The University of Kansas

Information and Telecommunication Technology Center

Technical Report

Digital Beamforming Receiver Design

Srikanth Gurrapu, Glenn Prescott,
and K. Sam Shanmugan

ITTC-FY1997-TR-10920-25B

May 1997

Sponsor: Defense Advanced Research Projects Agency/CSTO
AARPA Order No. 8195
Issued Contract #J-FBI-94-223
Sprint CK5007715

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless communications is enjoying its fastest growth period in history, due to enabling technologies which permit widespread deployment. The world is now in the early stages of a major telecommunications revolution that will provide ubiquitous personal communications. There has been enormous activity throughout the world to develop personal wireless systems that combine the network intelligence of today's PSTN (Public Switched Telephone Network) with modern digital signal processing and RF (Radio Frequency) technology. The concept, called Personal Communication Services (PCS) is aimed at developing new wireless systems and services for citizens. This surge of interest is due to demand for existing services like mobile radio systems, paging systems, cordless telephone systems and cellular communications, all sharing the available frequency spectrum.

1.1 Motivation

The frequency spectrum is, and always will be, a finite and scarce resource. Thus, there is a fundamental limit on the number of radio channels that are available for various kinds of wireless/mobile services. The increasing demand for mobile communication services without a corresponding increase in RF spectrum allocation motivates the need for new techniques to improve spectrum utilization. The current generation cellular communication networks which employ either omnidirectional, or broad sector-beam, base-station antennas, will be beset with the problem of severe spectral congestion as the subscriber community continues to expand.

A measure often used to assess the efficiency of spectrum utilization is the number of voice channels per MHz of available bandwidth per square kilometer [27]. This defines the amount of traffic that can be carried and is directly related to the ultimate capacity of the network. Hence, as traffic demands increase, the spectral efficiency of the network must also increase if the quality and availability of service is not to be degraded. At present this is overcome in areas with a high traffic density by employing a technique known as *cell splitting*. Cell splitting is the process of subdividing a congested cell into smaller cells, each with its own base station and a corresponding reduction in antenna height and transmitter power. Cell splitting increases the capacity of a cellular system since it increases the number of times the channels are used. However, the continuing growth in traffic demands has meant that cell sizes have had to be reduced to a practical minimum in many city centers in order to maintain the quality of service. Unfortunately, the number of subscribers able to simultaneously access these systems is still well below the long term service forecasts. This places the great emphasis on maximizing the spectral efficiency, or ultimate capacity, of future generation systems.

In this context, the concept of adaptive antenna arrays is an emerging research area. The benefits of adaptively controlled directional antennas to the cellular radio environment have been analyzed in a number of publications over the last few years [2][4][13]. There is considerable interest at the moment in the practical application of these adaptive antenna technologies both to enhance the present generation digital communication systems and to form an integral part of the air interface specification of future generation wireless networks.

The spatial filtering properties of adaptive antenna arrays in wireless networks make it possible to confine the radio energy associated with a given user to a small addressed volume. This reduces the overall level of co-channel interference within the service area and ultimately leads to an increase in capacity for interference limited systems (for e.g., CDMA[1] systems) [14]. In addition, by using the antenna array to spatially separate and reject multipath energy, the resulting delay spread can be significantly reduced offering the possibility of service in areas which would otherwise be restricted by time dispersion [14]. The reduction in multipath activity also supports higher bit rate services within a given operational environment. An adaptive array is also capable of steering radiation pattern maxima toward the desired mobiles i.e., forming individual beams towards each mobile contributing to spectral efficiency.

Despite all these advantages adaptive array antennas offer, their applications in commercial mobile communications systems are not common. This is partly due to system implementation cost/benefit [13]. Work on feasible, low cost and less complex adaptive array system architectures is an interesting research area.

1.1.1 Smart Antenna Concept

Adaptive antennas operate by exploiting some property of the signal environment present at the array aperture [4], and it is due to this ability that they are often referred to as 'smart' arrays. The realization of such an adaptive smart antenna requires an architecture capable of locating and tracking mobiles, and a beamforming network which is capable of producing the appropriate multiple independent beams.

The former requirement is broadly classified as that of a direction finding, or a spatial estimation problem.Various adaptive estimation techniques have been

[1]Code Division Multiple Access

proposed in literature[18][16][12].

The latter part of the problem is realization of a flexible hardware which forms more than one beam. The beamforming network of an adaptive antenna array can be implemented using either analog technology at Radio Frequency (RF) or Intermediate Frequency (IF) or digital circuit technology at baseband. RF and IF techniques, although once popular, are relatively inflexible. The digital approach offers the highest precision, flexibility and capacity for future development.

1.2 Our Contribution

The use of adaptive antenna systems with digital beamforming networks in communication systems is not new from the theoretical perspective. Over the years, Rome laboratory has developed DBF systems and concepts that dramatically improve array antenna performance [26]. However, the implementation cost/benefit has kept it away from commercial communications applications so far. One of the key objective is to develop technologies and system architectures which will make the use of adaptive antennas viable.

The advent of high speed, high dynamic range A/D and D/A converters, high speed VLSI (Very Large Scale Integration) CMOS (Complementary Metal Oxide Semiconductor) digital processing makes the use of adaptive antenna systems practically realizable.

The focus of this effort is to design a flexible and scalable digital beamforming receiver (DBF) which should be capable of forming more than one beam in different directions. This work is done as a part of the Rapidly Deployable Radio Network (RDRN) project.

1.2.1 Overview Of RDRN

The Rapidly Deployable Radio Networks (RDRN) project is a DARPA funded wireless ATM (Asynchronous Transfer Mode) research project at the University of Kansas. The goal of RDRN is to create an ATM-based wireless communication system that will be adaptive at both the link and network levels to allow for rapid deployment and reconfiguration in response to a changing environment. The objective of the architecture is to use an adaptive point-to-point topology to gain the advantages of ATM for wireless networks. Possible application areas for RDRN are in battlefield communications and in disaster relief operations.

Each node in the RDRN system is equipped with a Global Positioning System (GPS) receiver, a packet radio system used for out-of-band signaling, a phased-array steerable antenna and a wireless ATM interface to integrate seamlessly with a wide area ATM network. Digital beamforming is used to create directed beams and allow for spatial reuse of transmission frequencies. Location and call management information are passed using the out-of-band packet radio based orderwire network. The orderwire network thus handles all the pre-connection establishment signaling used to set up the high-speed ATM data paths. GPS based position updates are also transmitted over the orderwire network and help in tracking mobile nodes.

The use of digital beamforming allows the multiple beams formed by a specific transmitter to all be of the same center frequency. However, different frequencies are used on the uplink and downlink directions. Within each beam, a time division multiple access (TDMA) structure is used to partition the total bandwidth between multiple users. The beamforming in the receive direction is the focus of this work. This thesis describes the design of a digital beamforming receiver.

A proof-of-concept RDRN system has been built to demonstrate the key technologies. Details of this proof-of-concept system can be found in [8]. The operation and performance analysis of the orderwire subsystem are described in [3] while details of the RDRN transmitter beamforming can be found in [7].

1.3 Organization of the report

Chapter 2 provides some background on the beamforming concept and describes a typical beamforming communication system. Then, we go onto discussing the implementation aspects of the receiver explaining the digital beamforming technique. The advantages and the technical challenges in realizing this are also discussed.

Chapter 3 proposes an architecture for the digital beamforming receiver. We start with a block diagram and go on explaining the individual blocks of the receiver. Different concepts that are used in the architecture are explained. The hardware design of each section is presented. The scalability of such an architecture is also discussed.

Chapter 4 discusses the implementation aspects of the receiver. We have built and tested all the primary blocks of the design. The implementation details and the results are given in this chapter. An APTIX rapid prototyping board (MP3) is used to validate the design for the receiver.

Chapter 5 finally gives a summary of the work done and states the conclusions we have drawn. Suggestions for future extensions to this work are also made.

Chapter 2

Digital Beamforming

Beamforming allows for spatial reuse of a particular frequency. A flexible beamformer takes advantage of the location information to steer beams to each mobile node.

Spatial reuse requires the use of electronically steerable, directional multi-beam antennas at the transmitter. By manipulating the phase of the RF waveform on a per antenna element basis, it is possible to steer a beam in a particular direction. Summing individual beam's waveforms on a per element basis allows for a composite multi-beam pattern. In RDRN, beamforming is accomplished at the transmitter as shown in Figure 2.1[7]. The transmitting antenna array is capable of forming two beams to serve two users in two different directions.

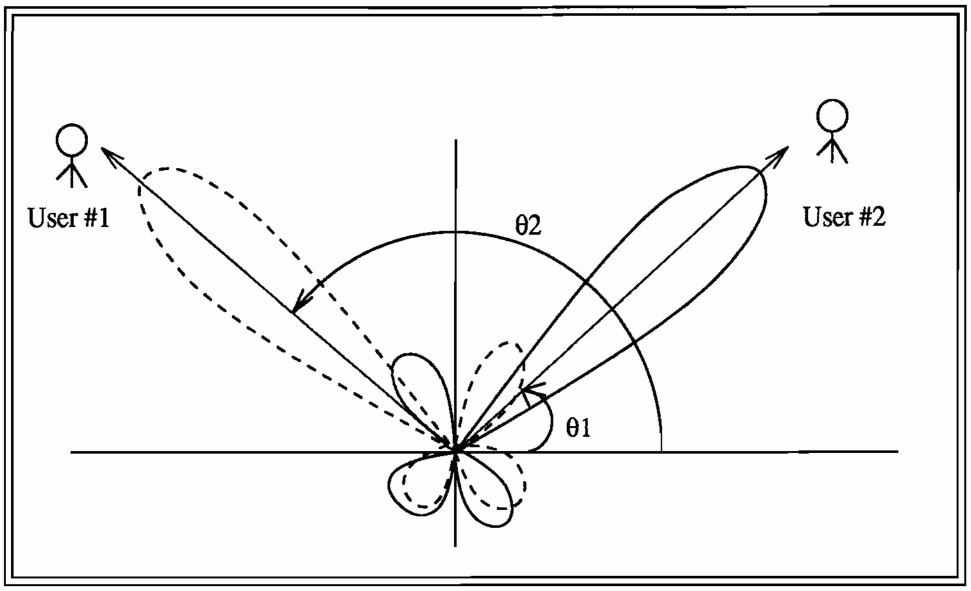

Figure 2.1: Spatial Reuse Using Beamforming

At the receiver, any node should be able to receive more than one signal that coincides in frequency and time but differs in direction. Beamforming enhances the performance of a communication system by directing most of the transmitted energy towards a specific direction rather than equal amounts in all directions. Beamforming also removes the requirement of having a mechanically steerable antenna since all beam steering is now performed through signal processing techniques applied on signals before reaching the antenna elements at the transmitter and after reaching the antenna elements at the receiver.

2.1 Receiver Beamforming Concept

To illustrate the beamforming at the receiver, consider an N-element linear antenna array as shown in Figure 2.2 [11]. All the antenna elements are equally spaced and let the spacing be *d*.

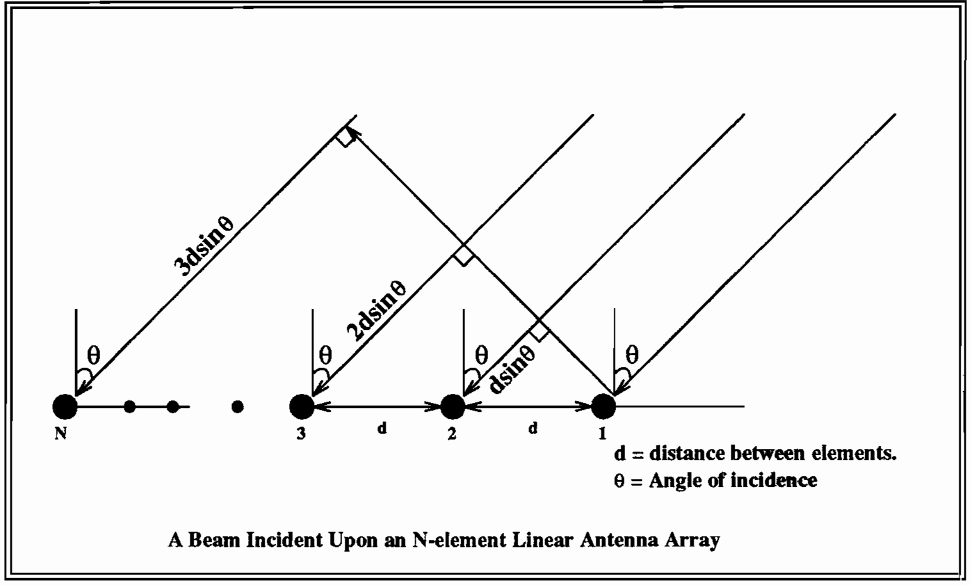
3dsinθ
2dsinθ
dsinθ
θ
N
3
d
2
d
1
d = distance between elements.
θ = Angle of incidence
A Beam Incident Upon an N-element Linear Antenna Array

Figure 2.2: An N-element array illustrating the Beamforming concept

To receive a beam in a particular direction (θ), the principle of constructive interference is used. Essentially, each antenna element receives a uniquely phase-shifted version of the signal being received. Each element's phase shift is determined by its location relative to the transmitter. In far-field situations, the angles of incidence on all the elements are equal. But, element 2 receives a delayed version of the same signal that element 1 receives, since it has to travel an extra distance of $dsin\theta$ before reaching the element. This path delay corresponds to a particular phase shift depending on the frequency of the narrow band signal we are trying to receive. Similarly, element 3 suffers twice the amount of path delay and hence, twice the amount of phase shift and, so on.

Hence, to receive the beam in the direction θ, we need to calculate the phase shifts undergone by the signals from each antenna element and then, multiply by a complex weight to compensate for the path delays associated with them, and then simply add them. To see how the complex weights are determined, from the Figure 2.2, we can do a simple analysis.The phase shift is due to the path difference of $dsin\theta$ and is given as $(dsin\theta)(2\pi/\lambda)$, where λ is the wavelength corresponding to the received signal. If we assume that the distance between the elements is half the wavelength, it becomes $\pi sin\theta$. This is a progressive phase shift for all the antenna elements. Hence, the complex weights are, $e^{j\pi sin\theta}$, $e^{j2\pi sin\theta}$, $e^{j3\pi sin\theta}$ and so on. This problem is similar to beamforming at the transmitter as an antenna is a reciprocal device. The MATLAB code for calculating complex weights for particular beam steering angles is given in Appendix A.

2.1.1 Beamforming System

The beamforming concept can be extended to support multiple beams at both the transmitter and receiver. Such an example is shown in Figure 2.3. An N-element linear antenna array is used in this example. The orthogonality property of beamforming allows one beam to be separated from the other at the receiver.

