Digital Beamforming Implementation on an FPGA Platform

(Projecte Fi de Carrera)

Digital Beamforming Implementation on an FPGA Platform

(Projecte Fi de Carrera)

July, 2007.

Author:
David Bernal Casas

SPCOM Group
Universitat Politècnica de Catalunya (UPC)
Escola Tècnica Superior d'Enginyers de Telecomunicacions de Barcelona (ETSETB)
Jordi Girona 1-3, Campus Nord, Edifici D5
08034 Barcelona, SPAIN

`albernal@gps.tsc.upc.edu`

Advisor:
Pau Closas Gómez

SPCOM Group
Universitat Politècnica de Catalunya (UPC)
Escola Tècnica Superior d'Enginyers de Telecomunicacions de Barcelona (ETSETB)
Jordi Girona 1-3, Campus Nord, Edifici D5
08034 Barcelona, SPAIN

`closas@gps.tsc.upc.edu`

Some words of my... I don't remember...

Te diré algo que ya sabes. El mundo no es sol ni arco iris. Es un sitio muy malo y desagradable, y no importa lo duro que seas, te pondrá de rodillas y te dejará ahí permanentemente si se lo permites. Tú, yo, nadie pega más duro que la vida. Pero no se trata de lo duro que pegues. Se trata de cuan duro te peguen y puedas seguir adelante. Se trata de cuanto aguantas y sigues adelante. Así es como se gana! Si sabes lo que vales, sal a buscarlo. Pero tienes que estar dispuesto a soportar los golpes. Y no acusar a nadie diciendo que no eres lo que quisieras por culpa de aquel, o de aquella o de nadie. Los cobardes hacen eso, y tú no eres un así! Eres mejor que eso! Siempre te querré no importa lo que pase.

Thanks to all

Contents

List of Figures

1 Introduction

The objective of the work presented is to implement a Digital Beamforming (DBF) platform for an antenna array receiver designed for the S-DMB system. Our project deals with the design of antenna arrays from a hardware point of view, in contrast to other theoretic studies regarding DBF algorithms. Hence, we will study practical aspects of DBF implementation such as signal quantization and required computational resources.

This work is part of UPC contribution to the CORPA (Cost-Optimised high performance active Receive Phase Array antenna for mobile terminals) project of ESA (European Space Agency). CORPA project is composed of several partners: Space Services CE Lda. (Portugal), Instituto de Telecomunicaçaos (Portugal), TriaGnoSys (Germany), Satellite Services BV (Netherland) and Technical University of Catalonia, UPC (Spain). The CORPA project started in May 2006 and is intended to end in December 2007. In the frame of the S-DMB system the project considers the design, manufacturing and test of a vehicle antenna, utilising Digital Beamforming for providing multimedia reception in S-band. In such antenna, each path will be equipped with a separate LNB, down-conversion, analogue-to-digital converter and some signal processing platform will be in charge of antenna array weights computation for interference mitigation.

A current problem of mobile terminals for data applications at the L/S/C frequency bands is the size and shape of the antenna. These antennas are currently mainly mechanically steerable in both azimuth and/or elevation, to be cost competitive. This makes them rather bulky and of limited interest for a larger market. Antenna systems specially designed for cars, trains, ships and airplanes are needed to receive signals to communicate via satellites on customer demand. Conformal, phased array antennas utilising Digital Beamforming can provide an attractive solution for medium gain antennas.

Figure 1.1 depicts the parts involved in CORPA project and our contribution to the project. In this diagram, it is shown the forming blocks of an antenna array based S-DMB receiver. For the CORPA project the geometry of the designed antenna array is conical, as it will be pointed later in this lines. After IF downconversion and ADC quantization, the digital signal is introduced in the digital platform, the FPGA. The FPGA is in charge of processing the signal, computing the beamforming weights and to deliver the output to a commercial S-DMB receiver. The computed weights are used to electronically modify the radiation pattern of the antenna array in order to track and point the desired signal and null all other sources of interference (whether other communication system or from intentional jammers).

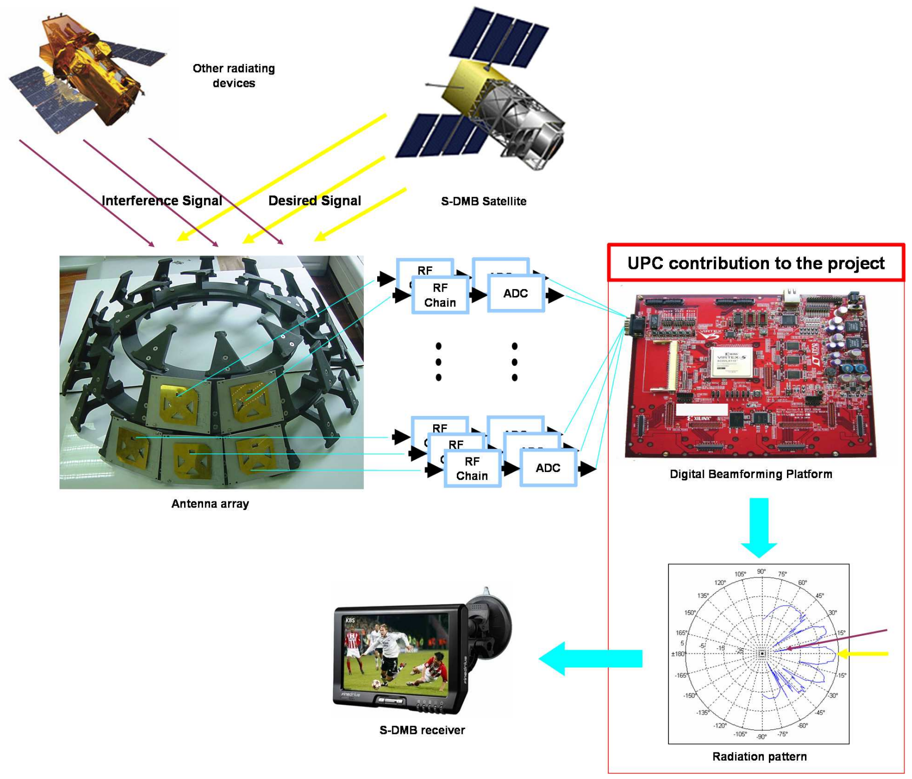


Fig. 1.1 Parts involved in the S-DMB system, as considered in CORPA.

As shown in Figure 1.2, for the conformal array a double cone geometry is considered to provide sufficient gain and low profile. An inner cone of 10 elements is placed in an outer cone comprising 30 elements. The elements of the inner cone provide additional gain for higher elevation angles and are not used for lower elevation angles. The base diameter is approximately 56 cm and height is 13 cm.

The receiver presented in Figure 1.1 must deal with the acquisition and tracking of synchronism. However, for the sake of simplicity the CORPA project only contemplates the DBF implementation since synchronism is considered to be given by an external communications receiver, as confirmed by ESA. Nevertheless, we have gone one step further

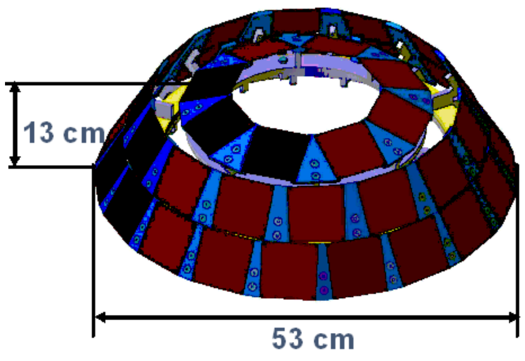


Fig. 1.2 Considered array geometry in CORPA.

in the work presented in the PFC, designing and implementing the acquisition of signal's synchronism. We have selected an FPGA as a Digital Platform because it allows to parallelize the operations for each antenna element. Finally, due to the amount of resources required for the project, it has been implemented on two FPGAs.

The antenna array success of our project has been the physical implementation of an independent receiver able to acquire the synchronism, perform the tracking of the signal and execute the Digital Beamforming according to the technique selected. In what follows, the contents of the document are briefly introduced.

A brief history about programmable devices for DSP implementation is presented in Chapter 2. We comment the state-of-the art of the digital platforms used nowadays: FPGAs, CPLDs and DSPs. We explain in more detail the features of the FPGAs: architecture, design, programming and applications. Finally, we provided an overview of the evaluation board used for this project: Xilinx Virtex-5 LX-220.

In Chapter 3, we introduce the standard S-DMB which is the standard that the CORPA project is focusing. We also present the signal model and we comment the different antenna array techniques to implement a Digital Beamforming, namely those techniques based on spatial information, those techniques based on temporal information and those techniques on both spatial and temporal information. In addition, we show that, for our system, techniques based on temporal information have the property to maximize the SINR (Signal-to-Interference-plus-Noise Ratio). We discuss the use of these techniques before and after the Pilot Channel Despreading. MATLAB simulations are provided in order to evaluate a preliminary study of DBF. Finally, we also study the operations involved in the processing and the computational cost.

In Chapter 4, we present a detailed implementation of our system. We have designed the architecture of the system in order to implement the acquisition and tracking modes. As said, two FPGAs have been considered due to the digital requirements. We have designed and tested hardware and software blocks for each FPGA. The conceptual description, schematics and simulations are presented for each block. All blocks have been designed in hardware and programmed in VHDL, except the block in charge of the computation of the antenna weights, which has been implemented using the MicroBlaze soft-processor of the FPGA and it have been programmed in C. Several studies, simulations and schematics blocks have been presented along Chapters 3 and 4. A summary of the project and conclusions are presented in Chapter 5.

2

State-of-the-art programmable devices for DSP implementation

2.1 Brief history of programmable devices

In 1970, Texas Instruments (TI) developed a mask-programmable IC (Integrated Chip) based on the IBM read-only associative memory or ROAM. This device, the TMS2000, was programmed by altering the metal layer during the production of the IC. TI coined the term Programmable Logic Array (PLA) for this device.

In 1973, National Semiconductor introduced a mask-programmable PLA device, the DM7575. This was more popular than the TI part but cost of making the metal mask limited its use. The device is significant because it was the basis for the Field Programmable Logic Array (FPLA) produced by Signetics in 1975, the 82S100.

