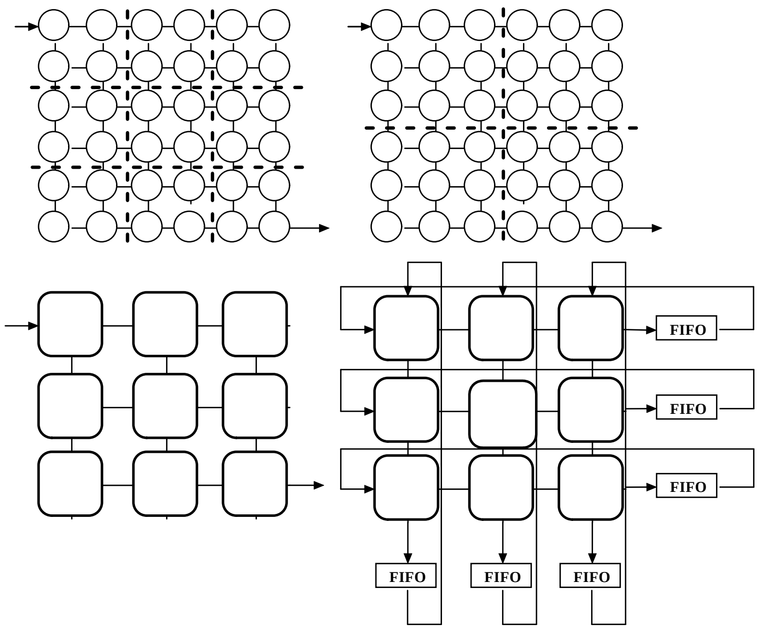


Figure 4.1 Systolic Array Partitioning Schemes

one computation to each processing element at the same time instant which brings out that simultaneous operations must not be on the same block Burleson (1991).

For the LSGP scheme the local memory of each processing element is lower bounded by the tile size since the local memories of the nodes get mapped to the same processing element. Internal buffers are required to hold the internal tile parameters for the sequential operation.

The main structure of the LPGS is again partitioned into tiles containing cells of the systolic array, but this time tiles execute in serial while cells inside the tiles operate in parallel. The array size in the LPGS scheme is equal to the tile size and the array is lower bounded by the number of tiles. Since the tiles are stacked, the internal communication inside the tiles requires external buffers and wires whose lengths are functions of the block size (Burleson, 1991).

Some limitations in scheduling of the tiles exist in LPGS scheme as; there must be no dependencies which point backward in time and no two nodes must be assigned to the same processor element at the same time (Burleson, 1991).

The LPGS scheme tends to achieve better load balancing and requires less local memory than LSGP scheme. However LPGS scheme imposes extra constraint on partitioning and requires much more inter-processor communication. As the communication processes aggravates, the LSGP scheme, having fewer inter-processor communications, will perform better than the LPGS scheme. (Hwang & Hen, 1992).

The LSGP partitioning scheme has been preferred for our application since it does not need any external buffering and having less inter processor communications for large arrays..

4.1.1 Reusage of Functions

Partitioning of the systolic arrays brings the benefit of reusage of common functions of the cells inside the tile. For the LSGP partitioning scheme if some or all of the functions of the cells inside the tiles are identical then they can be used as the functions of the corresponding cell at each time interval.

Figure 4.2 demonstrates a tile which includes four cells of the same kind. Since the partitioning scheme is chosen as the LSGP these cells will operate sequentially inside the tile. The sequential operation turn of the cells are given in parenthesis. Ports are also numbered according to the cells they are connected. Right hand side of the Figure is the functional diagram of the Figure which shows the reusage of the functions, inside the tile while inputs and outputs are multiplexed according to its turn.

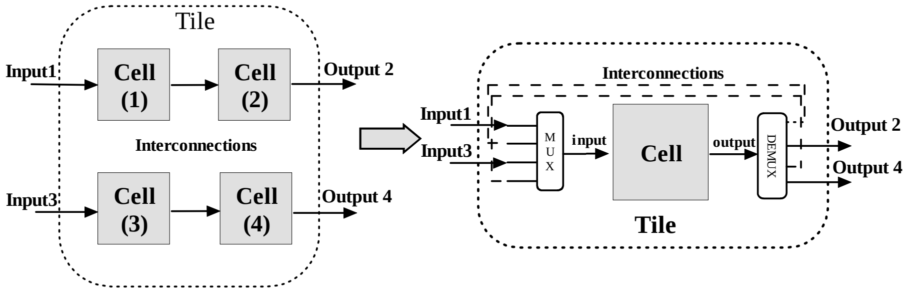


Figure 4.2 Reusage of the cell functions inside tiles

4.2 Folded Systolic Array Based MVDR Algorithm

The systolic array structure for the QRD CRLS algorithm presented in Chapter Three can be partitioned into tiles containing three cells as mentioned by Lijun & Parhi (2000). However this structure does not let us to build tiles containing more than three cells because of its rhomboid geometry. We can rearrange the structure of this systolic array, while preserving its functionality and systolic architecture. Since a rectangular structure is better than the rhomboid one, for building tiles including more cells, the structure of the systolic array will be rearranged to form a rectangular structure (Tamer & Ozkurt, 2007).

In order to make the systolic structure more appropriate for partitioning, we propose a new systolic structure for the QRD CRLS based MVDR beamformer algorithm which is presented in Figure 4.3. As can be seen in Figure 3.6 both C-2 and C-3 types of cells acquire the c and s parameters horizontally, which are generated by the C-1 type of cells on the boundary of the systolic array. Thus the C-3 type of cells can be shifted to the left of the first type of cells without effecting the functionality and the flow of the systolic array operation since none of the horizontal parameters, that the shifted cells require are modified by the C-2 cells. Nevertheless the C-2 cells still apply the Givens Rotations parameters to the data they received vertically so that the c and s parameters generated by the C-1 cells, are applied to the systolic array, properly.

Since the C-3 type of cells are moved to the left, the rest of the cells which remain on the right hand side in Figure 3.5, must be moved to the left also. The flow direction for the shifted part of the systolic array must be reversed, to avoid a change in the functionality. As presented in Figure 4.3 the data flow on the right of the first type of cells is unaffected while it is reversed on the left.

After rearranging the structure of the systolic array we can group the cells to form tiles that contain four cells each, as can be seen in Figure 4.4. Since the C-5 type of cells have the duty of normalizing the outputs by multiplying the weights by a variable, they are kept out of the tiles and used as additional processes.

As a result of grouping the cells, we have four types of tiles which can be seen in Figure 4.5; Tile-1, Tile-2 Tile-3 and Tile-4. Tile-1 contains C-3 and C-4 type of cells, Tile-2 contains C-1, C-2, C-3 type of cells while Tile-3 contains C-4 and Tile-4 contains C-2 type of cells only.

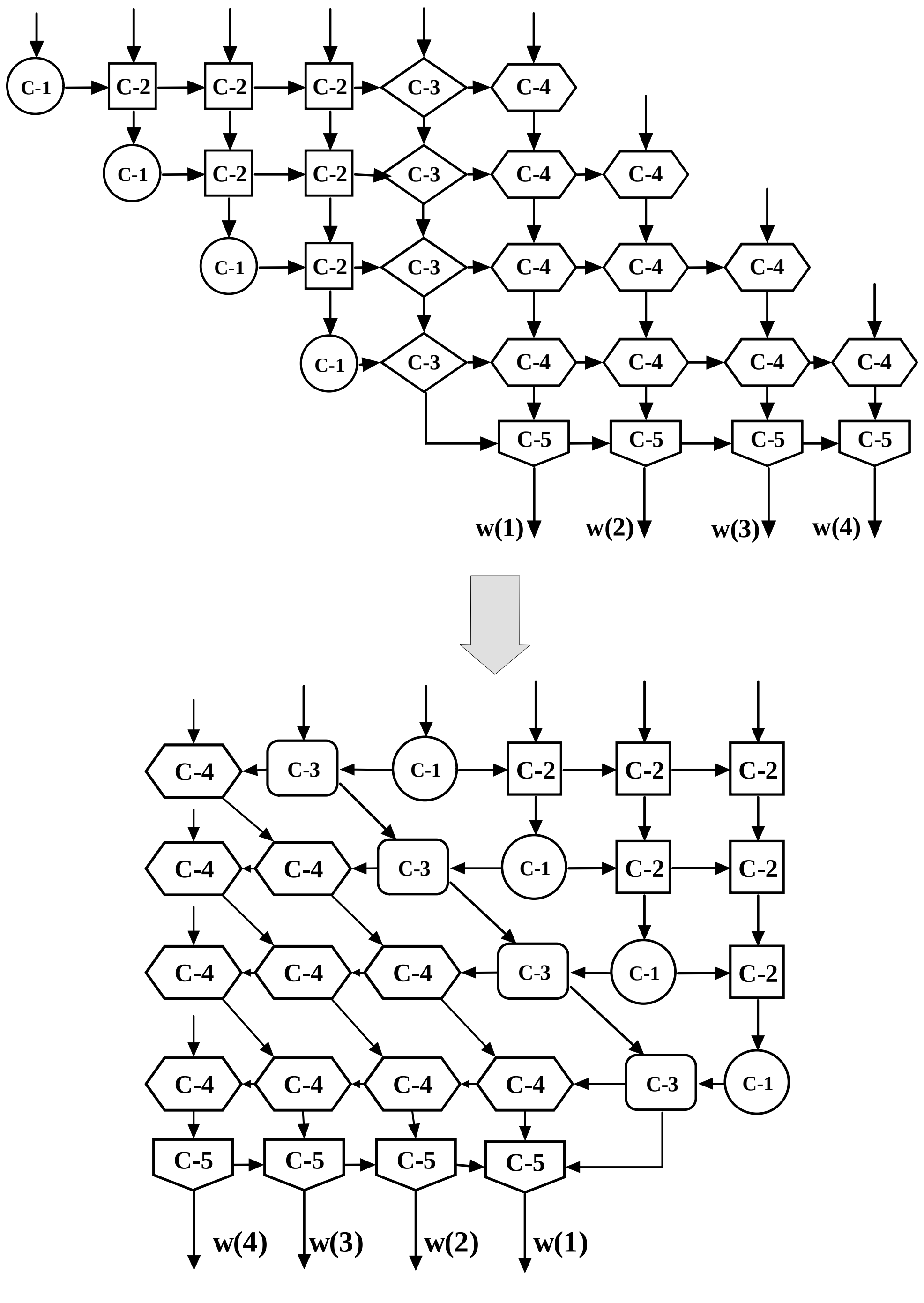


Figure 4.3 Transformation of the conventional MVDR systolic array to the rectangular one

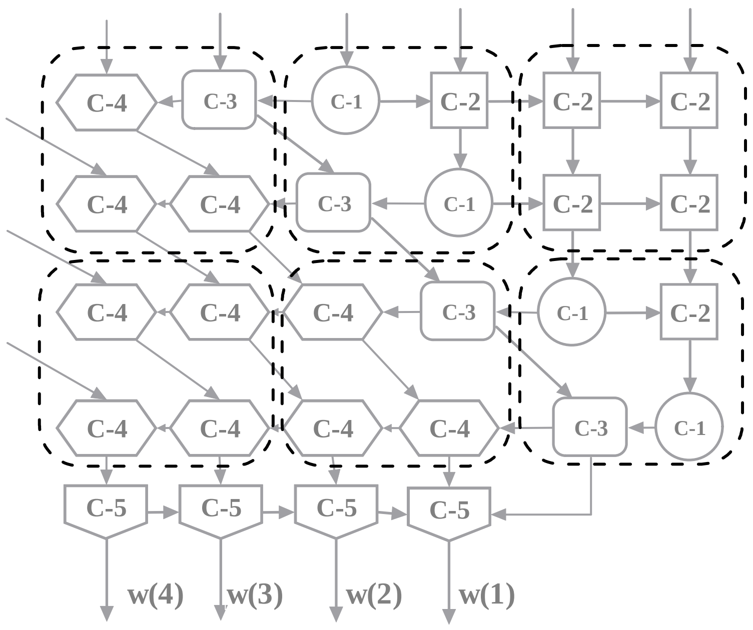


Figure 4.4 Grouping of the cells for the proposed systolic array

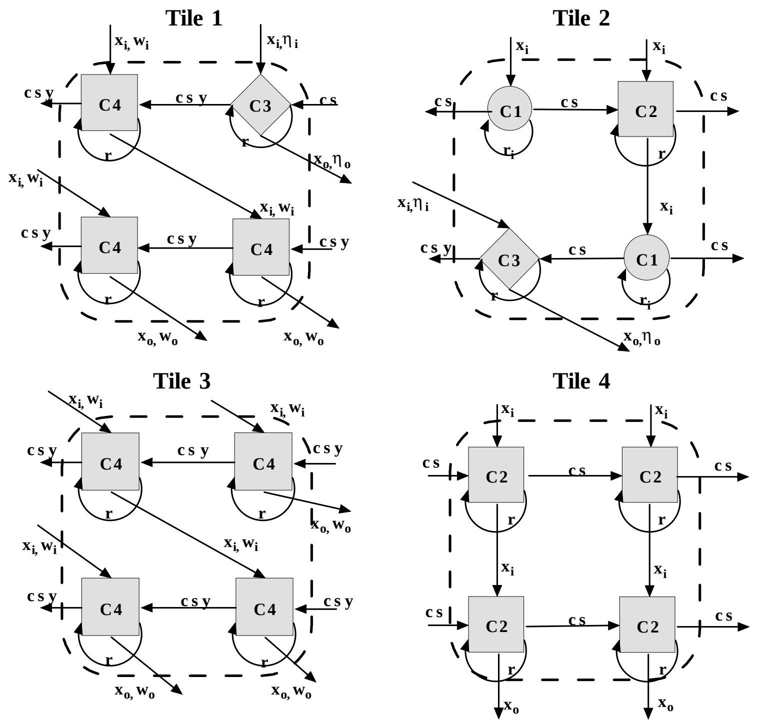


Figure 4.5 Internal structure of the tiles

As mentioned in subsection 4.1.1 identical functions in these cells can be used in common, so that each cell uses the same function at its turn. Thus, the number of functions inside a tile is less than the total number of functions of the included cells. For example Tile 4 contains four C-2 type cells and all the functions of these cells, certainly, are identical. Then it is possible to make each cell to use the same function during its turn and pass the parameters to the neighboring cell or outside of the tile. The reusage of the functions for Tile 4 can be seen in Figure 4.6 where the internal structure of it is given. The functions of the C-2 type of cell operate sequentially inside Tile 4 as shown in the Figure. Thus main operation inside Tile 4 becomes switching of the input and output parameters for the C-2 cell functions. This reusage of the functions reduces the number of operations inside the tiles.

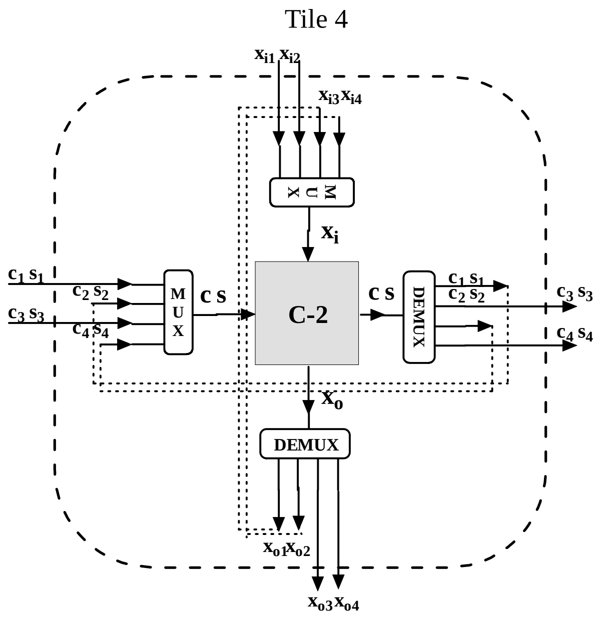


Figure 4.6 Reusage of functions inside Tile 4

Using the proposed tile structures the systolic array based QRD CRLS MVDR beamformer can be built as shown in Figure 4.7. The systolic array structure can easily be expanded for larger sensor arrays by just adding tiles presented in this work. As an example a systolic array structure for eight antennas is presented in Figure 4.8.

4.2.1 Operation of the Folded Systolic Array

The operation of the proposed folded systolic array structure for QRD CRLS algorithm is similar to the conventional one which is introduced in Section 3.4. The input of the systolic

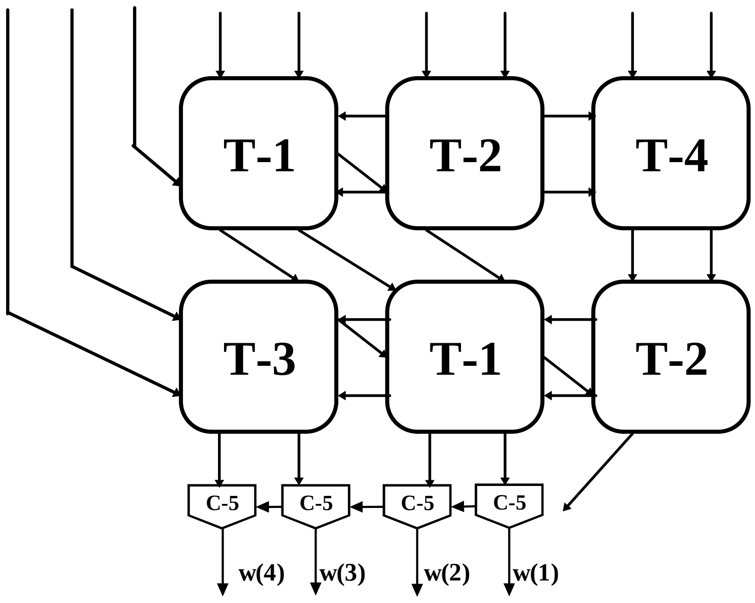


Figure 4.7 Folded sytolic array based MVDR beamformer

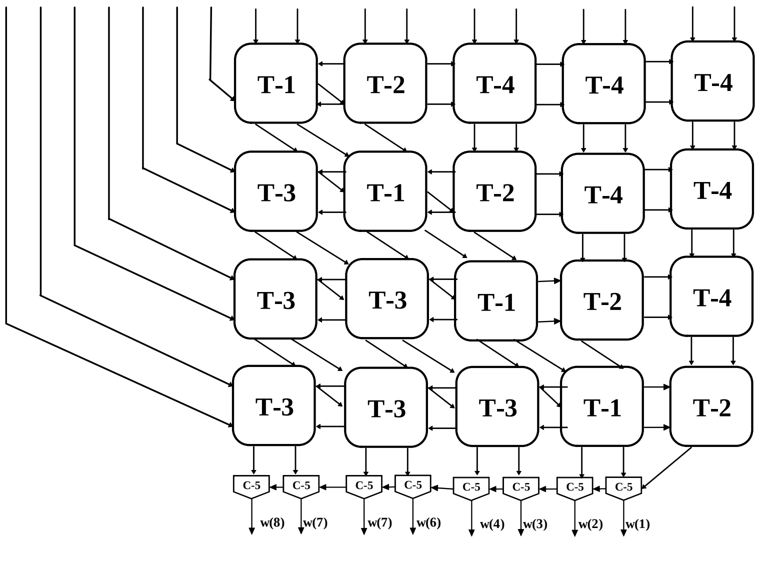


Figure 4.8 Folded systolic array structure for 8 antennas

array including the data and the identity matrices are changed according to the shifts of the cells which are described in section 4.2. So, the operation scheme for a systolic array for four antennas is presented in Figure 4.9

Upper part of the Figure denotes the input of the system. In the grey shaded part, the system operates in frozen mode and the following parts of the inputs denote the adaptive operation of the system. Inputs denoted by ’x’ denotes the measured data from the corresponding sensor while inputs denoted by ’c’ points out the constraint values. A more detailed description of the adaptive and frozen modes of the operation can be found in Chapter Three. Each line on the input table lasts for four clock cycles to let the serial operation of the cells inside the tiles. The optimal weights of the beamformer can be acquired at the outputs of the C-5 type of cells consequently. Arrays of larger size also have similar operation schemes.

The operation scheme evinces a major advantage of the proposed systolic array structure. For the conventional systolic structure first weight vector is obtained at 13^{th} clock cycle, however, for the rectangular systolic structure proposed in this thesis the weights are obtained at 10^{th} clock cycle since the shifted cells receive the c and s parameters earlier. This advantage brings a better throughput ratio for the rectangular systolic array.