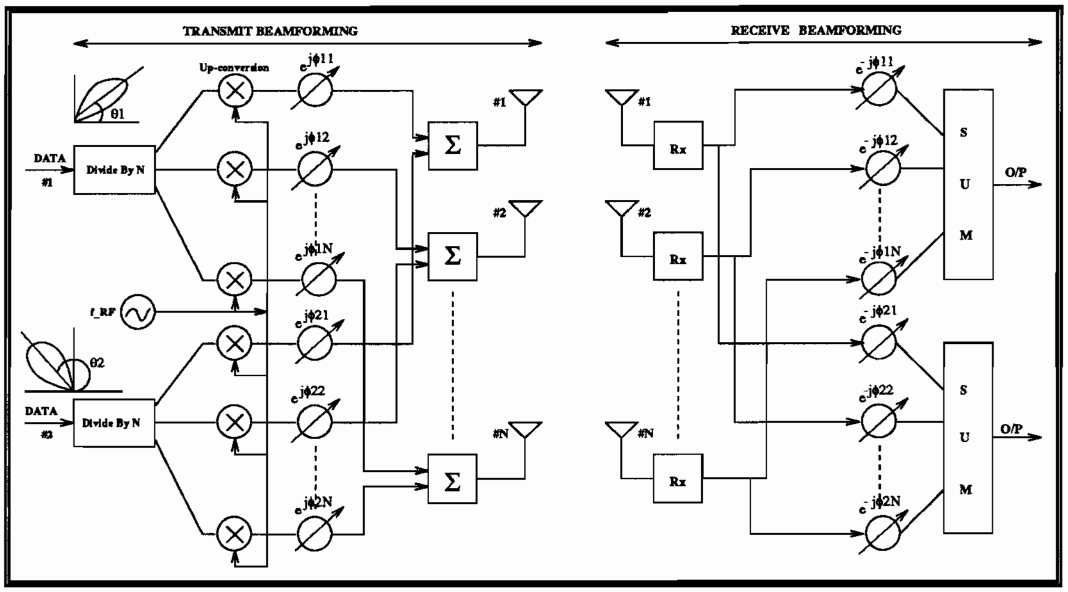

Figure 2.3: Beamforming Communication System

Let $X_1(t)$ and $X_2(t)$ be the two signals to be transmitted. Then, from the Figure 2.3, the signals at the transmitter antenna elements (assume N=4) can be written as:

$$A_1(t) = x_1(t)e^{j\phi_{11}} + x_2(t)e^{j\phi_{21}} \tag{2.1}$$
$$A_2(t) = x_1(t)e^{j\phi_{12}} + x_2(t)e^{j\phi_{22}} \tag{2.2}$$
$$A_3(t) = x_1(t)e^{j\phi_{13}} + x_2(t)e^{j\phi_{23}} \tag{2.3}$$
$$A_4(t) = x_1(t)e^{j\phi_{14}} + x_2(t)e^{j\phi_{24}} \tag{2.4}$$

where ϕ's are the phase shifts for two particular beam steering angles.

To receive $X_1(t)$ and $X_2(t)$, we have to use the complex weights which are the complex conjugates of the weights that are used at the transmitter. After the complex multiplication and summation at the receiver, we get:

$$r_1(t) = A_1(t)e^{-j\phi_{11}} + A_2(t)e^{-j\phi_{12}} + A_3(t)e^{-j\phi_{13}} + A_4(t)e^{-j\phi_{14}} \tag{2.5}$$

$$r_2(t) = A_1(t)e^{-j\phi_{21}} + A_2(t)e^{-j\phi_{22}} + A_3(t)e^{-j\phi_{23}} + A_4(t)e^{-j\phi_{24}} \tag{2.6}$$

Substituting the previous four equations in these two equations and simplifying, we get:

$$r_1(t) = 4x_1(t) + x_2(t)[e^{j(\phi_{21}-\phi_{11})} + e^{j(\phi_{22}-\phi_{12})} + e^{j(\phi_{23}-\phi_{13})} + e^{j(\phi_{24}-\phi_{14})}] \tag{2.7}$$

$$r_2(t) = 4x_2(t) + x_1(t)[e^{j(\phi_{11}-\phi_{21})} + e^{j(\phi_{12}-\phi_{22})} + e^{j(\phi_{13}-\phi_{23})} + e^{j(\phi_{14}-\phi_{24})}] \tag{2.8}$$

These equations can be written as:

$$r_1(t) = 4x_1(t) + I_2(t) \tag{2.9}$$

$$r_2(t) = 4x_2(t) + I_1(t) \tag{2.10}$$

$I_2(t)$ is the interference for the first beam due to the second beam and $I_1(t)$ is the interference for the second beam due to the first beam.

This analysis explains how a transmitted signal $X_1(t)$ can be extracted from multiple signals by beamforming. However, we still have interference due to $X_2(t)$. Advanced beamforming techniques assign amplitude values to the complex weights as well as modifying their phases to maximize the contributions of each element while minimizing the interference from other transmissions [11].

2.2 Digital Beamforming

In traditional analog beamforming, each antenna element receives a signal. And the analog beamformers output the weighted sum of these signals, reducing the signal dimensionality from N to 1, where N is the number of antenna elements in the array. The RF signal at each antenna element is normally down converted and the phase is changed by using analog phase shifters. At this stage, all the phase shifted signals are summed. This beamformed signal is then down converted to baseband and successfully transformed into digital words.

This is a well proven technique, but only a limited number of beams (less than 10, in general) can be formed at the RF or IF stage with passive phased-array antennas and analog technology [9]. The disadvantages of this technique are related to difficult control of the side lobe level, high loss, absence of individual beam shape control, complex construction and corresponding heavy equipment.

Unlike analog beamforming, digital beamforming receiver digitize the received signals at the element level, then processes these signals in a special-purpose digital processor to form the desired beam(s). The two approaches are shown in Figure 2.4 [26].

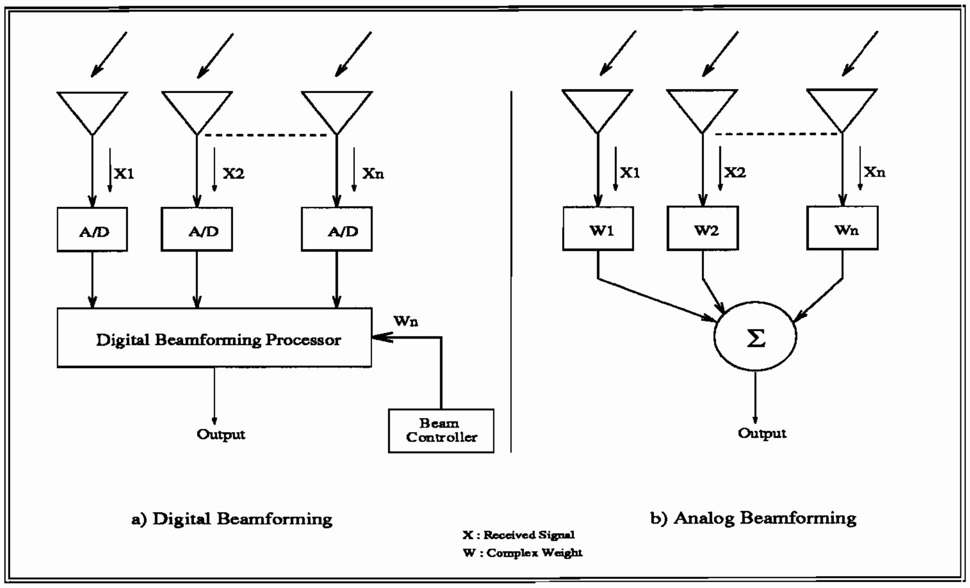

Figure 2.4: Digital & Analog Beamforming

2.2.1 Why Digital Beamforming ?

The main advantages of digital beamforming are:

- Once the physical input signals are properly digitized, they can be manipulated indefinitely without incurring any further degradation in signal-to-noise ratio (SNR) because a digital representation of the signals is used rather than the real received signal power and the SNR values have already been established in the low-noise amplifiers preceding the digital beamformer. So, any number of closed beams can be formed. The only limitation is the intensive digital processing required to do so [25].
- Separate control of each beam.
- Improved Adaptive Pattern Nulling : The antenna pattern can be shaped in an adaptive fashion giving rise to the concept of *smart antennas*. The antenna pattern can be adaptively shaped so as to place nulls in the direction of interfering radiation by using different weight sets (for the combination of antenna array elements). The weights control law involved in adaptive beamforming is quite complex, and can only be efficiently implemented in digital form [9].

2.3 Beamforming at the Baseband

In this work, DQPSK (Differential Quadrature Phase shift keying) modulation is implemented in conjunction with digital beamforming. A typical DQPSK system as shown in Figure 2.5 [17] and is briefly discussed in this section to help in understanding the beamforming concept at the baseband.

2.3.1 DQPSK modem

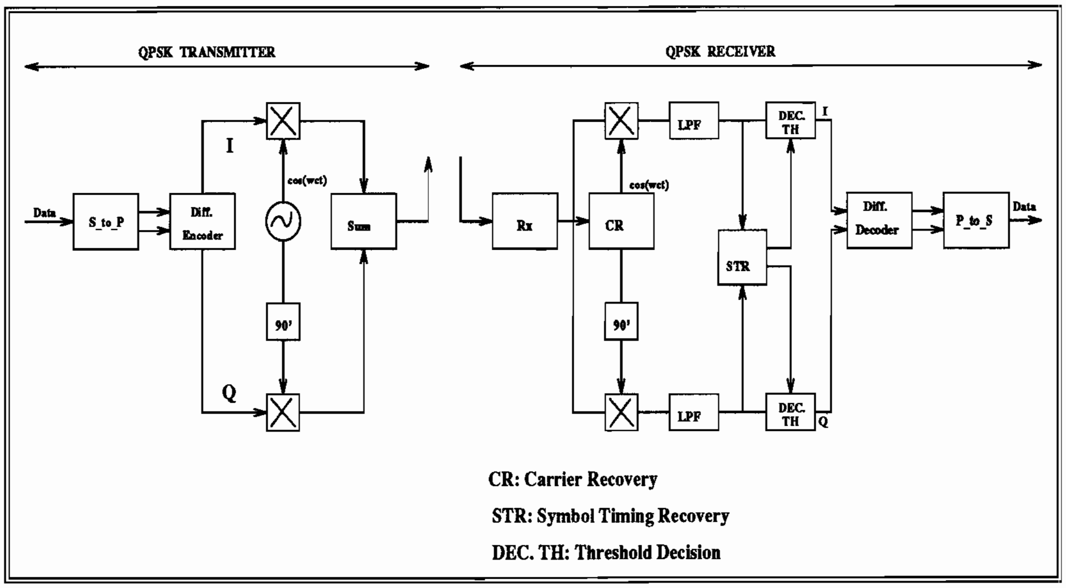

Figure 2.5: QPSK Transmitter and Receiver with Differential Coding

The incoming NRZ (non-return-to-zero) data stream entering the modulator is converted by a serial-to-parallel converter into two separate NRZ streams. One stream is in phase I(t) and the other is in quadrature Q(t), with a symbol rate equal to half of the incoming bit rate. A differential encoder, with a complementary differential decoder at the receiver, is used to remove the phase ambiguity, which may be introduced by the carrier recover circuitry. The output of this encoder, is applied to balanced multipliers. The second input to the I multiplier is $cos(\omega_t)$ and the input to the Q multiplier is the carrier signal shifted by exactly 90 degrees i.e., $sin(\omega_t)$. The outputs of both the multipliers are BPSK signals. The I multiplier output signal has phase 0 or 180 degrees relative to the carrier and the Q multiplier has phase 90 or 270 degrees relative to the carrier. The multiplier outputs are then summed to give a 4-phase signal. Thus QPSK can be regarded as two BPSK systems operating in quadrature.

The DQPSK signal at the modulator output is normally filtered to limit the radiated spectrum, amplified, then transmitted over the transmission channel to the receiver input. Because the I and Q modulated signals are in quadrature, the receiver is capable of demodulating and regenerating them independently of each other, operating effectively as two BPSK receivers.

The received signal is quadrature down converted to baseband frequencies. After conversion to baseband, each component passes through a matched filter (correlator), which essentially averages the signal over an entire symbol interval. This correlator output is passed to a decision device which estimates the symbol's value.

Looking into this QPSK modem, we observe that a QPSK signal can be represented as: $(I + jQ)e^{j2\pi f_c t}$. The first stage of the demodulator simply removes the

$e^{j2\pi f_c t}$ term from the received QPSK waveform. As usual, noise and interference from numerous sources can corrupt any received waveform and the correlators and the decision devices in the demodulator try to minimize this effect.

2.3.2 Beamforming at the baseband

Returning to the analysis previously done for the operation of the beamforming receiver we observe that the beamforming process is independent of the two received waveforms, $X_1(t)$ and $X_2(t)$. If we replace $X_1(t)$ and $X_2(t)$ with a QPSK waveform $(I + jQ)e^{j2\pi f_c t}$, the factor $e^{j2\pi f_c t}$ could be separated without affecting the beamforming operation. Since, this factor must be eventually removed through down-conversion to recover the I and Q data streams, we can remove this prior to beamforming. This eases the computational requirements on the digital beamforming processor. Thus, each antenna element's output can be down converted to baseband. The down converted baseband signal will have both I and Q components and can be treated as a complex signal (I+jQ). The phase of such signal can be altered using complex multiplication with complex weights corresponding to the beam direction in the beamforming processor. The second stage of the QPSK demodulation, correlators and decision devices, have to remain after the beamforming process.

2.3.3 Digital Beamforming Techniques

There are two technical approaches for digital beamforming differing in the scheme of digitization prior to beamforming.

1. Baseband Digitization [9]:

   As shown in Figure 2.6, this technique involves the analog down conversion of an IF signal to baseband using a pair of mixers to extract I and Q signals. These two baseband channels are then digitized by two A/D converters. The same process is to be done for all the antenna elements. Then, the baseband I and Q signals from all the antenna elements is given to the beamformer.

   The drawback of this approach is that the two A/D converters should sample the modulation envelopes at the same instant of time to ensure better matching of the phase and amplitude responses of the I and Q components.

2. Direct sampling and digitization of the IF signal:

   In this technique, a fast A/D converter is used to sample directly the IF waveform. IF subsampling techniques, discussed in the next chapter help in doing this. Mixing, low-pass filtering and other processing functions are then performed digitally. This approach offers better matching of the phase and amplitude responses of the I and Q components. With the advances in A/D converters, this approach is practically feasible and is used in this work.

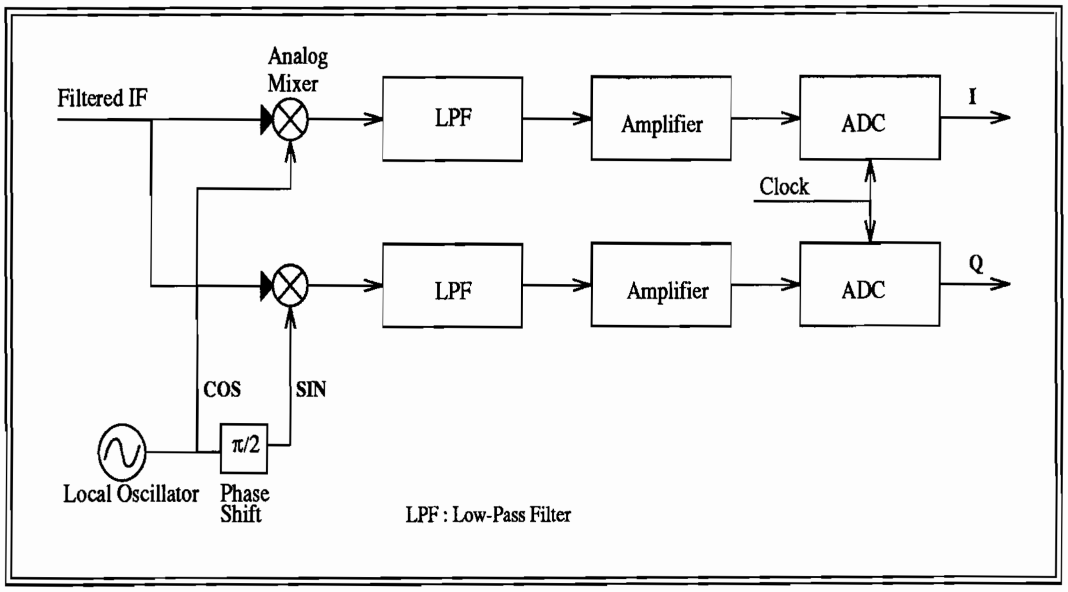

Figure 2.6: Baseband Digital Conversion in the receiving channel of a digital beamformer

2.3.4 Technical Challenges of Digital Beamforming

The main technical challenges in designing a digital beamforming receiver are:

- System complexity: As the number of antenna elements increase, the number of receiver sections needed increase. The performance requirements of the beamformer also increase in the same proportion. The system complexity also increases with the signal bandwidth and the number of simultaneous beams.
- Availability and cost of A/D converters.
- Throughput required for the digital beamformer.