In 1971, General Electric Company (GE) was developing a programmable logic device based on the new PROM technology. This experimental device improved on IBM's ROAM by allowing multilevel logic. Intel had just introduced the floating-gate ultraviolet (UV) erasable PROM so the researcher at GE incorpored that technology. The GE device was the first erasable Programmable Logic Device (PLD) ever developed. GE obtained several early patents on programmable logic devices.

In 1974, GE entered into an agreement with Monolithic Memories, Inc (MMI) to develop a mask-programmable logic device incorporating the GE innovations. The device was named the Programmable Associative Logic Array or PALA. The MMI 5760 was completed in 1976 and could implement multilevel or sequential circuits of over 100 gates. The device was supported by a GE design environment where Boolean equations would be converted to mask patters for configuring the device. The part was never brought to market.

MMI introduced a breakthrough device in 1978, the Programmable Array Logic or PAL. The architecture was simpler than that of Signetics FPLA because it omitted the programmable OR array. This made the parts faster, smaller and cheaper. The PALASM design software (PAL Assembler) converted the engineers' Boolean equations into the fuse pattern required to program the part.

An innovation of the PAL was the Generic Array Logic device, or GAL, invented by Lattice Semiconductor in 1985. This device has the same logical properties as the PAL but can be erased and reprogrammed. The GAL is very useful in the prototyping stage of design, when any bugs in the logic can be corrected by reprogramming. GALs are programmed and reprogrammed using a PAL programmer, or by using the in-circuit programming technique on supporting chips.

A similar device called a PEEL (Programmable Electrically Erasable Logic) was introduced by the International CMOS Technology (ICT) corporation.

PALs and GALs are available only in small sizes, equivalent to a few hundred logic gates. For bigger logic circuits, complex PLDs or CPLDs can be used. These contain the equivalent of several PALs linked by programmable interconnections, all in one integrated circuit. CPLDs can replace thousands, or even hundreds of thousands, of logic gates.

Some CPLDs are programmed using a PAL programmer, but this method becomes inconvenient for devices with hundreds of pins. A second method of programming is to solder the device to its printed circuit board, then feed it with a serial data stream from a personal computer. The CPLD contains a circuit that decodes the data stream and configures the CPLD to perform its specified logic function.

While PALs were busy developing into GALs and CPLDs, a separate stream of development was happening. This type of device is based on gate array technology and is called the Field-Programmable Gate Array (FPGA).

FPGAs use a grid of logic gates, similar to that of an ordinary gate array, but the programming is done by the customer not by the manufacturer. The term "Field-Programmable" means the array is done outside the factory, or "in the field".

FPGAs are usually programmed after being soldered down to the circuit board, in a manner similar to that of larger CPLDs. In most larger FPGAs the configuration is volatile, and must be re-loaded into the device whenever power is applied or different functionality is required. Configuration is typically stored in a configuration PROM (Programmable ROM) or EEPROM (Electrically Erasable PROM). EEPROM version may be in-system programmable (typically via JTAG).

Nowadays, FPGAs and CPLDs are often equally good choices for a particular task. Sometimes the decision is more an economic one than a technical one, or may depend on the engineer's personal preference or experience.

2.2 Digital Platforms

In this section, we will comment the three digital platforms more used nowadays: FPGA, CPLD and DSP. We will focus with more detail explaining features of the FPGAs since it is the device selected to implement the Digital Beamforming. We also explain the main differences between FPGA and DSP and we provide a guideline in order to make a good choice.

2.2.1 Field-Programmable Gate Array (FPGA)

A field programmable gate array is a semiconductor device containing programmable logic components and programmable interconnects. The programmable logic components can be programmed to duplicate the functionality of basic logic gates such AND, OR, XOR, NOT or more complex combinational functions such as decoders or simple mathematical functions. In most FPGAs, these programmable logic components also include memory elements, which may be simple flip-flops or more complete blocks of memories.

FPGAs are generally slower than their application-specific integrated circuit (ASIC) counterparts, as they can't handle as complex a design, and draw more power. However, they have several advantages such as a shorter time to market, ability to re-program in the field to fix bugs, and lower non-recurring engineering costs. Vendors can sell cheaper, less flexible versions of their FPGAs which cannot be modified after the design is committed. The designs are developed on regular FPGAs and then migrated into a fixed version that more resembles and ASIC. Another alternative are Complex Programmable Logic Devices (CPLDs).

A recent trend has been to take the coarse-grained architectural approach a step further by combining the logic blocks and interconnects of traditional FPGAs with embedded microprocessors and related peripherals to form a complete "system on programmable chip" (SOC). Examples of such hybrid technologies can be found in the Xilinx Virtex-II PRO and Virtex-4 devices, which include on or more PowerPC processors embedded within the FPGA's logic fabric. The Atmel FPSLIC is another such devices, which uses an AVR processor in combination with Atmel's programmable logic architecture. An alternate approach is to make use of "soft" processor cores that are implemented within the FPGA logic like in this project. These cores include the Xilinx Microblaze and PicoBlaze, the Altera Nios and Nios II processors, the lattice open source Mico32, as well as third-party (either commercial or free) processor cores.

2.2.1.1 Architecture

The typical basic architecture consists of an array of configurable logic blocks (CLBs) and routing channels. Multiple I/O pads may fit into the height of one row or the width of one column in the array. Generally, all the routing channels have the same width (number of wires).

An application circuit must be mapped into an FPGA with adequate resources.

The typical Xilinx FPGA logic block consists of a 4-input lookup table (LUT), and a flip-flop, as shown in Figure 2.1.

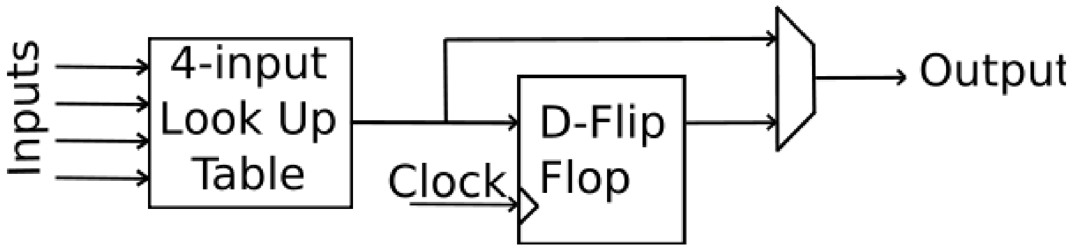


Fig. 2.1 Schematic Lookup table (LUT) block.

There is only one output, which can be either the registered or the unregistered LUT output. The logic block has four inputs for the LUT and a clock input. Since clock signals are normally routed via special-purpose dedicated routing networks in commercial FPGAs, they and other signals are separately managed. Modern FPGA families like Xilinx Virtex-5 have developed a 6-LUT logic block using technology of 65 nm.

2.2.1.2 FPGA design and programming

To define the behavior of the FPGA the user provides a hardware description language (HDL) or a schematic design. Common HDLs are VHDL and Verilog. Then, using an electronic design automation tool, a technology-mapped netlist is generated.

To simplify the design of complex systems, there exists libraries of predefined complex functions and circuits that have been tested and optimized to speed up the design process. These predefined circuits are commonly called *IP cores*, and are available from FPGA vendors and third-party IP suppliers (rarely free, and typically released under proprietary licenses). Other predefined circuits are available from developer communities such as OpenCores (typically free, and release under the GPL, BSD or similar license), and other sources.

In a typical design flow, the system is simulated at multiple stages throughout the design process. Initially the hardware description in VHDL or Verilog is simulated by creating test benches to simulate the system and observe results. Then, after the synthesis engine has mapped the design to a netlist, the netlist is translated to a gate level description where simulation is repeated to confirm the synthesis proceeded without error. Finally the design is laid out in the FPGA at which point propagation delays can be added and the simulation run again with these values back-annotated onto the netlist.

2.2.1.3 Applications

Applications of FPGAs include digital signal processor DSP, software-defined radio, aerospace and defense systems, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation and a growing range of other areas. FPGAs originally began as competitors to CPLDs and competed in a similar space. As their size, capabilities, and speed increased, FPGAs began to take over larger and larger functions to the state where some are now marketed as full systems on chips (SOC).

FPGAs are increasingly used in conventional High Performance Computing applications where computational kernels such as FFT or Convolution are performed on the FPGA instead of microprocessor. The use of FPGAs for computing tasks is known as reconfigurable computing. The inherent parallelism of the logic resources on the FPGA allows for considerable compute throughput event at sub-500 MHz clock rate. For example, the current (2007) generation of FPGAs can implement around 100 single precision floating point units, all of which can compute a result every single clock cycle. The flexibility FPGA allows for even higher performance by trading off precision and range in the number format for an increased number of parallel arithmetic units, This has driven a new type of processing called reconfigurable computing, where time intensive tasks are offloaded from software to FPGAs.

The adoption of FPGAs in high performance computing is currently limited by the complexity of FPGA design compared to conventional software and the extremely long turn-around times of current design tools, where 2-3 hours wait is necessary after even minor changes to the source code.

2.2.2 Complex Programmable Logic Device (CPLD)

A Complex Programmable Logic Device is a programmable logic device based on the PAL architecture. The building block of a CPLD is the macro cell, which contains logic implementing Disjuntive Normal Form (DNF) expressions and more specialized logic operations.

CPLDs and FPGAs include a relatively large number of programmable logic elements. CPLD logic gate densities range from the equivalent of several thousands to tens of thousands of logic gates, while FPGAs typically range from tens of thousands to several million. Hence, applications of CPLDs are the same that FPGAs but with lower digital requirements.

The primary differences between CPLDs and FPGAs are architectural. A CPLD has a somewhat restrictive structure consisting of one or more programmable sum-of-products logic arrays feeding a relatively small number of clocked registers. The result of this is less flexibility, with the advantage of more predictable timing delays and a higher logic-to-interconnect ratio. The FPGA architectures, on the other hand, are dominated by interconnect. This makes them far more flexible (in terms of the range of designs that are practical for implementation within them) but also far more complex to design for.

The most noticeable difference between a large CPLD and a small FPGA is the presence of on-chip non-volatile memory in the CPLD. This characteristic of non-volatility means that CPLDs are often used in modern digital design to perform 'boot loader' functions before handing over control to other devices not having this capability. A good example is when a CPLD is used to load configuration data for an FPGA from non-volatile memory.