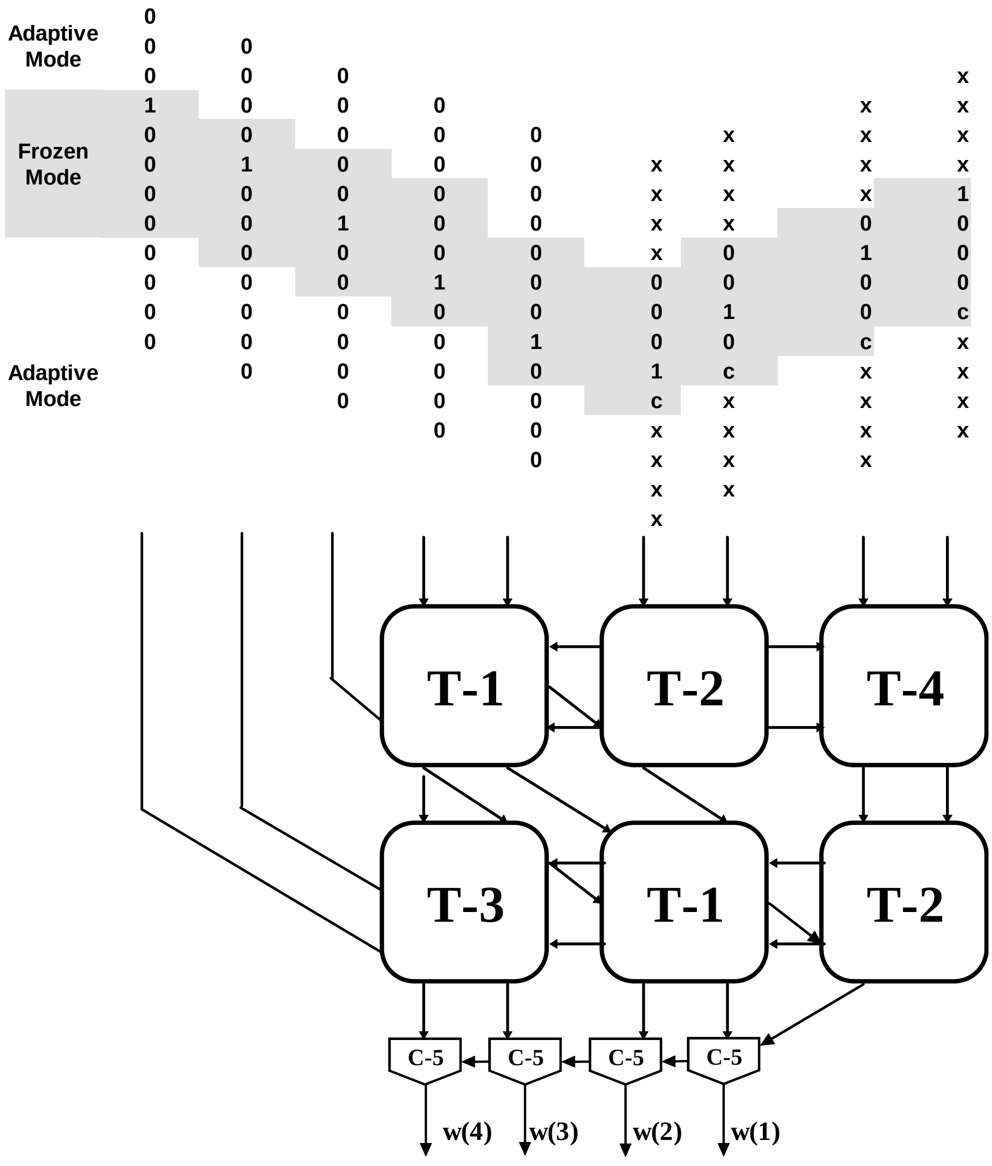


Figure 4.9 Operation of the Folded Systolic Array

CHAPTER FIVE
IMPLEMENTATION OF THE SYSTEM

A detailed block diagram of the implemented system can be seen in Figure 5.1. As you can observe the implementation mainly consists of three parts; the receiver part, the signal processing part, and the transmitter part.

At the receiver part the signal received by the antennas is amplified to a proper level for the I/Q demodulator. The task of the I/Q demodulator circuitry is to downconvert the signal to the baseband level with inphase (I) and quadriphase(Q) components. The representation of the baseband signal in I and Q components lets us to extract the phase and amplitude difference between the antennas which will be explained in detail in the following sections.

The I/Q demodulators are followed by analog to digital converters (ADC) which are the onboard equipments of the signal processing board. The Altera DSP Development Board is equipped with 2 ADC inputs. Since we have more than two signals to be processed, the onboard ADC's , will be connected to the analog blocks via multiplexers. So, the signal acquired by each of the antennas will be transferred to the ADC's when they are switched to them by the multiplexer.

The signal processing board consists of an Altera EP1S80 Stratix FPGA, memory modules and some interfaces which are given in detail in Appendix B. The inner diagram of the FPGA part denotes that the signals received by the I/O ports of the FPGA are first combined and converted into a complex form. These data are then applied to the QRD CRLS algorithm denoted by the systolic array block in the diagram. The outputs of the systolic array are the weights of the adaptive array and are applied to the input signals and combined for the output data.

The adaptive weights are also applied to the data which are received from the communication system and transferred to the transmitter part. At the transmitter part the weighted outputs from the I/O ports of the FPGA which are in inphase and quadriphase forms are converted into analog signals by the onboard DAC's. The I/Q modulators upconverts the signals to the operating frequency band and applies the necessary weights to the antennas for proper beamforming.

The implementation of the proposed system will be given in detail in two main sections;

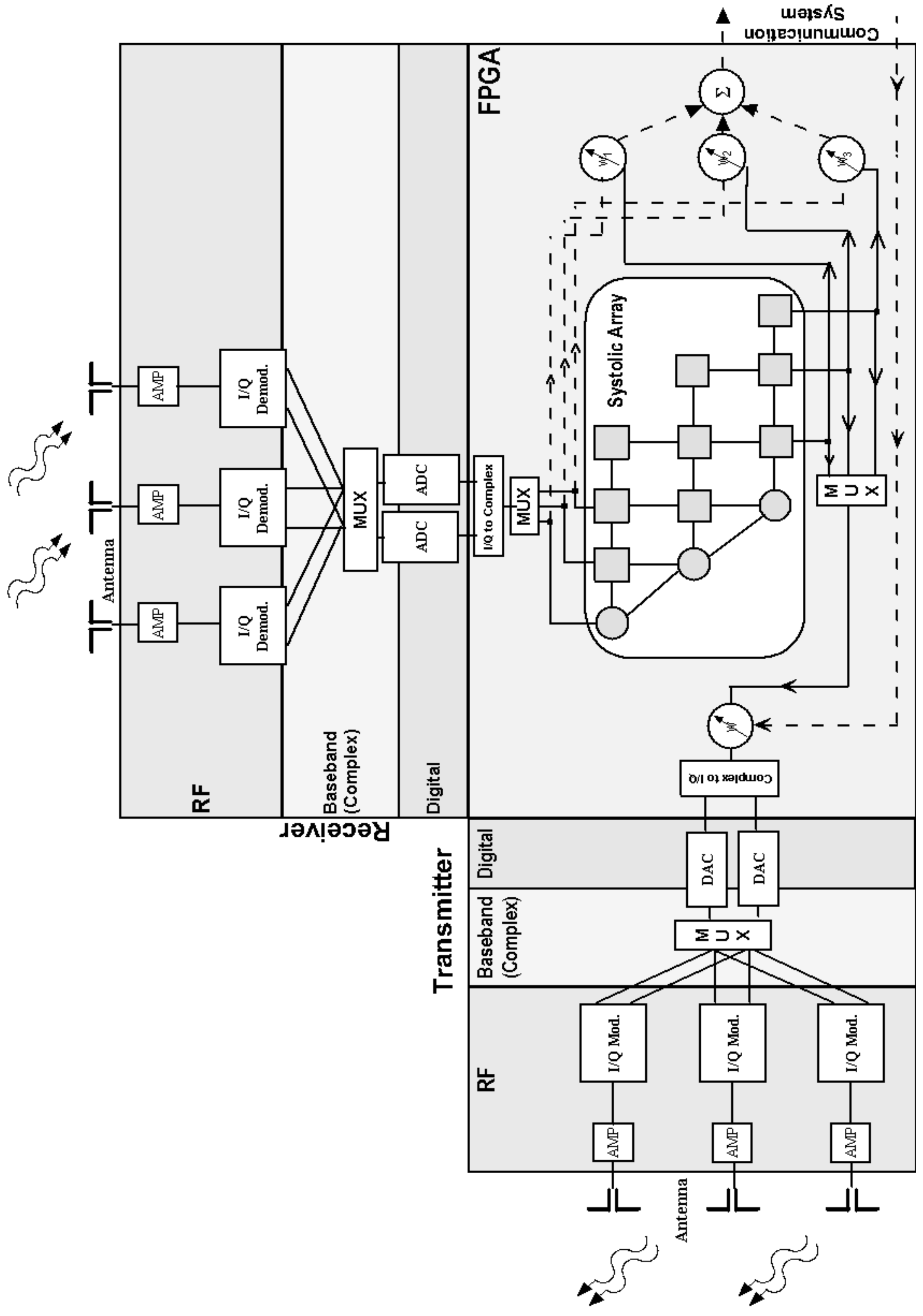


Figure 5.1 Block diagram of the implementation

hardware implementation and software implementation. The hardware implementation will present the RF parts such as the antennas, amplifiers and I/Q modulators and demodulators as well as multiplexers and ADC and DAC components. While the software implementation will focus on the development of the QRD CRLS algorithm on the SystemC environment and building the system with the developed parts in the Matlab Simulink environment..

5.1 Hardware Implementation

Hardware implementation of the system is the design and development process of the hardware parts which include antennas, RF blocks and multiplexers. Hardware parts can be analyzed in two pieces, the transmitter part and the receiver part.

The receiver and the transmitter parts use a direct conversion technique with an I/Q modulator/demodulator instead of a superheterodyne technique as explained in the following subsection.

5.1.1 The Direct Conversion Technique

The RF signals received by the antennas and amplified at the low noise amplifiers must be downconverted to a proper frequency for the signal processing units. The conventional way is to downconvert the signals to the intermediate frequency (IF) band and then digitize them by using the ADC’s. The signal processing unit compares and detects the relative phase and the amplitude of the digitized signals, with respect to a reference signal. The detection of the relative phase and amplitude are accomplished by the running software inside the signal processor, so the performance of the ADC and the software plays a critical role in the performance of the technique.

The second option, which is called the direct conversion technique is to directly convert the received RF signals into phase and amplitude by using phase and power detectors and then digitize these information by the ADC’s (Maskell & Woods, 2003). The signal processing unit does not deal with detecting the relative amplitude and phase of the signals. In the implementation of the thesis the direct conversion receiver technique is preferred to simplify the signal processing part and keep the resources of the FPGA for the other signal processing operations.

Main aim of the direct conversion receiver is to extract the phase and the magnitude information out of the received signal. First of all we need a phase detector circuitry to extract the phase from the RF signal. A basic phase detector consists of a mixer, local oscillator and a low pass filter (LPF). Illustration of this phase detector can be seen in Figure 5.2. The local oscillator ($\cos(\omega t)$) is at exactly the same frequency with the RF input signal ($\cos(\omega t + \varphi)$). The output of the mixer is given by;

$$\cos(\omega t + \varphi)\cos(\omega t) = \frac{1}{2}(\cos(\varphi) - \cos(2\omega t + \varphi)) \tag{5.1.1}$$

where ω is the operating frequency, and φ is the phase angle of the RF signal with respect to the local oscillator. If we filter the high frequency component $\cos(2\omega t + \varphi)$ by using a low pass filter, then we have only the component $\cos(\varphi)$ which contains the phase angle information in cosine form. By using an $arccos$ transformation in the signal processing part, the relative phase angle φ of the signal can be obtained (Maskell & Woods, 2003).

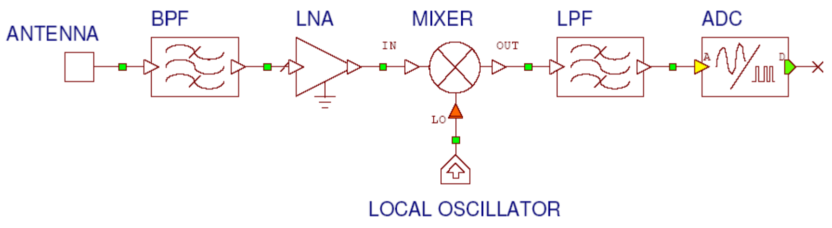


Figure 5.2 Block diagram of a phase petector

A major and well known problem in this phase determination method is that, the cosine function gives the same values for negative values of the angles between 0 and π. This means we can not determine whether the phase angle of the signal is between angles 0 and π or between angles π and 2π.

We can overcome this problem by adding a second phase detector circuitry with a 90^o phase shift at the local oscillator input as can be seen in Figure 5.3. The output of the second phase shifter is given by;

$$\cos(\omega t + \varphi)\cos(\omega t + \frac{\pi}{2}) = \cos(\omega t + \varphi)\sin(\omega t) = \frac{1}{2}(\sin(\varphi) + \sin(2\omega t + \varphi)) \tag{5.1.2}$$

and after the low pass filter we can evaluate $\sin(\varphi)$ at the output. This time we can determine if the signal is between angles 0 and π or between angles π and 2π and this lets us know the exact phase angle we evaluated from the first phase detector (Maskell & Woods, 2003).

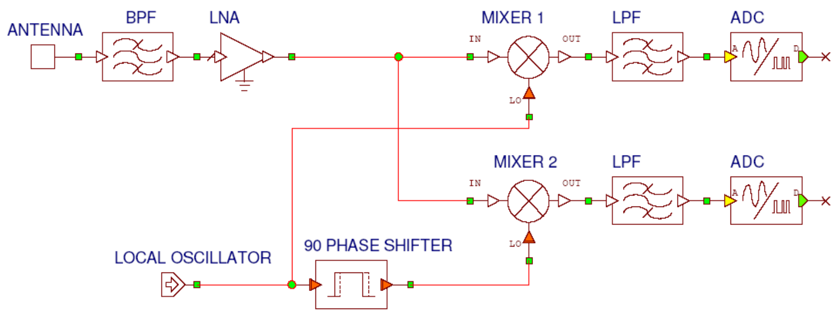


Figure 5.3 Block Diagram of the Direct Conversion Technique

If the input signal's amplitude is A and the local oscillator's amplitude is B, equations 5.1.1 and 5.1.2 become;

$$A\cos(\omega t+\varphi)B\cos(\omega t)=\frac{AB}{2}(\cos(\varphi)-\cos(2\omega t+\varphi)) \tag{5.1.3}$$

$$A\cos(\omega t+\varphi)B\sin(\omega t)=\frac{AB}{2}(\sin(\varphi)+\sin(2\omega t+\varphi)) \tag{5.1.4}$$

As we can observe from the equations, even if the local oscillator has a constant amplitude, outputs of the phase detector change with the input signal's amplitude. This problem can be overcome by calibrating the direct conversion receiver at various phase angles for a signal source.

5.1.2 The Receiver Block

The detailed block diagram of the receiver part is given in Figure 5.4. The received signals from the dipoles are converted to microstrip transmission line and fed to the quadrature demodulator integrated circuit which also includes necessary amplifiers.

Signals departing from the user arrive at the receiving antennas at different phases and amplitudes. The phase and amplitude difference between the antennas can be represented by complex signals. The quadrature demodulator converts the complex signals into inphase(I) and quadriphase(Q) signals by mixing them with the local oscillator signal. So that the complex signals can be reconstructed inside the signal processing unit. This phenomenon is also given in more detail in section 5.1.1.

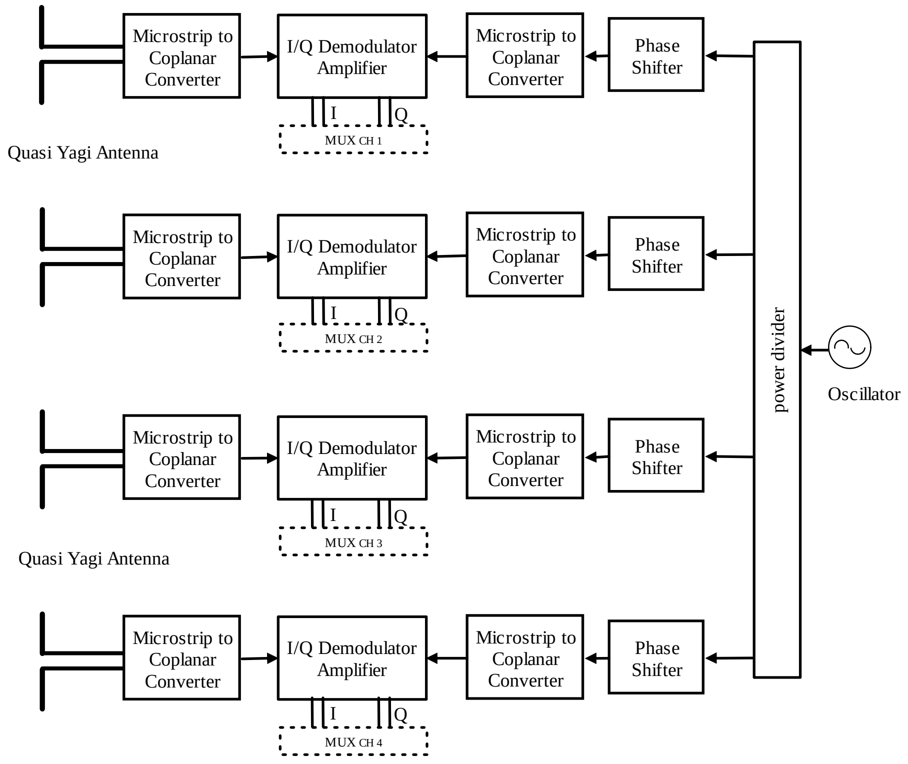


Figure 5.4 Receiver Block Diagram

The RF signal from the local oscillator is also fed to the quadrature demodulator by first dividing it into four branches and then equalizing the phases. The implemented receiver part can be seen in Figure 5.5. Major parts of the receiver; the quasi yagi antenna and the I/Q demodulator are explained in detail in the following subsections.

5.1.2.1 The Quasi Yagi Antenna

The antenna to be used as a part of the system must be printed and have a microstrip feed line so that the antenna can easily be fabricated with the RF blocks on a single substrate.

The Quasi-Yagi antenna which can be seen in Figure 5.6, is mainly a dipole antenna as the driver, with a reflector plate placed quarter wavelength apart, that increases the directivity of the antenna. Generally a director that increases the directivity and bandwidth of the antenna is placed in front of the dipole but in our work we did not use the director part since increasing the directivity, means a narrower beamwidth which is not preferred in most of the mobile communication systems (Deal et al., 2000).

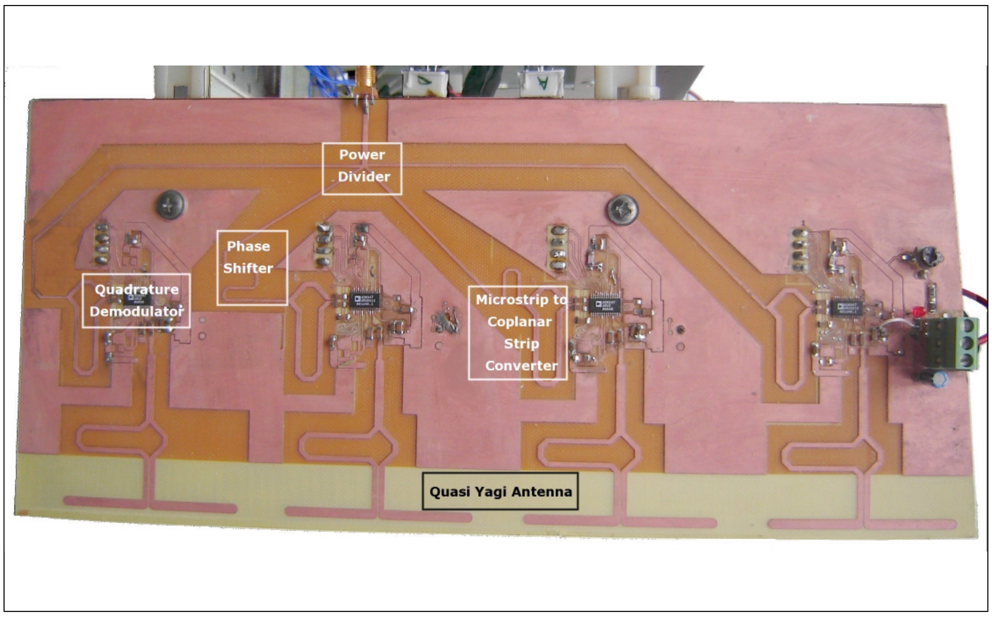


Figure 5.5 The receiver board and components

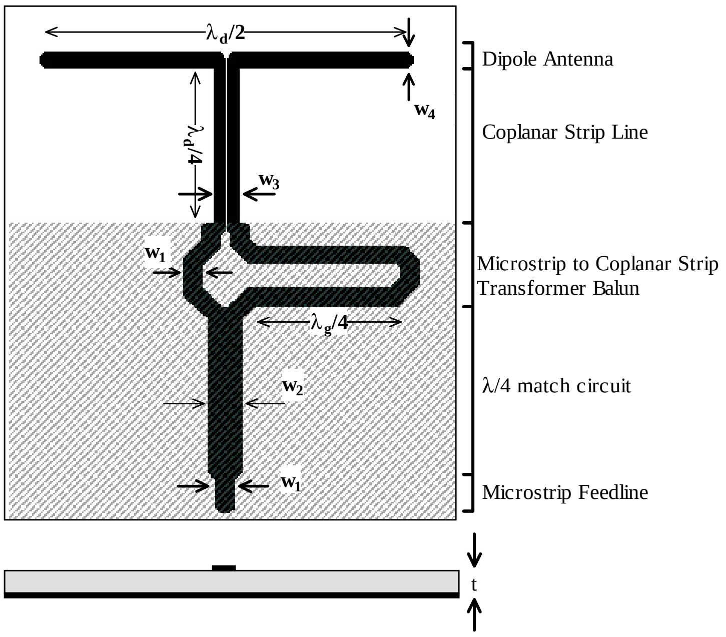


Figure 5.6 The Quasi Yagi antenna and the feed line

The driver of the Quasi Yagi antenna is fed with a coplanar strip line as can be seen in Figure 5.6. Since the transmission lines of our RF circuitry are microstrip, we have to convert the coplanar strip line to a microstrip line. This conversion is accomplished by the transformer balun which also balances the transmission line. The impedance of the transformer balun at the microstrip line is matched to the 50 Ω microstrip line by using a $\lambda_g/4$ match circuitry where λ_g is the effective wavelength inside the microstrip line.

The impedance of the microstrip line, used for feeding the coplanar strip line is 50 Ω. The junction point of the coplanar strip line makes the strip lines of the coplanar strip line act like parallel impedances; thus the impedance of the coplanar strip lines has to be adjusted for 100 Ω to avoid reflections.

Most critical point of the antenna design is transferring the power from the microstrip line to the driver antenna with minimum loss. This aim can be reached by adjusting the width of the coplanar strip and the line width of the phase shifter part. The characteristic impedance of the coplanar strip line depends on the dielectric constant of the media and the gap between the strip lines. Increasing the dielectric constant of the media decreases the impedance of the transmission line while increasing the gap increases the characteristic impedance of the transmission line (Pintzos, 1991). As mentioned before we designed the coplanar strip line with a characteristic impedance value of 100 Ω. The insertion loss and return loss graphs of the CPS to MS transformer balun are given in Figure 5.7 and Figure 5.8. As can be seen in Figure 5.7 insertion loss of the transformer at the operation band and mostly below 2 dB which means the power is transmitted to the antenna good enough. Also the return loss measurement shown in Figure 5.8 validate these results since at the operating band return losses below 10 dB are evaluated both from the CPS and the MS sides which means a power transmission ratio over 90%.