The advances in VLSI CMOS digital technology promise advances in digital beamforming architectures.

Chapter 3

Digital Beamforming Receiver Design

The Digital Beamforming Receiver (DBF) has to perform two functions: Beamforming and QPSK Demodulation. In terms of hardware, both of these functions are more efficiently performed using digital signal processing hardware than analog hardware. Particularly, digital hardware offers flexibility to the receiver in directing its antenna array in many directions.

The block diagram of the digital beamforming receiver is shown in Figure 3.1.

3.1 Block Diagram

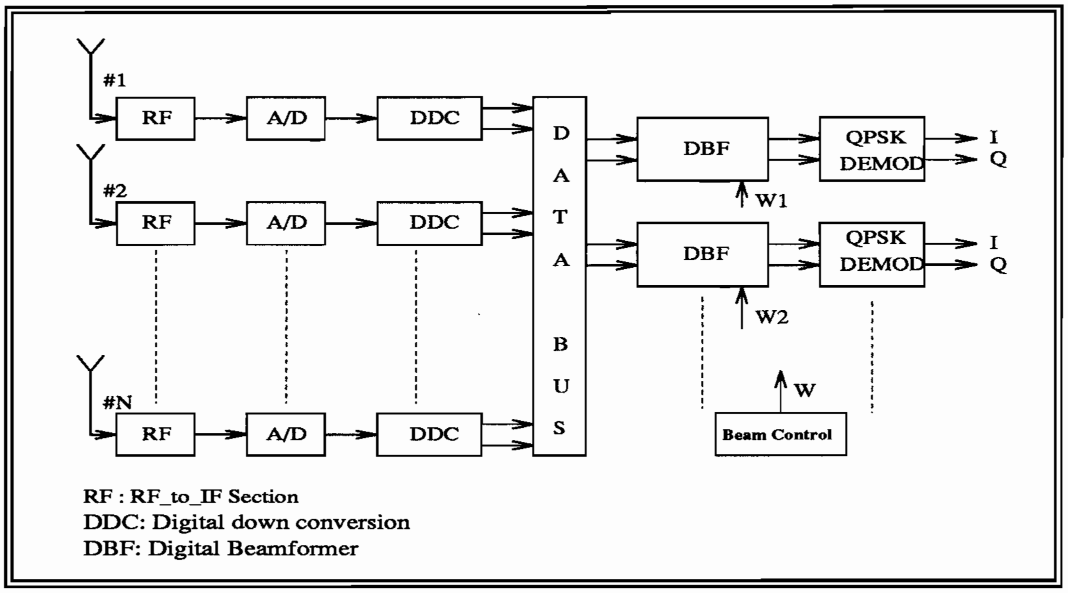

Figure 3.1: Block diagram of the digital Beamforming receiver

An N-element linear antenna array is considered. As discussed in the previous chapter, the received signal is digitized at the element level and there is one receiver section for each antenna element. After the input signals are down converted to baseband, they are processed in a digital beamformer to form a beam and then the signal is demodulated. Each block is now explained in detail.

3.1.1 RF section

The RF section is a superheterodyne receiver performing the following functions:

- Accurately translates the desired RF signal to a frequency band where processing required to extract the message is relatively cheap.
- Condition the signal (i.e., filtering and gain control) to remove adjacent channel interference while maintaining receiver sensitivity.

The superheterodyne principle is employed with the first IF is for image-rejection, and the second IF for selectivity. This principle is predominantly used in current receiver structures [15]. The IF frequency chosen for this application is 70 MHz.

3.1.2 A/D Section

The first task in a digital radio receiver is to digitize the analog received signal. We need one A/D converter for each antenna element. The A/D converter plays a key role because it determines the dynamic range of the receiver. When we digitize a signal we should guarantee that we can reconstruct the original signal from its digital representation. Here comes the role of Nyquist criterion.

The general Nyquist sampling theorem for sampling a band limited analog signal having no frequency components above f_{max} requires that the sampling rate be at least two times the highest frequency component of the signal $2f_{max}$. This ensures that the original signal can be reconstructed exactly from the digitized samples. This $2f_{max}$ is called Nyquist rate and sampling at rates greater than this is called oversampling. Oversampling eases the requirements on the anti-aliasing filter to be used when reconstructing the original signal. In our case, the IF signal typically will be centered around 70 MHz and has bandwidth around 5 MHz (data rate of 10 Mbps and QPSK modulation are assumed). Hence, to digitize this IF bandpass signal, we need a sampling rate greater than 140 MHz, irrespective of the small bandwidth of the signal. So not only we need an ADC (A/D converter) with this sampling frequency, but also we need to process its output at such a high rate. This seems to be a major hindrance for the digital radio design. But there is a solution for this problem which is known as *IF sub-sampling* [30][1].

This technique, if carefully used, allows sampling rates lower than $2f_{max}$ with an exact reconstruction of the information in the original signal. An ideal bandpass signal has no frequency components below f_l and above f_h. For this signal, the minimum sampling rate can be at least the twice of the bandwidth $(f_h - f_l)$ of the signal.

To ensure that spectrum overlap does not occur, the sampling frequency must satisfy the equation [30],

$$\frac{2f_h}{k} \leq f_s \leq \frac{2f_l}{k-1} \tag{3.1}$$

where k is restricted to integer values that satisfy,

$$2 \leq k \leq \frac{f_h}{f_h - f_l} \tag{3.2}$$

Bandpass sampling concept can also be used to down convert a signal from a bandpass signal at RF or IF to a bandpass signal at a lower IF. Since the bandpass signal is repeated at integer multiples of the sampling frequency, selecting the appropriate spectral replica of the original signal bandpass signal provides the down conversion function. This concept can be used to convert an IF signal to a lower IF simultaneously digitizing it.

This concept holds promise for radio receivers that digitize the signal directly at the IF. All this means is that ADCs with slower sampling rates (hence, potentially higher performance, lower power consumption, and lower cost) may be used. An important practical limitation, however, is that the ADC must still be able to effectively operate on the highest frequency component in the signal.

This kind of receiver is called a digital receiver and is characterized by the principle of using the inherent aliasing property of sampling to realize a down conversion. Thereafter, DSP techniques are used to extract the signal. The DSP techniques used include direct digital frequency synthesis (DDS), digital down conversion, high-speed digital filtering, and multi rate techniques such as decimation.

Chip details

The ADC input is centered around 70 MHz with 5 MHz bandwidth (10 Mbps data rate and QPSK modulation). A sampling rate of 25 MHz is selected that translates an image of the IF signal to a lower IF of 5 MHz. The frequency band of this signal is from 2.5 MHz to 7.5 MHz so the largest frequency in this signal is less than half of the sampling rate of 25 MHz and hence Nyquist criterion is maintained, as it is pointed out in chapter 3, that the ADC must still be able to operate at the highest frequency of the input signal.

The ADC used in the design is the HARRIS chip HI5703 [20]. HI5703 is a monolithic, 10-bit ADC and is designed especially for high speed applications where wide bandwidth and low power consumption are essential. It has got fully differential pipeline architecture with an internal sample and hold and provides a maximum sampling rate of 40 MHz. We are using this chip at a clock rate of 25 MHz. It also has got very good dynamic performance and consumes only 400 mW power at 40 MHz. Data output latches present valid data to the output bus with a latency of 7 clock cycles.

3.1.3 DDC Section

The digitized IF signal is now in the appropriate form to use digital signal processing techniques. So the next step is to perform frequency translation on the digitized IF signal to a complex baseband frequency. The complex baseband signal retains the magnitude and phase detail of the original real input signal at half the sample rate. Low-pass filtering and decimating this signal to a lower sample rate will produce a baseband signal. The Digital Down Conversion (DDC) process is shown in Figure 3.2 [29].

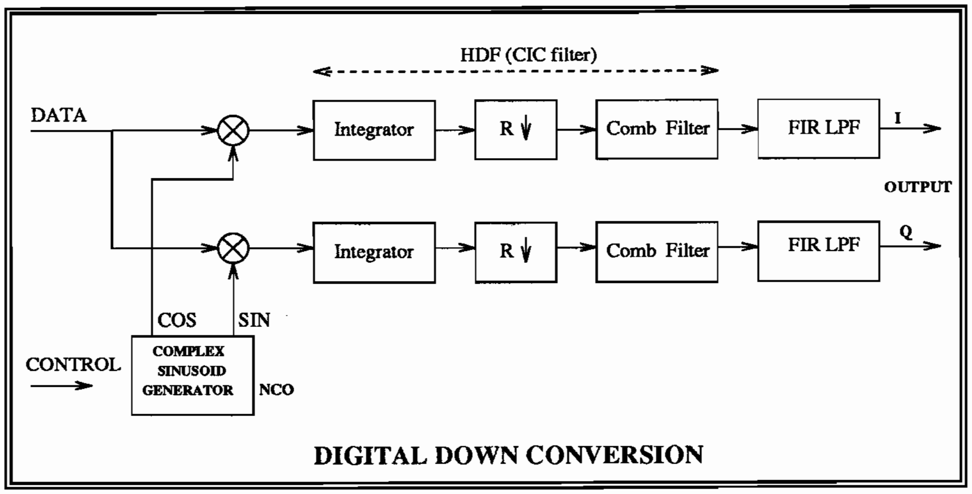

Figure 3.2: Digital Down Conversion block diagram

The major functional blocks of this section are :

1. **Numerically Controlled Oscillator (NCO):** The NCO implements the function of a local oscillator with programmable frequency and phase. At the heart of the NCO, there is a phase generator which increments its phase register on every clock cycle. This sets the phase angle to the sin/cos generator to be from 0 to 360 degrees. The frequency selectivity is set by the number of bits in the phase register. For the commercial HARRIS chips it is on the order of 0.012Hz [19]. This resolution in frequency control by the NCO offers excellent channel selectivity.

2. **Mixer & Down conversion :** In this process, the receiver needs to effect a frequency shift of the carrier frequency to baseband. To accomplish this, the real input signal is multiplied by the output of the NCO i.e., the phasor $e^{-j\omega_c}$, where w_c is the NCO frequency of the complex sinusoid. This complex phasor per Euler's formula can be written as [29]:

$$e^{-j\omega_c} = cos(\omega_c) - jsin(\omega_c) \tag{3.3}$$

Multiplication of the real input signal or data sequence $cos(\omega_k n)$ by this quadrature sinusoid using a digital multiplier will produce the sum and difference frequencies, u(n):

$$u(n) = 1/2[cos((\omega_k - \omega_c)n) + cos((\omega_k + \omega_c)n) \\ -j(sin((\omega_k + \omega_c)n) - sin((\omega_k - \omega_c)n))] \tag{3.4}$$

where ω_k and ω_c are the input data frequency and NCO frequency, respectively.

u(n) is a spectrally two-sided complex signal with the desired sideband at the baseband. Low-pass filtering will eliminate the high frequency component to yield a function:

$$v(n) = cos((\omega_k - \omega_c)n) + jsin((\omega_k - \omega_c)n) \tag{3.5}$$

Thus, the real input signal $cos(w_k n)$ has been down converted to baseband and is in quadrature format.

3. **DDC filtering:**

   Extraction of narrow band signals from wide band signal input in high-end applications require special filtering and sample rate reduction. Decimation is reducing the output sampling rate by ignoring all but every Mth sample. When a digital filter reduces the bandwidth of a signal of interest so the filter output is over-sampled and if the input sample rate is preserved, it is inefficient to compute outputs. Thus, there is a one-to-one correspondence between decimation rate and gain in computational efficiency. The reduction in sample rate for a narrow band signal has a noise reducing and an anti-aliasing effect. This filters the uniform spectral distribution of noise power and any out of band frequency components. The SNR improvements with this process is stated as [29]:

$$SNR\ improvement(dB) = 10\ log\frac{BW_i}{BW_o} \tag{3.6}$$

   where BW_i and BW_o are the input and output bandwidths, respectively.

   Filter requirements of linear phase and high decimation rates beyond 16000 make standard FIR filters with orders greater than 100,000 impractical, as they require many time consuming multiply and accumulate (MAC) operations.

   The DDC section circumvents this problem in both I and Q data branch by using a 2-stage filter architecture. It uses a high-decimation filter (HDF) for low pass filtering followed by a standard FIR (finite-impulse response) compensation filter for final wave shaping. The HDF is comprised of a 5-stage integrator, decimation register and a 5-stage comb filter. This kind of filters are called CIC (cascade-integrate-comb) filters and they lead to more economical hardware implementations. The economy of CIC filters derive from the following sources:

- no multipliers are required.
- no storage required for filter coefficients.
- intermediate storage is reduced by integrating at the high sampling rate and comb filtering at the low sampling rate, compared to the equivalent implementation using cascaded uniform FIR filters.
- little external control or complicated local timing is required.

Chip Details

The chip selected is HSP50110 (Digital Quadrature Tuner). The block diagram of this chip is shown in Figure 3.3. The chip can be configured over a general

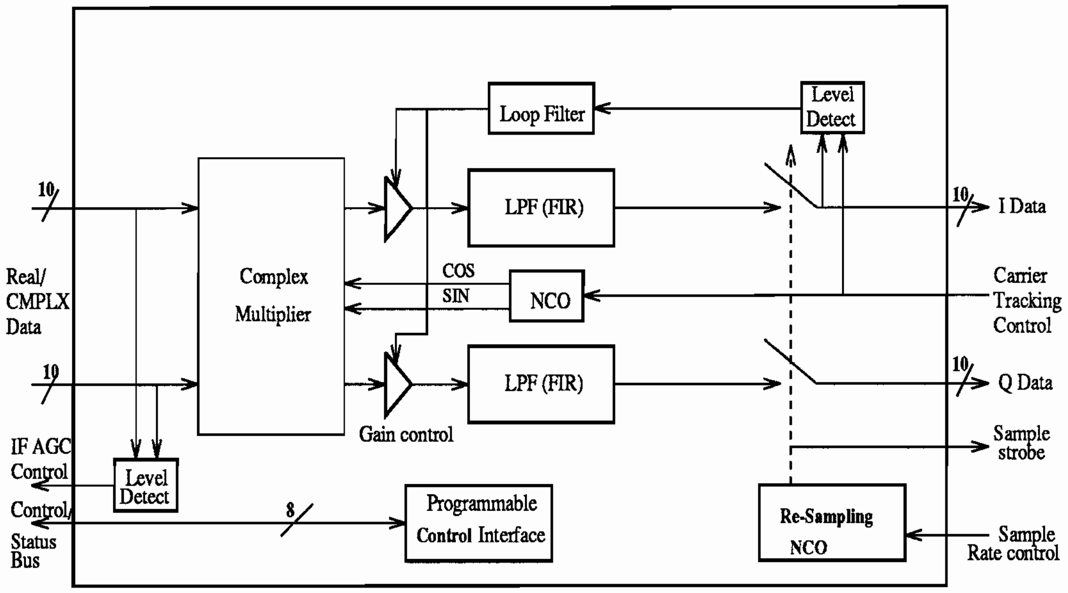

Figure 3.3: Block diagram of HSP50110 (Courtesy: HARRIS Semiconductor)

purpose 8-bit parallel bidirectional microprocessor interface. The programming of this chips involves writing data into nine 32-bit control registers to set the following parameters:

- local oscillator frequency.
- Low pass filter configuration.
- Re-sampler configuration.
- I/O control.

The microprocessor interface consists of a set of four 8-bit holding registers and one 8-bit address register. These registers are accessed via a 3-bit address bus (A0-2) and an 8-bit data bus (C0-7). The registers are loaded by setting up the address (A0-2) and data (C0-7) to the rising edge of WR (write signal). More details on timing constraints on programming this chip are found in [19].