Another notable difference between CPLDs and FPGAs is the presence in most FPGAs of higher-level embedded functions (such as adders and multipliers) and embedded memories. A related, important difference is the presence in most FPGAs of higher-level embedded functions (such as adders and multipliers) and embedded memories. A related, important difference is that many modern FPGAs support full or partial in-system reconfiguration, allowing their designs to be changed "on the fly" either for system upgrades or for dynamic reconfiguration as a normal part of system operation. Some FPGAs have the capability of partial re-configuration that lets one portion of the device be re-programmed while other portions continue running.

2.2.3 Digital Signal Processor (DSP)

DSP processors are microprocessors, typically programmed in C, that provides ultra-fast instruction sequences, such as shift and add, and multiply and add, which are designed to perform digital signal processing - the mathematical manipulation of digitally represented signals. Along with the rising popularity of DSP applications, the variety of DSP-capable processors has expanded greatly since the introduction of the first commercially successful DSP chips in the early 1980s. Today's DSP processors (or DSPs) are sophisticated devices with impressive capabilities. In this section, we introduce the features and applications of modern commercial DSP processors.

The most cited of these features is the ability to perform one or more multiply-accumulate operations (often called MACs) in a single instruction cycle. The multiply-accumulate operation is useful in DSP algorithms that involve computing a vector dot product, such as digital filters, correlation, and Fourier transforms. To achieve a single-cycle MAC, DSP processors integrate multiply-accumulate hardware into the main data path of the processor, as shown in Figure 2.2. Some recent DSP processors provide two or more multiply-accumulate units, allowing multiply-accumulate operations to be performed in parallel. For example, the Motorola DSP processor family examined in Figure 2.2 offers eight guard bits.

A second feature shared by DSP processors is the ability to complete several accesses to memory in a single instruction cycle. This allows the processor to fetch an instruction while simultaneously fetching operands and/or storing the result of a previous instruction

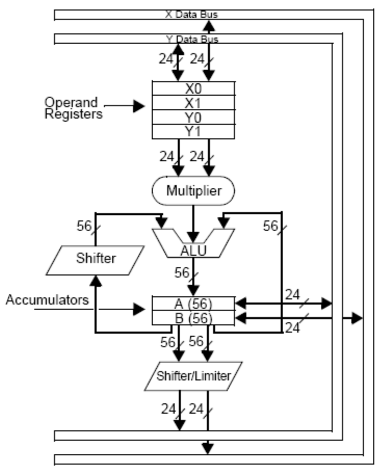


Fig. 2.2 A representative conventional fixed-point DSP processor (from the Motorola family).

to memory. For example, in calculating the vector dot product for an FIR filter, most DSP processors are able to perform a MAC while simultaneously loading the data sample and coefficient for the next MAC.

A third feature often used to speed arithmetic processing on DSP processors is one or more dedicated address generation units. Once the appropriate addressing registers have been configured, the address generation unit operates in the background (i.e., without using the main data path of the processor), forming the addresses required for operand accesses in parallel with the execution of arithmetic instructions. In contrast, general-purpose processors often require extra cycles to generate addresses needed to load operands.

One of the most fundamental characteristics of DSPs is the type of native arithmetic used in the processor. Most of DSPs use fixed-point arithmetic, where numbers are represented as integer or as fractions in a fixed range (usually -1.0 to +1.0). Other processors use floating-point arithmetic, where values are represented by a *mantissa* and an *exponent* as $mantissa \times 2^{exponent}$. The mantissa is generally a fraction in the range -1.0 to +1.0, while the exponent is an integer that represents the number of places that the binary point (analogous to the decimal point in a base 10 number) must be shifted left or right in order to obtain the value represented.

DSP processors find use in an extremely diverse array of applications, from radar systems to consumer electronics. In terms of dollar volume, the biggest applications for digital signal processors are inexpensive, high-volume embedded systems, such as cellular telephones, disk drives and portable digital audio players. A second important class of applications involves processing large volumes of data with complex algorithms for specialized needs. Examples include sonar and seismic exploration, where production volumes are lower, algorithms more demanding, and product designs larger and more complex.

2.2.4 FPGA or DSP

In the past, the use of digital signal processors was nearly ubiquitous, but with the needs of many applications outstripping the processing capabilities of digital signal processors, measured in millions of instructions per second (MIPS), the use of FPGAs is growing rapidly. Currently, the primary reason most engineers choose use a FPGA over a digital signal processors is driven by the application's MIPS requirements. The ability to manipulate the logic at gate level means you can construct a custom processor to efficiently implement the desired function. By simultaneously performing all of the algorithms's subfunctions, the FPGA can outperform a DSP by as much as 1000:1. As shown Figure 2.3, actual performance gains depend on algorithm efficiency, clock rates, degree of parallelism and other factors. Typical gains lie between 10:1 and 1000:1.

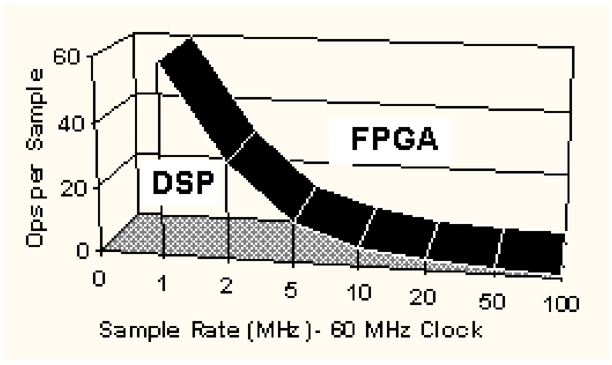


Fig. 2.3 DSP performance is limited by the serial instruction stream. FPGAs are a better solution in the region above the curve.

When the sample rates grow above a few MHz, a DSP has to work very hard to transfer the data without any loss. This is because the processor must use shared resources like memory busses, or even the processor core which can be prevented from talking interrupts for some time. An FPGA on the other hand dedicates logic for receiving the data, so can mantain high rates of I/O.

A DSP is optimized for use of external memory, so a large data set can be used in the processing. FPGAs have a limited amount of internal storage so need to operate on smaller data sets. However, FPGA modules with external memory can be used to eliminate this restriction.

A DSP is designed to offer simple re-use of the processing units, for example a multiplier used for calculating an FIR can be re-used by another routine that calculates FFTs. This is much more difficult to achieve in an FPGA, but in general there will be more multipliers available in the FPGA.

If a major context switch is required, the DSP can implement this by branching to a new part of the program. In contrast, an FPGA needs to build dedicated resources for each configuration. If the configurations are small, then several can exist in the FPGA at the same time. Larger configurations mean the FPGA needs to be reconfigured – a process which can take some time.

The DSP can take a standard C program and run it. This C code can have a high level of branching and decision making – for example, the protocol stacks of communications systems. This is difficult to implement within an FPGA.

Most signal processing systems start life as a block diagram of some sort. Actually translating the block diagram to the FPGA may well be simpler than converting it to C code for the DSP.

There are a number of elements to the design of the most processing systems. These all have an impact on the best choice of implementation. As a rough guideline, we have to try answer these questions:

- What is the sampling rate of this part of the system? If it is more than a few MHz, FPGA is the natural choice.
- Is your system already coded in C? If so, a DSP may implement it directly. It may not be the highest performance solution, but it will be quick to develop.
- What is the data rate of the system? If it is more than perhaps 20-30 MByte/second, then FPGA will handle it better.
- How many conditional operations are there? If there are none, FPGA is perfect. If there are many, a software implementation may be better.
- Does your system use floating point? If so, this is a factor in favor of the programmable DSP. None of the Xilinx cores support floating point today, although you can construct your own.
- Are libraries available for what you want to do? Both DSP & FPGA offer libraries for basic building blocks like FIRs or FFTs. However, more complex components may not be available, and this could sway your decision to one approach or the other.

In reality, most systems are made up of many blocks. Some of those blocks are best implemented in FPGA, others in DSP. Lower sampling rates and increased complexity suit the DSP approach; higher sampling rates, especially combined with rigid, repetitive tasks, suit the FPGA.

2.3 Virtex-5 LX220 application board

In this section we now comment the features of the evaluation board used for the CORPA project: TB-5V-LX220-DDR2. The TB-5V-LX220-DDR2 is an evaluation board that is equipped with high density, high performance Xilinx Virtex-5 LX series, DDR2 SDRAM chip and DDR2 SO-DIMM socket. Figure 2.4 shows the evaluation board used in this project and Figure 2.5 shows the block diagram.

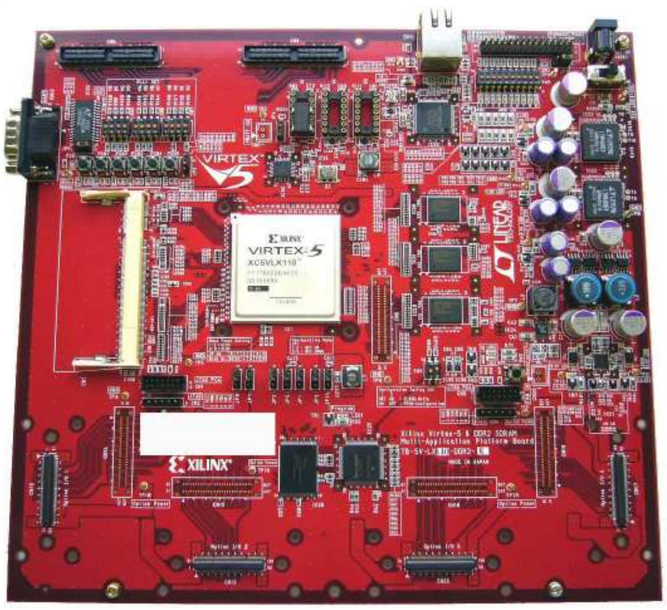


Fig. 2.4 TB-5V-LX200-DDR2 development board.

The main features of the board are:

- FPGA Xilinx XC5VLX220-1FF1760.
- DDR2 SDRAM component.
- DDR2 SDRAM SO-DIMM.
- TE7725PF enables to download a configuration data from PC into a Flash memory by using dedicated application software.
- 10/100 Base Ethernet MAC & PHY.
- RS232C.
- Soft-touch Connector enables to check independent modules.
- 2 LVDS connectors (22 pair × 2).
- Option I/O connectors (42 I/Os × 4).
- Multi-purpose pin-headers (20 I/Os).