The microstrip to coplanar strip converter is then connected to the quasi yagi antenna and performance of the antenna is measured. Figure 5.9 shows the return loss measurement of the antenna. The antenna impedance match is achieved at 1900 and 2100 MHz covering the UMTS operation bands.

In Figure 5.10 transmission measurement of the quasi yagi antenna is presented. The measurement procedure is transmitting the signal and receiving it with identical antennas as

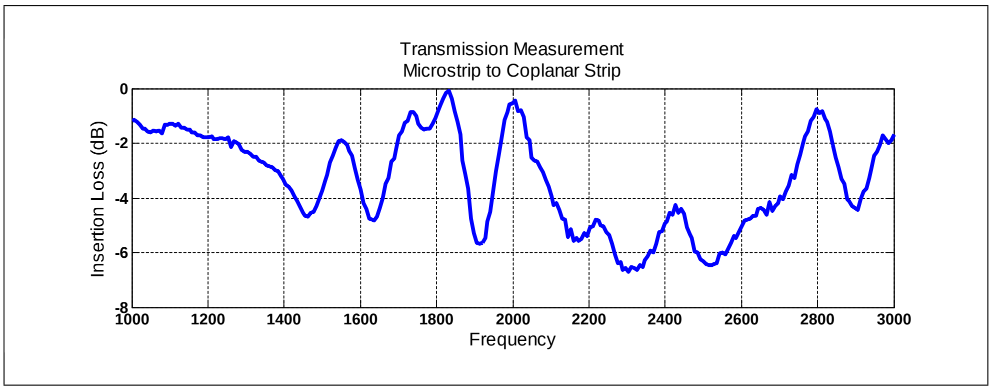


Figure 5.7 Insertion Loss graph of the CPS to MS transformer balun

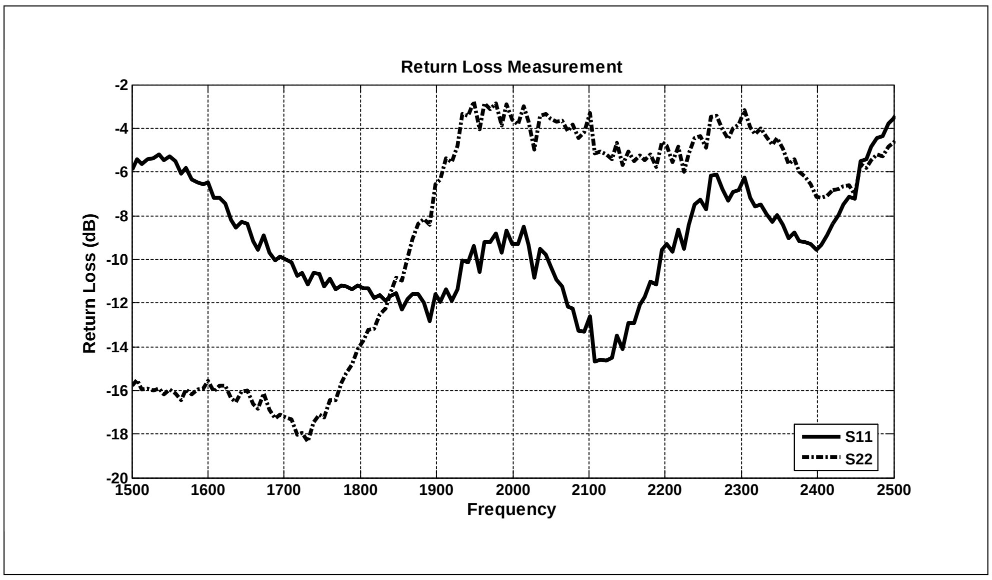


Figure 5.8 Return Loss Measurement of Microstrip to Coplanar Strip Transformer

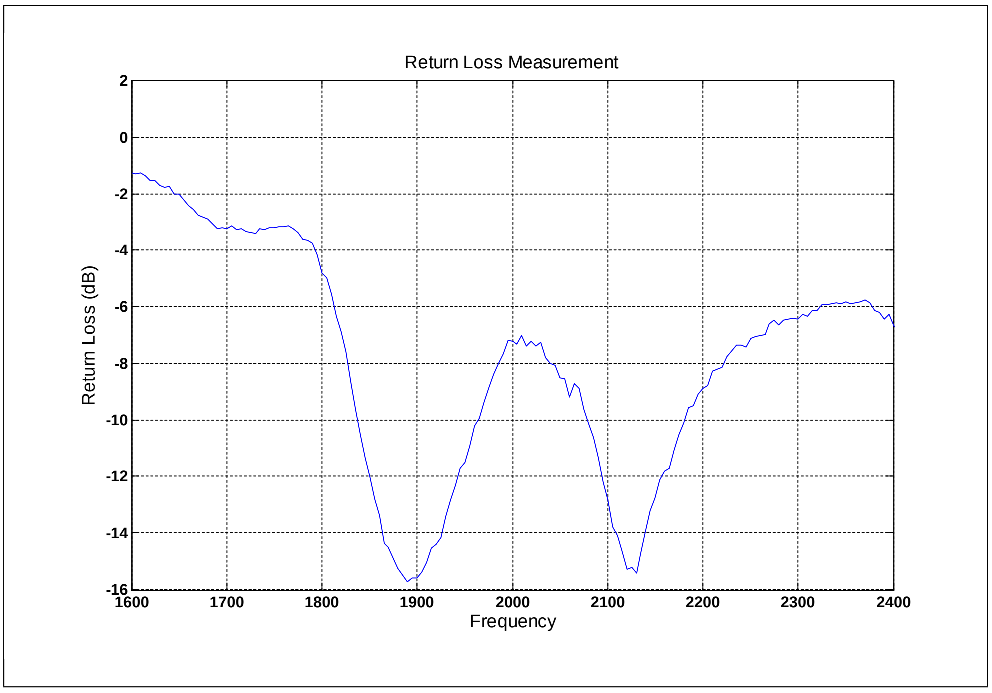


Figure 5.9 Return Loss Graph of the Quasi Yagi Antenna

described in The Standard Test Procedures for Antennas by IEEE (1979). The measurement results show that the antenna transmitted power between 1810 MHz and 1925 MHz and as a second band 2270 MHz and 2415 MHz. The antenna measurements were taken 8 cm. apart and with a free space loss of 17.7 dB at 2300 MHz 15.6 dB at 1800 MHz. Even if the measured received power level is at -11.4 dBm with a 0 dBm transmitted power we also must include the cable and connector losses which are also measured as 2.5 dB. So that the quasi yagi antenna implemented has a 4.4 dB gain at 2300 MHz and 3.35 dB gain at 1800 MHz which also consistent with the theoretical values. When we compare the results with the return loss graph we can observe that the transmission frequency is shifted from 2150 MHz to 2300 MHz. This shift is due to the fact that the second antenna in the measurement procedure has a shifted operation band.

5.1.2.2 The I/Q Demodulator

A detailed block diagram of the I/Q demodulator is given in Figure 5.3. The I/Q demodulator is implemented using the AD8347 integrated circuit which includes a 0.8 GHz 2.7 GHz direct conversion IQ demodulator. As can be seen from the data sheet given in Appendix A the AD8347 is composed of a 90^o phase shifter connected to the local oscillator input, a mixer and

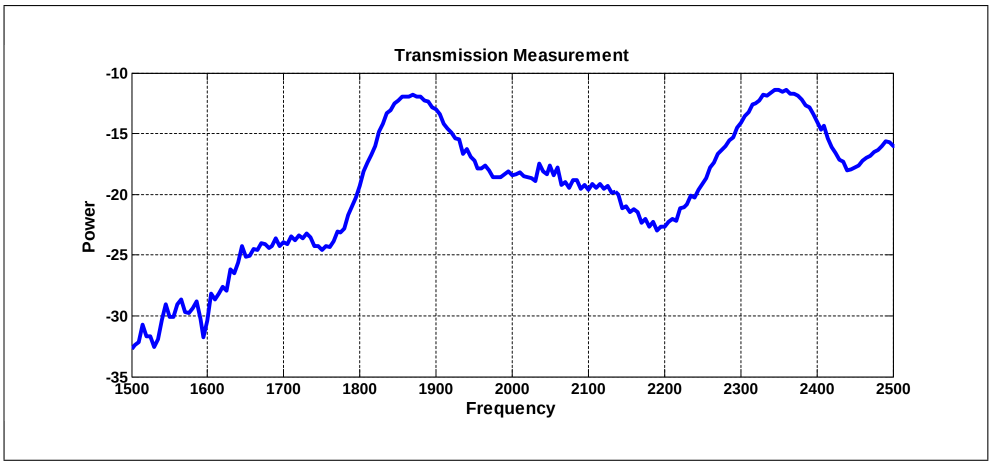


Figure 5.10 Transmission measurement of the quasi yagi antenna

an automatic gain control amplifiers at the RF input which makes it proper for our application.

The local oscillator input of the AD8347 needs a coplanar feed line for proper operation. The microstrip to coplanar strip transformer mentioned in section 5.1.2.1 is used to convert the microstrip local oscillator feed to coplanar strip feed line.

The implemented circuit was tested with the I/Q modulator circuitry. First the I/Q modulator was fed with sinusoidal waveforms from both of the channels and the RF signal source at the local oscillator. The outputs of the modulator are then connected to the corresponding input ports of the demodulator. The I and Q outputs of the demodulator are connected to an oscilloscope and compared with the input signal. Results of this measurement can be seen in Figure 5.11.

5.1.3 The Transmitter Block

The detailed block diagram of the transmitter part can be seen in Figure 5.12. The structure of the transmitter block is very similar to the receiver block. It includes a power divider module, which splits the RF signal received from the oscillator to the I/Q demodulators at equal phase. The phase of the RF signals are kept equal by using phase shifting transmission lines after the power divider circuitry.

Quadrature modulators used in the implementation process, have coplanar strip line at the local oscillator input port, so that we must convert the microstrip transmission line to coplanar

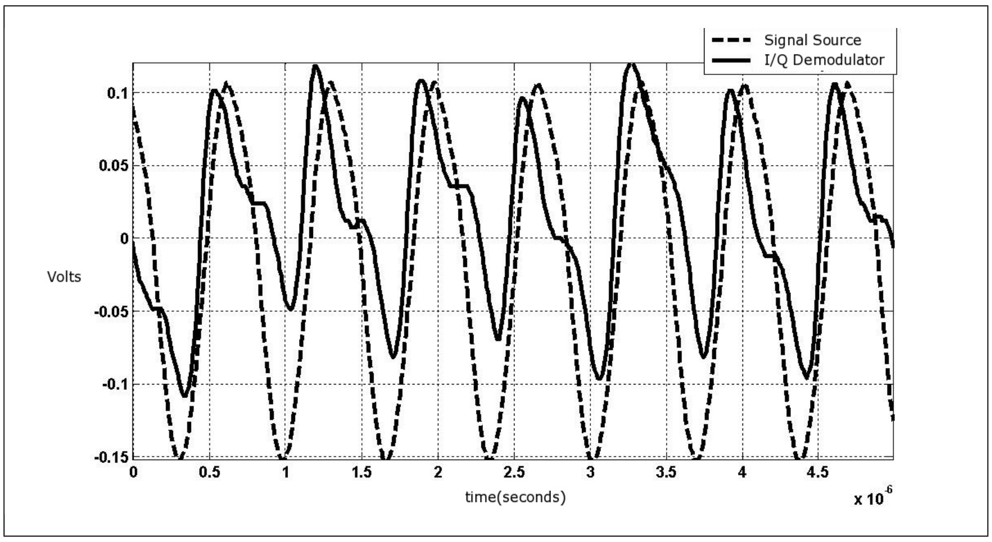


Figure 5.11 Oscilloscope plot of the Q channel input od the quadrature modulator and Q channel output of the Quadrature demodulator

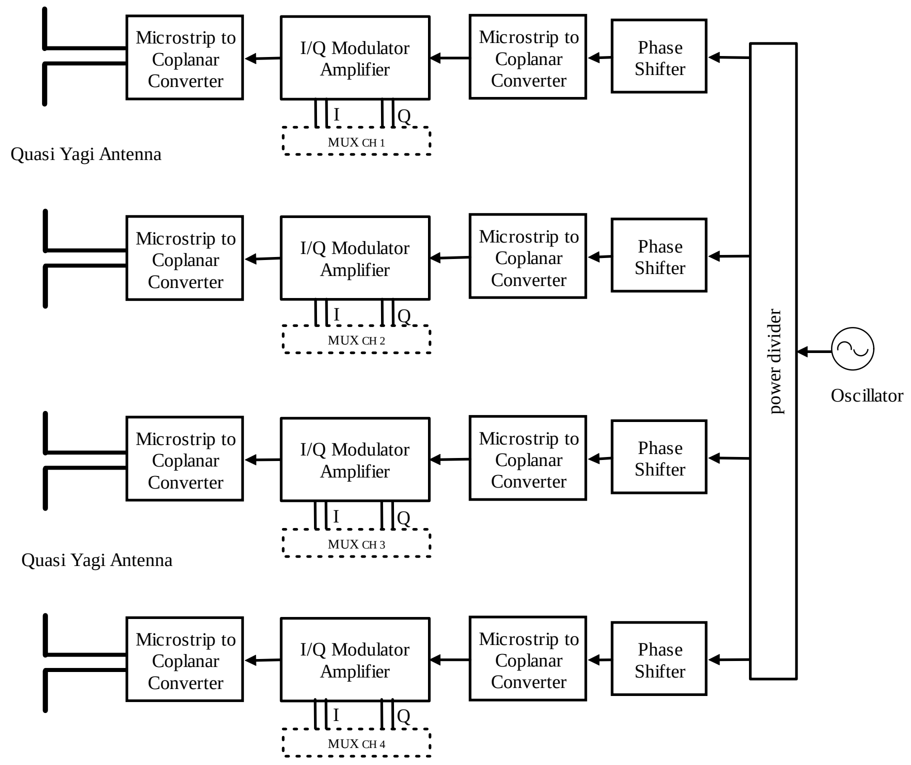


Figure 5.12 Transmitter block diagram

transmission line. A similar problem has been overcome in designing a Quasi Yagi antenna as denoted in section 5.1.2.1. We can use the same converting structure for feeding the quadrature modulator with the RF signal.

The I/Q modulator integrated circuit also includes, automatic gain controlled amplifiers, which meets our need for an external amplifier. I and Q inputs of the modulator are used to adjust the output weights of the antennas and fed from the multiplexer.

Output of the quadrature modulator is connected to the Quasi Yagi antenna via a microstrip to coplanar strip converter explained in detail in section 5.1.2.1. An image of the designed and implemented transmitter board can be seen in Figure 5.13. The Quasi Yagi antenna is one of the major parts of the transmitter block and was explained in detail in subsection 5.1.2.1. The I/Q modulator part is explained in detail in the following subsection.

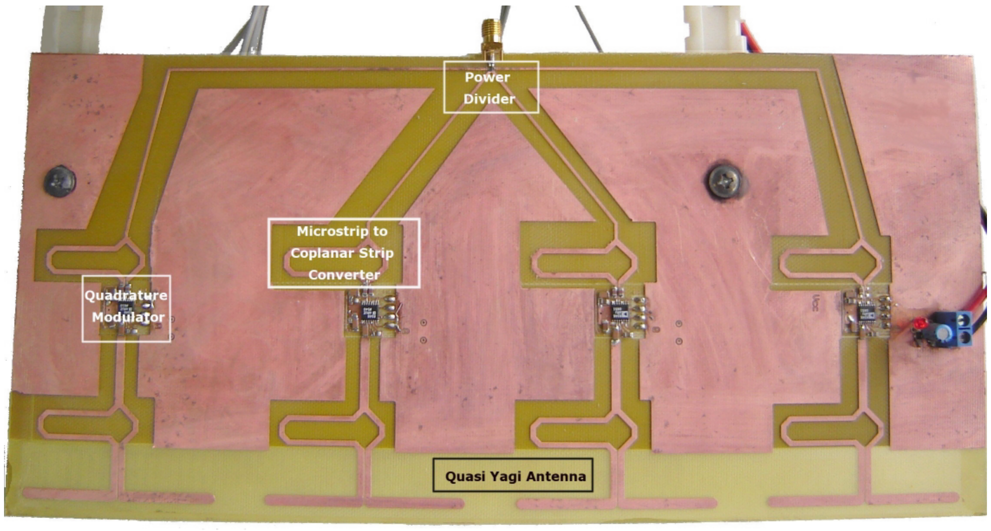


Figure 5.13 The transmitter board and components

5.1.3.1 The I/Q modulator

The I/Q modulator circuitry has the function to apply the necessary phase and amplitude changes to each RF signal to feed the antennas. The outputs of the FPGA are in the forms of I and Q signals and given by;

$$\begin{aligned} I &= A\cos(\omega t+\varphi)B\cos(\omega t) = \frac{AB}{2}(\cos(\varphi)-\cos(2\omega t+\varphi)) && (5.1.5)\\ Q &= A\cos(\omega t+\varphi)B\sin(\omega t) = \frac{AB}{2}(\sin(\varphi)+\sin(2\omega t+\varphi)) && (5.1.6)\end{aligned}$$

where A represents the amplitude of the signal and B represents the amplitude of the local oscillator and φ represents the phase angle with respect to the reference signal.

The block diagram of the I/Q modulator can be seen in Figure 5.14. When the I port is fed with $\cos(\varphi)$ and the Q port is fed with $\sin(\varphi)$ the RF output becomes;

$$I\cos(\omega t) - Q\sin(\omega t) = \cos(\varphi)\cos(\omega t) - \sin(\varphi)\sin(\omega t) = \cos(wt+\varphi) \qquad (5.1.7)$$

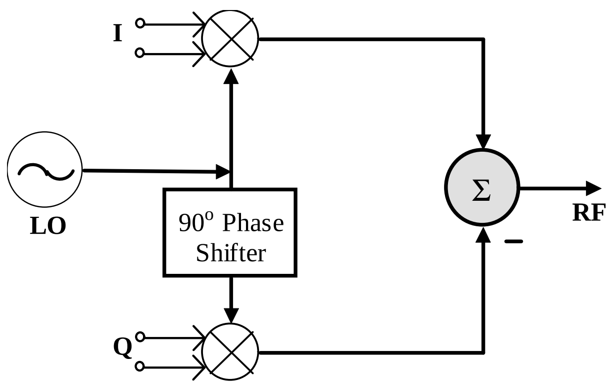


Figure 5.14 I/Q modulator

We can see that the phase shift of the output is adjusted at the RF output port of the modulator. Amplitude of the output can also be adjusted by amplifying (or attenuating) the I and Q ports. Also the amplitude of the LO is effective at the output and must be taken care at the output of the FPGA.

The I/Q modulator was implemented with the AD8346 integrated circuit from Analog Devices which is the conjugate of the AD8347 I/Q demodulator. The datasheet of the AD8346 can be found at Appendix A. As it can be observed from the datasheet this integrated circuit also has coplanar feed lines at the input ports so that the coplanar strip to microstrip converter circuitry is also needed for the local oscillator feed line of the device.

5.2 Software Implementation

Software implementation is defined as developing and uploading the Folded Systolic Array Based QRD CRLS Beamforming algorithm into the FPGA.

The implementation procedure is as follows briefly. The cells and the tiles of the systolic array and surrounding components such as internal multiplexers are first developed using the SystemC library as the first step. After testing and simulating these components they are transferred to the MATLAB Simulink environment by using the Altera DSP Builder software and assembled for the complete system. The developed code by using the MATLAB Simulink environment is then synthesized and uploaded to the FPGA core.

A brief flow chart of software development and implementation is given in Figure 5.15. Each step in the flowchart is explained in detail in the following sections.