3.1.4 Latches and Data Bus

If we see the hardware design of the beamformer (Figure ??), we have 8 ADC's and DQT's connected in parallel corresponding to the 8 antenna elements. Now, as the I and Q samples are at the baseband, we have to do the beamforming operation on them. For that, we have to treat I and Q data together as a complex number and then multiply that with a complex weight stored in the ROM. This is to be done for 8 data values for the 8 elements and then we have to add them all together to get the beamformed I and Q output signals. The CMAC to be used in the next stage is fast enough so that we can use a single CMAC for all these 8 complex multiplications. So we can latch all these I and Q outputs and then we have to take an (I,Q) complex number at a time and then multiply that with a complex weight in the ROM and then store the result in the accumulator of the CMAC. This is to be repeated for all the 8 (I,Q) signals from 8 elements. Also, the same 8 (I,Q) signals are needed to form other beams. Only the ROM contents will be different for each beam with the beamformers added in parallel sharing the same data bus. This encourages the use of a common data bus for I and Q data from the digital tuners. Moreover, additional HSP50110's can be added into the receiver should more antenna elements be desired.

So, each DQT's output is buffered with a bus interface latch. Each receiver section is enabled sequentially by a 3-to-8 decoder that is controlled by a 3-bit counter which provides the address of the receiver section. It is obvious that the clock that drives the counter should be 8 times the sample rate so that all 8 I,Q data values will be available on the bus during a single sample period.

3.1.5 Beamformer

The digital beamformer, a very fast digital processor, forms multiple beams by finding the product of the set of received samples from the array elements and the sets of weights that differently shape the beams originating from the antenna. The complex weights are determined from the beam location. The core of the beamforming operation is to do the complex multiplication and accumulation of the complex weights and the received samples as explained in the last chapter.

We considered three ways of implementing this task.

1. **DSP Processor [11]**

   If we are using an 8-element antenna array and our target data rate is 10Mbps, and QPSK modulation is being used, then the I and Q output samples are passed at a rate of 5 Msps (Mega samples per second) into the beamformer, as we are doing the beamforming at the baseband. In order to form one beam, we have to do 8 complex multiplications and each complex multiplication is equivalent to 4 real multiplications and 2 additions. Then, all these 8 complex products are to be added, which has 16 real additions. And hence, to form one beam, 32 real multiplications, 32 real additions to be done. That is, 32 real MAC operations are to be done in 1 symbol duration.

In addition, we need to read 16 real I and Q data in to the DSP processor and write the beamformed I and Q data to the next stage of the receiver. So in total, we have 50 operations (32 MACs, 16 read, 2 write operations) to be done in one symbol duration i.e., 200 ns.

Even if we assume that a DSP processor has a performance of 1 instruction per clock cycle, we need 50x5=250 MHz clock frequency for the DSP processor to form just 1 beam. This requirement gets linearly multiplied by the number of beams we want to receive. A DSP system that operates at such high speeds is not realizable without using a multi-processing configurations, which in turn, increases the complexity of the system.

2. **FPGAs**

   In general, FPGAs (Field Programmable Gate Arrays) provide higher speeds than DSP processors and lower costs than an ASIC for moderate volume applications and more flexibility than the alternate approaches.

   For our purpose, we need 32 real MAC operations per one symbol duration i.e., 200 ns. That means, we need one real MAC operation in 6.25 ns. So, if we use two real MAC units, we can form one beam and the design problem is to design a real MAC unit with a worst case delay of 12.5 ns. It has been reported that using pipelined array architectures [6], real Multiply and Accumulator (MAC) units have been designed on XC4000 series with a worst case delay of 12.1 ns. Moreover, it has been suggested that two such MAC units can be accommodated in one single XC4013 FPGA [5].

3. **Special Purpose CMACS**

   A feasible solution can be arrived at using high-speed ASICs (Application Specific Integrated Circuits) that perform CMAC (Complex Multiply and Accumulate) operation within one clock cycle. As we need 8 CMAC operations in 1 symbol duration, the ASIC has to operate at a minimum of 40 MHz. ASICs are available that operate above 50 MHz and hence this seems to be a cost-effective approach for the design. To form B number of beams, we require the same number of CMAC ASIC chips. In this work, this approach is used.

   The chip used is HSP45116A. HSP45116A is actually a high performance quadrature numerically controlled oscillator and modulator with an accumulator. We can use this chip as a high speed 16-bit CMAC (Complex Multiplier and Accumulator) [24]. The chip interfacing in the receiver is shown in Figure ??. This chip is used as a CMAC operating at 40 MHz clock. It takes two complex inputs. Each complex input includes a real and an imaginary component. While the first complex input being clocked into the HSP45116A through the parallel input ports XRe and XIm, the second complex input vector is being clocked from a single input port Y, by clocking one complex component at a time. The implication is that the clock of the second complex vector must be twice of the clock for the first complex vector. So the baseband samples are to be clocked in at 80MHz.

3.1.6 QPSK Demodulator

The final stage is a standard QPSK demodulator. It's input is the I and Q output from the beamformer. The HARRIS chip, HSP50210 (DCL) performs many of the baseband processing tasks required for the demodulation of BPSK, QPSK and OQPSK (Offset-QPSK) waveforms. These tasks include matched filtering, carrier tracking, symbol synchronization, AGC (Automatic Gain Control), and soft decision slicing. It also has a NCO and a complex multiplier so that we can, in fact, give a QPSK bandpass signal to it. But, we have already done the downconversion in HSP50110 and hence, this section has to be bypassed.

Chip details

The DCL processes the In-phase(I) and quadrature-phase(Q) components of a baseband signal which have been digitized to 10 bits. The functional block diagram of this chip is shown in Figure 3.4 [23].

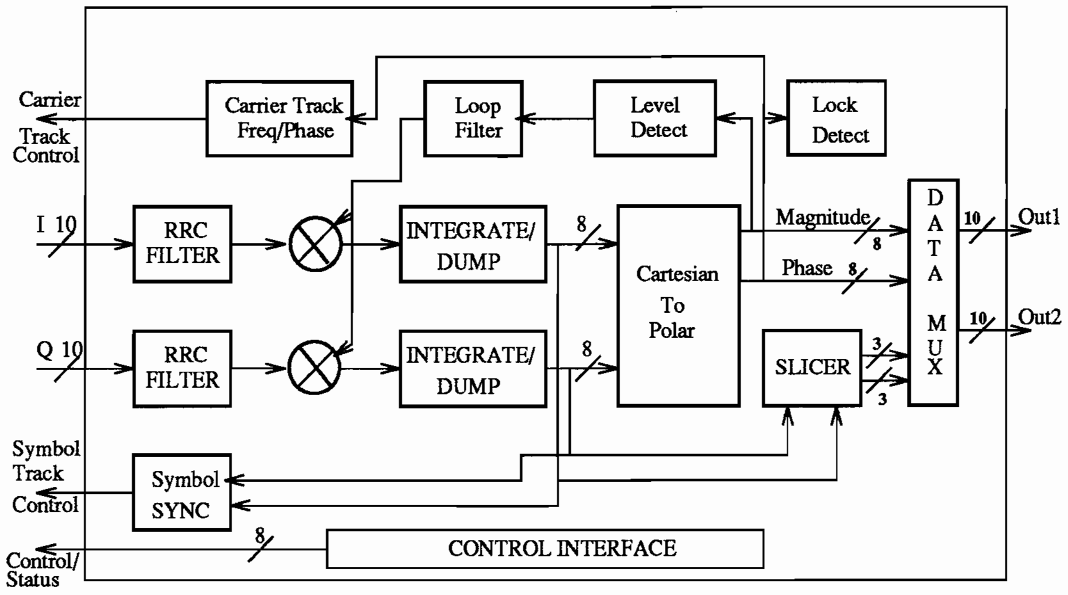

Figure 3.4: Block diagram of HSP50210 (Courtesy: HARRIS Semiconductor)

The major functions we are using are:

- Matched Filtering:

  The DCL provides two selectable matched filters. They are Root raised Cosine Filter (RRC) and an Integrate & Dump (I&D) filter. The RRC filter is provided for shaped pulses and the I&D filter is for square wave data. The filters may be cascaded for better adjacent channel rejection for square wave data. As an additional feature, it provides a means to bypass these filter filters if they do not meet the baseband filtering specifications, and use external filters such as HSP43168 Dual FIR filter or the HSP43214 serial I/O filter, still using the same DCL. This is possible by setting the desired filter configuration in the data path configuration register. The sample rate of the baseband input depends on the symbol rate and filtering configuration chosen. The I&D filter itself decimates the input sample rate down to two samples per symbol. This configuration supports decimation factors of 1 (by-passing the filter), 2, 4, 8, 16, 32. This corresponds to the data rates of 10 Mbps, 5 Mbps, 2.5 Mbps, 1.25 Mbps, 625 kbps and 312.5 kbps, respectively.

But, in the case of 10 Mbps, we really have only one sample per one symbol. And hence, the performance will be poor for this case. But, the speed of the CMAC restricts the use of sampling frequency more than 5 Msps for the output of DQT. So, this design can be used for the data rate of 1.25 Mbps with satisfactory performance as we have 8 samples per one symbol in this case, to make a decision.

- Carrier and Symbol Tracking:

  The carrier tracking loop removes the frequency and phase uncertainties in the carrier due to local oscillator inaccuracies and doppler. The symbol tracking loop removes the frequency and phase uncertainties in the data and generates a recovered clock synchronous with the received data. Each loop consists of an error detector, a loop filter and a frequency or gain adjustment/control. These loops are external to DCL and are closed around the baseband filtering to center the signal in the filter bandwidth. In our application, the loops are closed through a serial interface around HSP50110 (DQT).

- Soft Decision Slicing:

  The soft decision slicer encodes the I,Q data end-symbol samples into 3-bit soft decisions. The input to the slicer is assumed to be a bi-polar (2-ary) baseband signal representing the encoded values of either '1' or '0'. The MSB of the 3-bit soft decision represents a hard decision with respect to the mid point between the expected symbol values. The 2 LSBs represent a level of confidence in the decision. They are determined by comparing the magnitude of the slicer input to multiples (1x,2x and 3x) of a programmable soft decision threshold.

3.1.7 Programming

The chip can be configured over a general purpose 8-bit parallel bidirectional microprocessor interface. The programming of this chips involves writing data into 31 32-bit control registers to set the following parameters:

- Matched filter configuration.
- Carrier tracking loop configuration.
- Symbol tracking loop configuration.
- I/O control.

The microprocessor interface is similar to that of DQT and consists of a set of four 8-bit holding registers and one 8-bit address register. These registers are accessed via a 3-bit address bus (A0-2) and an 8-bit data bus (C0-7). The registers are loaded by setting up the address (A0-2) and data (C0-7) to the rising edge of WR (write signal). More details on timing constraints on programming this chip are found in [23].

3.2 Scalability Of the Receiver Design

Scalability is the right touchstone for the efficiency and effectiveness of a beamforming receiver design. This problem is three dimensional in this case as we have to consider this issue in three cases.

1. **Increase in Antenna Elements:** If we increase the number of elements in the receiving antenna array, we have to increase the number of digital receiver sections (RF/IF section, A/D section, DQT section) to be used. As we have used the data bus, to pass the I and Q data from the receiver section to beamformer section, this is easy. However, it increases the amount of data we are getting at the input of the beamformer. And hence, the number of complex multiplications we have to do in each symbol interval increases for the same data rate. It demands extra performance from beamformer. For example, we calculated earlier that for an 8-element case and at 10 Mbps data rate, we require a CMAC which operates at 40 MHz. Now, if we go for 10-element case, we require the CMAC to operate at 50 MHz. This is easy as our CMAC can operate at this speed. However, if we increase the number of elements to 16, we need 80 MHz speed for the CMAC. This may be difficult. However, we can use two CMACs in parallel to serve the purpose.

   This is illustrated in Figure 3.5.

2. **Increase in Data Rate:** If we increase the data rate, the situation again is the same in that the amount of data we get to the beamforming section increases. And hence, the performance requirements of the beamformer increases. The performance of the beamformer can be characterized in terms of the number of complex multiplication and accumulations to be done per second. Our prototype system can handle data rates of 1.25 Mbps and it can easily be scaled to accommodate higher data rates. For example, to design a system which can operate uptp 2.5 Mbps with 8-element antenna array, we can just

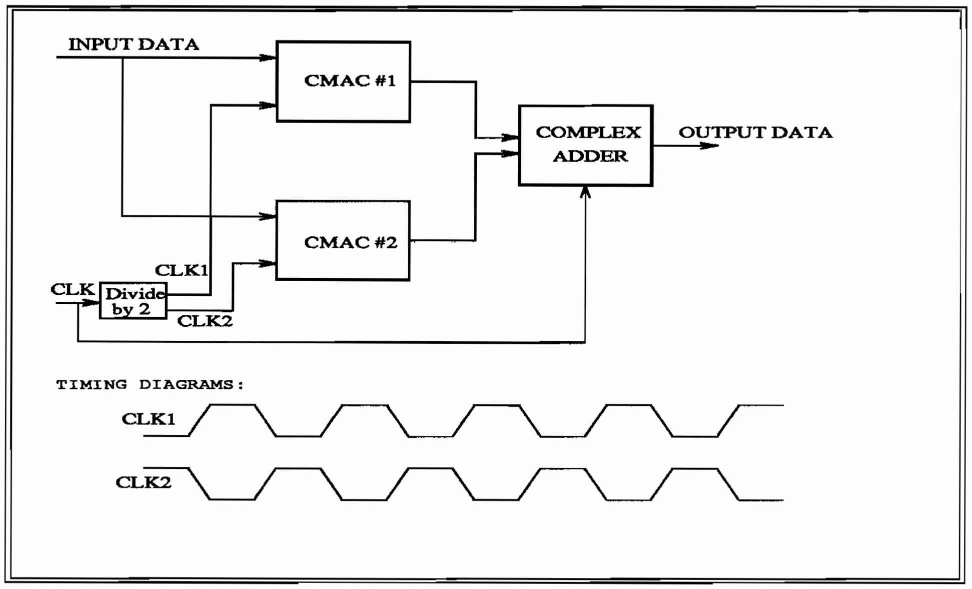

Figure 3.5: Beamforming Receiver Scalability: Using 2 CMACs in parallel

use the approach described earlier, that is using two CMACs with the same specifications in parallel for each beam.

3. **Increase in the number Of beams:** As the number of beams to be received increase, we just need to add each beamformer consisting of a ROM (to store weights) and a CMAC, in parallel. As we have used data bus to transfer the data from the DQT's to the beamformer, the system is inherently scalable with the number of beams.

Chapter 4

Implementation

In this chapter, the functionality of the design proposed in the previous chapter is validated by designing & testing each individual section of the beamforming receiver.

4.1 A/D Section

An evaluation board (HI5703-EV) is used for the purpose of digitizing the incoming IF signal at 70 MHz. It uses HI5703 10-bit A/D converter which can operate upto 40 MHz. It includes clock driver circuitry, reference voltage generators, and the analog input drive circuits. To verify the correct operation of the A/D converter, a 10-bit D/A is included in the test setup. In the test setup as shown in Figure 4.1, the A/D output is given to the D/A input. The various control settings on the evaluation boards are set for the proper digitization/reconstruction of the input signal. The D/A board used was the evaluation board from HARRIS semiconductor, DACRECON-EV, which uses a 12-bit D/A converter, AD9713B.