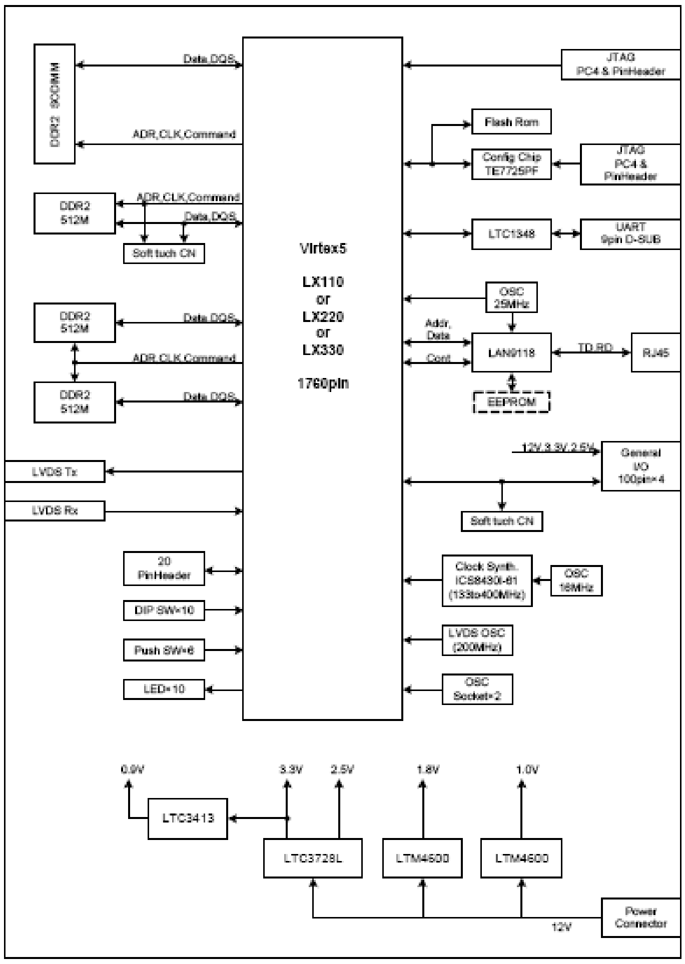


Fig. 2.5 TB-5V-LX220-DDR2 Block Diagram.

3

An Antenna Array Receiver for the S-DMB system

In this Chapter, we study several aspects related with the solution proposed in CORPA. First, we provide an overview of the S-DMB standard. Section 3.2 presents a number of techniques that can be used in order to compute weights in a digital antenna array. Finally, section 3.3 presents different architecture designs for the DBF implementation, simulation and some considerations regarding the processing power required and operations involved in a DBF platform.

3.1 Signal characteristics of the S-DMB system

This section is intended to give an overview of the signal structure, whose ITU-R Recommendation can be found in reference [1] under the nomenclature of *Digital System E* and a summary of its main features can be found in [2]. This section provides an exhaustive overview of the S-DMB standard in terms of signal coding, modulation and multiplexing, also featuring important parameters for link budget calculation. Indeed, modulation parameters are a relevant issue to take into account in the beamforming design process.

3.1.1 General Issues

The system is designed to provide high-quality audio and multimedia and data broadcasting services to mobile receivers, handheld and vehicular. The geostationary satellite specially designed for this system was launched succesfully in March 2004, followed by the inaguration of regular broadcasting services on 20 October 2004. Due to the nature of satellite broadcasting, it covers the Japanese area widely. The system also adopts complementary terrestrial on-channel repeaters for shadowed areas from the direct satellite broadcasting signal. It should be noted that the system is the world first broadcasting satellite services for handheld and vehicular receivers with high-quality audio and multimedia and data receiving capability.

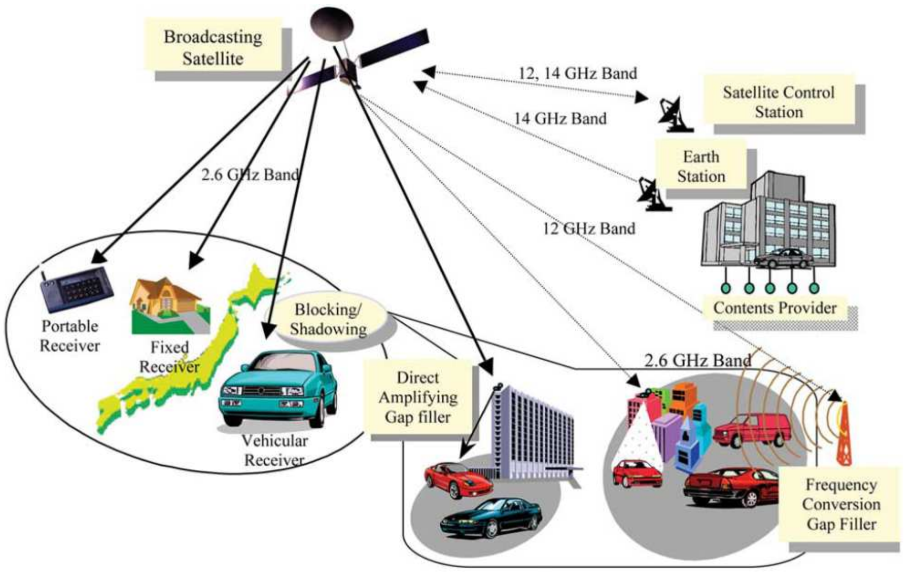


Fig. 3.1 Overview of Digital System E.

It has been designed to optimize performance for both the Geostationary satellite constellation and the terrestrial on-channel repeaters services delivery in the S-band. This is achieved through the use of Code Division Multiple Access (CDMA) based on QPSK modulation with concatenated code using Reed-Solomon (RS) code and convolutional error correcting coding.

A block diagram of the broadcasted signal generation can be seen in Figure 3.3. As explained later, broadcasted data is encoded and interleaved before simulation. In the corroborative testing of the Standard, 30 multiplexed channels have been considered. A detailed block diagram description of CDMA modulation module, found in Figure 3.2, is shown in Figure 3.3, which also will be explained later.

3.1.1.1 Allocated Frequency Resources

The service link in Digital System E uses the S frequency band. Specifically, a 25 Mhz bandwidth and a central frequency of 2642.5 MHz have been proposed. Hence, the system uses the band of frequencies between 2630 and 2655 MHz.

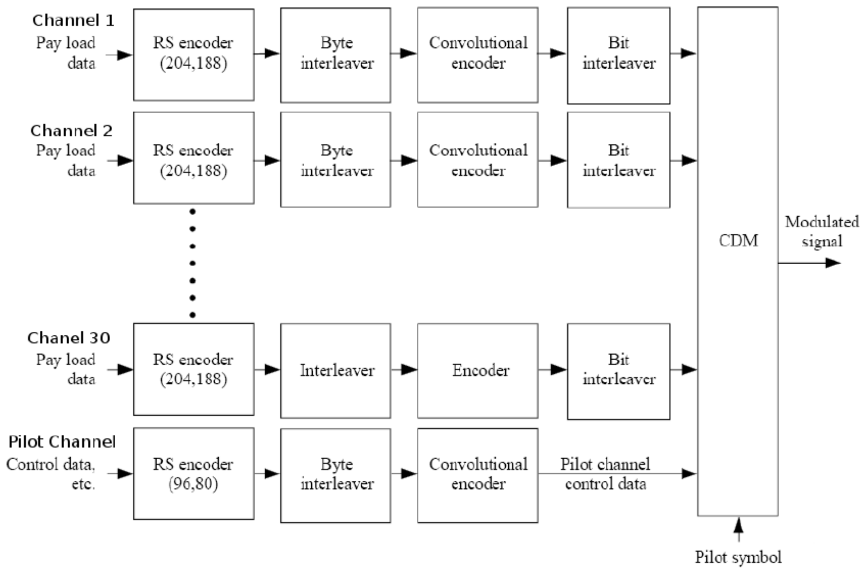


Fig. 3.2 Block Diagram of Broadcasting system.

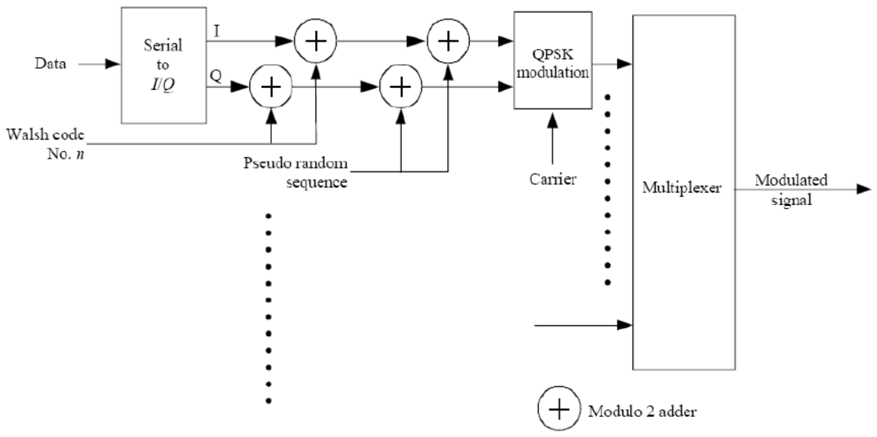


Fig. 3.3 Block Diagram of the CDMA module in Figure 3.2.

3.1.1.2 EIRP

The minimum *Effective Isotropically Radiated Power*, herein referred to as EIRP, required for mobile reception is of 67 dBW. The EIRP takes into account the losses in transmission line and connectors and the gain of the antenna.

3.1.1.3 Polarization

Polarization is circular-polarization; however a complementary terrestrial repeater may use either circular-polarization or linear polarization.

3.1.2 Source Coding

The broadcasted data adopts the following systems:

- **Audio coding:** MPEG-2 AAC (ISO/IEC 13818-7) is selected for this system. To use AAC bit stream in MPEG-2 Systems environment, audio data transport stream (ADTS) is adopted.
- **Data coding:** Various types of data broadcasting are applicable including monomedia (e.g. video source coding, text) and multimedia (mixture of audio, video, text and data) as long as these data structures are MPEG-2 Systems compliant.

3.1.3 Channel Coding

The system incorporates channel codification techniques to deal with the nuisances in the scenario and provide bit error correction. Thus, data to be broadcasted is processed with two types of blocks, as seen from Figure 3.3. On the one hand, error correction codes are applied to the stream of data and on the other hand, interleaving stages are adopted to palliate bursts errors.