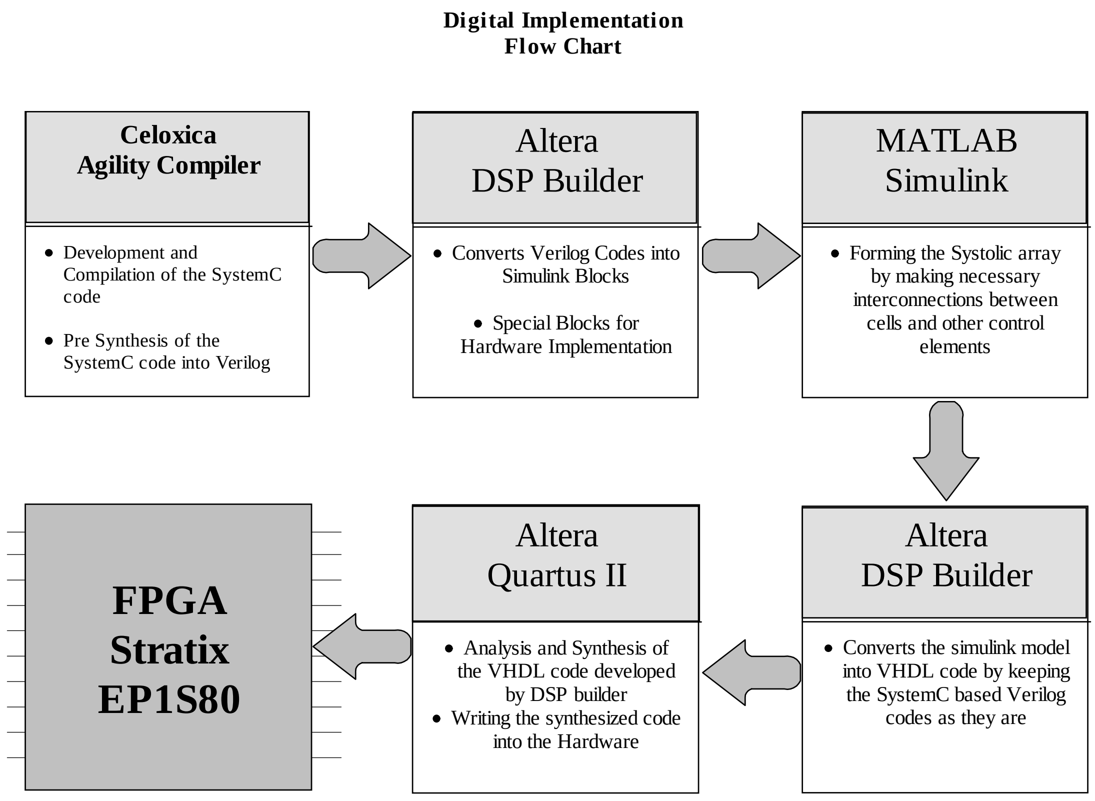

Digital Implementation
Flow Chart
Celoxica
Agility Compiler
• Development and Compilation of the SystemC code
• Pre Synthesis of the SystemC code into Verilog
Altera
DSP Builder
• Converts Verilog Codes into Simulink Blocks
• Special Blocks for Hardware Implementation
MATLAB
Simulink
• Forming the Systolic array by making necessary interconnections between cells and other control elements
Altera
DSP Builder
• Converts the simulink model into VHDL code by keeping the SystemC based Verilog codes as they are
Altera
Quartus II
• Analysis and Synthesis of the VHDL code developed by DSP builder
• Writing the synthesized code into the Hardware
FPGA
Stratix
EP1S80

Figure 5.15 Software development and implementation flowchart

5.2.1 The SystemC Library

Several languages have emerged to address the various aspects of system design. "C/C++" is the language which is predominately used for embedded system software. The hardware description languages (HDLs), "VHDL" and "Verilog", are used for simulating and synthesizing digital circuits. "Vera" and "e" are the languages for functional verification of complex application-specific integrated circuits (ASICs). "SystemVerilog" is a new language that evolves the Verilog language to address many hardware-oriented system design issues. "Matlab" and several other tools and languages such as "SPW" and "System Studio" are widely used for capturing system requirements and developing signal processing algorithms. Figure 5.16 shows the comparison of "SystemC" with other languages. One of the most important advantages of the SystemC based hardware design is that it is the only environment which gives the opportunity of both an architectural design language and a HDL (Black & Donovan, 2004).

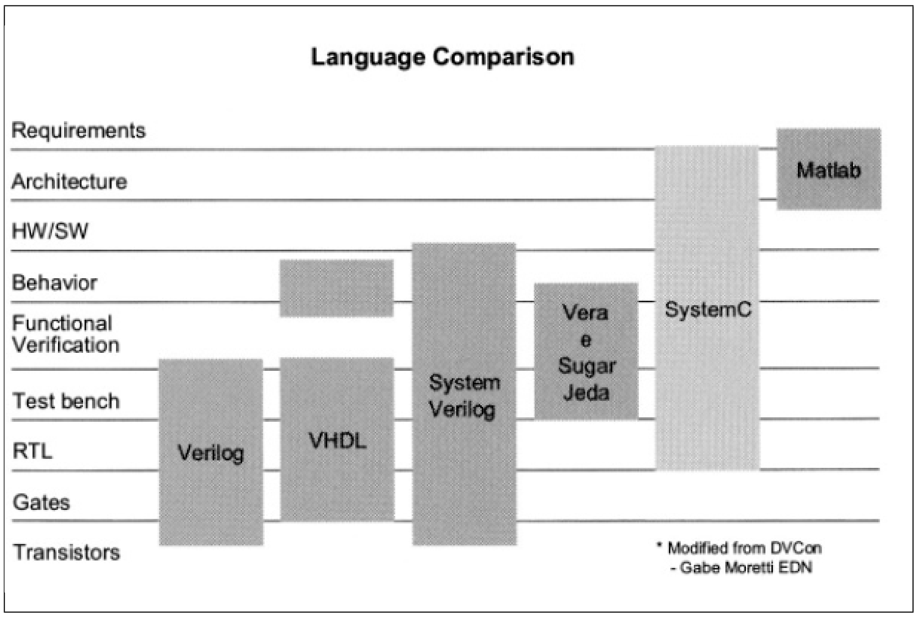


Figure 5.16 SystemC vs other design languages

Today's integrated circuits (ICs) often exceed 10 million gates, which conservatively translates to one hundred thousand lines of RTL code. Today's designs are practical because of the methodologies that apply RTL synthesis for automated generation of gates by using behavioral design languages. But as the systems get more complex behavioral design languages also become unpractical which brings out the need for architectural design languages. Integrated development of hardware and software is also a major advantage of the SystemC since it is a C++ library and enables communication with other C++ designs (Black & Donovan, 2004).

SystemC is preferred in our software implementation for expediting the software design and implementation process.

Recently, IEEE published the standard titled "IEEE 1666^{TM} -2005 Standard SystemC Language Reference Manual (LRM)" (SystemC Standard, 2006) and accepted SystemC as a standard library for system development environment. Version of the SystemC library published by the IEEE as a standard is 2.1 which is the version used in this thesis also.

Recently, some software have been developed for behavioral synthesis of the SystemC codes by both eda vendors and research groups but these are limited synthesis tools which demand specially developed codes for synthesis. After behavioral synthesis RTL synthesis tools and logic synthesis tools are widely used without any limitations and provided mainly by chip vendors (Black & Donovan, 2004).

5.2.2 Developing Systolic Arrays Using the SystemC library

The main structure of the beamformer hardware is the systolic array on which the QRD CRLS algorithm runs and evaluates the weights as the outputs. The conventional systolic array structure consists of components called cells, which are explained in detail in chapter three. The folded systolic structure, proposed in this thesis, includes tiles containing cells, which are mentioned in Chapter Four. SystemC is used in developing and simulating these components.

5.2.2.1 Development of the Conventional Systolic Array

The conventional systolic array structure contains five type of cells; C-1, C-2, C-3, C-4, C-5, as mentioned in chapter three. As an example to the developed SystemC code for modeling and synthesis, the C-2 cell will be explained in detail.

The C-2 cell and internal functions are presented in Figure 5.17. It has a data input shown as x_{in} , inputs and outputs of Givens rotation parameters as c and s and data output as x$_{out}$. The cell also has an internal parameter r which is updated at each clock cycle, and shown as a loop from an output to an input as presented in Figure 5.17.

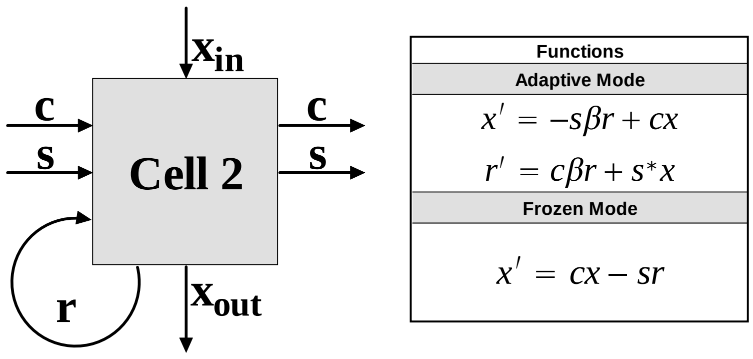


Figure 5.17 C-2 cell and internal functions

All of the parameters used in the functions of C-2 cells are complex valued except the forgetting factor β which is preferred as 0.5 in this work. Since the parameters are complex, we need to use complex operations for the internal functions inside the cell. In order to expedite the development process and ease the traceability of the code, a complex operations library is developed. Complex operations defined in this library are given in table 5.1. .

Table 5.1 Complex Operations

Operation	Complex Representation
$x = (a, b), y = (c, d),\ x, y \in \mathbb{C}$	
$x + y$	$(a + b, c + d)$
$x - y$	$(a - b, c - d)$
xy	$(ac - bd, ad + bc)$
$\frac{x}{y}$	$\left(\frac{ac+bd}{c^2+d^2}, \frac{bc-ad}{c^2+d^2}\right)$
$\sqrt{x}$	$\frac{\sqrt{2}}{2}(\sqrt{\sqrt{a^2+b^2}+a}, sgn(b)\sqrt{\sqrt{a^2+b^2}-a})$
$\lvert x\rvert$	$\sqrt{a^2+b^2}$

A comparison of the code developed with and without the complex operations library is given in Figure 5.18. It can be clearly observed that the operations using the complex library are more apparent than the standard operations which enables rapid and efficient implementation of the signal processing algorithm.

Complex division and complex square root operations include more division and square root operations than for the real numbers. These operations require too many logic elements

SystemC Code Using Standard Libraries

```
sc_out<sc_int<8> > c_re, c_im, s_re, s_im;                                  // output ports
sc_int<8>        betareal, betaimag, rreal, rimag, dreal, dimag             //parameters

/*calculate of output values c and s and write to the ports */
c_re.write(((betareal*rreal-betaimag*rimag)*dreal+(betareal*rimag+betaimag*rreal)*dimag)/(dreal*dreal+dimag*dimag));
c_im.write(((betareal*rimag+betaimag*rreal)*dreal-(betareal*rreal-betaimag*rimag)*dimag)/(dreal*dreal+dimag*dimag));
s_re.write((binreal*dreal+binimag*dimag)/(dreal*dreal+dimag*dimag));
s_im.write((binimag*dreal-binreal*dimag)/(dreal*dreal+dimag*dimag));
```

SystemC Code Using the Complex Library

```
sc_out<sc_int<8> > c_re, c_im, s_re, s_im;              // output ports
complex<sc_int<8> > c, s, rout, x, beta, rin;           // parameters

/*calculation of output values*/
rout= sqrt(beta*beta+abs(x)*abs(x));
c=beta*rout/rin;
s=x/rin;

/*write the calculated parameters to the ports*/
c_re.write( c.get_real() );
c_im.write( c.get_imag() );
s_re.write( s.get_real() );
s_im.write( s.get_imag() );
```

Figure 5.18 Developed SystemC codes with and without the complex libraries

and may reduce resources of the FPGA chip dramatically, when used. To prevent the lack of resources with these operations, division and square root operations contained in the library are defined using the CORDIC algorithm.

5.2.2.2 The CORDIC algorithm

The CORDIC acronym stands for **CO**ordinate **R**otation **DI**gital **C**omputer. It is a class of shift-add algorithms for rotating vectors in a plane.

The CORDIC algorithm provides an iterative method of performing vector rotations by arbitrary angles using only shift and adds. The rotation transform given by;

$$\begin{aligned} x' &= x\cos\phi - y\sin\phi \qquad (5.2.1) \\ y' &= y\cos\phi + x\sin\phi \end{aligned}$$

rotates a vector in a Cartesian plane by the angle ϕ. We can reexpress this transform as;

$$\begin{aligned} x' &= \cos\phi(x - y\tan\phi) \qquad (5.2.2) \\ y' &= \cos\phi(y + x\tan\phi) \end{aligned}$$

If the rotation angles are restricted so that $\tan\phi = \pm 2^{-i}$ the multiplication by tangent term reduces to a simple shift operation. Since at each iteration the direction of rotation is determined by the sign of ϕ , the $\cos\phi$ term becomes constant as its value does not change with the direction. The angular shift is kept constant at each iteration so value of the cosine term does not change. As a result, we have the following equations for each iteration;

$$\begin{aligned} x_{i+1} &= K_i[x_i - y_i d_i 2^{-i}] \\ y_{i+1} &= K_i[y_i + x_i d_i 2^{-i}] \end{aligned} \tag{5.2.3}$$

where:

$$\begin{aligned} K_i &= \cos(\arctan(2^{-i})) = \frac{1}{\sqrt{1+2^{-2i}}} \\ d_i &= \pm 1 \end{aligned} \tag{5.2.4}$$

The scaling constant K_i can be applied elsewhere in the system without loss of generality. As number of iterations increase K_i approaches to 0,6073. So we have a gain for this algorithm given by;

$$A_n = \prod_n \sqrt{1+2^{-2i}} \tag{5.2.5}$$

where n is the number of iterations. A_n approaches to 1,647 as n increases. By adding or subtracting elementary rotation angles depending on the direction of rotation, the rotation angle can be accumulated. This angular accumulator adds a third difference equation to the CORDIC algorithm;

$$z_{i+1} = z_i - d_i \arctan(2^{-i}) \tag{5.2.6}$$

The CORDIC rotator has two basic operation modes. In the rotation mode we rotate the input vector by a specified angle. In the vectoring mode we rotate the input vector to the x axis while recording the angle required to make that rotation (Andraka, 1998).

In the rotation mode CORDIC algorithm, the angle accumulator, z, is initialized with the desired rotation angle. At each iteration the residual angle in the angle accumulator is made to diminish. The CORDIC operations for the rotation mode are given by;

$$\begin{aligned} x_{i+1} &= x_i - y_i d_i 2^{-i} \\ y_{i+1} &= y_i - x_i d_i 2^{-i} \\ z_{i+1} &= z_i - d_i \tan^{-1}(2^{-i}) \end{aligned} \tag{5.2.7}$$

where

$$d_i = -1 \text{ if } z_i < 0 \text{ else } d_i = 1 \tag{5.2.8}$$

and as a result of the algorithm we evaluate;

$$\begin{aligned} x_n &= A_n[x_0 \cos z_0 - y_0 \sin z_0] \\ y_n &= A_n[y_0 \cos z_0 + x_0 \sin z_0] \\ z_n &= 0 \end{aligned} \tag{5.2.9}$$

The absolute value of the complex parameter can be evaluated by using the vectoring mode by letting the y_n equal to zero. The vectoring algorithm is based on minimizing the y component of the residual vector at each iteration. The CORDIC equations for the vectoring mode are given as;

$$\begin{aligned} x_{i+1} &= x_i - y_i d_i 2^{-i} \\ y_{i+1} &= y_i - x_i d_i 2^{-i} \\ z_{i+1} &= z_i - d_i \tan^{-1}(2^{-i}) \end{aligned} \tag{5.2.10}$$

where;

$$d_i = -1 \text{ if } y_i < 0 \text{ else } d_i = 1 \tag{5.2.11}$$

and we evaluate the following results;

$$\begin{aligned} x_n &= A_n\sqrt{x_0^2 + y_0^2} \\ y_n &= 0 \\ z_n &= z_0 + \arctan(y_0/x_0) \\ A_n &= \prod_n \sqrt{1 + 2^{-2i}} \end{aligned} \tag{5.2.12}$$

The CORDIC algorithms are limited between angles -$\pi/2$ and $\pi/2$. This limitation is due to the use of 2^o for the tangent in the first iteration. An initial rotation of either π or 0 can be made according to the sine of the y value in order to find the correct rotation angle of the vector (Andraka, 1998).

The CORDIC based complex division and complex multiplication operations are based on transforming the relevant operands into phasor forms and then accomplishing the operation. The advantage of using phasor division is it requires less operations compared with the conventional complex division as given in table 5.2. The complex multiplication operation conventionally needs 4 multiplications and 2 additions whereas the phasor operation needs only one multiplication and one addition operations. However the transformation operations are also needed before and after the multiplication in order to transform the complex numbers into phasor forms and back transform the result into complex form. These transformations also need too much resources for implementation. However the vectoring mode CORDIC algorithm can also be used for the complex notation to phasor notation transformation operation and the rotation mode CORDIC algorithm can be used for phasor to complex notation transformation.

Table 5.2 Comparison of Complex Division and Phasor Division

	Complex	**Operations**	**Phasor**	**Operations**
Multiplication $(a+bi)\times(c+di)$	$(ac-bd)+i(ad+bc)$	4 Mult. 2 Add.	$r=r_{ab}r_{cd}$ $\theta=\theta_{ab}+\theta_{cd}$	1 Mult. 1 Add. 3 Trans.
Division $\frac{(a+bi)}{(c+di)}$	$\frac{ac+bd}{c^2+d^2}+i\frac{bc-ad}{c^2+d^2}$	2 Div. 6 Mult. 4 Add.	$r=\frac{r_{ab}}{r_{cd}}$ $\theta=\theta_{ab}-\theta_{cd}$	1 Div. 1 Add. 3 Trans.

To find the square root we first assign a reference value which starts from the square of the MSB and compare the parameter with this reference value. If it is greater than the MSB, the result is set to 1 and the reference is subtracted from the parameter, if not, it is set to 0. The reference is then set to the square of the (MSB-1) bit and compared with the parameter again until the LSB is reached. The result is the square root of the parameter. Table 5.3 gives an example to the square root operation with a 8 bit reference and 8700 as the parameter value. Note that width of the reference is always the half of the parameter.

Table 5.3 Square Root by CORDIC Algorithm

Reference	**Result**	**Parameter**	**Operation**
128	0	$128\times128<8700$	do nothing
64	64	$64\times64<8700$	$result=64$
32	64	$96\times96<8700$	do nothing
16	80	$80\times80<8700$	$result=64+16=80$
8	88	$88\times88<8700$	$result=80+8=88$
4	92	$92\times92<8700$	$result=88+4=92$
2	92	$94\times94<8700$	do nothing
1	93	$93\times93<8700$	$result=93$

5.2.2.3 The Fixed Point Arithmetic

Because of the basic logic elements they are composed of Field Programmable Gate Arrays are capable of performing integer arithmetic only. However in signal processing applications acquired data from the real world is never integer. Applying the integer arithmetic operations results in the loss of data and improper operation of the systolic array. For example if the acquired data for a multiplication operation are 0.5 and 2 the result of the operation should be 1 however in integer arithmetic the first operand becomes 0 and so is the result.

The `int` format represents integers from 0 up to the largest integer that can be represented with the available number of bits. Fixed point format is used to include numbers that lie between 0 and 1; with a 'binary point' assumed to lie just after the most significant bit. A 8 bit binary word x interpreted as; $\begin{bmatrix} b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \end{bmatrix}$ equals to

$$x = \sum_{i=0}^{7}(2^i b_i) \tag{5.2.13}$$

for an unsigned integer representation which varies between values 0 to 255. For a signed representation x equals

$$x = \pm\sum_{i=0}^{6}(2^i b_i) \tag{5.2.14}$$

and can get values from -128 to 128.

If x is a 4.4 bit fixed point data then it becomes;

$$x = \sum_{i=-4}^{3}(2^i b_i) \tag{5.2.15}$$

for the unsigned case and it will have a range between 16 and 0 which increases with 0.0625 step size. For the signed case x equals;

$$x = \pm\sum_{i=-4}^{2}(2^i b_i) \tag{5.2.16}$$

and can get values between -8 and 8 with the same step size.

For the addition of two fixed point numbers the binary point must be casted at the same digit for a proper result. Same rule is also valid for subtraction operation.

The fixed point multiplication operation can be accomplished by multiplying two fixed point numbers as if they were integers and then shift the result twice the length of the fractional part. However this kind of operation uses too much resources since the result is twice the length of the operands. Another method is somehow more complicated but it is a more effective one. First of all the operands are separated in integer and fractional parts. First the integer parts are multiplied and the result is stored, afterwards the integer part of the first operand is multiplied by the fractional part of the second operand, and the integer part of the second operand is multiplied with the fractional part of the first operand, however since the multiplication operation includes a fractional part, the result is shifted as much as the length of the fractional part before adding the results to the stored multiplication value. The multiplication result of the fractional parts of the operators are shifted twice the length of the fractional part before adding the result to the previous ones. Figure 5.19 shows a 4.4 bit fixed point multiplication operation briefly. Here the result is defined to be 4.4 bit fixed point, so that the 4 most significant bits of the integer part of the result are saturated while the 4 least significant bits of the fractional part are truncated to have a 4.4 bit fixed point result.

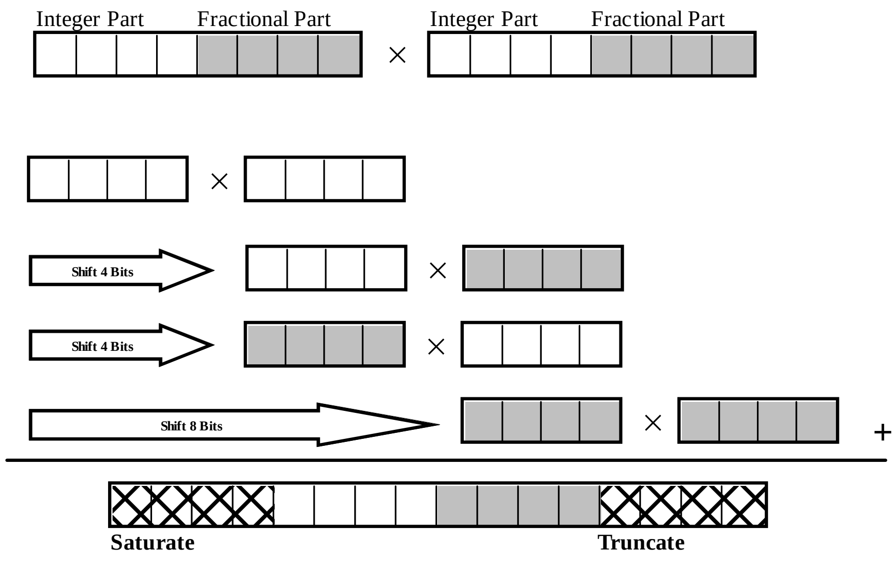


Figure 5.19 4.4 Fixed Point multiplication operation

The division operation is also accomplished in a similar manner while this time dividing an operand with a fractional part would cause a left shift of the result.

These fixed point arithmetic issues are also defined in the complex operations library, so that no additional operations inside the main code are required for converting the results to the fixed point representations.

The r parameter of the cell is updated at each clock cycle of the array, so that the update of the triangularization matrix is accomplished. Outputs of the components developed using the SystemC are updated at the end of each clock cycle and inputs which are updated at previous clock cycle are read with the triggering of the clock. So that connecting the input and output of the r parameter will provide the update of it at each triggering of the clock signal.