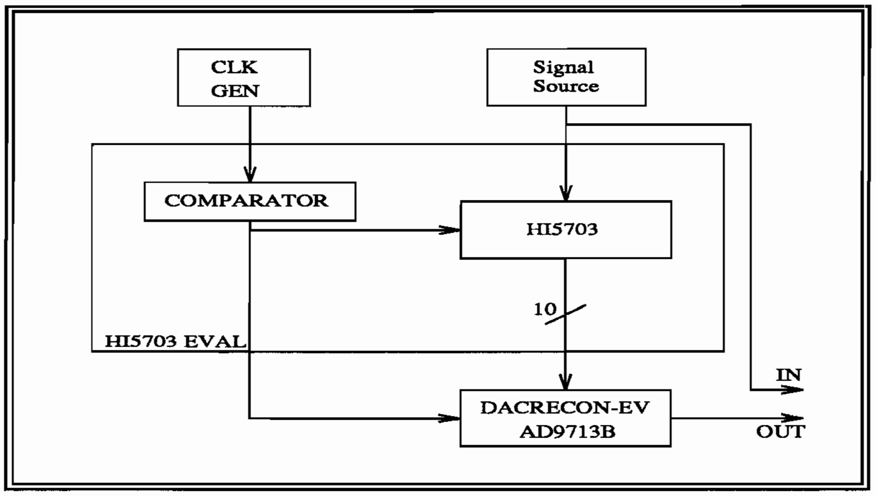

Figure 4.1: A/D, D/A test setup

4.2 Prototyping of a Reconfigurable Digital Receiver

A reconfigurable radio is prototyped on APTIX rapid prototyping board in order to validate the design of the demodulator section using HARRIS chips. A 70 MHz BPSK modulated IF signal is given as the input to the receiver. This is first digitized using the A/D board designed in the first step. The next step is to build a receiver section, which digitally down converts the IF signal to baseband, decimates the output and demodulates it to recover the original data.

4.2.1 Design

The receiver is designed using HARRIS chips HSP50110 (DQT) and HSP50210 (DCL). The functions of these chips are explained in chapter 4. The reconfigurable digital radio design is shown in Figure 4.2.

The major functional blocks in the design are:

1. A/D : The board designed in section 5.1 is used.

2. Programming section : (XC4013 and 27C64)
   The HSP50110 and HSP50220 chips are to be programmed for the particular specifications of the receiver by loading the registers in these chips via the microprocessor interface. The main programmable parameters of the receiver are:

   - Local Oscillator (HSP50110): The center frequency and the phase offset of the local carrier can be programmed. The carrier frequency can be adjusted dynamically to account for the carrier phase offset effects by connecting the COF/COFsync output of the HSP50210 to the COF/COFsync input of the HSP50110. The width of the COF is to be selected.
   - Input Level Detector Threshold (HSP50110): This is used for IF AGC control. The level detector output is to be externally averaged to set the gain of an amplifier in front of the A/D which closes the AGC loop.
   - Baseband AGC (HSP50110): The level of the Mixer output is gain adjusted by baseband AGC around the Low pass Filtering. This baseband AGC provides the coarse gain correction necessary to help maintain the output of the HSP50110 at a signal level which maintains an acceptable dynamic range. The programmable parameters of the AGC are:
     - AGC level detector threshold.
     - Loop filter lower limit and upper limit.
     - Loop gain.
   - Low Pass Filter configuration (HSP50110): The low pass filter can be configured as a 1-stage or 3-stage CIC filter.

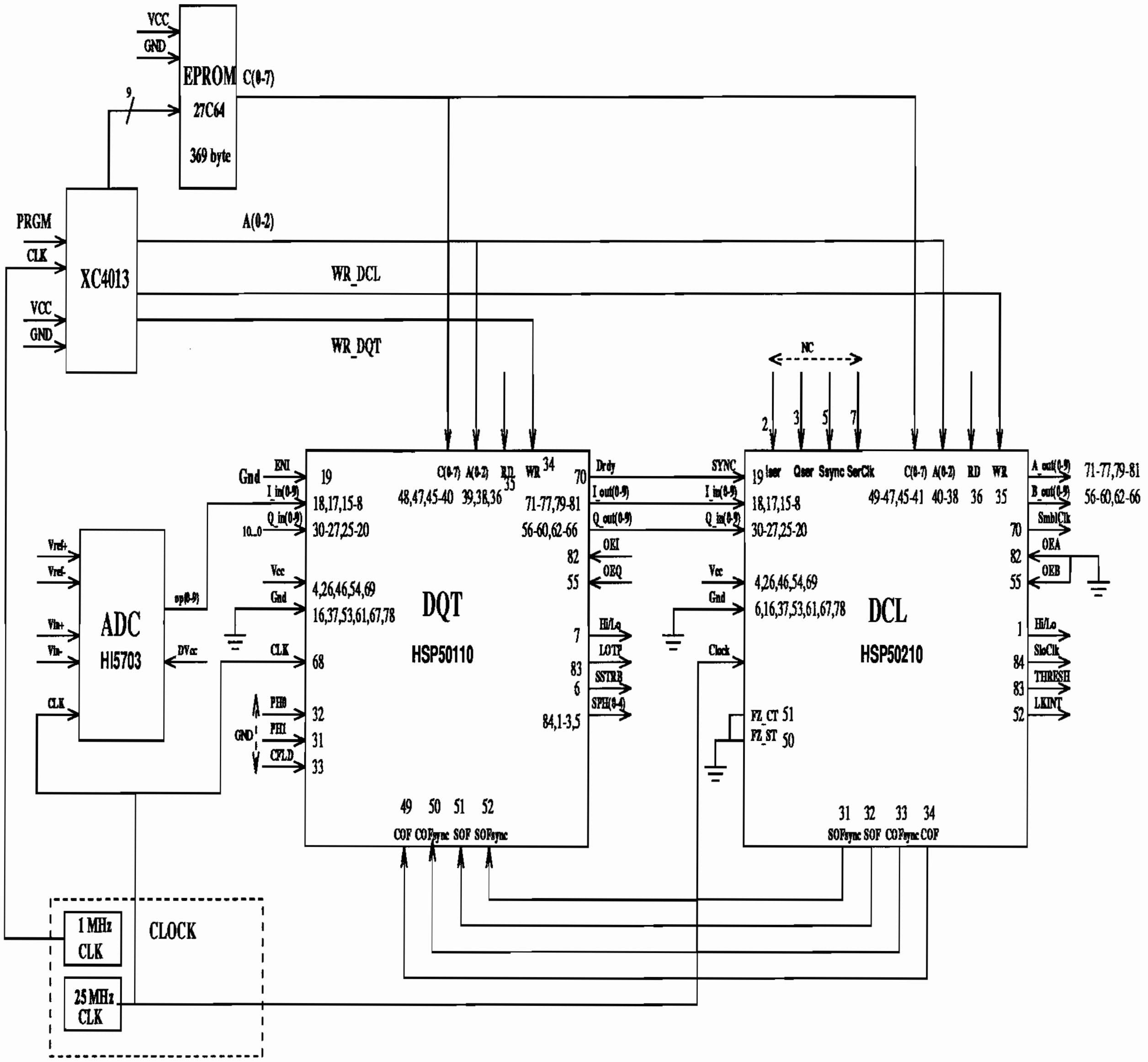

Figure 4.2: Reconfigurable Digital Receiver

- Re-sampler configuration (HSP50110): The re-sampler sets the output sample rate by controlling the sample rate of the decimation filters. The output sample rate can be adjusted dynamically to synchronize with baseband waveforms. This is done by connecting the SOF/SOFsync output of the HSP50210 to the SOF/SOFsync input of the HSP50110. The width of the SOF is to be selected.
- I/O control (HSP50110): Input controller operates in gated input mode with ENI (#19) tied at logic '0' and hence the input samples will be read at the rate of CLK (25 MHz).
- Matched Filtering (HSP50210): We can use Raised Root Cosine (RRC) or Integrate & Dump (I/D) filter for matched filtering. The number of samples to be integrated per symbol can also be programmed.
- Baseband AGC (HSP50210): This is used to maximize the dynamic range, adjust for signal to noise variations and maintain an optimal signal level at the input to the soft decision slicer. This provides smooth AGC whereas HSP50110 provides coarse AGC. The parameters are:
  - AGC level detector threshold.
  - Loop filter lower limit and upper limit.
  - Loop gain.
- Symbol & Carrier Tracking Loop (HSP50210): The tracking loop order, bandwidth and damping ratio need to be configured.

The HARRIS DEMODEVB [22] software is used to generate the configuration data for both DQT and DCL. This configuration data is stored in an EPROM (27C64). The programming logic given in Appendix C, loads the configuration data into the HSP50110 and HSP50210 chips, supplying the necessary control signals. This section is designed using XC4013 FPGA and an AM27C64 EPROM. The logic is designed using XLS [1] [28]. The receiver programming logic is shown in Figure 4.3. More details on the timing requirements for loading the Harris chips is given in [19][23].

3. HSP50110/210: The interconnection of HSP50110 and HSP50210 are shown in Figure 4.2. The chips are to be programmed once before the normal operation to configure the receiver to the given specifications.

4.2.2 Aptix Rapid Prototyping Board

Field Programmable Circuit Boards (FPCBs)[2] provide an ideal vehicle for rapid system prototyping by utilizing Field Programmable Interconnect Components

[1]Xilinx Logic Synthesizer (XLS) is a tool developed at University Of Kansas for programming Xilinx FPGAs.

[2]FPCB, FPIC are registered trademarks of Aptix corporation

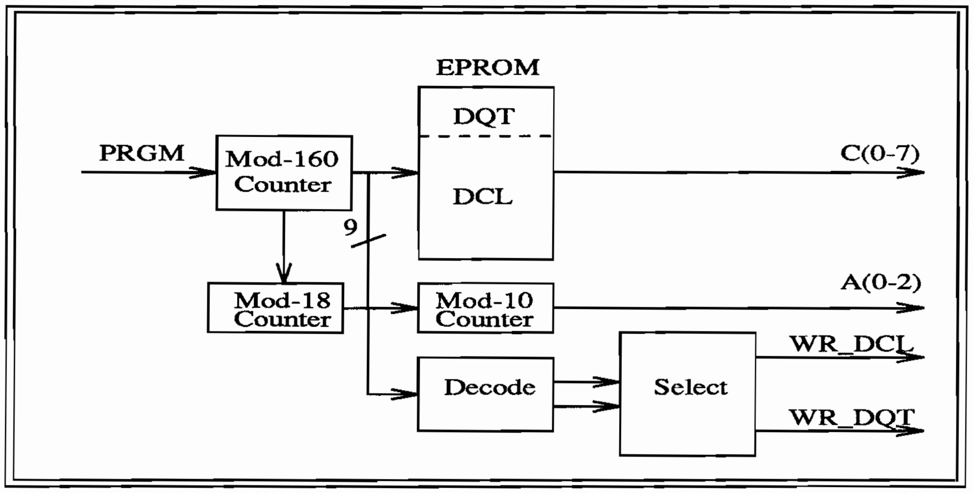

Figure 4.3: Receiver Programming Logic

(FPICs). FPIC devices provide the high density interconnect architecture and performance and make programmable interconnect a reality. Thus, the FPCB can be used as a programmable bread-board where the interconnections between various components of a system are done by using the FPIC devices. Programming FPIC devices is handled through a connector on the FPCB that interfaces to a Workstation via a cable and host interface module (HIM).

FPIC devices deliver high speed I/O interconnects as fast as 5 ns. Interconnect programming elements based on CMOS SRAM technology allow them to be quickly reprogrammed. FPIC/D is a type of FPIC device which has a diagnostic cable providing an observability-window for selectively accessing any of the signals which are routed through the device. This provides software-driven debugging capability by allowing real-time observability of all design signals. Hence, it takes less time to test a particular design.

The Aptix board used in this work is MP3 board, which is designed for the verification of complex digital systems. The MP3 supports up to three AX1024AD and/or AX1024AR FPIC devices.

4.2.3 Design Flow

The design flow for prototyping the reconfigurable digital receiver on APTIX board is shown in Figure 4.4.

A top-level netlist file (given in Appendix B of [10]) for the receiver design is first written in XNF[3] format showing the components that are used in the design and the interconnections between them as shown in Figure 4.2. The A/D output is to be given to the FPCB via IO-FPGA. The logic is designed for IO-FPGA and is simulated in Quicksim-II[4]. The logic for the programming FPGA is designed and simulated in Quicksim-II. The bit files for configuring the IO-FPGA and the

[3]XNF stands for Xilinx Netlist Format

[4]Quicksim is a tool from Mentor Graphics to simulate the logic in VHDL.

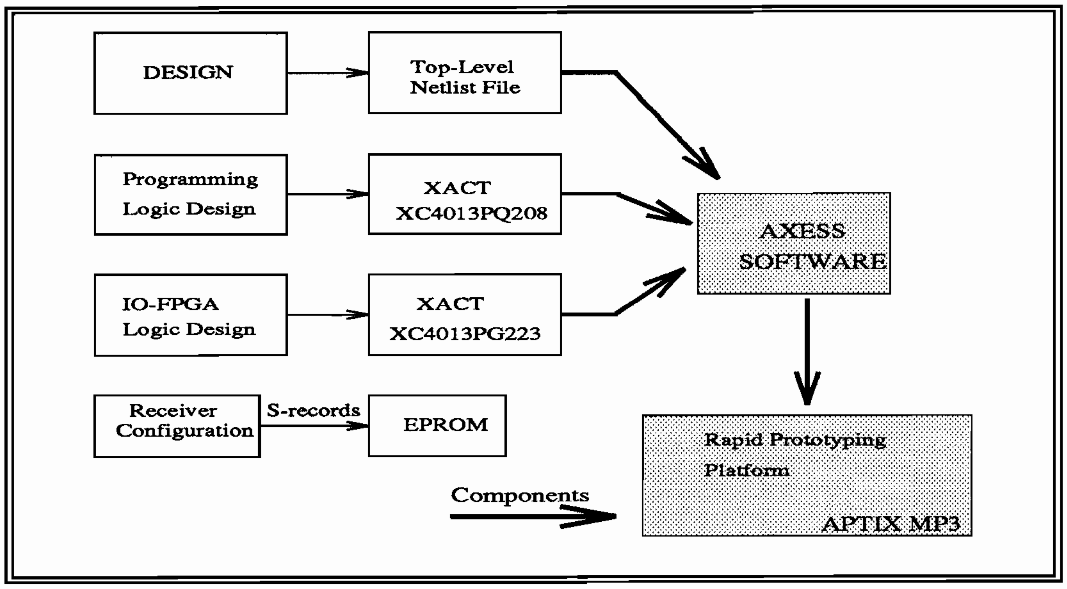

Figure 4.4: Design Flow

programming FPGA are then generated by using makebits tool from XILINX. The receiver configuration is stored in an EPROM.

To test the design, the top-level netlist file is given to the AXESS[5] software using which we can place the components and do the routing. The receiver setup on the APTIX board is shown in Figure 4.5. The FPIC devices are programmed to connect the components.

4.2.4 Results

The configuration of the receiver is given in Appendix C.1 of [10].

The corresponding S-records to program the EPROM are given in Appendix B. The test setup of the receiver is shown in Figure 4.5. The BERT (Bit error rate tester) is used to generate a pseudo random bit sequence which is BPSK modulated by the RDRN transmitter. The RDRN transmitter IF card output is at 70 MHz. This is given as input to the receiver. That is subsampled by HI5703 A/D board and the digital output is given to the digital demodulator on the APTIX board. The HSP50110 downconverts the IF signal to baseband digitally and this signal is demodulated in HSP50210. The output bit stream is again routed back to the BERT to compare with the input data.

Before operating the receiver, the receiver has to be programmed. A switch (PRGM) is provided to initiate the programming operation. To verify the programming operation, there are three probe signals in the design. They have to be monitored for correct operation of the receiver after programming. They are:

- LOTP (Local Oscillator Test Point) : This is the pin 83 of HSP50110. The

[5] AXESS is the software interface to APTIX MP3 prototyping board.