3.1.3.1 Error correction coding

To provide bit error correction, concatenated code comprised of a $K = 7$ convolutional code as inner code and shortened $RS(204, 188)$ code as outer code are adopted.

Reed-Solomon code:

Outer code is the same as for other digital broadcasting systems. The original $RS(255, 235)$ code is defined as follows:

Code generator polynomial: $g(x) = (x + \lambda_0)(x + \lambda_1)(x + \lambda_2)(x + \lambda_{15}), \quad where \quad \lambda = 02_h$
Field generator polynomial: $P(x) = x^8 + x^4 + x^3 + x^2 + 1$

The shortened RS code can be implemented by adding 51 bytes, all set to zero, in front of the information bytes at the input of $RS(255, 239)$ encoder. After the RS coding procedure, these null bytes are discarded.

Convolutional code:

A $K = 7$ convolutional code is adopted as the inner code of this system. Any code rate can be selected from among 1/2, 2/3, 3/4, 5/6 and 7/8 by a puncturing technique for each broadcasting channel. These code rates are signalled through control data of the pilot channel. Rate 1/2 convolutional code is used for the pilot channel.

3.1.3.2 Interleaving

To cope with shadowing and blocking caused by small objects, phenomenas that appear in a vehicular reception condition as solid bursts of noise in the received signal of up to approximately a second. The specific configuration of those blocks do no affect the design of the beamforming module but the Bit Error Rate (BER) performance of the whole receiver. For that reason, these aspects are not explained in detail in this section, as can be consulted in the Standard [1].

Byte-wise interleaving: is the same as for other digital broadcasting systems for example, DVB-s, DVB-Y, ISDB-S and ISDB-T.

Bit-wise interleaving: the working mechanism of the bit-wise interleaver is not of interest in the beamforming considerations and can be found in the Standard. The time delay of a bit-wise interleaver can be selected from eight possible positions defined for each broadcasting channel by using control data in pilot channel. In the corroborative testing, this position was selected and fixed to a given value; hence the bit-wise interleaver has about a 3.257 s delay to recover up to 1.2 s blackout of the received signal.

3.1.4 Pilot Channel

Payload data is transmitted through broadcasting channels, while this system adopts a pilot channel to simplify the receiver's synchronization and to transmit system control data. A pilot channel has three functions. The first is to transmit the unique word for frame synchronization and frame counter for super frame synchronization. The second is to send a pilot symbol. The third is to transmit control data to facilitate the receiver functions. Hence, the pilot channel plays an important role in the beamforming design.

3.1.4.1 Frame and Super-frame

A pilot symbol is inserted every 250 μs as described in the next section. One transmission frame comprises 51 times of one pilot symbol insertion period that has a 12.75 ms time period. The first symbol D_1 (4 bytes or 32 bits) other than pilot symbols is the unique word.

Six times of transmission frame makes a super transmission frame that has a 76.5 ms time period. The second symbol D_2 is the frame counter, which assists the receiver to establish super frame synchronization. Any broadcasting channel with an arbitrarily punturing rate can be synchronized in one super frame time period because this is the least common multiplies of unit time intervals of each broadcasting channel with any possible puncture rate of convolutional code.

3.1.4.2 Pilot Symbol

Special data embedded in the pilot channel are pilot symbols that are composed of 32-bit length continuing run of data 1. Using these pilot symbols, a receiver can analyse received signal profiles (path-search analysis) and these results are used to assist a RAKE receiver function. Pilot symbols are transmitted every 250 μs. In order to improve the accuracy of path-search analysis, the pilot channel may have more signal power than a broadcasting channel. In the corroborative testing, the pilot channel had twice the signal power of a broadcasting channel.

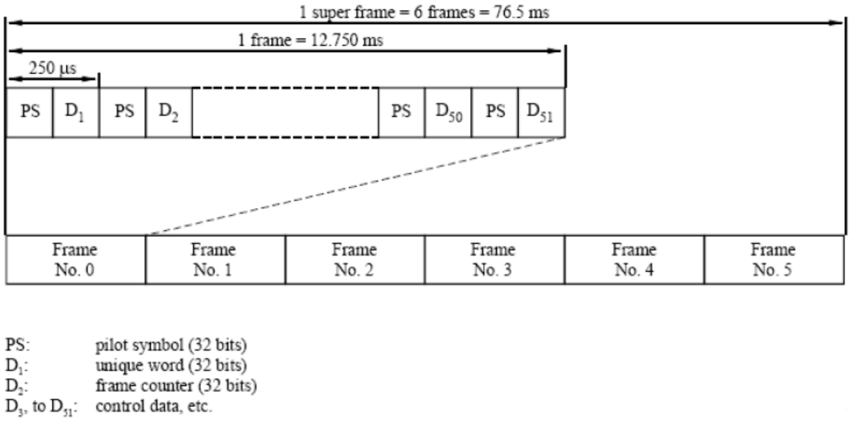

1 super frame = 6 frames = 76.5 ms
1 frame = 12.750 ms
250 µs
PS
D_1
PS
D_2
PS
D_{50}
PS
D_{51}
Frame No. 0
Frame No. 1
Frame No. 2
Frame No. 3
Frame No. 4
Frame No. 5
PS: pilot symbol (32 bits)
D_1: unique word (32 bits)
D_2: frame counter (32 bits)
D_3, to D_{51}: control data, etc.

Fig. 3.4 Frame and super-frame in pilot channel.

3.1.5 Modulation

The CDMA scheme is adopted for modulation both of the satellite link and the terrestrial gap filler link. As shown in Figure 3.3, one data sequence is converted from serial bit stream to I and Q data sequences at first. After that, each I and Q data are spread by the same unique Walsh code and a truncated M-sequence. Signature sequences and spreading sequences are modulo-2 added to the original I and Q sequence. These spread data are modulated into a QPSK signal. Modulated signals, each signal being identified by its Walsh code, are multiplexed with each other in the same frequency band. Thus, one pilot channel and several broadcasting channels comprise one whole CDMA modulated broadcasting system. The main characteristics of the modulation block are itemized now:

- Each broadcasting channel and part of the pilot channel data stream uses QPSK modulation for the component modulation, while the rest of the pilot channel (pilot symbols, frame synchronization symbols and frame counter) are modulated using BPSK. In this system, QPSK is demodulated using coherent phase detection. Signal constellation of both modulations is shown in Figure 3.5.

- The signature sequence is a Walsh code of 64-bit long.
- The spreading sequence adopted is a truncated M-sequence of 2048-bit length. This spreading sequence is obtained by truncating maximum length sequences of 4095-bit length generated using 12-stage feedback shift register sequence.
- The chip rate is 16.384 MHz.
- The processing gain is 64.
- The transmitted signal is filtered by square-root raised cosine filter. The roll-off factor is 0.22.
- Although theoretically the system can multiplex 64 CDMA channels, because a 64-chip length Walsh code is adopted, a broadcasting system is intended to deliver up to 30 CDMA channels to achieve stable reception in multipath environment.

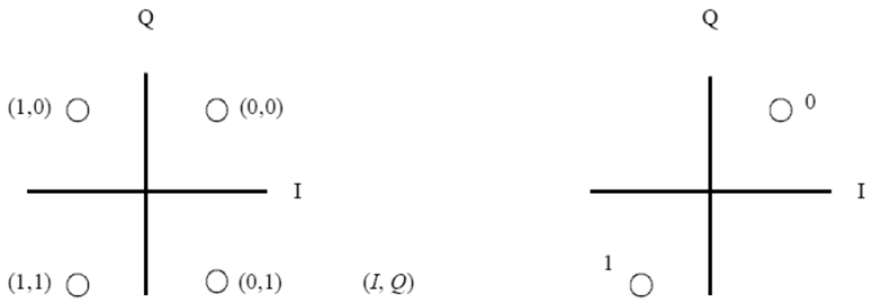


Fig. 3.5 Symbol mappings of QPSK and BPSK modulations.

3.1.6 Properties of the Pseudo-Random Noise Codes

We now consider the properties of the sequences used in the system, assuming them to be polar, that is ± 1 valued. In this section, some considerations regarding the correlation and autocorrelation functions of the two sequences used in the system are done.

3.1.6.1 Walsh Codes

Walsh-sequences have the advantage to be orthogonal, so that any multiple-access interference should be overcame. Theses codes are columns or rows from Walsh matrices that are constructed from Walsh functions. The Walsh matrix can also be obtained from a

Hadamard matrix of the same dimension by rearranging the rows so that the number of sign-changes is in increasing order, taking into account that the dimension must be a power of two. This is called sequency ordering. Since a Walsh matrix can be obtained from Hadamard matrix solely by exchanging rows it retains the property that the dot product of any two distinct rows (or columns) is zero, ensuring orthogonality. There are some properties of these codes to be highlighted:

- The codes do not have a single, narrow autocorrelation peak.
- The codes have a null Cross correlation.
- The spreading is not over the whole bandwidth, instead the energy is spread over a number of discrete frequency-components.
- These codes are used as the signature sequences in the system with a code length of $2^6 = 64$ chips per bit. Thus, the processing gain is 64, which is equivalent to assume a gain in the despreading process of 18 dB with respect to other in-band signals.

3.1.6.2 Truncated M-Sequences

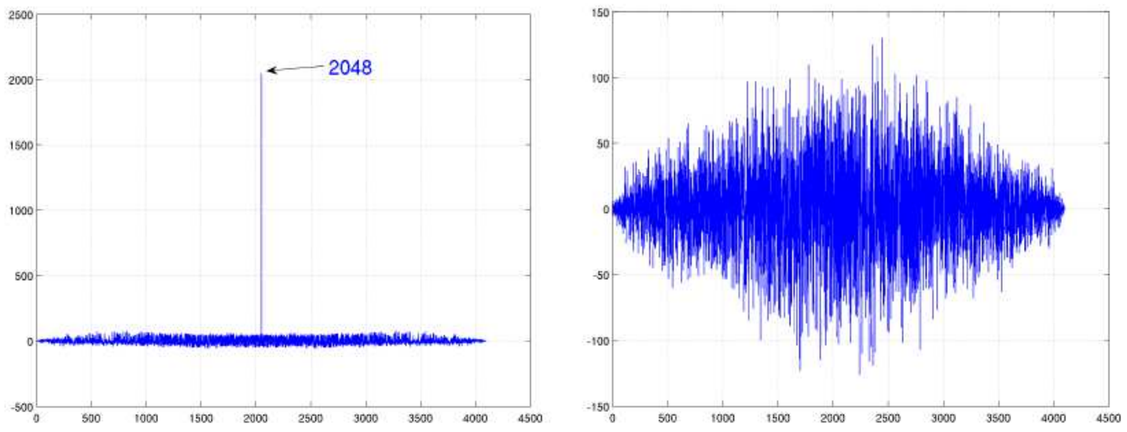
A type of Pseudo-Random Noise codes are the maximum-length shift register sequences, or M-sequences for short [3]. An M-sequence has a length $L = 2^M - 1$ chips and is generated by an M-stage shift register with linear feedback (LFSR). The sequence is periodic with period L. Each period has a sequence of 2^{M-1} ones and $2^{M-1} - 1$ zeros. In the case of M-sequences, the autocorrelation sequence is:

$$R(m) = \begin{cases} L, & m = 0 \\ -1, & 1 \leq \ m \leq L-1 \end{cases}$$

According to [1], a 12-stage LFSR will be used to generate maximum-length sequences of 4095-chip length and these sequences will be truncated to obtain 2048-chip length sequences. Autocorrelation and Cross correlation functions are similar to those obtained for M-sequences, equation 3.1, as shown in Figure 3.6.