5.2.2.4 Development of the Folded Systolic Array

The proposed folded systolic array structure contains 4 types of tiles; T-1, T-2, T-3, T-4. The internal structure of T-2 tile is given in Figure 5.20. Each cell inside the tile must operate sequantially for a proper operation of the systolic array. The mode input shown in the Figure determines the routing of the inputs and the outputs of the tile, to the cells in order. So that each cell acquires its inputs and delivers its outputs to the ports of the tile at its turn. The multiplexers are the critical parts of the tiles in this manner, since they are responsible in routing of the data from/to the cells to/from the ports of the tiles.

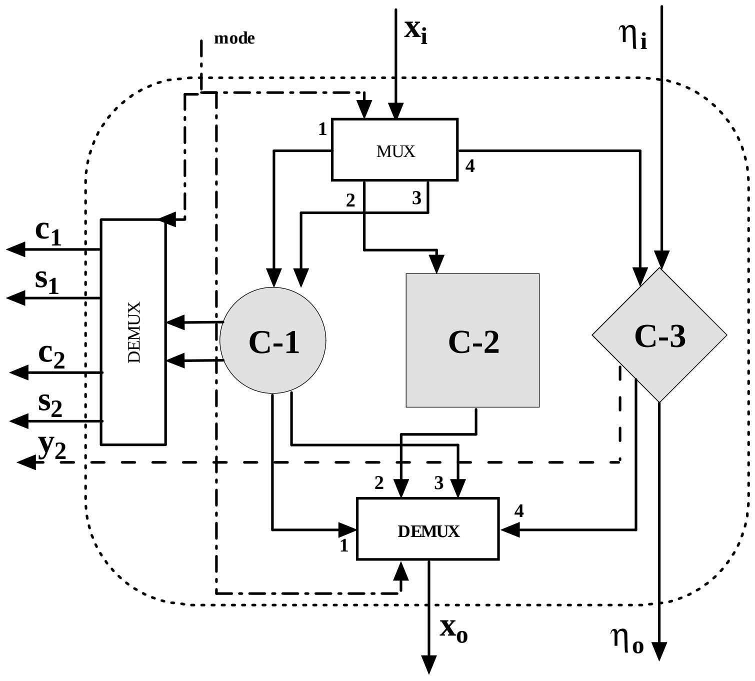
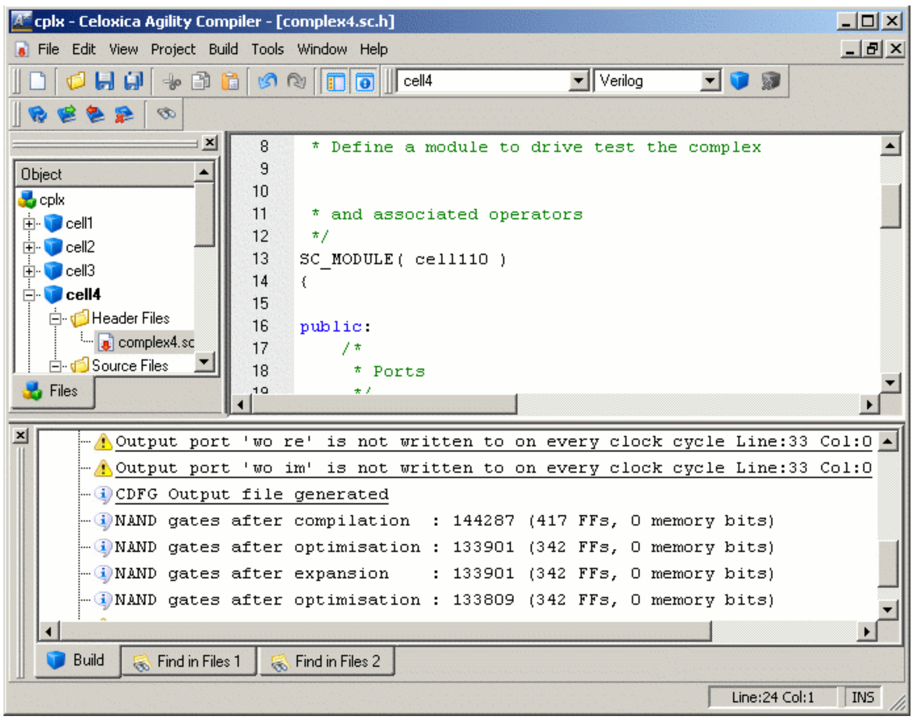


Figure 5.20 Internal Structure of the T2 Tile

5.2.3 Assembling Software Components in MATLAB Simulink

The process of converting a digital design written in a hardware description language (HDL) into a low-level implementation consisting of primitive logic gates is called the synthesis. Synthesis of the components developed using the SystemC is accomplished by using the Celoxica Agility Compiler. A screenshot of the synthesis environment is given in Figure 5.21.



```
 * Define a module to drive test the complex


 * and associated operators
 */
SC_MODULE( cell110 )
{

public:
    /*
     * Ports
     */
```

```
Output port 'wo re' is not written to on every clock cycle Line:33 Col:0
Output port 'wo im' is not written to on every clock cycle Line:33 Col:0
CDFG Output file generated
NAND gates after compilation  : 144287 (417 FFs, 0 memory bits)
NAND gates after optimisation : 133901 (342 FFs, 0 memory bits)
NAND gates after expansion    : 133901 (342 FFs, 0 memory bits)
NAND gates after optimisation : 133809 (342 FFs, 0 memory bits)
```

Figure 5.21 Celoxica Agility Compiler for SystemC synthesis

For a more flexible software development each of the cells and tiles are synthesized separately which gives us the freedom to connect as many cells of each type as needed. So that the systolic array can easily be converted to a larger or smaller one.

The synthesized cells are then transferred to the Matlab Simulink environment as distinct blocks by using the Altera DSP builder software. The interconnections between the cells and other simulink interface blocks are established to form a MVDR beamformer for 4 antennas. Figure 5.22 presents a systolic array for two antennas structure while in Figure 5.23 a systolic array for 4 antennas is presented. As can be seen here cells are displayed as simulink blocks with necessary interconnections between them.

As presented in Chapter Three the systolic array operates in two different modes namely the adaptive and the frozen modes. The cells switches between these modes at different time intervals since inputs to the array elements impinge at different time intervals. An array generator is placed for the switching operation of the cells to the appropriate mode. Time delay elements are placed between mode ports of the cells to ensure the switch the correct cell at the specified time interval. These elements are seen as small boxes in the Figure 5.22 and 5.23.

The internal structure of a cell, imported to the Matlab Simulink, can be seen in Figure 5.24. Here the large block that is placed at the center with light gray color is the imported block into simulink as a model from the SystemC code. Ports of this block are connected to the input and output ports of the simulink interface.

For concurrent operation, all time delays between the inputs and outputs of the cells must be equal. We placed delays to the ports with less time delay and made the outputs ready at the same instant. The dark gray cells indicate the necessary delays for the synchronous output of the ports.

Acquiring the signals from the ADC's and feeding the systolic array with this data, and transferring the evaluated weights to the DAC's, need multiplexing and routing mechanisms, which should operate synchronously with the external muliplexers and demultiplexers. Figures 5.25 and 5.26 present these multiplexing and routing structures. As can be seen in Figure 5.25, inputs from the ADC's are shown as SMA connectors in Simulink. These inputs are first converted to 4.4 bit fixed point data format from 12 bit ADC output and are connected to the input multiplexer. The input multiplexer block is also developed in SystemC environment and is used to acquire the constraint and data vectors from the ADC's and feed the cells with these data separately.

The constraint vector is also acquired from the antennas but its acquisition is controlled by the pushbutton that is also presented in the Figure. The constraint vector is acquired and stored when the button is pressed and it is routed to the systolic array during the operation. Internal multiplexers following the input multiplexer blocks are used to form the input data matrix of the cells.

The output part of the model consists of an output multiplexer block as can be seen in Figure 5.26. The output multiplexer block is also developed in the SystemC environment to route the weight vector to the onboard DAC's while controlling the output multiplexer to acquire the

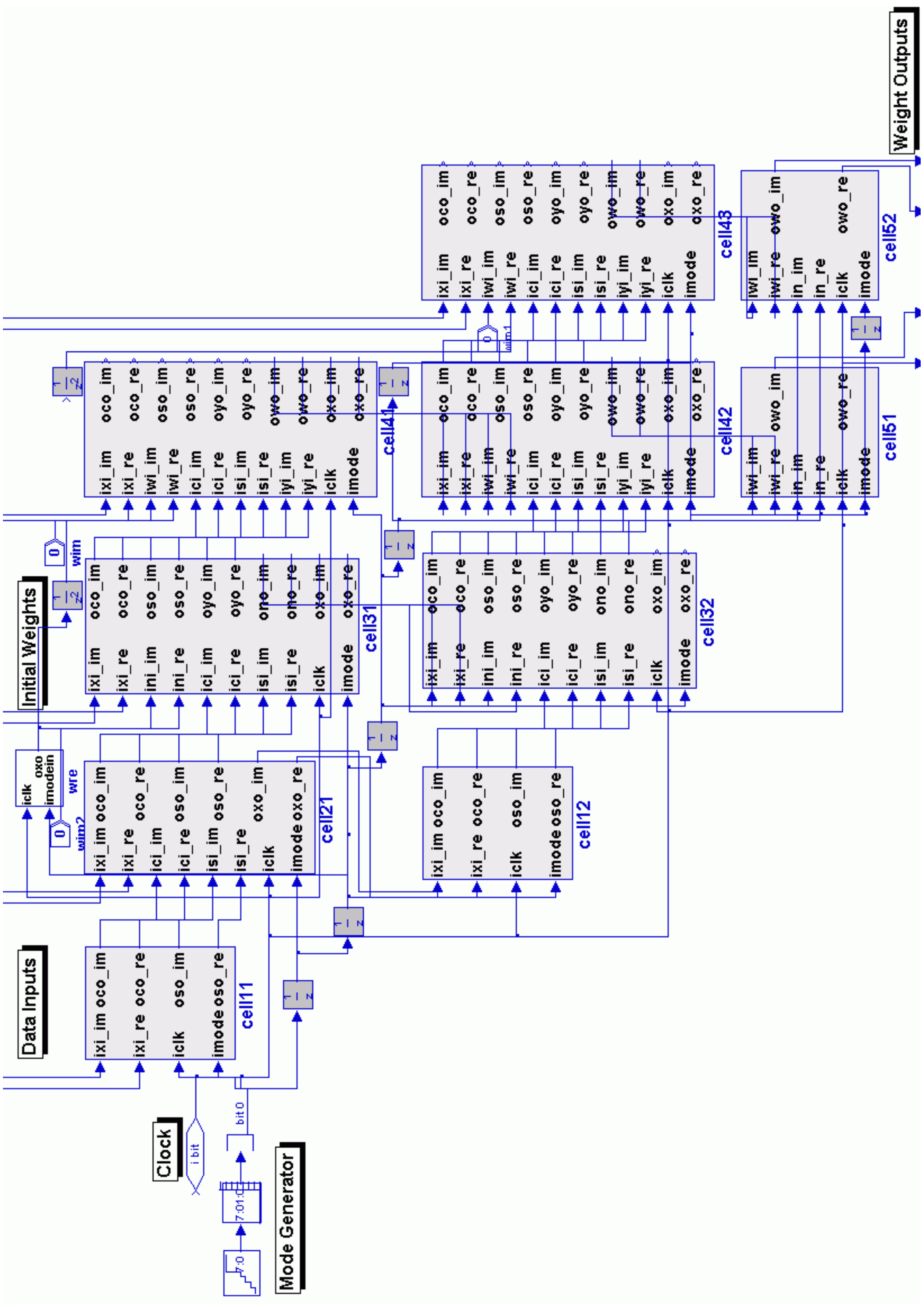


Figure 5.22 Systolic array for 2 antennas in Matlab Simulink

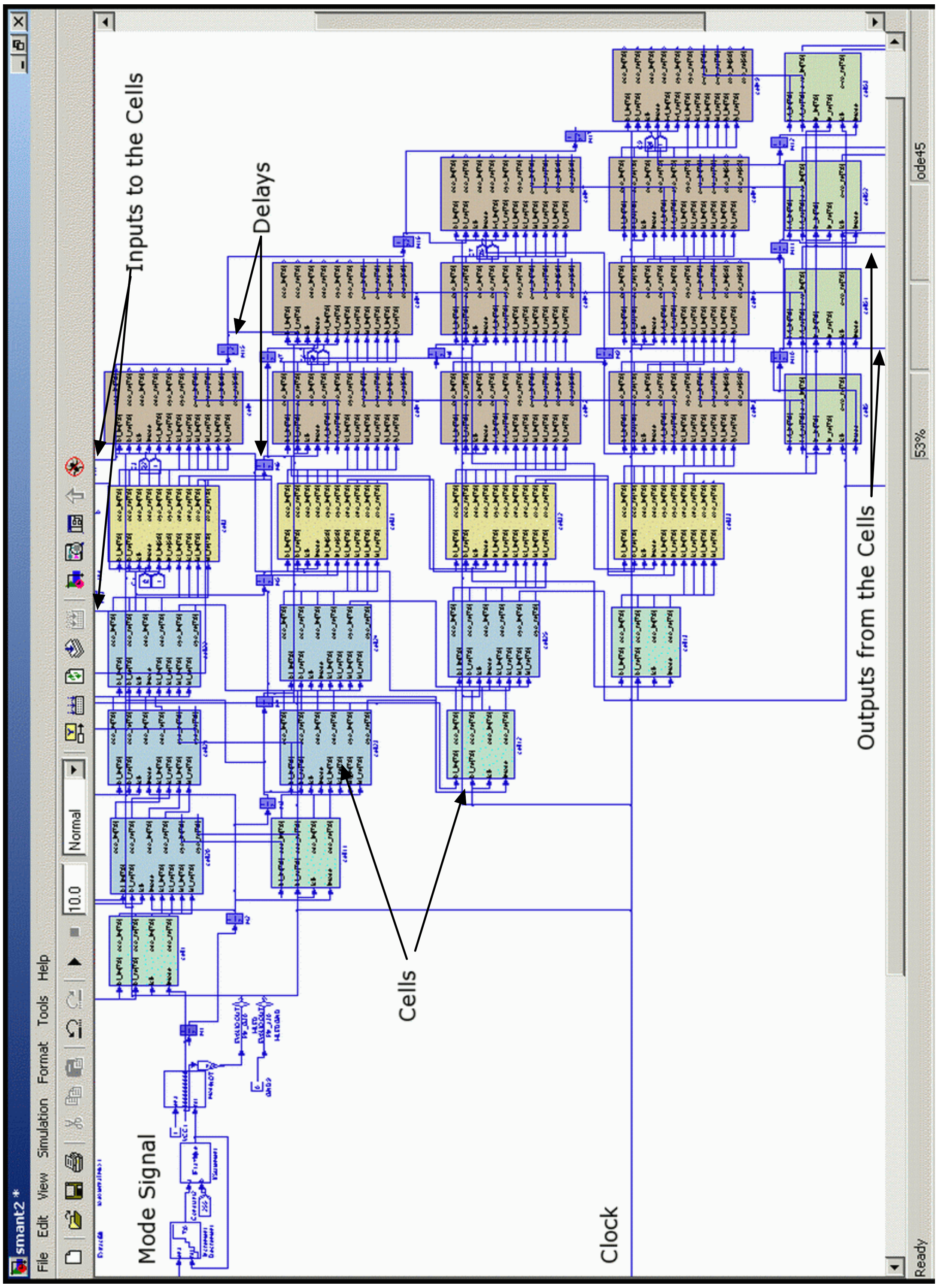


Figure 5.23 Systolic array in Matlab Simulink

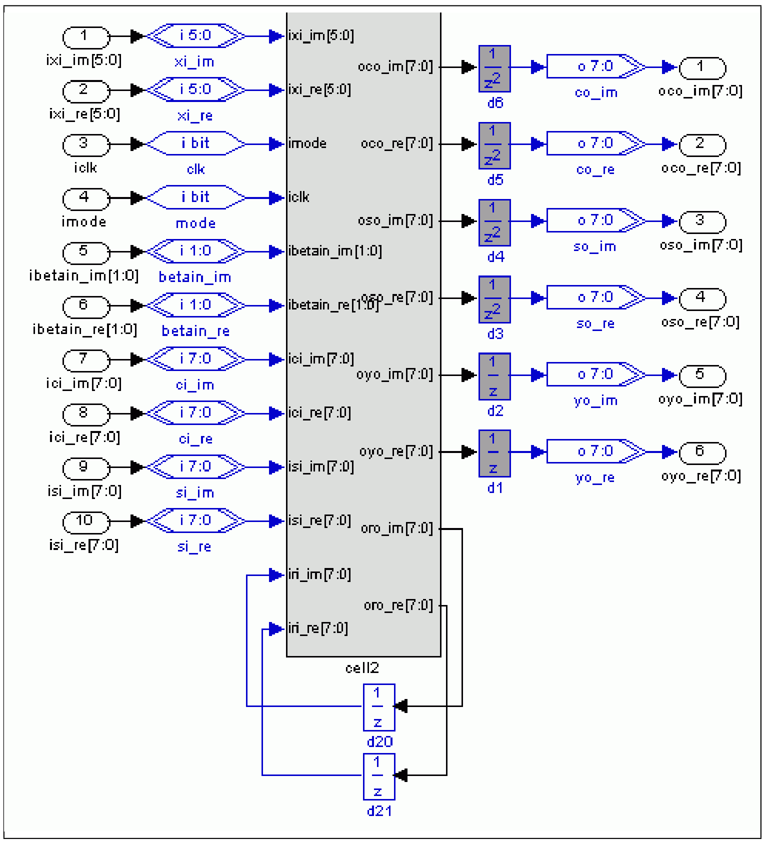


Figure 5.24 Internal Structure of C2 Cell in simulink environment

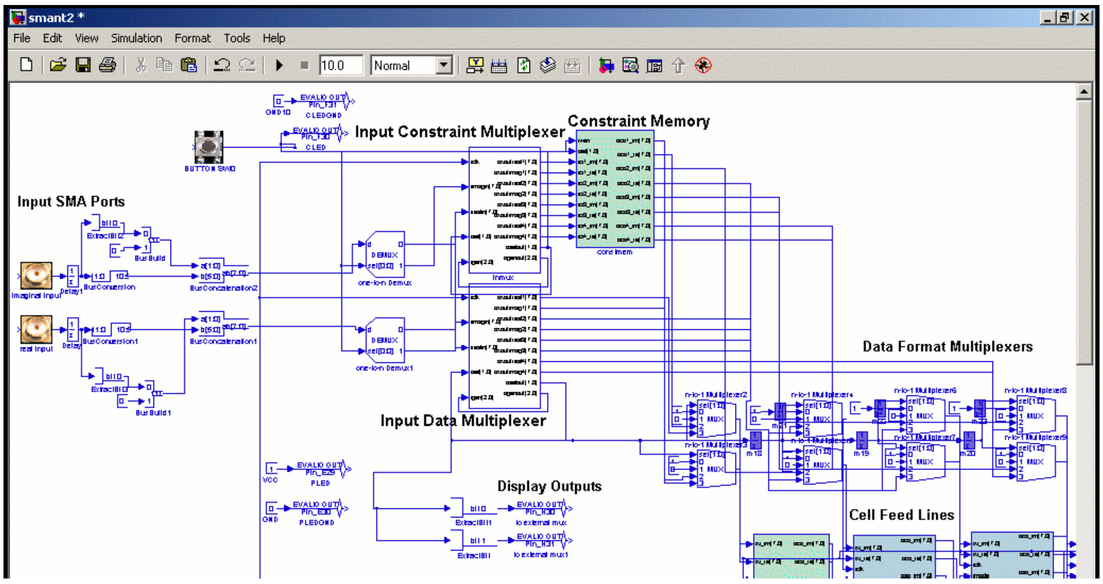


Figure 5.25 Input block of the simulink model

proper weight for the correct antenna array element.

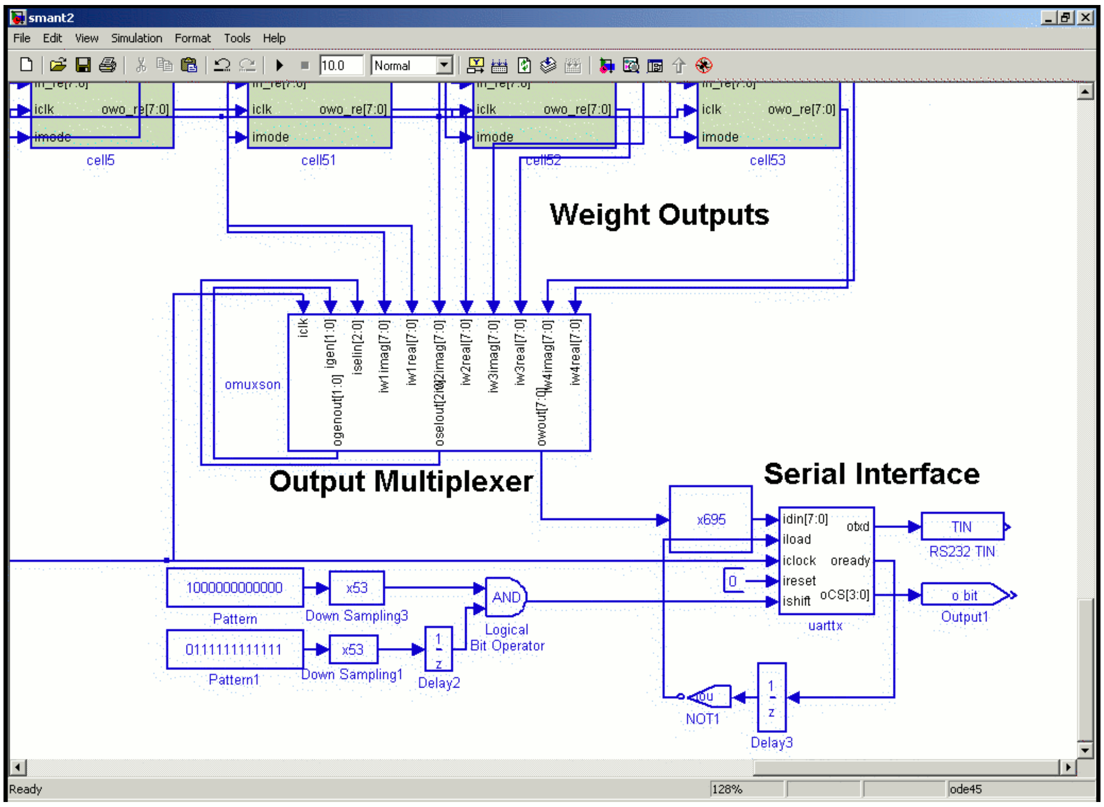


Figure 5.26 Output block of the systolic model

The folded systolic array structure, which is proposed in this work, is also implemented in a similar manner. As the initial step the cells are combined to form the tile structures and pre synthesized by using the Celoxica Agility Compiler. Afterwards the tiles are transferred into Matlab Simulink environment as distinct blocks by using the Altera DSP Builder software. Figure 5.27 demonstrates the internal structure of a tile transferred to the Matlab Simulink environment.