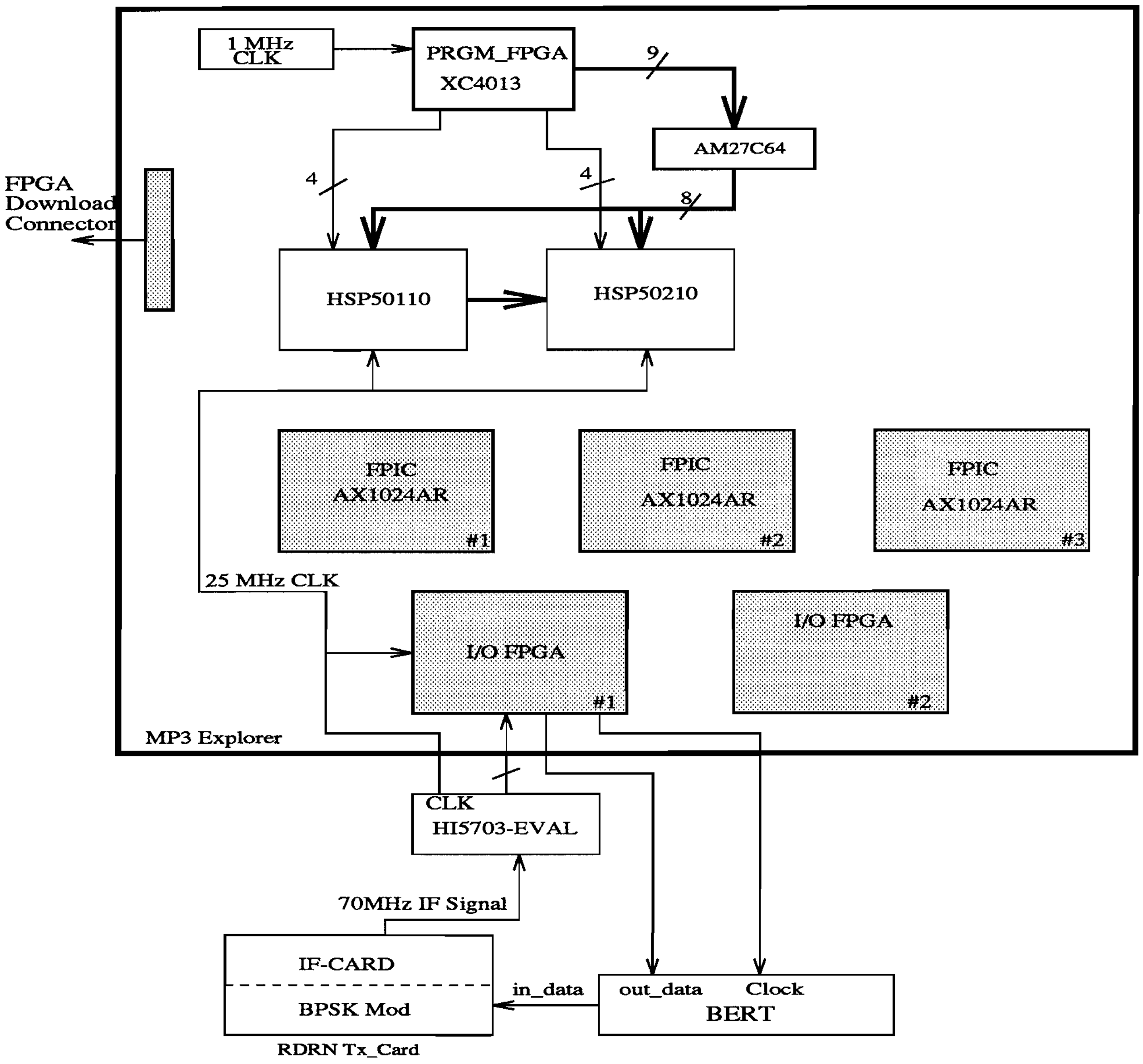

Figure 4.5: Receiver Setup On The Aptix Board

frequency of the signal at this point must be equal to the local oscillator frequency (5MHz).

- DQT output rate : This is the pin 70 of HSP50110. The frequency of this signal must be equal to the DQT output rate (2 MHz).
- DCL Symbol Clock : This is the pin 70 of HSP50210. This is the output symbol clock. The signal at this pin should be of 50% duty cycle clock (1 MHz).

In the testing, the programming is done properly and the receiver is correctly demodulating the BPSK modulated IF signal. The bit rate used was 1 Mbps.

4.3 Digital Beamforming Receiver

Having tested all the individual blocks of the beamforming receiver, a complete design of an 8-element digital beamforming receiver is presented in this section.

4.3.1 Parts List

Table 4.1 gives the list of parts.

PART	Manufacturer	Quantity
HI5703-EV	HARRIS SEMICONDUCTOR	8
HSP50110	HARRIS SEMICONDUCTOR	8
HSP50210	HARRIS SEMICONDUCTOR	1
HSP45116A	HARRIS SEMICONDUCTOR	1
XC4013PG223	XILINX	1
AM27C64	AMD	1
XC3195A	XILINX	1
22v10B	Lattice	1

Table 4.1: List of parts for the beamforming Receiver

4.3.2 Schematics

Figure 4.6 shows the 8 receiver sections comprised of an RF section, HI5703 (ADC) and HSP50110 (DQT). The exact pin connections for each HSP50110 is similar to that shown in Figure 4.2.

Each receiver section consists of an RF section which down converts the RF signal to IF signal. The HI5703 digitizes the IF signal at 25 MHz. The digitized samples are given to HSP50110 operating at 25 MHz. The output of HSP50110 is the I and Q baseband samples of the received signal. All these signals are tied together to form a bus and the output is given to the next section. The OUTPUT

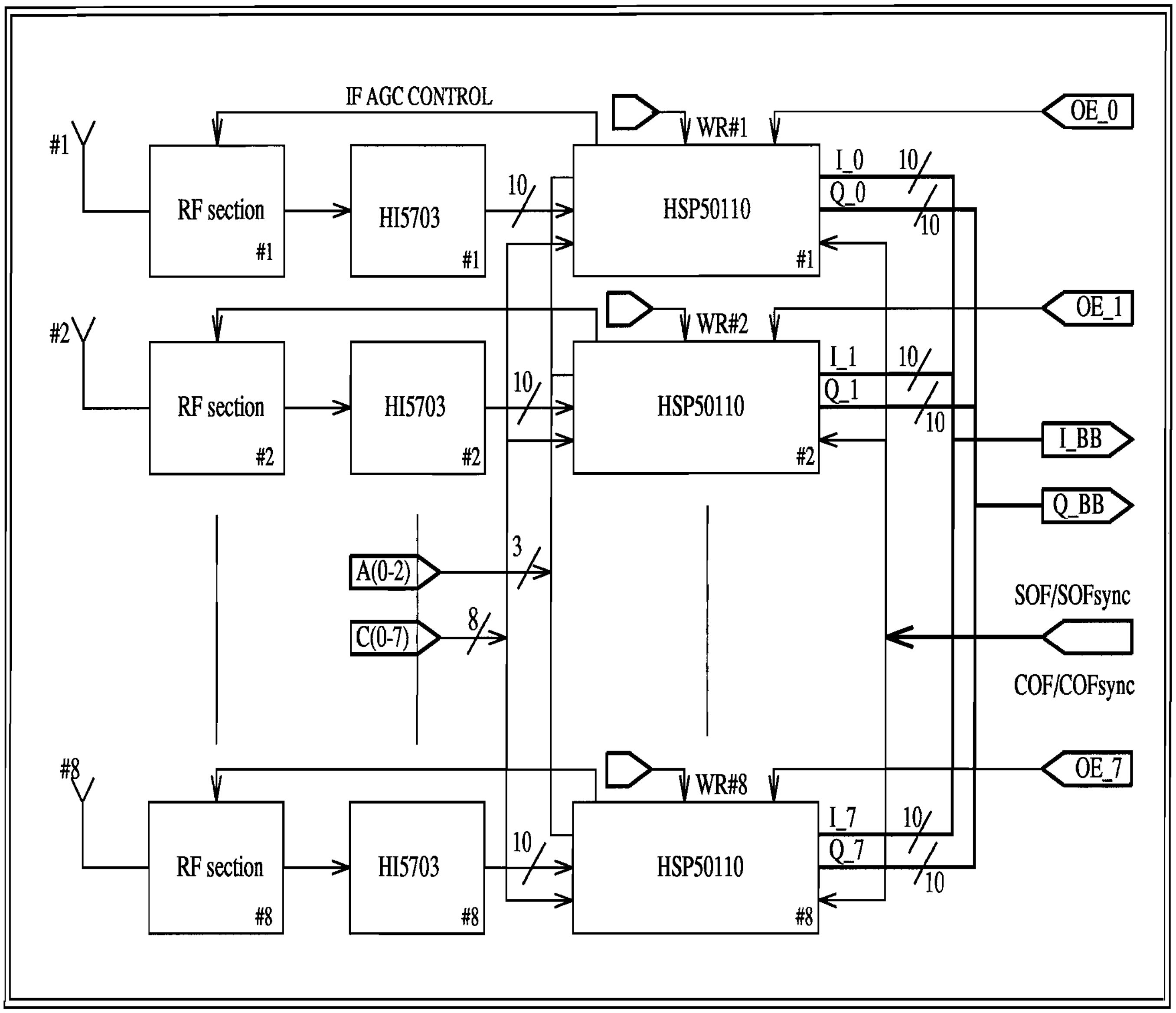

Figure 4.6: DBF Receiver-1

ENABLE (OE) signal which is provided from the next section selects the output from a single HSP50110, tri-stating all the other outputs. The HSP50110s are programmed using the programming section shown in Figure 4.7.

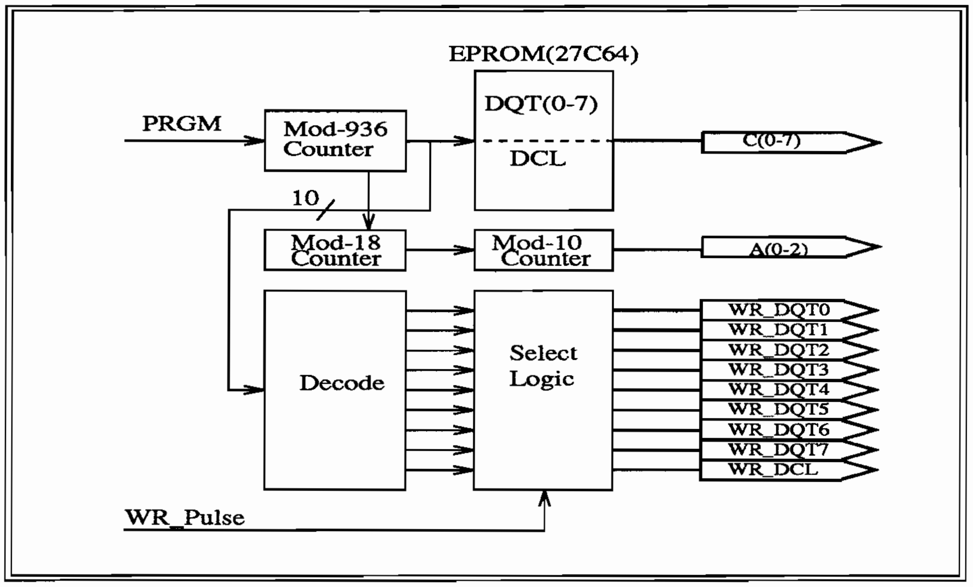

Figure 4.7: Programming Logic Design For DBF Receiver

Figure 4.8 shows the beamforming and demodulator section. The dotted portion of the Figure 4.8 is designed to fit in a Xilinx FPGA XC3195A. It provides the OUTPUT ENABLE (OE) signals to HSP50110s. The HSP45116A chip operating at 40 MHz does the complex multiplication and accumulation of this data with the complex weights stored in sequential logic. The PAL logic to control the 10-bit multiplexer operates at 80 MHz.

4.3.3 Directions for future implementation

To build the complete beamforming receiver, following directions are given.

- Prepare the schematics with the help of Figures 4.6, 4.8, 4.7 and 4.2, depending on the number of elements used and the number of beams to be used.

- Build more A/D, D/A boards for each antenna element. More documentation on building this can be found in the Wireless Communications and Digital Signal Processing Lab.

- Build the RF_to_IF section to downconvert the RF signal to 70 MHz. The design used in the RDRN receiver can be used for this purpose.

- A PCB layout is to be made from the schematics developed.

- Programming Logic Design:

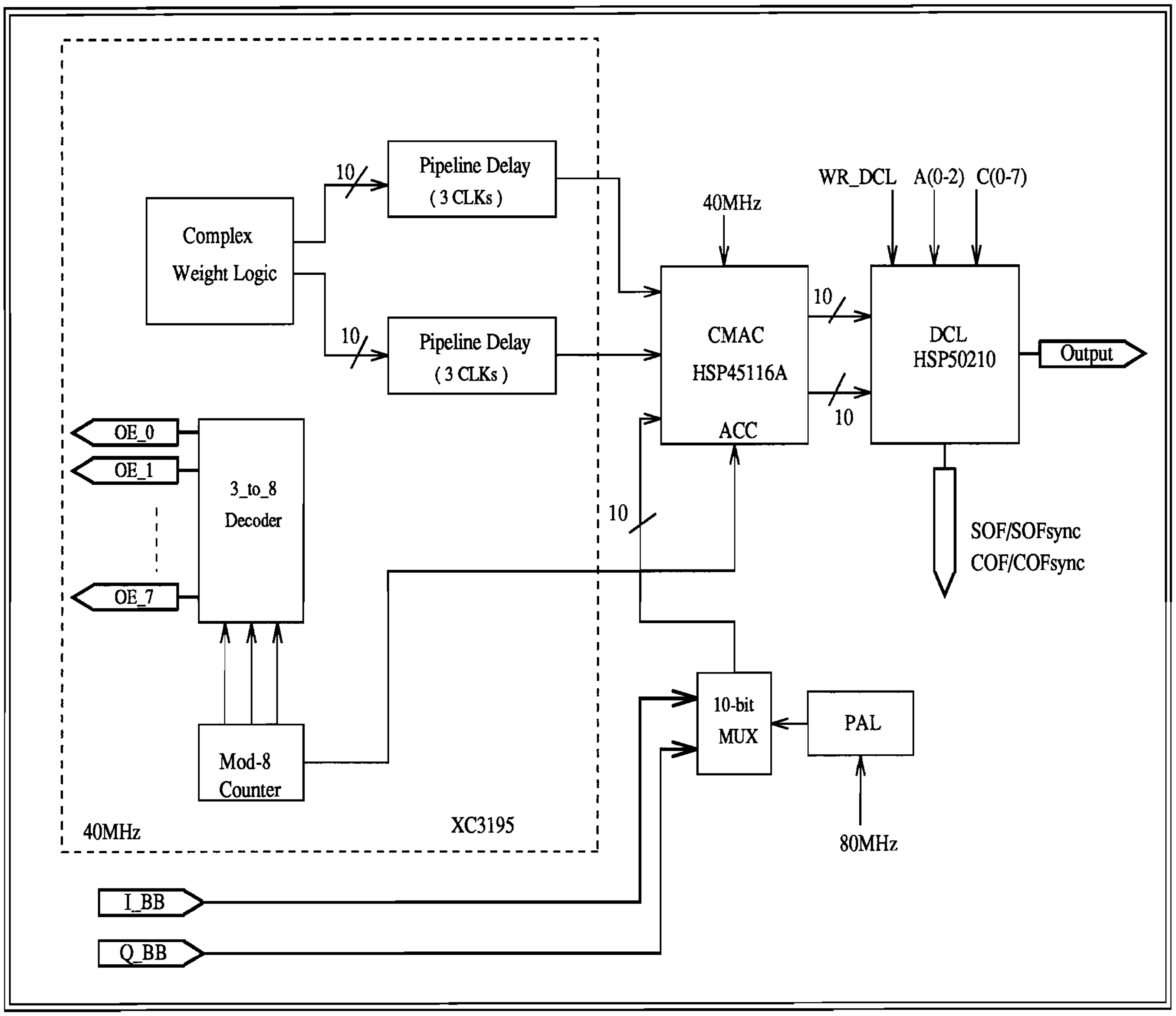

Figure 4.8: DBF Receiver-2

1. For programming the 8 HSP50110's and 1 HSP50210, use the programming logic given in Appendix D. First see the logic output in Quicksim.
2. For more details on programming, refer to Figures 23 and 26 in HSP50110 data sheet.
3. To use a low-cost memory, the programmable logic is designed for 1 MHz clock. This clock should be derived from the global clock (80 MHz).
4. The setup and hold times for the control signals WR and A(0-2) and C(0-7) are to be verified.
5. A switch (PRGM) is provided for programming the receiver. This is to give a rising edge to the programming logic to initiate the programming.
6. When storing the receiver configuration in the EPROM, follow the directions given in Table 5 of the HSP50110 data sheet in page 5-67. The LSB 8 bits (7-0) are to be stored in holding register # 0. Check the programming logic output in Quicksim to see the signals A(0-2), WR and COUNT (EPROM address). The output of the EPROM of C(0-7). So, in timing analysis, EPROM memory access time is to be included.
7. In designing the programming logic, after loading one 32-bit register, 4 clock cycles are left to ensure proper loading of the register as required in Figure 23 of the HSP50110 data sheet. This is to be observed.