There are some considerations of interest for synchronization and beamforming purposes:

- The codes do not have a single, narrow autocorrelation peak.
- A 2048-chip sequence has a length equal to 32 bits, which is the word unit considered, see Figure 3.4.
- Recalling the signal structure of the pilot channel introduced in section 3.1.4, we know that the pilot symbol is composed of a 32-bit length all 1 data. This means that during the pilot symbol, the spreading gain is provided by the Truncated M-sequence as it longs for the whole symbol duration. Thus, when dealing with



(a) Autocorrelation

(b) Cross Correlation

Fig. 3.6 Autocorrelation and Cross correlation representations for the family of Truncated M-sequence of 2048-bit length codes.

the Pilot Channel, the considered processing gain is 2048, which corresponds to 33 dB after the despreading process, this factor can be used to deal with both interferences from other system satellites and narrow-band interferences from other systems.

3.1.7 Signal Propagation Issues

The major issues related to signal propagation in the 2.6 GHz band are shadowing and blocking of the direct satellite path. The system uses two techniques to cope with various types of shadowing and blocking.

The first one is based on an interleaver technique in the receiver to deal with shadowing and blocking caused by small objects. This shadowing and blocking appears in a vehicular reception condition as solid bursts of noise in the received signal of up to approximately a second. A solid burst of noise is distributed over a time period of several seconds using interleaving to fit error-correcting capabilities of this system.

The second method to alleviate signal fading caused by shadowing and blocking is the inclusion of gap-fillers in the system design, that is terrestrial repeaters. Such gap-fillers retransmit the satellite signal. These gap-fillers are expected to cover the area blocked by, for example, buildings and large constructions. There are two types of gap-fillers in this system, the so-called direct amplifying gap-filler and the frequency conversion gap-filler to cover different types of blocked areas.

The direct amplifying gap-filler only amplifies the 2.6 GHz band signal broadcast from the satellite. This type of gap-filler is inherently limited to low gain amplifier to avoid undesired oscillation caused by signal coupling between transmitting and receiving antennas. This gap-filler covers a narrow area of direct path up to a 500 m long LOS (line-of-sight) area.

However, a frequency conversion gap-filler is intended to cover a large area within 3 km radius. The satellite fed signal using a different frequency than the 2.6 GHz, for example, the 11 GHz band.

In such circumstances, multipath fading appears in the area where more than two broadcasting signals are received. In this broadcasting system, the CDMA technique is adopted to secure a stable reception of the multipath-faded signal. By using a RAKE technique and antenna diversity in the receiver, a large improvement in the receiver's performance is expected in a multipath fading environment.

Spotlight type gap-filler also could improve the multipath environments where CDMA and RAKE receiver cannot decode properly without this gap-filler. This is a major feature of the CDMA system. Spotlight gap-filler can either use amplification or frequency conversion to satisfy the specific requirement of the target area to be improved.

3.1.8 Interfering signals

Interferences coming from broadcasters others than the one being tracked must be handled as nuisances. These satellites, which are assumed to have equal EIRPs, cause co-channel interference [3]. The use of CDMA greatly improves the system performance against this kind of interferences due to the despreading process, since different broadcasters will use different orthogonal codes for spreading the signal in order to broadcast their own programmes independently. On the one hand, the CDMA system proposed in [1] provides the receiver with 18 dB of desired signal gain with respect to possible inband interferences. On the other hand, for synchronization and beamforming purposes, as explained in section 3.1.6, the Pilot Channel provides the receiver with more than 18 dB (TBD) of process gain.

In addition to these features of CDMA systems, the use of antenna array based receivers can drastically improve the performance of the system against co-channel interferences. If the antenna array has enough resolution (TBD) the receiver will be able to spatially combat interferences. The gain obtained when using antenna array receivers depends on the number of elements in the antenna array.

On the downlink transmission from satellite to terminal in urban areas, the interference experienced by each terminal is primarily due to the prevailing channel conditions which are generally location-dependent. As the downlink is power-limited, to overcome poor channel conditions, this system is especially designed for multipath environment conditions. It works on the basis of receiving power summation of multipath using a RAKE receiver. This feature allows the use of on-channel repeaters to cover shadowed areas.

Also, more than 1-second blackout will be recovered using segmented convolutional bit wise interleaver. The system is based on the simultaneous reception from both satellite and complementary on-channel repeaters (gap-fillers) to fed a RAKE receiver with enough replicas of the signal. Hence, it allows the use of the same receiver for both broadcasters (satellite and terrestrial), from the RF front end to the audio and data output. Thus, the diversity provided by the gap-fillers is used to increase the Bit Error Rate (BER) of the system with a RAKE receiver. Another possible scenario to take into account is the case of having two in-view gap-fillers retransmitting signal from two different satellites. This will be the same problem of co-channel interferences, exposed before in this section, and must be handled in that way.

3.2 DBF Techniques

In this chapter, we will explain the considered signal model in antenna array receiver, we will discuss three possible techniques to implement the Digital Beamforming and we will demonstrate the most suitable technique in order to maximize the $SINR$.

3.2.1 Signal model

The problem under study concerns the extraction of information from measurements using an array of antennas. The measurements are considered to be a plane wave corrupted by noise and, eventually interferences and multipath. Given the measurements, the objective is to estimate a set of parameters associated with the wavefront. An antenna receives a scaled, time-delayed and Doppler-shifted version of the direct-sequence spread-spectrum (DS–SS) signal coming from the satellite, herein denoted as $\hat{s}(t)$. The receiving complex baseband signal at each antenna can be modeled as

$$x(t) = \alpha_d \hat{s}(t-\tau) \exp\{j2\pi f_d t\} + n(t) \tag{3.1}$$

where α_d is the complex amplitude the desired signal, τ is the time-delay, f_d is the Doppler shift and $n(t)$ is additive white Gaussian noise, and all other disturbances. In an N element antenna array, each antenna element receives a different replica of this signal, with a different phase depending on the array geometry and the Directions Of Arrival (DOA). This can be expressed by a vector signal model, where each row corresponds to one antenna:

$$\mathbf{x}(t) = \alpha_d a_d(t) \mathbf{s}_d(\theta, \psi) + \mathbf{n}(t) \tag{3.2}$$

where

- $\mathbf{x} \in \mathbb{C}^{N\times 1}$ is the observed signal vector,
- $\mathbf{s}_d(\theta, \psi) \in \mathbb{C}^{N\times 1}$ is the steering vector of the desired signal, related to the array geometry and the DOA,
- $a_d(t) = \hat{s}(t-\tau) \exp\{j2\pi f_d t\}$ the delayed Doppler shifted narrowband signals envelope and

- $\mathbf{n}(t) \in \mathbb{C}^{N\times 1}$ represents additive noise and all other disturbing terms, like multipath of each signal or interferences.

This model is built upon the narrowband array assumption, consisting of taking the time required for the signal to propagate along the array as much smaller than its reverse bandwidth. Thus, a phase shift can be used to describe the propagation from one antenna to another. In the same way, we have assumed that the Doppler effect can be modeled by a frequency shift, which is commonly referred as the narrowband signal assumption.

Suppose that K snapshots of the impinging signal are taken at a suitable sampling interval T_s. Then the sampled data can be expressed as

$$\mathbf{X} = \alpha_d \mathbf{s}_d(\theta,\psi)\mathbf{a}_d + \mathbf{N} = \alpha_d \mathbf{s}_d \mathbf{a}_d + \mathbf{N} \tag{3.3}$$

using the following definitions:

- $\mathbf{X} = [\mathbf{x}(t_0) \quad \ldots \quad \mathbf{x}(t_{K-1})] \in \mathbb{C}^{N\times K}$, referred as the spatiotemporal data matrix,
- $\mathbf{a}_d = [a_d(t_0) \quad \ldots \quad a_d(t_{K-1})] \in \mathbb{C}^{1\times K}$, known as the basis-function matrix,
- $\mathbf{N} = [\mathbf{n}(t_0) \quad \ldots \quad \mathbf{n}(t_{K-1})] \in \mathbb{C}^{N\times K}$, a matrix containing all the undesired inputs,
- for the sake of simplicity, $\mathbf{s}_d$ is used as a shortcut for $\mathbf{s}_d(\theta,\psi)$.

3.2.2 Antenna Array Beamforming techniques

There are two technical approaches to steerable antennas: mechanically moved dishes and electronically steerable antenna array. In the case of dishes, the satellite tracking is performed by means of a mechanical engine, i.e., the antenna is physically moved to point to desired satellite. This solution implies high mechanical complexity. In addition, this kind of antenna does not provide any capability in spatial processing, for instance nulling the reception of other unwanted signals or adaptative processing, and they have limited interference rejection.

On the other hand, antenna arrays are pointed electronically: while the antenna remains physically immobile, the underneath signal processing virtually steers the radiation pattern to the desired direction. Moreover, they provide interesting capabilities of automatic tracking and adaptive nulling. An array of sensors has the potencial to improve the overall reception performance of the relied signals in an environment having several sources of interference, multipath propagation or weak signal reception, providing spatial diversity to enhance the desired signal reception. Beamforming is the combination of radio signals from a set of small non-directional antennas to simulate a large directional antenna.