The simulink blocks representing the tiles are then connected properly to form the folded systolic array structure which is presented in Figure 5.28.

The input and output multiplexing circuitry presented in Figures 5.25 and 5.26 are used without any changes in the implementation procedure of the folded systolic array structure.

The interconnections between systolic array structures and the multiplexing interfaces are transferred to a VHDL code by using the Altera DSP Builder software. The SystemC blocks together with the VHDL definitions of the interconnections are then synthesized by the Altera

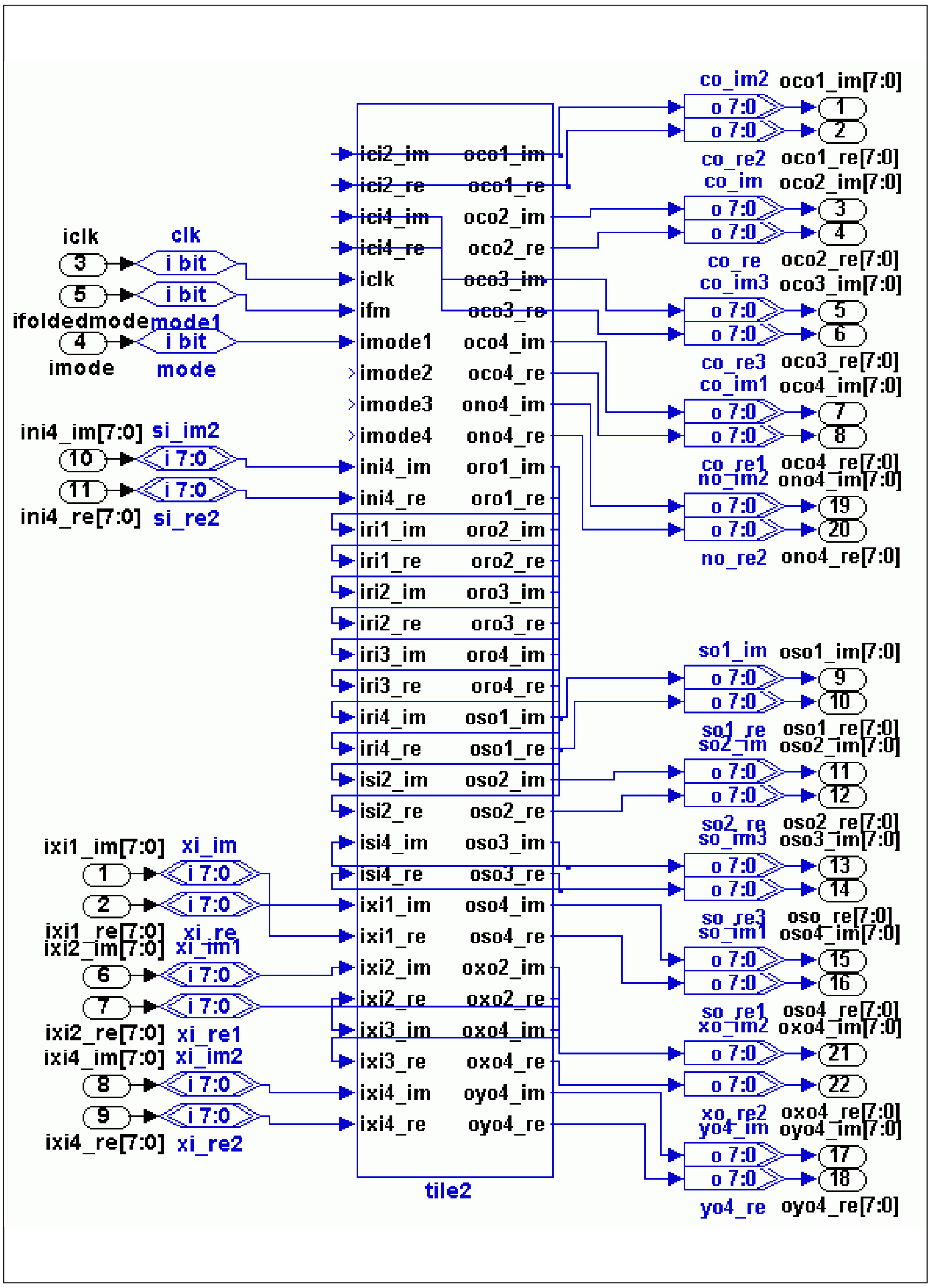


Figure 5.27 Internal Structure of Tile2 in simulink environment

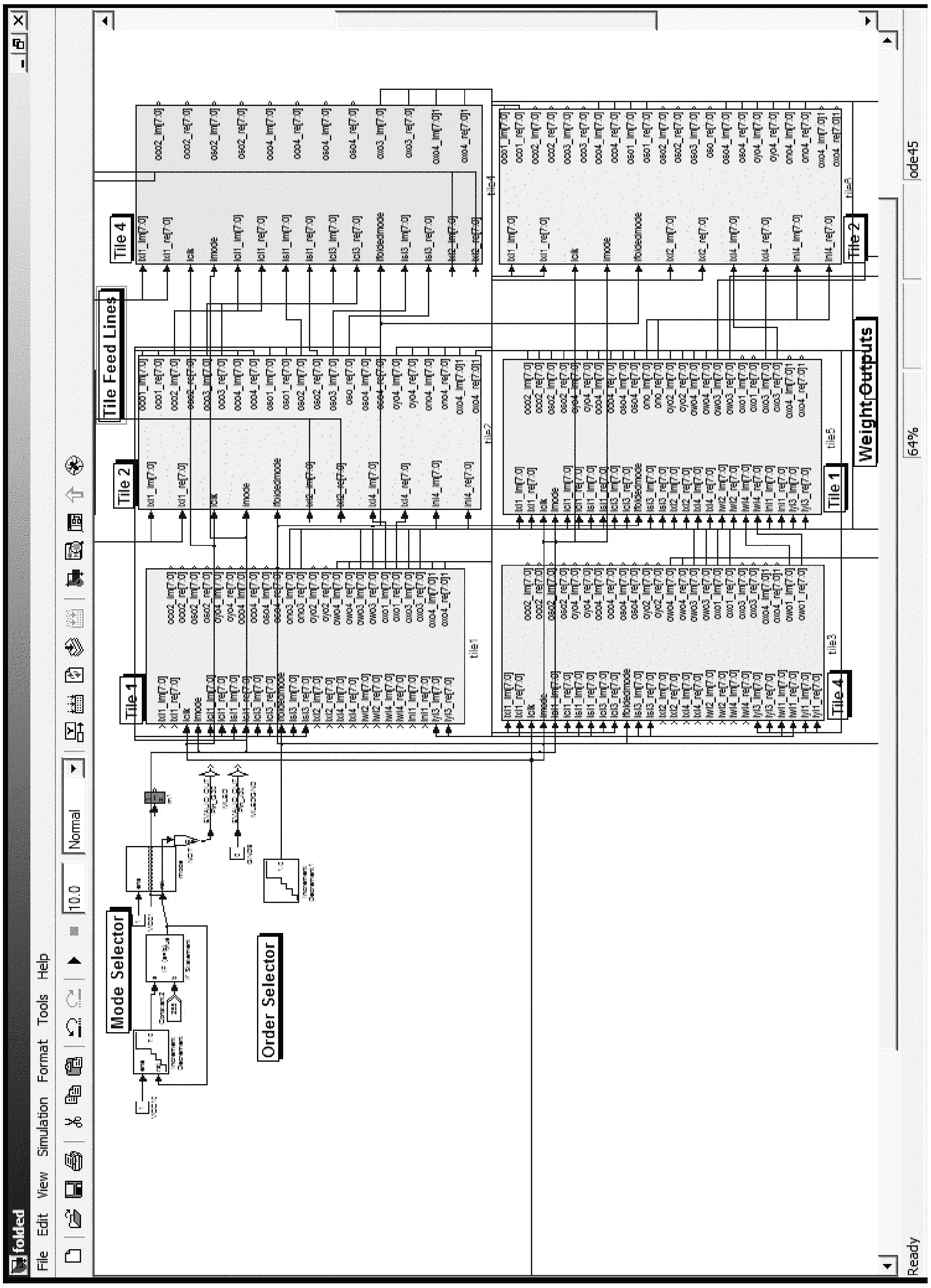
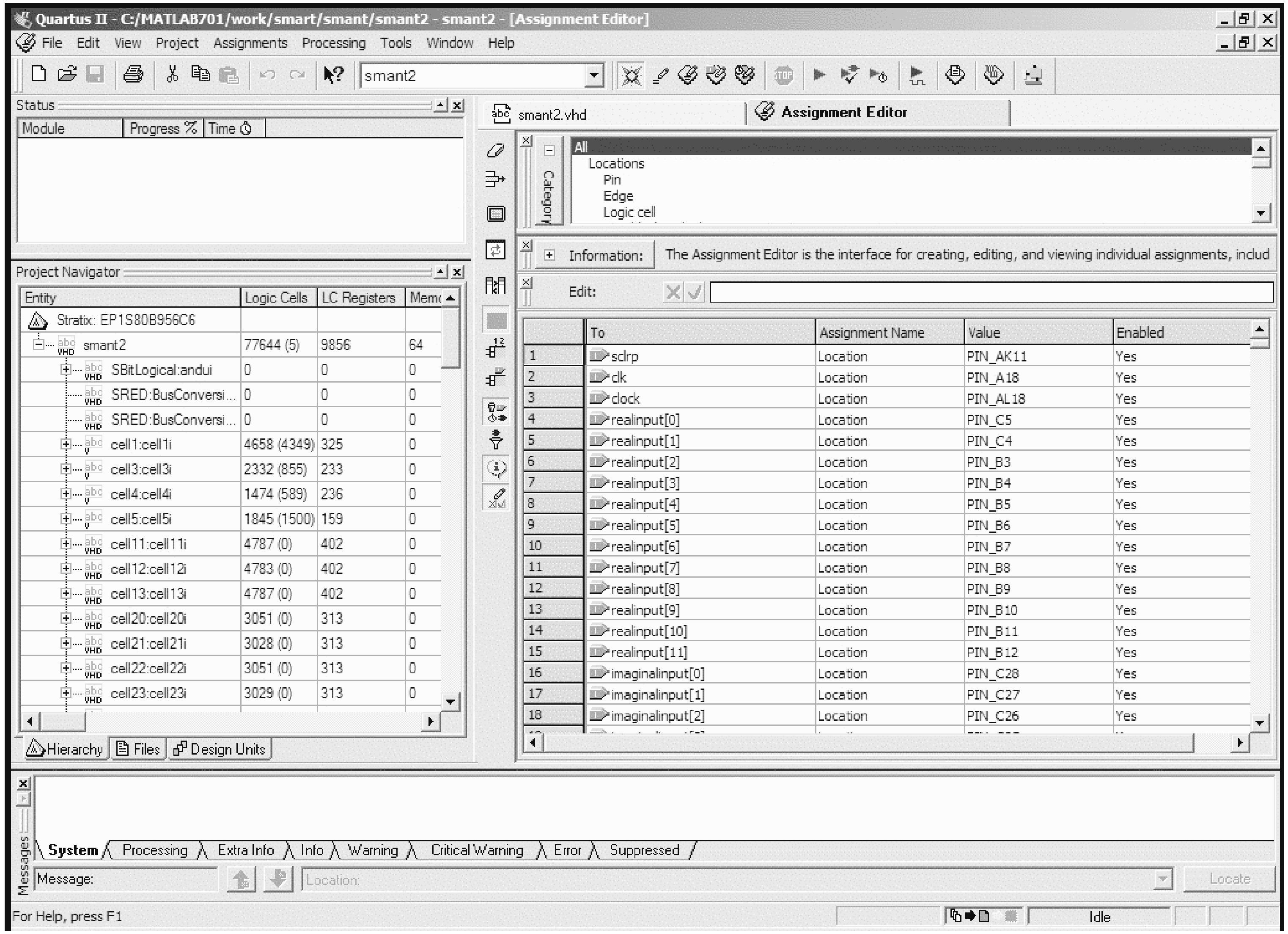


Figure 5.28 Folded Systolic Array Structure in Matlab Simulink

Quartus II software to generate a programming file for the FPGA chip. Figure 5.29 presents a screenshot of the Quartus II software. Results of the synthesis operation of both the conventional and the folded systolic array structures will be presented in the following chapter.



The Assignment Editor is the interface for creating, editing, and viewing individual assignments, includ

Entity	Logic Cells	LC Registers	Mem
Stratix: EP1S80B956C6			
smant2	77644 (5)	9856	64
SBitLogical:andui	0	0	0
SRED:BusConversi...	0	0	0
SRED:BusConversi...	0	0	0
cell1:cell1i	4658 (4349)	325	0
cell3:cell3i	2332 (855)	233	0
cell4:cell4i	1474 (589)	236	0
cell5:cell5i	1845 (1500)	159	0
cell11:cell11i	4787 (0)	402	0
cell12:cell12i	4783 (0)	402	0
cell13:cell13i	4787 (0)	402	0
cell20:cell20i	3051 (0)	313	0
cell21:cell21i	3028 (0)	313	0
cell22:cell22i	3051 (0)	313	0
cell23:cell23i	3029 (0)	313	0

	To	Assignment Name	Value	Enabled
1	sclrp	Location	PIN_AK11	Yes
2	clk	Location	PIN_A18	Yes
3	clock	Location	PIN_AL18	Yes
4	realinput[0]	Location	PIN_C5	Yes
5	realinput[1]	Location	PIN_C4	Yes
6	realinput[2]	Location	PIN_B3	Yes
7	realinput[3]	Location	PIN_B4	Yes
8	realinput[4]	Location	PIN_B5	Yes
9	realinput[5]	Location	PIN_B6	Yes
10	realinput[6]	Location	PIN_B7	Yes
11	realinput[7]	Location	PIN_B8	Yes
12	realinput[8]	Location	PIN_B9	Yes
13	realinput[9]	Location	PIN_B10	Yes
14	realinput[10]	Location	PIN_B11	Yes
15	realinput[11]	Location	PIN_B12	Yes
16	imaginalinput[0]	Location	PIN_C28	Yes
17	imaginalinput[1]	Location	PIN_C27	Yes
18	imaginalinput[2]	Location	PIN_C26	Yes

Figure 5.29 Altera Quartus II software for final synthesis

CHAPTER SIX
RESULTS

The conventional and the folded systolic array structures for the QRD RLS algorithm based MVDR beamforming system are implemented as presented in Chapter Five. As denoted in Chapter Four the folded structure needs less logic elements for the implementation since it enables the reusage of functions inside the cells. Following section will present the synthesis results and comparison of the conventional and the folded systolic array structures. The test bed results of the smart antenna system will be given afterwards.

6.1 Synthesis Results

Synthesis is a translation process, taking a design description from one level of abstraction to a lower level. Behavioral synthesis of SystemC is the process of translating, the SystemC code into a register-transfer-level (RTL) structure. This entails determining the number of clock cycles and separating out data paths, memories, and control units, ruling the same data paths and memories. Although the exact nature of data path operators may be left to the ensuring synthesis step, the data path architecture (e.g. number of data path registers, pipeline depths etc.) becomes fixed in this step. The end result is normally written out as RTL code. Following synthesis steps are the RTL synthesis and the logic synthesis.

Signal processing applications usually require floating point operations for better resolution and to avoid rounding and cropping errors. However floating point data types cause too much delay in signal processing systems and require too many logic elements for the implementation as well as it is not defined as a synthesizable data type of the SystemC library. Eventhough fixed point operations lack of precision they do not consume as much resources as floating point data types. In our application fixed point operations using integer data types are defined in the complex operations library. The parameters used are 8 bit word length with 4 signed integer bits and 4 fractional bits, so that their values range from $\pm\ 8$ with step size $0,0625$.

The pre-synthesis operation of the SystemC code is achieved by using the Celoxica Agility Compiler. Pre-synthesis is defined as the operation of translating the SystemC code, to a suitable code for the Altera Quartus II software, like EDIF or Verilog. The pre-synthesis results of the developed cells and tiles are given in Table 6.1 and in Table 6.2. These results are the estimated logic element usage of the developed code by the Celoxica Agility Compiler. Eventhough these are not the exact logic resources used by the cells and tiles, it can be evaluated that for a four antenna array the conventional systolic array structure needs 94404 look up tables(LUT) and 6948 flip flops(FF) for the implementation. However for the folded systolic array structure we only need 38475 LUT's and 4739 FF's for the implementation of a same size smart antenna which corresponds to a 40 % reduction in LUT's and 68 % reduction in flip flops. Eventhough these synthesis results are only estimations for the pre-synthesis operation and optimization for the full synthesis operation can affect the results.

Table 6.1 Resources needed for the implementation of cells (pre-synthesis results)

	C-1	C-2	C-3	C-4	C-5
LUT	13175	1272	1347	1318	3876
FF	468	181	223	241	172

Table 6.2 Resources needed for the implementation of tiles (pre-synthesis results)

	Tile1	Tile2	Tile3	Tile4
LUT	6414	3098	1369	2578
FF	1017	514	452	537

The developed and simulated code is synthesized by the Quartus II software supplied by the FPGA manufacturer Altera Corporation. The target hardware for the synthesis is selected EP1S80B956C6, which is a member of the Altera Stratix Family. The Altera Stratix family includes DSP Multipliers which reduce the number of logic elements needed for the implementation especially for signal processing operations. The EP1S80B956C6 FPGA has 79,040 Logic Elements, 7,427,520 RAM bits, 22 DSP blocks, 176 Embedded DSP, multipliers, 12 PLLs and 956 User I/O pins.

The synthesis result of the SystemC code for the cells and the tiles can be seen in Table 6.3 and 6.4. These results are the final synthesis results after optimization procedure. The C-1 type cell uses the resources more than the other cells since it is responsible for calculating the c and s parameters, which include division and square root operations. The C-2, C-3 and C-4 types of cells occupy similar amount of logic area with each other since they accomplish analogous tasks, as processing the received data and forwarding the c and s parameters as defined in Chapter Three. The C-5 type cell uses the least resources since it is only responsible of normalizing the weights.

Table 6.3 Resources needed for the implementation of cells (final synthesis results)

	C-1	C-2	C-3	C-4	C-5
Logic Elements	5142	3545	3326	3541	2033
DSP Multipliers	5	0	0	0	0

Table 6.4 Resources needed for the implementation of tiles (final synthesis results)

	Tile1	Tile2	Tile3	Tile4
Logic Elements	7807	11764	1171	4282
DSP Multipliers	2	5	0	0

As can be seen in Table 6.4, Tile2 occupies the largest number of logic elements since it includes C-1 type of cells which also has the same property when compared with other cells.

The reduction ratio of the number of logic elements of the tiles and the total number of logic elements of the cells they include, is given in Table 6.5. Tile3 and Tile4 are compressed better than the other tiles as expected, since cells inside these tiles are the same and only some switching and routing mechanisms are added. Tile 2 has the worst ratio since it has three different types of cells and each of these cells is coded inside the tile distinctly.

Table 6.5 Resources needed for the implementation of tiles (final synthesis results)

	L.E.	Total L.E.	Ratio
Tile1	7807	13949	55 %
Tile2	11764	17155	68 %
Tile3	4278	14164	30 %
Tile4	4282	14180	30 %

The synthesis results of the implemented smart antenna with a conventional systolic array occupies 72732 logic elements with 17 Embedded DSP multipliers (9 bit), while same size of an array running on a folded structure only occupies 29725 logic elements and 8 Embedded DSP multipliers which corresponds to a reduction ratio of 40% for the logic elements. Note that these are the result of the fitter operation which also employs some optimization techniques.

The proposed folding scheme becomes more efficient for larger antenna arrays since the tiles Tile3 and Tile4 are used more frequently in larger arrays and these tiles benefit more from the reusage approach. This phenomenon makes the die size of the large arrays smaller compared with the conventional systolic array as can be seen in Table 6.6. As the number of antennas increase from 4 to 16 the ratio of the used logic elements needed for implementation of folded and conventional systolic arrays decrease from 52% to 38%.

Table 6.6 Resources needed for the implementation for systolic arrays with differens sizes)

	Folded		Conventional		Ratio	
Number of Antennas	LE's	DSP's	LE's	DSP's	LE	DSP
4	47684	14	90552	20	53	70
6	84339	21	178344	30	47	70
8	129536	28	294480	40	44	70
10	183275	35	438960	50	42	70
12	245556	42	611784	60	40	70
14	316379	49	812952	70	39	70
16	395744	56	1042464	80	38	70

6.2 Implementation Results

Implementation of both the conventional and the folded systolic array based smart antenna systems were presented in Chapter Five. These implementations are synthesized and then downloaded to the FPGA chip on the Altera DSP Development board. The FPGA chip with the systolic array is then connected to the other hardware presented in Chapter Five via the ADC's and DAC's existing on the DSP Development board.

A block diagram of the smart antenna testbed is presented briefly in Figure 6.1. The local oscillator inputs of the receiver and the transmitter blocks of the smart antenna system are connected to a RF generator which operates at 2.0 GHz. The RF generators of the mobile units are also tuned to 2.0 GHz for a reliable operation of the smart antenna system. As presented in chapter three the smart antenna separates the desired user from the interfering user according to the angular placement of the users. The smart antenna system treats the user inside the look direction as the desired user and the other user(s) as the interfering user(s).