- In the beamforming section, the complex weights are to be stored in some form of memory. We need a memory of 8 10-bit complex words. This memory needs to be accessed at 40 MHz. Instead of going for fast memory chips, a simple logic is designed in XC3195A to store the components. A mod-8 counter is designed to represent the 8 different complex words. A karnagh-map can be made for each bit of the complex word to do logic reduction.

- The complex weights are to be represented in digital form. The real and imaginary values vary from -1 to 1. So, in binary-offset format, 0000000000 represents -1.00, 1000000000 represents 0.0 and 1111111111 represents 1.00.

- After programming is done, verify the programming operation by monitoring the following probe signals.

  1. pin 83 of HSP50110 : The frequency of this signal must be equal to 5 MHz corresponding to the local oscillator frequency of the digital downconverter.

2. pin 70 of HSP50210 : This signal must be 1 MHz, 50% representing the output symbol clock.

- The beamforming logic is designed to fit in one XC3195 FPGA. It does the decoding and pipeline delay functions necessary to do prior to beamforming.
- For more number of beams, same FPGA can be used. To form more than one beams, we just need to add the CMAC and HSP50210 chips in parallel. The logic to incorporate more number of beams can be implemented in this FPGA. Also, when we increase the number of elements, the logic can be changed in this FPGA along with the addition of the receiver sections, avoiding the change of physical hardware.

Chapter 5

Conclusions

5.1 Summary of Results and Conclusions

An architecture for a digital beamforming processor is presented which incorporates the concept of doing beamforming at the baseband. Various approaches for the hardware implementation are considered. A scalable and flexible digital hardware design is proposed. The receiver is capable of receiving more than one beam. The design has got the advantages of low complexity and low cost which have been the two most required features in beamforming receivers. The receiver can be built with commercially available VLSI integrated circuits.

The individual sections of the beamforming receiver are designed. The designs are validated by prototyping on APTIX MP3 rapid prototyping board. To verify the functionality of the receiver section, the primary section of the beamforming receiver, a reconfigurable digital receiver is designed and implemented. The performance of the receiver is completely evaluated and then the design is extended for 8 such receiver sections in the beamforming receiver.

The beamforming section of the receiver includes a CMAC ASIC and the control logic. The control logic is designed and tested on the Aptix prototyping board.

5.2 Suggestions for Future Work

Future directions include the practical implementation of the complete beamforming receiver. In this work, the architecture is proposed and the hardware design is presented along with the validation of the design by testing the primary blocks of the receiver. The work presented here concentrated on the design starting from IF stage. The analog RF front end for each receive section is to be built. It is a standard superheterodyne type receiver and can be built from commercially available RF components.

To build the compete beamforming receiver, 7 more copies of A/D sections and the receiver sections are to be built. The logic is already designed and tested. A printed circuit-board (PCB) has to be made with all the digital hardware on it.

This work used an A/D evaluation board in implementation. The beamforming receiver uses 8 such A/D converters. Hence, it would be efficient to put 8 A/D converters on a single PCB in order to reduce the size and complexity of the system. All the eight A/D converters and the 9 HARRIS ICs have to operate at a single clock of 25 MHz. Hence, proper clock driving circuitry has to be inserted on the PCB design.

The design proposed in this work uses CMAC ASICs for doing the beamforming. We have also highlighted the importance and ways of doing beamforming in field programmable gate arrays (FPGAs), which offer flexible and low cost solution. It is worthwhile to investigate into efficient multiplier implementations on FPGAs.

The work presented here concentrates on designing a hardware system which forms more than one beam in different directions, given the corresponding complex weights. In practice, the real importance of the system comes when we have an adaptive beamforming receiver, which adaptively controls the locations of the beams. So, the advances in adaptive beamforming techniques might also influence the applications of this work.

Finally, reducing the size of beamforming receiver is very much desired keeping in mind the number of components in the system. The complexity of the digital beamforming receiver is inherent in the concept itself to have different receiver sections for different antenna elements. An idea of integrating the complete system into one chip (probably an ASIC) would be of enormous interest in future generation wireless communication systems.

Bibliography

[1] Carl Andren and John Fakatselis. Digital IF Sub Sampling Using The HI5702, HSP45116 and HSP 43220. Technical Report AN9509.1, Harris DSP and Data Acquisition, April 1995.

[2] Ayman F Naguib and Arogyaswamy Paulraj. Receursive Adaptive Beamforming For Wireless CDMA. *Proceedings Of IEEE*, pages 1515–1519, 1995.

[3] Stephen F. Bush, Sunil Jagannath, Ricardo Sanchez, Joseph B. Evans, Victor Frost, and K. Sam Shanmugan. Rapidly Deployable Radio Networks (RDRN) Network Architecture. Technical Report 10920-09, Telecommunications & Information Sciences Laboratory, July 1995.

[4] C R Ward and J E Hudson and J G Searle. Smart Antenna Solutions For Mobile Radio Systems. *9th International Conference On Antennas and Propagation*, 1:344–347, ICAP 1995.

[5] Chi-Jhi Chou, Satish Mohanakrishnan, and Joseph B Evans. FPGA Implementation Of Digital Filters. Online version available at http://-www.tisl.ukans.edu/∼ evans/fpga.

[6] Joseph B Evans. Efficient FIR Filter Architectures Suitable For FPGA Implementation. Online version available at http://www.tisl.ukans.edu/-∼ evans/fpga.

[7] Benjamin Ewy, Craig Sparks, Joseph Evans, Victor S Frost, and Glenn Prescott. A Flexible Beamforming Digitally Synthesized IF Modulator. Technical Report 10920-10, Telecommunications & Information Sciences Laboratory, July 1995.

[8] Benjamin Ewy, Craig Sparks, K.Sam Shanmugan, Joseph Evans, and Glenn Prescott. An Overview Of The Rapidly Deployable Radio Network Proof Of Concept System. Technical Report 10920-16, Telecommunications & Information Sciences Laboratory, July 1995.

[9] Alfonso Ferina. *Digital Beamforming Antennas*. Prentice Hall, New Jersey, second edition, 1993.

[10] Srikanth Gurrapu. Digital Beamforming Receiver Design. *Masters Thesis, University Of Kansas, Available in Telecommunications & Information Sciences Laboratory*, May 1997.

[11] Srikanth Gurrapu, Harvind Samra, Glenn Prescott, and K.S.Sam Shanmugan. Digital Beamforming Receiver Architecure. Technical Report 10920-24, Telecommunications & Information Sciences Laboratory, October 1996.

[12] Wookom Lee and Raymond L Rickholtz. Maximum Likelihood Multi User Detection with Use Of Linear Antenna Arrays. Technical report, Department Of Electrical Engineering and Computer Sciences, George Washington University.

[13] M A Beach and H Xue and J P McGeehan. Adaptive Antenna Technologies For Third Generation Systems. *9th International Conference On Antennas and Propagation*, 1:344–347, ICAP 1995.

[14] M A Beach and R L Davies and P Guemas and H Xue and J P McGeehan. Capacity and Service Extension For Future Wireless Networks Using Adaptive Antennas. *9th International Conference On Antennas and Propagation*, 1:125–129, ICAP 1995.

[15] Heinrich Meyr and Ravi Subramanian. Advanced Digital Receiver Principles and Technologies for PCS. *IEEE Communications Magazine*, pages 68–78, January 1995.

[16] Ayman F Naguib. Adaptive Antennas For Wireless CDMA Networks. *PhD Dissertation, Stanford University*, August 1996.

[17] Theodore S Rappaport. *Wireless Communications-Principles and Practice*. Prentice Hall, New Jersey, first edition, 1995.

[18] Zhigang Rong, Theodore S Rappaport, Paul Petrus, and Jeff H Reed. Simulation of Multitarget Adaptive Array Algorithms for Wireless CDMA Systems. Technical report, Mobile and Portable Research Group, Virginia Polytech Institute & State University.

[19] Harris Semiconductor. HSP50110: Digital Quadrature Tuner Datasheet. October 1995.

[20] Harris Semiconductor. HI5703: 10-bit, 40MSPS A/D Converter Datasheet. August 1996.

[21] Harris Semiconductor. HSP45116A: Numerically Controlled Oscillator/Modulator Datasheet. May 1996.

[22] Harris Semiconductor. HSP50110/210 Evaluation Board: Digital Costas Loop Datasheet. March 1996.

[23] Harris Semiconductor. HSP50210: Digital Costas Loop Datasheet. March 1996.

[24] Harris Semiconductor. Using HSP45116A as a Complex Multiplier and Accumulator. Technical Report TB317, May 1996.

[25] Hans Steyskal. Digital Beamforming Antennas: An Introduction. *Microwave Journal*, 11:107–124, January 1987.

[26] Hans Steyskal. Digital Beamforming At Rome Laboratory. *Microwave Journal*, pages 100–124, February 1996.

[27] S C Swales, M A Beach, and D J Edwards. Adaptive Antenna Arrays For Future Generation Cellular Communication Networks. *Sixth International Conference On Antennas and Propagation*, 301:321–325, ICAP 1989.

[28] Venugopal Thammanna and Satish Mohanakrishnan. Xilinx Logic Synthesizer. *Telecommunications & Information Sciences Laboratory, University Of Kansas*, May 1995.

[29] Jeffery A Wepman. Considerations In The Development Of A Low Cost, High Performance Receiver Based On DSP Techniques. *DSP Applications*, pages 15–28, December 1993.

[30] Jeffery A Wepman. Analog To Digital Converters And Their Applications In Radio Receivers. *IEEE communications magazine*, pages 39–45, May 1995.

Appendix A

MATLAB code for generating complex weights

```
% Code for generating complex temps.

clear;
N = input('number of elements in the array  ');
B = input('number of beams to be formed  ');
for I = 1:B,
        input('Steering angle, theta  ');
        Theta = ans*(pi/180);
        A = input('Signal Amplitude  ');
        phi = (0.5*sin(Theta))*(2*pi);
        for H = 1:N,
 temp = (N+1)/2;
 phase(H, I) = (temp-H)*phi;
 weight(H,I) = A*cos((temp-H)*phi)+j*A*sin((temp-H)*phi);
        end;
end;
```

Appendix B

S-Records To Store Digital Receiver Configuration In An EPROM

```
-- S-Records for the Receiver design : radio
-- Time Stamp :  03-15-97.

S1130000003333333300000000000047E17A14010069
S1130010000000002600000200000000200C0C017B
S113002003000000008303000004000000001A0025
S113003020000500000000F83F000006000000005A
S1130040000000000700000000000000000800009D
S1130050000030852600000000000034000000018C
S1130060000000003030680F0200000000010000B2
S11300700003000000000000000004000000000075
S11300800000000500000000610000000600000000
S1130090000A0000000700000000A8A44E000800A9
S11300A0000000585BB1FF0900000000002C0100B3
S11300B00A00000000EEA900000B00000000C0C010
S11300C096050C00000000000000000D0000000078
S11300D0080100000E00000000169F02000F00003F
S11300E00000EA60FDFF10000000004E6D010011E9
S11300F000000000EC690200120000000000000093
S1130100001300000000FE79010014000000006CE0
S1130110E1ABFA1500000000CDFCCDFC1600000098
S113012000903000001700000000000000001800DC
S11301300000000000000019000000002000000082
S11301401A00000000660200001B00000000070007
S113015000001C00000000000000001D0000000062
S1130160000000001E00000000000000001F00004E
S1130170000000000000000000000000000000007B
S9030000FC
```