Beamforming with antenna arrays consists of several antennas which outputs are controlled in phase and gain, i.e., multiplied by complex weights, in order to achieve a gain pattern that can be manipulated electronically. Then, all the weighted signals are combined to obtain a single output. Considering an N-element array, these mentioned weights can be stacked in a complex-valued vector $\mathbf{w} \in \mathbb{C}^{N\times 1} = [w_0 \quad \ldots \quad w_{N-1}]^T$, and the output signal of the beamformer can be computed as $\mathbf{y} = \mathbf{w}^H\mathbf{X}$. Weight vector $\mathbf{w}$ can be designed following several criteria [4, 5, 6].

3.2.2.1 Minimum Variance Beamformer (MVB)

The classical Minimum Variance Beamformer (MVB), also known as Capon beamformer, consists on minimizing the total output power while forcing the beamformer to always point to the desired signals DOAs. Considering the signal model expressed in 3.3, the MVB can be stated as

$$\hat{\mathbf{w}}_{MVB} = \arg\min_{\mathbf{w}} \left[E\{\mid \mathbf{w}^H\mathbf{X} \mid^2\} = E\{\mathbf{w}^H\mathbf{X}\mathbf{X}^H\mathbf{w}\} = \mathbf{w}^H\hat{\mathbf{R}}_{XX}\mathbf{w}\right] \tag{3.4}$$

$$\mathbf{w}^H\mathbf{s}_d = \mathbf{1} \tag{3.5}$$

where $\hat{\mathbf{R}}_{XX} = \frac{1}{K}\mathbf{X}\mathbf{X}^H$ is the sample spectral matrix. Applying the Lagrange multipliers method, MVB results in

$$\hat{\mathbf{w}}_{MVB} = \hat{\mathbf{R}}_{XX}^{-1}\mathbf{s}_d\left(s_d^H\hat{\mathbf{R}}_{XX}^{-1}\mathbf{s}_d\right)^{-1} \tag{3.6}$$

This approach implies the prior knowledge of the steering matrix s_d. Thus, in addition to the beamformer, algorithms for estimating the DOA of the desired signal must be studied. Also, proper array calibration must be performed.

3.2.2.2 Temporal Reference Beamformer (TRB)

Another classical approach to beamforming is based on the minimization of the mean square error, understanding error as the mismatch between the actual output signal and a reference signal. In this case, the temporal diversity is exploited provided that the signal waveform is a priori known. If spatial signatures are not taken into account, the temporal reference can be expressed as $\alpha_d\mathbf{a}_d$ and this criterion can be written as

$$\hat{\mathbf{w}}_{TRB} = \arg\min_{\mathbf{w}} E\{\mid \mathbf{w}^H\mathbf{X} - \alpha_d\mathbf{a}_d \mid^2\} \tag{3.7}$$

A straightforward gradient computation leads to the following weights expression

$$\hat{\mathbf{w}}_{TRB} = \hat{\mathbf{R}}_{XX}^{-1}\hat{\mathbf{p}}\alpha_d^* \tag{3.8}$$

where $\hat{\mathbf{R}}_{XX}$ is defined as before, and $\hat{\mathbf{p}} = \frac{1}{K}\mathbf{X}\mathbf{a}_d^H$. The complex amplitude vector α_d can be estimated by some other method or initialized to an arbitrary value (for example, the expected receiving amplitude of the overall system). Notice that the temporal reference is provided by the known structure of the pilot channel data. This technique is unsensitive to calibration errors and do not requires knowledge of the steering vector. Notice that $\hat{\mathbf{p}}$ is an estimate of the steering vector of the signal as the number of samples considered tends to ∞.

$$\lim_{k\to\infty} \hat{\mathbf{p}} = \mathbf{s}_d \tag{3.9}$$

3.2.2.3 Hybrid Space-Time Reference Beamforming (HB)

Space reference can be combined with time reference in order to obtain an improved performance. This section describes a type of beamforming which exploits both diversities, following the same modular and parallelized structure of the previous section. However, the computation of the weighting vectors takes into account temporal information and, as will be shown immediately, it needs the incoming sampled signal stored in matrix $\mathbf{X}$. In addition, their values are expected to vary much faster than in the previous beamforming, Therefore, a totally digital weighting architecture seems a suitable structure for the implementation.

The derivation of the beamforming is as follows. Firstly, we define the following estimated correlation matrices based on the signal model (3.1):

$$\begin{aligned} \hat{\mathbf{R}}_{XX} &= \frac{1}{K}\mathbf{X}\mathbf{X}^H & \hat{\mathbf{R}}_{XD} &= \frac{1}{K}\mathbf{X}\mathbf{a}_d^H \\ \hat{\mathbf{R}}_{DX} &= \hat{\mathbf{R}}_{XD}^H & \hat{\mathbf{R}}_{DD} &= \frac{1}{K}\mathbf{a}_d\mathbf{a}_d^H \end{aligned} \tag{3.10}$$

The mean square error (MSE) between the output of a beamformer with weights $\mathbf{w}$ and a temporal reference signal $\alpha_d\mathbf{a}_d$ is

$$J(\mathbf{w}) = \frac{1}{K}\left|\left|\mathbf{w}^H\mathbf{X} - \alpha_d\mathbf{a}_d\right|\right|^2 \tag{3.11}$$

In this case, the temporal reference is not completely known but parameterized by the amplitude α_d, the Doppler shift f_d and the time delay τ. In order to take advantage of the knowledge of the steering matrix, a spatial constraint is imposed to force the beamformers to always point the desired signal direction of arrival. The criterion combining temporal and spatial information could be stated as follows:

$$\min_{w} J(\mathbf{w}) \tag{3.12}$$

$$\mathbf{w}^H\mathbf{s}_d = 1 \tag{3.13}$$

The amplitudes-vector components that minimize J for fixed $\mathbf{w}$, f_d and τ can be straightforwardly computed as

$$\hat{\alpha}_d = \frac{\mathbf{w}^H\hat{\mathbf{R}}_{XD}}{\hat{\mathbf{R}}_{DD}} \tag{3.14}$$

where $\mathbf{a}_d$ is computed from previous estimations of f_d and τ or proper initializations. The beamvector in equation (3.14) is a weight vector computed in an early iteration, being the hybrid beamformer designed as an iterative algorithm [7, 8]. The expression for the weight vectors is obtained as

$$\hat{\mathbf{w}}_{hybrid} = \hat{\mathbf{R}}_{XX}^{-1}\hat{\mathbf{R}}_{XD}\alpha_d^* + \hat{\mathbf{R}}_{XX}^{-1}\mathbf{s}_d\left(\mathbf{s}_d^H\hat{\mathbf{R}}_{XX}^{-1}\mathbf{s}_d\right)^{-1}\left(1-\mathbf{s}_d^H\hat{\mathbf{R}}_{XX}^{-1}\hat{\mathbf{R}}_{XD}\alpha_d^*\right) \tag{3.15}$$

This result is a multiple beamforming which is a linear combination of two previously known results. On one hand,

$$\hat{\mathbf{w}}_{TRB} = \hat{\mathbf{R}}_{XX}^{-1}\hat{\mathbf{R}}_{XD}\alpha_d^* \tag{3.16}$$

is the multiple beamforming under the MSE criterion taking into account only the temporal reference. On the other hand,

$$\hat{\mathbf{w}}_{MVB} = \hat{\mathbf{R}}_{XX}^{-1}\mathbf{s}_d\left(\mathbf{s}_d^H\hat{\mathbf{R}}_{XX}^{-1}\mathbf{s}_d\right)^{-1} \tag{3.17}$$

is the minimum variance beamforming considering only the spatial information. These solutions have a different behaviour against multipath and interferences: while $\mathbf{w}_{TRB}$ tries to combine constructively the desired signal with the other replicas in order to increase the $SINR$, $\mathbf{w}_{MVB}$ combines destructively such signals to minimize the output signal power [9]. The presented hybrid beamforming combines these two behaviors to mitigate multipath and interferences.

3.2.3 Interferences and Multipath Mitigation

This section is devoted to the study of interferences and fading multipath impact in the overall performance. We first extend the signal model in equation (3.2) to include these phenomenons and then obtain the expression of the beamvector that maximizes the signal-to-interference-plus-noise ratio ($SINR$), which is a parameter to be maximized by the system.

In an N element antenna array, the received complex baseband signal is expressed in equation (3.2), this model can be extended to take into account the presence of interferences and fading multipath conditions. Hence,

$$\mathbf{x}(t) = \alpha_d a_d(t)\mathbf{s}_d(\theta,\psi) + \alpha_{mp}a_{mp}(t)\mathbf{s}_{mp}(\theta,\psi) + \alpha_I a_I(t)\mathbf{s}_I(\theta,\psi) + \mathbf{n}(t) \tag{3.18}$$

where

- $\mathbf{x} \in \mathbb{C}^{N\times 1}$ is the observed signal vector,
- α_d, α_{mp} and α_I are the amplitudes of the desired, multipath and interference signals respectively,
- $a_d(t)$ and $a_I(t)$ are the narrowband signals envelopes of the desired and interference signals respectively,
- $\mathbf{s}_d(\theta,\psi)$, $\mathbf{s}_{mp}(\theta,\psi)$, $\mathbf{s}_I(\theta,\psi) \in \mathbb{C}^{N\times 1}$ are the steering vectors of the desired, multipath and interference signals respectively, and
- $\mathbf{n}(t) \in \mathbb{C}^{N\times 1}$ represents additive noise and all other disturbing terms, like multipath of each signal or interferences.