Eventhough the employment of the smart antenna improves the communication quality, it is not easy to observe the performance and the operation of it. The beamform monitor shown in the Figure is a personal computer that receives the weights of the beamformer via serial port and displays the resultant radiation pattern. So that the operation of the system is visible to the observers and its performance for various cases can easily be established. The beamform monitor updates its display continuously to present the instantaneous performance of the smart antenna system.

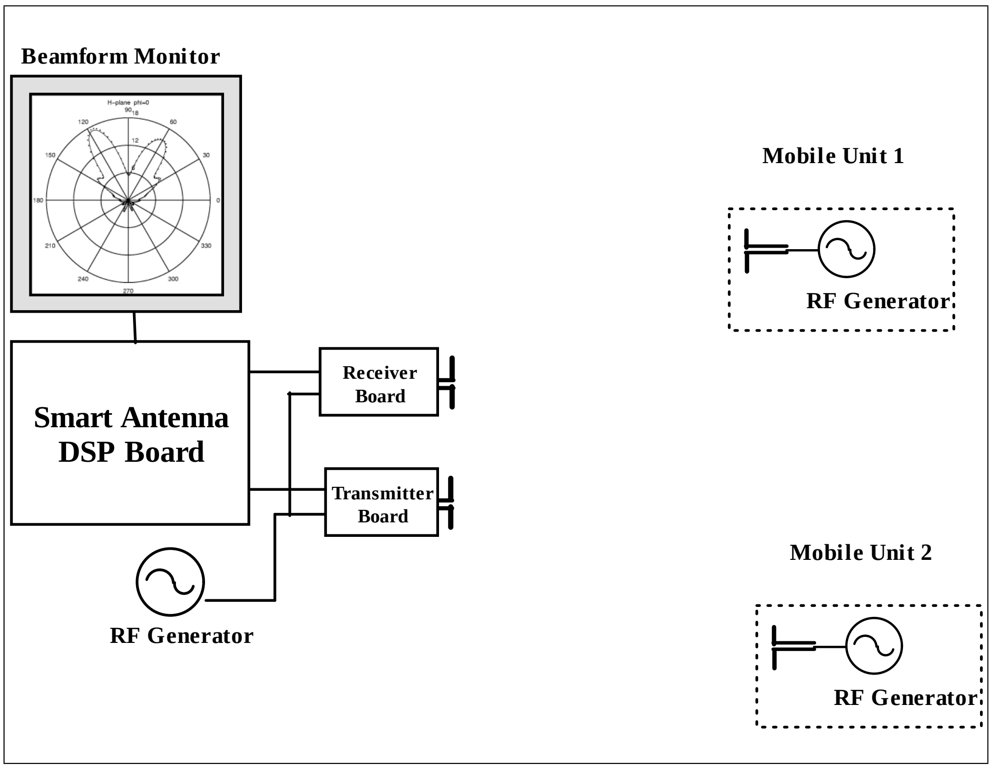


Figure 6.1 Brief diagram of the smart antenna testbed

The look direction of the beamforming algorithm is determined by acquiring and storing the data from the single user when the SW1 push button on the DSP Development Boards is pressed. The stored data is then used as the constraint data vector for the systolic array which determines the look direction. For multiple constraint applications other onboard switches can be used for storing data for other look directions and use as different constraint vectors.

The test procedure of the smart antenna is as follows; first the beamforming algorithm to be tested is downloaded to the FPGA chip. Then for single user case the constraint data vector is stored in the FPGA, by first placing the mobile user to the look direction of the antenna and then pressing the onboard pushbutton "SW1" so that the signal impinging on the receiver board is stored as the constraint vector. The desired user is then located to the angle of the constraint vector and the resultant radiation pattern is obtained at the beamform monitor.

For the two user case the constraint data vector is stored in the FPGA. Then one of the users is located at or close to the look direction and the other one is placed to another angular location and the radiation pattern formed by the beamforming algorithm is viewed from the display of the beamform monitor. Performance of the smart antenna system for various locations of the users are presented in the following figures. The figures denote the array factor plot corresponding to the weight output of the system, therefore, only the part between 0^o and 180^o must be taken into consideration since the antennas do not propagate backwards because of the reflecting ground plane.

Figure 6.2 presents the radiation pattern of the smart antenna system for a single user thest, when the look direction is adjusted to 90^o,and the user is located at 90^o. As can be seen in the Figure main beam of the antenna array is directed to the user.

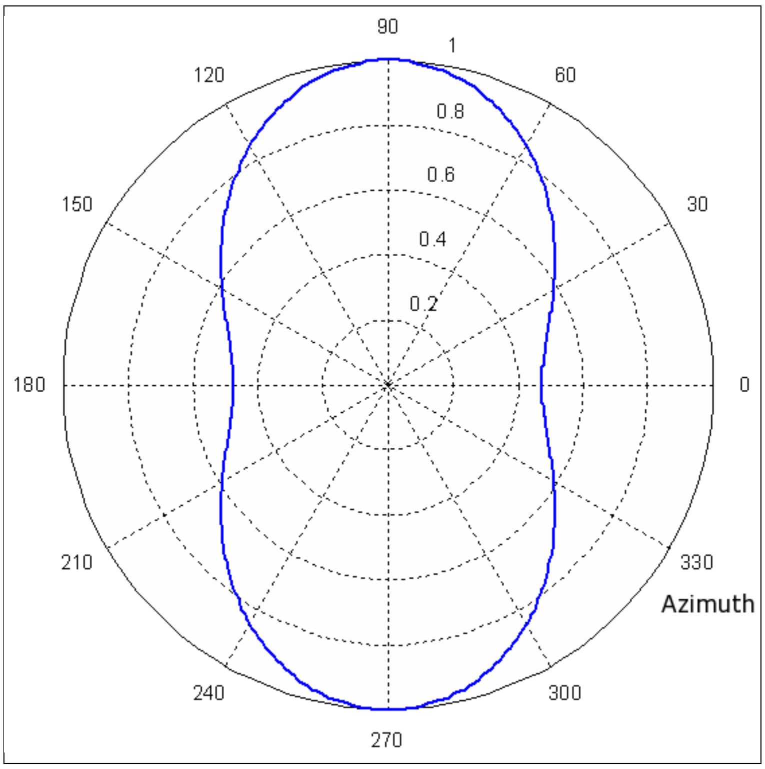


Figure 6.2 Plot of the radiation pattern when user 1 is located at 90^o and user 2 is located at 30^o

For the second case the location of the user and the constraint vector are set to 60^o. The radiation pattern of the antenna array becomes as presented in Figure 6.3. The main lobe of the radiation pattern is slightly (5^o) shifted to the left from 60^o.

When we change the constraint vector and the location of the user to 30^o the radiation

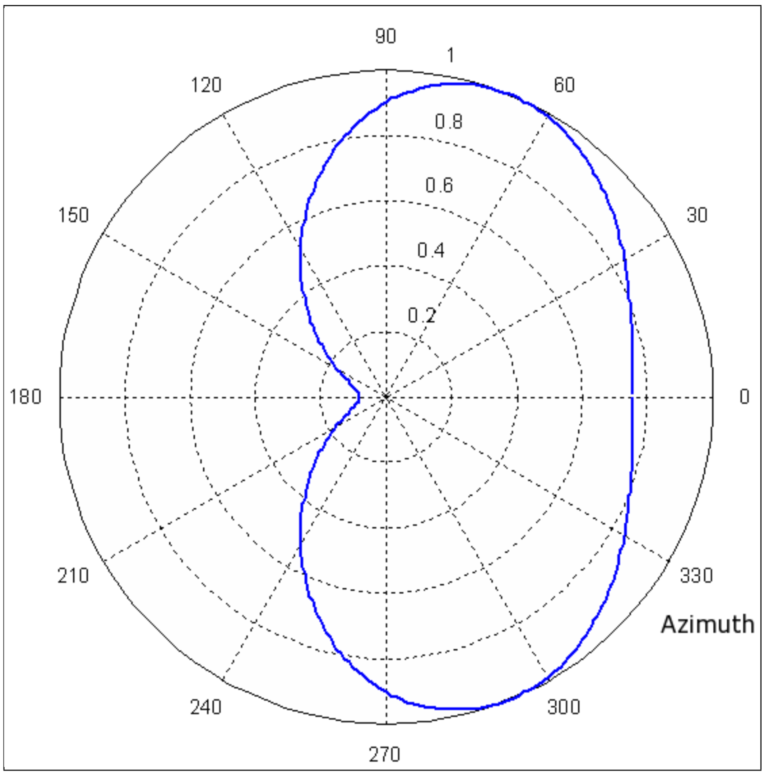


Figure 6.3 Plot of the radiation pattern when user 1 is located at 90^o and user 2 is located at 60^o

pattern of the antenna array becomes as can be seen in Figure 6.4. Eventhough the main beam of the antenna array is not exactly directed towards the user , we can observe that there is less difference betwen the gain at the maxima of the main beam and the location of the desired user.

Figure 6.5 presents a theradiation pattern for a two user case. The constraint vector is adjusted for a look direction of 90^o while the users are kept at 90^o and 45^o is given. The null is steered to the user at 45^o, while the main beam is close to the user at 90^o but not exactly at the same direction (shifted 1^o-2^o to the left). The attenuation of the desired user with respect to the interfering user is -8 dB.

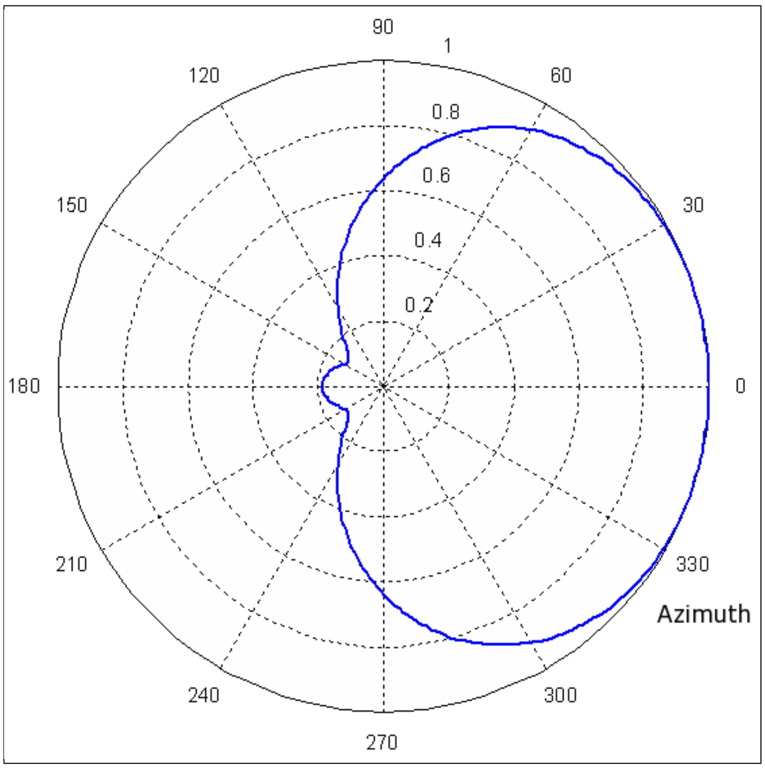


Figure 6.4 Plot of the radiation pattern when user 1 is located at 60^o and user 2 is located at 120^o

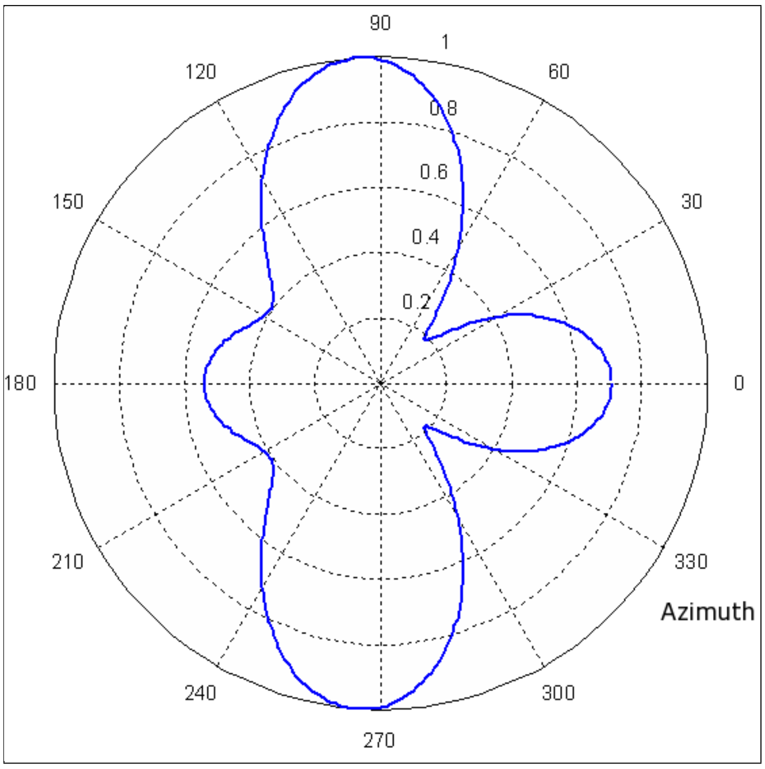


Figure 6.5 Plot of the radiation pattern when user 1 is located at 120^o and user 2 is located at 60^o

CHAPTER SEVEN
CONCLUSION

Throughout this thesis work, a smart antenna system running on a proposed folded rectangular systolic array for QRD RLS algorithm based MVDR beamformer is designed and implemented. Systolic array architectures have attracted attention for array processing applications for two decades. However conventional systolic arrays include many arithmetic operations like multiplication, division, square root which makes their implementation difficult because of the large size of their die. Folding of the systolic structures have the potential of reducing die size by enabling reusage of functions of the elements of the array.

The rectangular systolic array structure, which is proposed in this thesis work, aims to eliminate the poor folding nature of the conventional systolic algorithm, as depicted in Chapter Three. The rectangular systolic structure is more adequate for partitioning and folding schemes, since the folded tiles can include more than three cells, which improves the benefits of the folding approach by reducing the resources needed for implementation. The rectangular systolic array is then folded into tiles that include four cells each. Therefore, functions of the cells inside the tiles can be reused, and the die size of the implementation can be reduced significantly.

The rectangular systolic array structure, does not have any drawbacks, when compared with the conventional rhomboid systolic array, from which it has been derived from. However its structure is more adequate for partitioning and folding. Thus the folded tiles include more cells, let the signal processing algorithms to be implemented into smaller and cheaper FPGA chips, at the same time algorithms with very high operation count, that are hard to implement, will also have the possibility for an implementation.

In accordance with theoretical expectations, the rectangular systolic array could easily be folded into tiles including more cells. Eventhough, in this work each tile included four cells, depending on the number of antennas, tiles composed of six, nine or even sixteen cells could be used upon request. Tiles including more tiles could of course result with a systolic array that uses the resources of the hardware more effectively.

After the folding process of the systolic array, the resources used by T2 and T4 tiles are reduced to 30% of the total logic elements of the cells they include. This is because all of the cells are identical and full reusage of them has become possible. T1 and T3 tiles include less identical cells and the reduction ratio of them are 55% and 68% respectively. However reduction ratio of the folded systolic array beamformer system when compared with the conventional one is 40% which means that a beamformer using the folded structure could be implemented to a FPGA chip with one over third capacity of the conventional systolic array.

The implementation procedure is composed of two main parts hardware and software design. The design of the quadrature modulators and demodulators, the antenna and the antenna arrays and their implementations compose the hardware part of this work. The quadrature demodulator is used as the direct downconversion receiver, which also extracts the phase and amplitude information of the channels, relative to a reference channel, in forms of inphase (I) and quadriphase (Q) signals. The phase and amplitude information is then used for representing the acquired data in complex forms in the signal processing part. This method overcomes the resolution problems of the analog to digital converters as mentioned in chapter five. The transmitter part of the antenna employs quadrature modulators to convert the signals in I and Q forms into complex signals at RF frequency and transmit to the antennas.

The quasi yagi antenna is preferred for both the receiver and the transmitter part of the system. It is a printed dipole antenna with a microstrip feed line which makes it easy to fabricate with other components of the system. The array is designed as a uniform linear array with four antennas printed on the same circuit board with the modulators and demodulators.

The Open SystemC Initiative (OSCI) defines the SystemC as a language built in C++ that spans from concept to implementation in hardware and software. Eventhough it is a new platform for hardware and software modeling, design, verification; it has been approved by the IEEE as a standard language for its functions, which makes it a promising language for system level applications and designs. SystemC brings out the advantage of testing and developing of related systems on a single platform, over other hardware description languages. The SystemC model developed for the smart antenna system could easily be embedded in a communication system and the performance of the whole system could be evaluated in a single environment in a more realistic way.

The SystemC library for the C++ programming language is preferred as the hardware development environment for this thesis work. The SystemC code, which is developed for cells or tiles is then presytnhesized by using Celoxica Agility Compiler software and transferred to the Matlab Simulink to realize the interconnections between cells or tiles and form a systolic array structure. The Altera DSP builder software is then used to generate the VHDL code for these interconnections. The resultant codes and interconnections are then synthesized by using the Altera Quartus II software and downloaded in to the FPGA chip.

The array signal processing algorithms require complex operations to represent the phase and amplitude differences acquired from the sensors. So, a complex operations library needed for easy development of the SystemC code is built as a part of the thesis work for software implementation. Operations like multiplication, division and square root are defined in forms of CORDIC operations, in order to reduce the resources required for implementation as mentioned in Chapter Five.

Measurement results which were evaluated at the test bed and are presented in Chapter Six, are also in accordance with the theoretical expectations. The adaptive antenna array has chosen the user which is inside the look direction as the desired user and directed the main beam of the array towards that user for both single user and multi user tests. For tests with two users the algorithm places null to the direction of the interfering user and attenuating it up to 8 dB. The smart antenna system better performed when the look direction is determined as ninety degrees and placed the null at the interferers direction, however for look directions like sixty and one hundred and twenty the main beam and the null placement of the smart antenna has shifts.

The smart antenna system presented in this work has the potential to be applied to actual mobile communication technologies, by adjusting the RF parts for the frequency band of the system. However the signal processing part is a beamformer which is independent of the application, thus with proper sensor array systems and data acquisition elements, this part could be employed in applications such as RADAR or SONAR where array signal processing applications count.

7.1 Futureworks

Eventhough the results of the system are in accordance with the theoretical expectations, they are evaluated using a four antenna array which will not give the best results. Using an antenna array with more elements will exhibit more reliable results.

Evaluating the results for tiles composed of more cells for different size of antenna arrays could also present valuable information for similar applications.

A uniform linear antenna array is used for the implementation. However using different array geometries could also turn better results out.

The test bed included one desired user and one interfering user however the antenna array used in this thesis could separate two interfering users. An antenna array composed of more antennas even can separate more users, and results of such a system can be evaluated.

REFERENCES

Andraka, R. (1998). A survey of CORDIC algorithms for FPGAs. In *ACM/SIGDA sixth international symposium on Field programmable gate arrays*.

Black, D. C., & Donovan, J. (2004). *SYSTEMC: FROM THE GROUND UP*. Kluwer Academic Publishers.

Boukalov, A., & Haggman, S. (2000). System aspects of smart-antenna technology in cellular wireless communications-an overview. *IEEE Transactions on Microwave Theory and Techniques*, 48, 919–929.

Burleson, W. (1991). The partitioning problem on VLSI arrays: I/O and local memory complexity. In *International Conference on Acoustics, Speech, and Signal Processing*.

Deal, W., Kaneda, N., Sor, J., Qian, Y., & Itoh, T. (2000). A new quasi-yagi antenna for planar active antenna arrays. *IEEE Transactions On Microwave Theory And Techniques*, 48, 910–918.

Dietrich, C. B. (2000). *Adaptive Arrays and Diversity Antenna Configurations for Handheld Wireless Communication Terminals*. Ph.D. thesis, Virginia Polytechnic Institute and State University.

Frost, O. L. (1972). An algorithm for linearly constrained adaptive array processing. *Proceedings of the IEEE*, 60, 926–935.

Fuhl, J., & Bonek, E. (1998). High-quality and high-speed wireless multimedia transmission technology for personal handy phone system. *Wireless Personal Communications*, 9, 271–293.

Gentleman, W., & Kung, H. (1981). Matrix triangularization by systolic arrays. *Real-Time Signal Processing Proceedings of the Society of PhotoOptical Instrumentation Engineers*, 298, 19–26.

Huo, J., & Lung, Y. H. (1998). Numerical properties of the linearly constrained QRD-RLS adaptive filter. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Hwang, Y.-T., & Hen, H. Y. (1992). Novel scheduling scheme for systolic array partitioning problem. In *Workshop on VLSI Signal Processing*.

Jainandunsing, K. (1986). Optimal partitioning scheme for wavefront/systolic array processors. *Proc. IEEE Int. Symp. Circuits and Systems*, 45, 154–161.

Kung, H., & Leiserson, C. (1978). Systolic arrays. *Sparse Matrix Proceedings*, pp. 256–282.

Kung, S. (1985). VLSI array processors. *IEEE ASSP Magazine*, 2, 4– 22.

Laurila, J. (2000). *Semi Blind Detection of Co-Channel Signals in Mobile Communications*. Ph.D. thesis, Technische Universitat Wien.

Liberti, J. C., & Rappaport, T. S. (1999). *Smart Antennas for Wireless Communications: IS-95 and Third Generation CDMA Applications*. Upper Saddle River, NJ, USA: Prentice Hall.

Lijun, G., & Parhi, K. K. (2000). Hierarchical pipelining and folding of QRD-RLS adaptive filters and its application to digital beamforming. *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, 47, 1503–1519.

Manolakis, D. G., Ingle, V., & Kogon, S. (2005). *Statistical and adaptive signal processing; spectral estimation, signal modeling, adaptive filtering, and array processing*. Artech House.

Maskell, D., & Woods, G. (2003). Adaptive subsample delay estimation using a windowed quadrature phase detector. In *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region*.