Appendix C

Programming Logic Design for the Receiver

```
// To program HSP50110 and HSP50210.
// Output Signals are: C(0-7), A(0-2), WR
// Input is, PRGM

#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <stdio.h>
#include "lca.h"
#include "header.h"
#include "clkbuffer.h"

int main(void)
{
       lca pr3("pr3","4013PQ208-4");
       net NIL;
       char* NILS="\0";
       net Clock("Clock",'L','C');
       net p2b("p2b");
       clkbuffer globclk(Clock,p2b,"BUFGP_BR",NILS);

// Assigning inpads and outpads.
createiob("inpad1","P6","PRGM",NILS ,NILS ,NILS,NILS ,NILS );
createiob("outpad1","P10",NILS, "A0",NILS, NILS, NILS,NILS);
createiob("outpad2","P16",NILS, "A1",NILS, NILS,NILS,NILS);
createiob("outpad3","P12",NILS, "A2",NILS, NILS,NILS,NILS);
createiob("outpad4","P18",NILS,"q1",NILS,NILS,NILS,NILS);
createiob("outpad5","P34",NILS,"q2",NILS,NILS,NILS,NILS);
createiob("outpad6","P36",NILS,"q3",NILS,NILS,NILS,NILS);
createiob("outpad7","P21",NILS,"q4",NILS,NILS,NILS,NILS);
createiob("outpad8","P39",NILS,"q5",NILS,NILS,NILS,NILS);
createiob("outpad9","P23",NILS,"q6",NILS,NILS, NILS,NILS);
createiob("outpad10","P41",NILS,"q7",NILS,NILS,NILS,NILS);
createiob("outpad11","P43",NILS,"q8",NILS,NILS,NILS,NILS);
createiob("outpad12","P28",NILS,"q9",NILS,NILS,NILS,NILS);
createiob("outpad13","P32",NILS,"WR_DQT",NILS, NILS,NILS,NILS);
createiob("outpad14","P30",NILS,"WR_DCL",NILS, NILS,NILS,NILS);

// Probe Signals.
createiob("outpad15","P59",NILS, "q0",NILS, NILS,NILS , NILS);
createiob("outpad16","P72",NILS, "enb",NILS, NILS, NILS, NILS);
createiob("outpad17","P198",NILS, "s1",NILS, NILS, NILS, NILS);
createiob("outpad18","P196",NILS, "en_5",NILS, NILS,NILS , NILS);
createiob("outpad19","P61",NILS, "done",NILS, NILS,NILS , NILS);

// State Machine Logic For Iniating the Programming.
createclb("clb1","AA","s1=(PRGM*PRGM)","s2=(s1*s1)",NILS,"ffx",
"ffy","Clock",NILS);
createclb("clb2","AB","start=(s1*(~s2))","enb=(start+((~start)*
(~done)*enb))",NILS,"ffx","ffy","Clock",NILS);
createclb("clb4","BA","q0=((~q0)*enb)","q1=((q1@q0)*enb)",NILS,
"ffx","ffy","Clock",NILS);
createclb("clb5","BB","q2=((q2@(q1*q0))*enb)","q31=(q2*q1*q0)",
NILS,"ffx",NILS,"Clock",NILS);
createclb("clb6","BC","q3=((q3@q31)*enb)","q41=(q3*q2*q1*q0)",
NILS,"ffx",NILS,"Clock",NILS);
createclb("clb7","BD","q4=((q4@q41)*enb)","q5=((q5@(q4*q41))*enb)"
,NILS,"ffx","ffy","Clock",NILS);
createclb("clb8","CA","q61=(q5*q4*q41)","q6=((q6@q61)*enb)",NILS,
NILS,"ffy","Clock",NILS);
createclb("clb9","CB","q7=((q7@(q6*q61))*enb)","q81=(q61*q6*q7)",
NILS,"ffx",NILS,"Clock",NILS);
createclb("clb10","CC","q8=((q8@q81)*enb)","q9=((q9@(q8*q81))*enb)
",NILS,"ffx","ffy","Clock",NILS);
createclb("clb11","CD","count1=((~q8)*q7*q6*(~q5))",
"count2=(q4*q3*q2*q1)",NILS,NILS,NILS,"Clock",NILS);
createclb("clb13","DA","done=(count1*count2*q9)",NILS,NILS,
"ffx",NILS,"Clock",NILS);
createclb("clb18","DB","decode1=((q7*q5)+(q7*q6))",
"decode=(decode1+q8+q9)",NILS,NILS,"ffy","Clock",NILS);
createclb("clb19","DC","WR_DQT=(en_5*(~decode)*(~q0)*enb)",
"WR_DCL=(en_5*decode*(~q0)*enb)",NILS,"ffx","ffy","Clock",NILS);

// MOd-10 Counter.
```

```
createclb("clb20","DD","A0=((A0@q0)*en_a*en_5)","A1_1=(A0*q0)",
NILS,"ffx",NILS,"Clock",NILS);
createclb("clb21","EA","A1=((A1@A1_1)*en_a*en_5)","A2_1=(A1*A0*q0)",
NILS,"ffx",NILS,"Clock",NILS);
createclb("clb22","EB","A2=((A2@A2_1)*en_a*en_5)","en_a=
(~(A2*(~A0)*(~A1)))",NILS,"ffx","ffy","Clock",NILS);


// MOD-18 counter.
createclb("clb12","EC","d0=((~d0)*enb2*enb)",
"d1=((d1@d0)*enb2*enb)",NILS,"ffx","ffy","Clock",NILS);
createclb("clb3","ED","d21=(d2@(d1*d0))","d31=(d0*d1*d2)",NILS,
NILS,NILS,"Clock",NILS);
createclb("clb14","EE","d3=((d3@d31)*enb2*enb)","d2=(d21*enb2*enb)",
NILS,"ffx","ffy","Clock",NILS);
createclb("clb15","EF","d41=(d4@(d3*d31))","d4=(d41*enb2*enb)"
,NILS,NILS,"ffy","Clock",NILS);
createclb("clb16","EG","enb1=(d4*(~d3)*(~d2)*(~d1))",
"enb2=(~((~d0)*enb1))",NILS,NILS,"ffy","Clock",NILS);
createclb("clb17","EH",
"temp1=(((~d3)*(~d2)*(~d1)*d0)+(d3*(~d2)*d1*(~d0)))",
"temp2=(((~d2)*(d1@d3))+((~d3)*d2))",NILS,NILS,NILS,"Clock",NILS);
createclb("clb24","EI","en_5=((~d4)*(temp1+temp2))",NILS,NILS,
"ffx",NILS,"Clock",NILS);

pr3.writeXNF();
pr3.writeVHDL();
return 0;
}
```

Appendix D

Programming Logic Design for DBF Receiver

```
// To program eight  HSP50110's and one HSP50210.
// Output Signals are: C(0-7), A(0-2), WR0, WR1, WR2, WR3, WR4, WR5,
// WR6, WR7, WR_DCL
// Input is, PRGM

#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <stdio.h>
#include "lca.h"
#include "header.h"
#include "clkbuffer.h"

int main(void)
{

       lca pr("pr","4013PQ208-4");
       net NIL;
       char* NILS="\0";
       net Clock("Clock",'L','C');
       net p2b("p2b");
       clkbuffer globclk(Clock,p2b,"BUFGP_BR",NILS);

// Assigning inpads and outpads.
createiob("inpad1","P6","PRGM",NILS ,NILS ,NILS,NILS ,NILS );
createiob("outpad1","P10",NILS, "A0",NILS, NILS, NILS,NILS);
createiob("outpad2","P16",NILS, "A1",NILS, NILS,NILS,NILS);
createiob("outpad3","P12",NILS, "A2",NILS, NILS,NILS,NILS);
createiob("outpad4","P18",NILS,"q1",NILS,NILS,NILS,NILS);
createiob("outpad5","P34",NILS,"q2",NILS,NILS,NILS,NILS);
createiob("outpad6","P36",NILS,"q3",NILS,NILS,NILS,NILS);
createiob("outpad7","P21",NILS,"q4",NILS,NILS,NILS,NILS);
createiob("outpad8","P39",NILS,"q5",NILS,NILS,NILS,NILS);
createiob("outpad9","P23",NILS,"q6",NILS,NILS, NILS,NILS);
createiob("outpad10","P41",NILS,"q7",NILS,NILS,NILS,NILS);
createiob("outpad11","P43",NILS,"q8",NILS,NILS,NILS,NILS);
createiob("outpad12","P28",NILS,"q9",NILS,NILS,NILS,NILS);
createiob("outpad27","P28",NILS,"q10",NILS,NILS,NILS,NILS);
createiob("outpad13","P32",NILS,"WR1",NILS, NILS,NILS,NILS);
createiob("outpad14","P32",NILS,"WR2",NILS, NILS,NILS,NILS);
createiob("outpad15","P32",NILS,"WR3",NILS, NILS,NILS,NILS);
createiob("outpad16","P32",NILS,"WR4",NILS, NILS,NILS,NILS);
createiob("outpad17","P32",NILS,"WR5",NILS, NILS,NILS,NILS);
createiob("outpad18","P32",NILS,"WR6",NILS, NILS,NILS,NILS);
createiob("outpad19","P32",NILS,"WR7",NILS, NILS,NILS,NILS);
createiob("outpad20","P32",NILS,"WR8",NILS, NILS,NILS,NILS);
createiob("outpad21","P30",NILS,"WR_DCL",NILS, NILS,NILS,NILS);

// State Machine Logic For Iniating the Programming.
createclb("clb1","AA","s1=(PRGM*PRGM)","s2=(s1*s1)",NILS,"ffx",
"ffy","Clock",NILS);
createclb("clb2","AB","start=(s1*(~s2))",
"enb=(start+((~start)*(~done)*enb))",NILS,"ffx","ffy","Clock",NILS);

createclb("clb4","BA","q0=((~q0)*enb)","q1=((q1@q0)*enb)",NILS,
"ffx","ffy","Clock",NILS);
createclb("clb5","BB","q2=((q2@(q1*q0))*enb)","q31=(q2*q1*q0)",
NILS,"ffx",NILS,"Clock",NILS);
createclb("clb6","BC","q3=((q3@q31)*enb)","q41=(q3*q2*q1*q0)",
NILS,"ffx",NILS,"Clock",NILS);
createclb("clb7","BD","q4=((q4@q41)*enb)",
"q5=((q5@(q4*q41))*enb)",NILS,"ffx","ffy","Clock",NILS);
createclb("clb8","CA","q61=(q5*q4*q41)",
"q6=((q6@q61)*enb)",NILS,NILS,"ffy","Clock",NILS);
createclb("clb9","CB","q7=((q7@(q6*q61))*enb)",
"q81=(q61*q6*q7)",NILS,"ffx",NILS,"Clock",NILS);
createclb("clb10","CC","q8=((q8@q81)*enb)",
"q9=((q9@(q8*q81))*enb)",NILS,"ffx","ffy","Clock",NILS);
createclb("clb10","CC","q101=(q81*q8*q9)",
"q10=((q10@q101)*enb)",NILS,NILS,"ffy","Clock",NILS);

createclb("clb11","CD","count1=(q8*(~q7)*q6*(~q5))",
"count2=((~q4)*q3*q2*(~q1))",NILS,NILS,NILS,"Clock",NILS);
createclb("clb13","DA","done=(count1*count2*q9*q10)",NILS,
NILS,"ffx",NILS,"Clock",NILS);

// Mod-81 counter
createclb("clb4","BA","e0=((~e0)*enb*(~enb_81))",
"e1=((e1@e0)*enb*(~enb_81))",NILS,"ffx","ffy","Clock",NILS);
createclb("clb5","BB","e21=(e1*e0)","e2=((e2@e21)*enb*(~enb_81))",
NILS,NILS,"ffy","Clock",NILS);
createclb("clb6","BC","e31=(e2*e21)","e3=((e3@e31)*enb*(~enb_81))",
NILS,NILS,"ffy","Clock",NILS);
createclb("clb7","BD","e41=(e3*e31)","e4=((e4@e41)*enb*(~enb_81))",
NILS,NILS,"ffy","Clock",NILS);
createclb("clb7","BD","e51=(e4*e41)","e5=((e5@e51)*enb*(~enb_81))",
NILS,NILS,"ffy","Clock",NILS);
createclb("clb8","CA","e61=(e51*e5)","e6=((e6@e61)*enb*(~enb_81))",
NILS,NILS,"ffy","Clock",NILS);
createclb("clb8","CA","e71=(e61*e6)","e7=((e7@e71)*enb*(~enb_81))",
NILS,NILS,"ffy","Clock",NILS);
createclb("clb8","CA","enb_81x=(e7*(~e6)*e5*(~e4))",
"enb_81y=((~e3)*(~e2)*(~e1)*(~e0))",NILS,NILS,NILS,"Clock",NILS);
createclb("clb8","CA","enb_81=(enb_81x*enb_81y)",NILS,NILS,"ffx",NIL
"Clock",NILS);
createclb("clb5","BB","f0=((f0@enb_81)*enb)","f1=((f1@(f0*enb_81))*e
NILS,"ffx","ffy","Clock",NILS);
createclb("clb5","BB","f21=(f1*f0*enb_81)","f2=((f2@f21)*enb)",NILS,
"ffy","Clock",NILS);
createclb("clb5","BB","f31=(f2*f1*f0*enb_81)","f3=((f3@(f2*f21))*enb
NILS,NILS,"ffy","Clock",NILS);
createclb("clb5","BB","decode1=((~f3)*(~f2)*(~f1)*(~f0))",
"decode2=((~f3)*(~f2)*(~f1)*f0)",NILS,"ffx","ffy","Clock",NILS);
createclb("clb5","BB","decode3=((~f3)*(~f2)*f1*(~f0))",
"decode4=((~f3)*(~f2)*f1*f0)",NILS,"ffx","ffy","Clock",NILS);
createclb("clb5","BB","decode5=((~f3)*f2*(~f1)*(~f0))",
"decode6=((~f3)*f2*(~f1)*f0)",NILS,"ffx","ffy","Clock",NILS);
createclb("clb5","BB","decode7=((~f3)*f2*f1*(~f0))",
"decode8=((~f3)*f2*f1*f0)",NILS,"ffx","ffy","Clock",NILS);

// WR signals.
createclb("clb19","DC","WR1=(en_5*decode1*(~q0)*enb)",
"WR2=(en_5*decode2*(~q0)*enb)",NILS,"ffx","ffy","Clock",NILS);
createclb("clb19","DC","WR3=(en_5*decode3*(~q0)*enb)",
"WR4=(en_5*decode4*(~q0)*enb)",NILS,"ffx","ffy","Clock",NILS);
createclb("clb19","DC","WR5=(en_5*decode5*(~q0)*enb)",
"WR6=(en_5*decode6*(~q0)*enb)",NILS,"ffx","ffy","Clock",NILS);
createclb("clb19","DC","WR7=(en_5*decode7*(~q0)*enb)",
"WR8=(en_5*decode8*(~q0)*enb)",NILS,"ffx","ffy","Clock",NILS);
```

```
createclb("clb19","DC","WR_DCL=(en_5*f3*(~q0)*enb)",NILS,NILS,
"ffx",NILS,"Clock",NILS);

// EPROM ADDRESSES. en_5 as the count enable.
createclb("clb4","BA","g0=((~g0)*enb)","g1=((g1@g0)*enb)",NILS,"ffx"
,"ffy","Clock",NILS);
createclb("clb5","BB","g2=((g2@(g1*g0))*enb)","g31=(g2*g1*g0)",NILS,
"ffx",NILS,"Clock",NILS);
createclb("clb6","BC","g3=((g3@g31)*enb)","g41=(g3*g2*g1*g0)",NILS,
"ffx",NILS,"Clock",NILS);
createclb("clb7","BD","g4=((g4@g41)*enb)","g5=((g5@(g4*g41))*enb)",
NILS,"ffx","ffy","Clock",NILS);
createclb("clb8","CA","g61=(g5*g4*g41)","g6=((g6@g61)*enb)",NILS,NIL
"ffy","Clock",NILS);
createclb("clb9","CB","g7=((g7@(g6*g61))*enb)","g81=(g61*g6*g7)",NIL
"ffx",NILS,"Clock",NILS);
createclb("clb10","CC","g8=((g8@g81)*enb)","g9=((g9@(g8*g81))*enb)",
NILS,"ffx","ffy","Clock",NILS);
createclb("clb10","CC","g101=(g81*g8*g9)","g10=((g10@g101)*enb)",NIL
NILS,"ffy","Clock",NILS);

// MOd-10 Counter.
createclb("clb20","DD","A0=((A0@q0)*en_a*en_5)","A1_1=(A0*q0)",NILS,
"ffx",NILS,"Clock",NILS);
createclb("clb21","EA","A1=((A1@A1_1)*en_a*en_5)","A2_1=(A1*A0*q0)",
NILS,"ffx",NILS,"Clock",NILS);
createclb("clb22","EB","A2=((A2@A2_1)*en_a*en_5)",
"en_a=(~(A2*(~A0)*(~A1)))",NILS,"ffx","ffy","Clock",NILS);

// MOD-18 counter.
createclb("clb12","EC","d0=((~d0)*enb2*enb)","d1=((d1@d0)*enb2*enb)"
NILS,"ffx","ffy","Clock",NILS);
createclb("clb3","ED","d21=(d2@(d1*d0))","d31=(d0*d1*d2)",NILS,NILS,
NILS,"Clock",NILS);
createclb("clb14","EE","d3=((d3@d31)*enb2*enb)","d2=(d21*enb2*enb)",
NILS,"ffx","ffy","Clock",NILS);
createclb("clb15","EF","d41=(d4@(d3*d31))","d4=(d41*enb2*enb)",NILS,
NILS,"ffy","Clock",NILS);
createclb("clb16","EG","enb1=(d4*(~d3)*(~d2)*(~d1))",
"enb2=(~((~d0)*enb1))",NILS,NILS,"ffy","Clock",NILS);
createclb("c","EH","temp1=(((~d3)*(~d2)*(~d1)*d0)+(d3*(~d2)*d1*(~d0)
"temp2=(((~d2)*(d1@d3))+((~d3)*d2))",NILS,NILS,NILS,"Clock",NILS);
createclb("clb24","EI","en_5=((~d4)*(temp1+temp2))",NILS,NILS,"ffx",
NILS,"Clock",NILS);
pr.writeXNF();
pr.writeVHDL();
return 0;
}
```