McWhirter, J. (1990). Adaptive signal processing. In *Tutorial Meeting on Digital Signal Processing for Radar and Sonar Applications*, vol. 17, pp. 63–94.

McWhirter, J., & Shepherd, T. (1989). Systolic array processor for MVDR beamforming. *IEE Proceedings on Radar and Signal Processing*, 136.

Moldovan, D. I., & Fortes, J. A. B. (1986). Partitioning and mapping algorithms into fixed size systolic array. *IEEE Trans. Comput.*, c-35(39), 1–12.

Muroga, S. (2000). *The VLSI Handbook*, chap. 44pp. 44.1–44.11. CRC Press.

Pintzos, S. G. (1991). Full-wave spectral-domain analysis of coplanar strips. *IEEE Transactions On Microwave Theory And Techniques*, 39, 239 – 246.

Proudler, I., McWhirter, J., Moonen, M., & Hekstra, G. (1996). The formal derivation of a systolic array for recursive least squares estimation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43, 247–254.

Quinton, P., & Robert., Y. (1991). *Systolic algorithms & architectures*. Prentice Hall.

Robert, S. (1998). *Receiver Imperfections and Calibration of Adaptive Antennas*. Ph.D. thesis, Technischen Universität Wien.

Roy, R., Paulraj, A., & Kailath, T. (1986). Direction-of-arrival estimation by subspace rotation methods - ESPRIT. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '86.*

Roy, R., Paulraj, A., & Kailath, T. (1987). Comparative performance of ESPRIT and MUSIC for direction-of-arrival estimation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '87.*

Swindlehurst, A., & Kailath, T. (1992). A performance analysis of subspace-based methods in the presence of model errors. i. the MUSIC algorithm. *IEEE Transactions on Signal Processing*, 40, 1758 – 1774.

SystemC Standard, I. (2006). IEEE std 1666 - 2005 IEEE standard systemc language reference manual.

Tamer, O., & Kokturk, G. (2004). Analysis of the application of the stationary wavelet transform to the direction of arrival estimation. *Frequenz Journal of Telecommunications*, 58, 246–249.

Tamer, O., & Ozkurt, A. (2007). Analysis of the folded systolic array based MVDR beamformer. *Frequenz Journal of RF Engineering and Telecommunications*, 59, 152–156.

Tang, C. E. T., Liu, K. J. R., & Tretter, S. A. (1994). Optimal weight extraction for adaptive beamforming using systolic arrays. *IEEE Trans. Aerosp. Electron*, 30, 367–385.

The Standard Test Procedures for Antennas by IEEE (1979). IEEE antennas & propagation society.

Veen, B. V., & Buckley, K. M. (1999). *Beamforming Techniques for Spatial Filtering*, chap. 61pp. 1301–1323. CRC Press.

Zooghby, A. E. (2005). *Smart Antenna Engineering*. ARTECH HOUSE, INC.

APPENDIX A
SAMPLE SYSTEMC CODES

8.1 C1 Type Cell

The header and the main code of the C1 type of cell is given below.

8.1.1 Header

```
#ifndef _COMPLEX_H_
#define _COMPLEX_H_
#include "cplxopsfixedphasor.sc.h"
/** Module Definition**/
SC_MODULE( cell11 )
{
public:
/* * Ports* */
sc_in<bool> clk;
sc_in <sc_int<8> > xi_re, xi_im, co_re, co_im, so_re, so_im;
sc_in<sc_uint<1> > mode;
private:
/** Process**/
void cell11_thread();
public:
/** Constructor**/
SC_CTOR( cell11 )
{
  SC_THREAD( cell11_thread );
  sensitive_pos << clk;
}
};
#endif
```

8.1.2 Main Code

```
#include "complex.sc.h"
void cell11::cell11_thread()
{
  /*Reset cycle, before the first wait()*/
  complex<sc_int<8> > c, ta ,tb, s, r,ro, x;//tc,
  sc_uint<1> m;
  ro.set_real(16); // initialize r to 1.0
  ro.set_imag(0);
  wait();
  while(1)
 {
    x.set_real( xi_re.read() );
    x.set_imag( xi_im.read() );
    r=ro;
    wait();
    m=mode.read();
    if( m ) //frozen mode
    {
      s=x/r;
      c.set_real(1);
      c.set_imag(0);
      ro=r;
    }
    else if( !m ) //adaptive mode
    {
      ta.set_real(6);
      ta.set_imag(0);
      ro = sqrt(x*x+ta);
      ro=sqrt(ro);
      if ( ro.get_real() == 0&& ro.get_imag() == 0)
      {
       ro.set_real(1);
      }
     tb.set_real(ro.get_real()>>1);
     tb.set_imag(ro.get_imag()>>1);
     c=tb/r;
     s=x/ro;
   }
   co_re.write( c.get_real() );
   co_im.write( c.get_imag() );
   so_re.write( s.get_real() );
   so_im.write( s.get_imag() );
   wait();
 }
}
void ag_main()
{
  cell11 toplevel( "top_level" );
}
```

8.2 Tile 4

The header and the main code of the Tile 4 type of tile is given below.

8.2.1 Header

```
#ifndef _TILE4_H_
#define _TILE4_H_
#include "cplxopsfixedphasor.sc.h"
/*Module Definition **/
SC_MODULE( tile )
{
public:
/** Port Definitions **/
sc_in<bool> clk;
sc_in <sc_int<8> > xi1_re, xi1_im, ci1_re, ci1_im, si1_re, si1_im;
sc_in <sc_int<8> > xi2_re, xi2_im, ci2_re, ci2_im, si2_re, si2_im;
sc_in <sc_int<8> > xi3_re, xi3_im, ci3_re, ci3_im, si3_re, si3_im;
sc_in <sc_int<8> > xi4_re, xi4_im, ci4_re, ci4_im, si4_re, si4_im;
sc_in<sc_uint<1> > mode1,mode2,mode3,mode4;
sc_in<sc_uint<2> > fm;
sc_out<sc_int<8> > xo1_re, xo1_im, co1_re, co1_im, so1_re, so1_im;
sc_out<sc_int<8> > xo2_re, xo2_im, co2_re, co2_im, so2_re, so2_im;
sc_out<sc_int<8> > xo3_re, xo3_im, co3_re, co3_im, so3_re, so3_im;
sc_out<sc_int<8> > xo4_re, xo4_im, co4_re, co4_im, so4_re, so4_im;
private:
/** Process Definition**/
void tile_thread();
public:
/** Constructor Definition**/
SC_CTOR( tile )
{
  SC_THREAD( tile_thread );
   sensitive_pos << clk;
}
};
#endif
```

8.2.2 Main Code

```
#include <systemc.h>
#include "tile4.sc.h"
void tile::tile_thread()
{
  /** Reset cycle, before the first wait()**/
  complex<sc_int<8> > xi, xo, ri, ro, c , s , r, y;
  sc_int<1> mode;
  ro_set.real(16);
  ro_set.imag(0);
  wait();
  while (1)
  {
    if (fm==0)              //read ports for first cell
    {
     ro=ri;
     wait();
      xi.set_real( xi1_re.read() );
      xi.set_imag( xi1_im.read() );
      c.set_real( ci1_re.read() );
      c.set_imag( ci1_im.read() );
      s.set_real( si1_re.read() );
      s.set_imag( si1_im.read() );
      mode=mode1.read();
   }
   if (fm==1)               //read ports for second cell
    {
     ro=ri
     wait();
     xi.set_real( xi2_re.read() );
     xi.set_imag( xi2_im.read() );
     c.set_real( ci2_re.read() );
     c.set_imag( ci2_im.read() );
     s.set_real( si2_re.read() );
     s.set_imag( si2_im.read() );
     mode=mode2.read();
    }
    if (fm==2)              //read ports for third cell
    {
     ro=ri
     wait();
      xi.set_real( xi3_re.read() );
      xi.set_imag( xi3_im.read() );
      c.set_real( ci3_re.read() );
      c.set_imag( ci3_im.read() );
      s.set_real( si3_re.read() );
      s.set_imag( si3_im.read() );
      mode=mode3.read();
    }
    if (fm==3)              //read ports for fourth cell
    {
     ro=ri;
     wait();
      xi.set_real( xi4_re.read() );
      xi.set_imag( xi4_im.read() );
      c.set_real( ci4_re.read() );
      c.set_imag( ci4_im.read() );
      s.set_real( si4_re.read() );
      s.set_imag( si4_im.read() );
      mode=mode4.read();
    }
/*Operation of the Cell2*/
   y = c*x-s*r;
   if( !mode )//adaptive mode
   {
     ro= c*r+ conj(s)*x;
   }
   if( mode ) //frozen mode
   {
     ro=r;
   }
   if (fm==0)            //write ports for first cell
   {
     co1_re.write( c.get_real() );
     co1_im.write( c.get_imag() );
     so1_re.write( s.get_real() );
     so1_im.write( s.get_imag() );
     xo1_re.write( y.get_real() );
     xo1_im.write( y.get_imag() );
   }
   if (fm==1))           //write ports for second cell
   {
     co2_re.write( c.get_real() );
     co2_im.write( c.get_imag() );
     so2_re.write( s.get_real() );
     so2_im.write( s.get_imag() );
     xo2_re.write( y.get_real() );
     xo2_im.write( y.get_imag() );
   }
   if (fm==2))           //write ports for third cell
   {
     co3_re.write( c.get_real() );
     co3_im.write( c.get_imag() );
     so3_re.write( s.get_real() );
     so3_im.write( s.get_imag() );
     xo3_re.write( y.get_real() );
     xo3_im.write( y.get_imag() );
   }
   if (fm==3))           //write ports for fourth cell
   {
     co4_re.write( c.get_real() );
     co4_im.write( c.get_imag() );
     so4_re.write( s.get_real() );
     so4_im.write( s.get_imag() );
     xo4_re.write( y.get_real() );
     xo4_im.write( y.get_imag() );
   }
  wait();
  }
}
void ag_main()
{
  tile toplevel( "top_level" );
}
```

APPENDIX B

ALTERA DSP DEVELOPMENT BOARD

ALTERA®

Stratix EP1S80 DSP Development Board

December 2004, ver. 1.3 Data Sheet

Features

The Stratix® EP1S80 DSP development board is included with the *DSP Development Kit, Stratix Professional Edition* (ordering code: DSP-BOARD/S80). This board is a powerful development platform for digital signal processing (DSP) designs, and features the Stratix EP1S80 device in the speed grade (-6) 956-pin package.

Components

- Analog I/O
  - Two 12-bit 125-MHz A/D converters
  - Two 14-bit 165-MHz D/A converters
  - Single-ended or differential inputs, and single-ended outputs
- Memory subsystem
  - 2 Mbytes of 7.5-ns synchronous SRAM configured as two independent 36-bit buses
  - 64 Mbits of flash memory
- Configuration options
  - On-board configuration via the 64-Mbits flash memory, plus the Altera® EPM7064 programmable logic device (PLD)
  - Download configuration data using ByteBlasterMV™ download cables
- Dual seven-segment display
- One 8-pin dual in-line package (DIP) switch
- Three user-definable pushbutton switches
- One 9-pin RS-232 connector
- Two user-definable LEDs
- On-board 80-MHz oscillator
- Single 5-V DC power supply (adapter included)

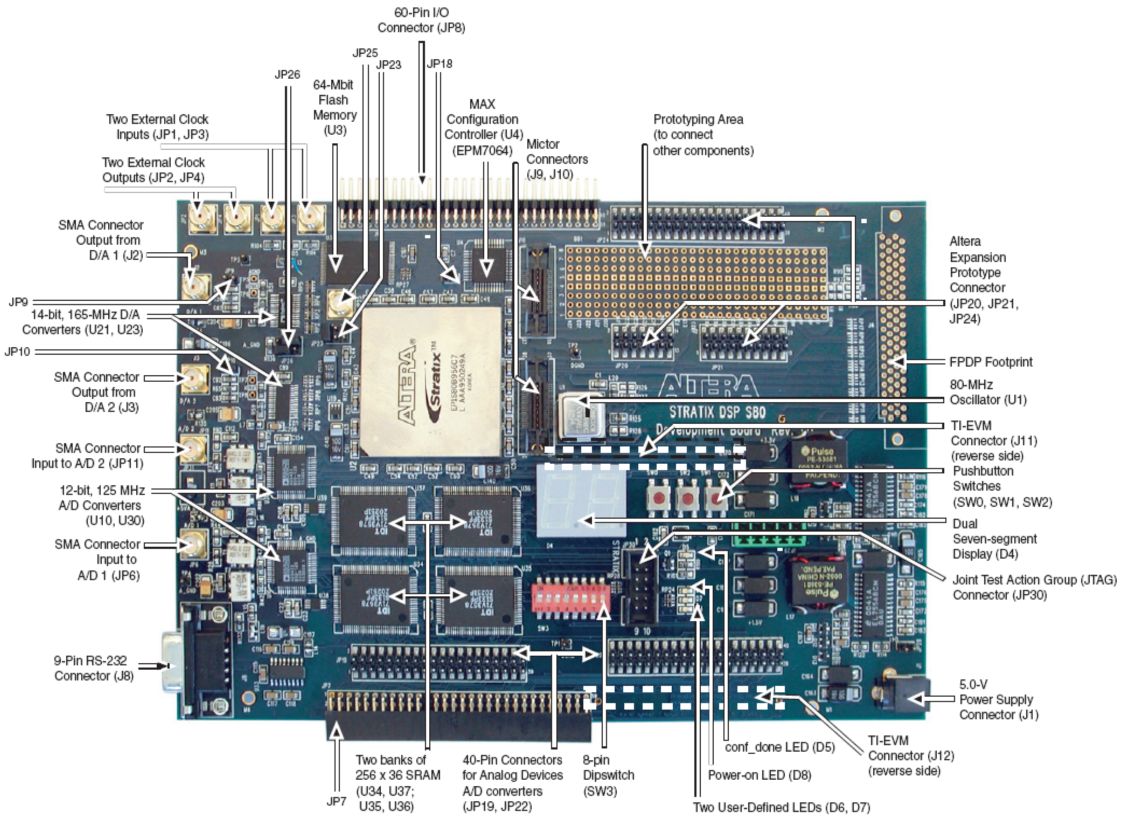
Debugging Interfaces

- Two Mictor-type connectors for Agilent Technologies logic analyzers
- Several 0.1-inch headers

Expansion Interfaces

- Two connectors for Analog Devices A/D converter daughter cards
- Connector for Texas Instruments Evaluation Module (TI-EVM) daughter cards
- Altera Expansion Prototype Connector

The Stratix EP1S80 DSP development board provides a hardware platform designers can use to start developing DSP systems based on Stratix devices immediately. Combined with DSP intellectual property (IP) from Altera and Altera Megafunction Partners Program (AMPPSM) partners, users can quickly develop powerful DSP systems. Altera's unique OpenCore® Plus technology allows users to try out these IP cores in hardware prior to licensing them. DSP Builder (version 2.2.1) includes a library for the Stratix EP1S80 DSP development board. This library allows algorithm development, simulation, and verification on the board, all from within The MathWorks MATLAB/Simulink system-level design tools. Additionally, the Stratix DSP development board has a Texas Instruments' EVM (cross-platform) daughter card connector, which enables development and verification of FPGA co-processors.


60-Pin I/O Connector (JP8)
JP25
JP23
JP18
JP26
64-Mbit Flash Memory (U3)
MAX Configuration Controller (U4) (EPM7064)
Mictor Connectors (J9, J10)
Prototyping Area (to connect other components)
Two External Clock Inputs (JP1, JP3)
Two External Clock Outputs (JP2, JP4)
SMA Connector Output from D/A 1 (J2)
JP9
14-bit, 165-MHz D/A Converters (U21, U23)
JP10
SMA Connector Output from D/A 2 (J3)
SMA Connector Input to A/D 2 (JP11)
12-bit, 125 MHz A/D Converters (U10, U30)
SMA Connector Input to A/D 1 (JP6)
9-Pin RS-232 Connector (J8)
Altera Expansion Prototype Connector (JP20, JP21, JP24)
FPDP Footprint
80-MHz Oscillator (U1)
TI-EVM Connector (J11) (reverse side)
Pushbutton Switches (SW0, SW1, SW2)
Dual Seven-segment Display (D4)
Joint Test Action Group (JTAG) Connector (JP30)
5.0-V Power Supply Connector (J1)
TI-EVM Connector (J12) (reverse side)
conf_done LED (D5)
Power-on LED (D8)
Two User-Defined LEDs (D6, D7)
JP7
Two banks of 256 x 36 SRAM (U34, U37; U35, U36)
40-Pin Connectors for Analog Devices A/D converters (JP19, JP22)
8-pin Dipswitch (SW3)

APPENDIX C

DATA SHEETS OF THE USED INTEGRATED CIRCUITRY

ANALOG DEVICES

0.8 GHz to 2.7 GHz Direct Conversion Quadrature Demodulator

AD8347

FEATURES

Integrated RF and baseband AGC amplifiers
Quadrature phase accuracy 1° typ
I/Q amplitude balance 0.3 dB typ
Third-order intercept (IIP3) +11.5 dBm @ min gain
Noise figure 11 dB @ max gain
AGC range 69.5 dB
Baseband level control circuit
Low LO drive −8 dBm
ADC-compatible I/Q outputs
Single supply 2.7 V to 5.5 V
Power-down mode
28-lead TSSOP package

APPLICATIONS

Cellular base stations
Radio links
Wireless local loop
IF broadband demodulators
RF instrumentation
Satellite modems

FUNCTIONAL BLOCK DIAGRAM

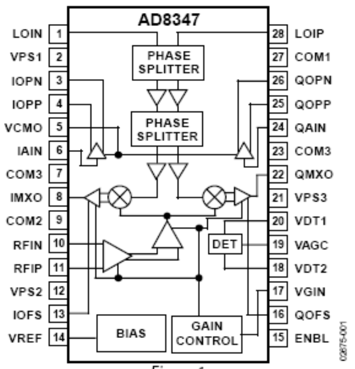


Figure 1.

GENERAL DESCRIPTION

The AD8347[1] is a broadband direct quadrature demodulator with RF and baseband automatic gain control (AGC) amplifiers. It is suitable for use in many communications receivers, performing quadrature demodulation directly to baseband frequencies. The input frequency range is 800 MHz to 2.7 GHz. The outputs can be connected directly to popular A-to-D converters such as the AD9201 and AD9283.

The RF input signal goes through two stages of variable gain amplifiers prior to two Gilbert-cell mixers. The LO quadrature phase splitter employs polyphase filters to achieve high quadrature accuracy and amplitude balance over the entire operating frequency range. Separate I and Q channel variable gain amplifiers follow the baseband outputs of the mixers. The RF and baseband amplifiers together provide 69.5 dB of gain control. A precision control circuit sets the linear-in-dB RF gain response to the gain control voltage.

Baseband level detectors are included for use in an AGC loop to maintain the output level. The demodulator dc offsets are minimized by an internal loop, whose time constant is controlled by external capacitor values. The offset control can also be overridden by forcing an external voltage at the offset nulling pins.

The baseband variable gain amplifier outputs are brought off-chip for filtering before final amplification. By inserting a channel selection filter before each output amplifier, high level out-of-channel interferers are eliminated. Additional internal circuitry also allows the user to set the dc common-mode level at the baseband outputs.

ANALOG DEVICES

0.8 GHz to 2.5 GHz Quadrature Modulator

AD8346

FEATURES

High accuracy
- **1 degree rms quadrature error @ 1.9 GHz**
- **0.2 dB I/Q amplitude balance @ 1.9 GHz**

Broad frequency range: 0.8 GHz to 2.5 GHz
Sideband suppression: –46 dBc @ 0.8 GHz
Sideband suppression: –36 dBc @ 1.9 GHz
Modulation bandwidth: dc to 70 MHz
0 dBm output compression level @ 0.8 GHz
Noise floor: –147 dBm/Hz
Single 2.7 V to 5.5 V supply
Quiescent operating current: 45 mA
Standby current: 1 μA
16-lead TSSOP

FUNCTIONAL BLOCK DIAGRAM

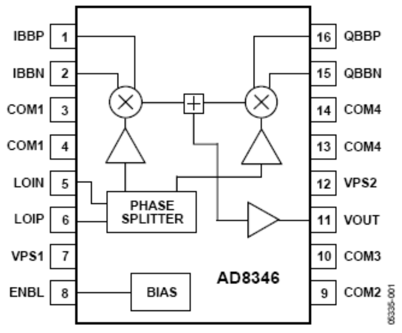


Figure 1.

APPLICATIONS

Digital and spread spectrum communication systems
Cellular/PCS/ISM transceivers
Wireless LAN/wireless local loop
QPSK/GMSK/QAM modulators
Single-sideband (SSB) modulators
Frequency synthesizers
Image reject mixer

GENERAL DESCRIPTION

The AD8346 is a silicon RFIC I/Q modulator for use from 0.8 GHz to 2.5 GHz. Its excellent phase accuracy and amplitude balance allow high performance direct modulation to RF.

The differential LO input is applied to a polyphase network phase splitter that provides accurate phase quadrature from 0.8 GHz to 2.5 GHz. Buffer amplifiers are inserted between two sections of the phase splitter to improve the signal-to-noise ratio. The I and Q outputs of the phase splitter drive the LO inputs of two Gilbert-cell mixers. Two differential V-to-I converters connected to the baseband inputs provide the baseband modulation signals for the mixers. The outputs of the two mixers are summed together at an amplifier which is designed to drive a 50 Ω load.

This quadrature modulator can be used as the transmit modulator in digital systems such as PCS, DCS, GSM, CDMA, and ISM transceivers. The baseband quadrature inputs are directly modulated by the LO signal to produce various QPSK and QAM formats at the RF output.

Additionally, this quadrature modulator can be used with direct digital synthesizers in hybrid phase-locked loops to generate signals over a wide frequency range with millihertz resolution.

The AD8346 comes in a 16-lead TSSOP package, measuring 6.5 mm × 5.1 mm × 1.1 mm. It is specified to operate over a −40°C to +85°C temperature range and a 2.7 V to 5.5 V supply voltage range. The device is fabricated on Analog Devices' high performance 25 GHz bipolar silicon process.