For the sake of simplicity we define $\mathbf{s}_d = \mathbf{s}_d(\theta, \psi)$, $\mathbf{s}_{mp} = \mathbf{s}_{mp}(\theta, \psi)$ and $\mathbf{s}_I = \mathbf{s}_I(\theta, \psi)$. Additionally, the amplitude of the desired signal is normalized ($\alpha_d = 1$) being the amplitudes of the rest of the signals relative to that. Thus the signal model is

$$\begin{aligned} \mathbf{x}(t) &= a_d(t)\mathbf{s}_d + \alpha_{mp} a_d(t)\mathbf{s}_{mp} + \alpha_I a_I(t)\mathbf{s}_I + \mathbf{n}(t) \\ &= a_d(t)\left(\mathbf{s}_d + \alpha_{mp}\mathbf{s}_{mp}\right) + \alpha_I a_I(t)\mathbf{s}_I + \mathbf{n}(t) \\ &= a_d(t)\mathbf{s} + \alpha_I a_I(t)\mathbf{s}_I + \mathbf{n}(t) \end{aligned} \tag{3.19}$$

where be have defined an equivalent steering vector of the desired and multipath fading as $\mathbf{s} = \mathbf{s}_d + \alpha_{mp}\mathbf{s}_{mp}$

3.2.3.1 Maximization of the $SINR$

The output of the antenna array is given by the next expression

$$\mathbf{y}(t) = \mathbf{w}^H\mathbf{x}(t) = a_d(t)\mathbf{w}^H\mathbf{s} + \alpha_I a_I(t)\mathbf{w}^H\mathbf{s}_I + \mathbf{w}^H\mathbf{n}(t) \tag{3.20}$$

defining the desired signal and the noise plus interference contribution as

$$\begin{aligned} \mathbf{y}_d(t) &= a_d(t)\mathbf{w}^H\mathbf{s} \\ \mathbf{y}_{IN}(t) &= \alpha_I a_I(t)\mathbf{w}^H\mathbf{s}_I + \mathbf{w}^H\mathbf{n}(t) \end{aligned} \tag{3.21}$$

we can obtain its corresponding powers. The desired signal power is

$$P_d = E\{y_d(t)y_d(t)^*\} = \sigma_d^2\mathbf{w}^H\mathbf{s}\mathbf{s}^H\mathbf{w} \tag{3.22}$$

where $\sigma_d^2 = Ea_d(t)a_d(t)^*$ is the power of the desired signal. The power of the interference and noise contribution is then

$$P_{IN} = E\{y_{IN}(t)y_{IN}(t)^*\} = \mathbf{w}^H\mathbf{R}_{IN}\mathbf{w} \tag{3.23}$$

where $\mathbf{R}_{IN} = \sigma_I^2\mathbf{s}_I\mathbf{s}_I^H + \sigma_n^2\mathbf{I}$ and $\sigma_I^2 = |\alpha_I|^2E\{a_I(t)a_I(t)^*\}$ and σ_n^2 are the power of the interference signal and the noise respectively. The signal-to-interference-plus-noise radio (SINR) is defined as the relation between the power of the desired signal and the interferences plus noise

$$SINR = \frac{P_d}{P_{IN}} = \frac{\sigma_d^2\mathbf{w}^H\mathbf{s}\mathbf{s}^H\mathbf{w}}{\mathbf{w}^H\mathbf{R}_{IN}\mathbf{w}} \tag{3.24}$$

The problem of maximizing the $SINR$ is closely related to the *Rayleigh Quotient*. The Rayleight Quotient is expressed as

$$r(\mathbf{z}) = \frac{\mathbf{z}^H\mathbf{A}\mathbf{z}}{\mathbf{z}^H\mathbf{B}\mathbf{z}} \tag{3.25}$$

where A and B are symmetric matrices and $\mathbf{z}$ is a vector of proper dimension. A straightforward gradient computation with respect $\mathbf{z}^H$ yields the following relation

$$\mathbf{A}\mathbf{z} = r(\mathbf{z})\mathbf{B}\mathbf{z} \tag{3.26}$$

which is recognized as a *Generalised Eigenproblem.* The positive scalar $r(\mathbf{z})$ and the corresponding vector $\mathbf{z}$ are called the generalized eigenvalue and eigenvector, respectively, of the matrix pencil $(\mathbf{A}, \mathbf{B})$. The maximum value of $r(\mathbf{z})$ is given by the eigenvectors of the Generalized Eigenproblem. Identifying terms in equations (3.24) and (3.25) it is shown that the vector $\mathbf{w}$ that maximizes the $SINR$ is given by the eigenvector of the Generalized Eigenproblem. Thus, equation (3.26) is

$$\underbrace{\sigma_d^2 \mathbf{s}\mathbf{s}^H}_{\mathbf{A}} \underbrace{\mathbf{w}_0}_{\mathbf{z}} = \underbrace{SINR_0}_{r(\mathbf{z})} \underbrace{\mathbf{R}_{IN}}_{\mathbf{B}} \underbrace{\mathbf{w}_0}_{\mathbf{z}} \tag{3.27}$$

where $SINR_0$ is the optimum $SINR$ and $\mathbf{w}_0$ is the optimum beamvector in terms of maximizing the $SINR$. Substitution of $SINR_0$ in equation (3.27) yields to the following expression

$$\begin{aligned} \mathbf{s}\mathbf{s}^H \mathbf{w}_0 &= \sigma_d^{-2} \left(\frac{\sigma_d^2 \mathbf{w}_0^H \mathbf{s}\mathbf{s}^H \mathbf{w}_0}{\mathbf{w}_0^H \mathbf{R}_{IN} \mathbf{w}_0} \right) \mathbf{R}_{IN} \mathbf{w}_0 \\ &= \frac{\mathbf{w}_0^H \mathbf{s}\mathbf{s}^H \mathbf{w}_0}{\mathbf{w}_0^H \mathbf{R}_{IN} \mathbf{w}_0} \mathbf{R}_{IN} \mathbf{w}_0 \\ &= \kappa \mathbf{s}^H \mathbf{w}_0 \mathbf{R}_{IN} \mathbf{w}_0 \end{aligned}$$

where we have defined the scalar $\kappa = \frac{\mathbf{w}_0^H \mathbf{s}}{\mathbf{w}_0^H \mathbf{R}_{IN} \mathbf{w}_0}$ to obtain $\mathbf{s} = \kappa \mathbf{R}_{IN} \mathbf{w}_0$, from where it is straightforward to obtain the optimum beamvector in terms of $SINR$ maximization as

$$\mathbf{w}_0 = \frac{1}{K} \mathbf{R}_{IN}^{-1} \mathbf{s} \tag{3.28}$$

which can be proofed that is equivalent to the Temporal Reference Beamformer (TRB). This results shows that in the presence of interferences, the TRB provides the optimum $SINR$.

3.3 DBF Architecture and Digital Requirements

In section 3.2, three digital beamforming techniques have been analyzed theoretically. Every technique has advantages and inconvenients depending, basically, on the signal scenario. One of them, Minimum Variance Beamforming (MVB), uses spatial information, mainly the direction of arrival of the desired signal. Temporal Reference Beamforming (TRB) is based on the temporal structure of the desired signal. It computes the weight vector by minimizing the mean square error between the array output and a reference signal highly correlated with the desired signal. The third technique, Hybrid Beamforming (HB), is a combination of the two other techniques and it uses both, spatial reference and temporal reference. The three techniques obtain the same optimum solution for the weight vector, except for a scale factor, under ideal conditions for the signal scenario, that is, the direction arrival of the desired signal is perfectly known, multipath signal is not present, there are no antenna elements imperfections and the signal structure is also perfectly known. However, when the direction of arrival of the desired signal is roughly known, the pattern of the array elements have some imperfection, multipath is present or the mutual coupling is appreciable and unknown, the performance of MVB can be seriosly degraded. Moreover, MVB requires knowledge of the direction of arrival of the desired signal, so in mobile applications, an additional technique has to be used in other to estimate that angle. TRB does not suffer from these drawbacks but it requires additional temporal processing and some extra-hardware. HB, retains some properties of the other two techniques but it also suffer from the same problems that MVB, though in a less grade. Its interest lies in that while retaining some properties of the TRB, it presents a predetermined pattern in the quiescent state, that is, in absence of interferences.

S-DMB system, as described in section 3.1, is very appropiated for TRB as S-DMB standard transmits a pilot channel, which has three functions. The first is to transmit the unique word for frame synchronization and frame counter for super frame synchronization. The second is to send a pilot signal. The third is to transmit control data to facilitate the receiver functions. The pilot signal is perfectly known to the receiver so it can be used as a temporal reference signal which can be generated after the chip and frame timings are recovered.

As S-DMB uses CDMA modulation, there are two possibilities for computing the array weight vector: before despreading the pilot signal and with despread pilot signal.

3.3.1 DBF Architecture

Digital Beamforming cannot be considered as an independent subsystem of the entire receiver. It shares many elements with the whole system, as it will be shown, being application dependent.

The incident signals to the antenna elements are amplified and down converted to baseband or to a convenient intermediate frequency. All these operation are performed in the analogue domain by the front ends and they are identical to all the elements, maintaining the same phase reference. The output signal of each front end is digitized by the A/D converter. Digital beamforming is performed by a set of complex weights which are computed according to some criteria depending on the chosen technique. The output of the beamformer is then despread and data are obtained after demodulation.

The update of the weight vector is performed using the signals from the array elements and some side information which depends on the beamforming technique. For spatial reference beamforming as MVB and part of HB, the basic information is the steering vector associated to the desired signal direction of arrival. In addition, some other spatial constraints can be imposed, like derivative constraints, on the radiation pattern. In any case, the steering vector has to be known and in a mobile scenario it should be adaptively computed by some angle of arrival estimation technique. This issue along with the drawbacks mentioned above as element imperfections, mutual coupling, pointing errors, etc, make spatial reference technique quite unsuitable for beamforming in mobile communications systems. Moreover, these techniques are very sensitive to multipath environments which are inherent to all the mobile communications systems. For Temporal Reference Beamforming (TRB) the side information consist in a reference signal highly correlated with the desired signal and uncorrelated with the interference signals. This technique minimizes the mean square error between the array output and the reference and if the reference is the desired signal itself, it is also known as MMSE (Minimum Mean Square Error). It is insensitive to all the drawbacks of the spatial reference technique and obtains the best signal-to-interference-plus-noise ratio. In this sense TRB is the technique that best results achieves in a multipath environment. Eventually, the array output can be also used to update the weight vector depending on the implemented algorithm.

3.3.1.1 DBF After signal Despreading

As depicted in Figure 3.7, the TRB algorithm performs two DBF: one with the despread pilot signal and the other with the whole spreaded incoming signal to be delivered to the receiving system. The first is used to compute the optimum weights, which are copied in the latter. Actually, the output of the DBF for the despreaded pilot signal must only be computed when close-loop adaptive algorithms are used for computing the weights, which need the output of the beam former. When weights are computed by Direct Matrix Inversion, the calculation of the output is not necessary and can be avoided. The DBF module comprises weights calculation and reference generation. The synchronization of the signal can be provided by the receiver itself or self generated by the antenna array (a desirable property in cold-acquisition scenarios).

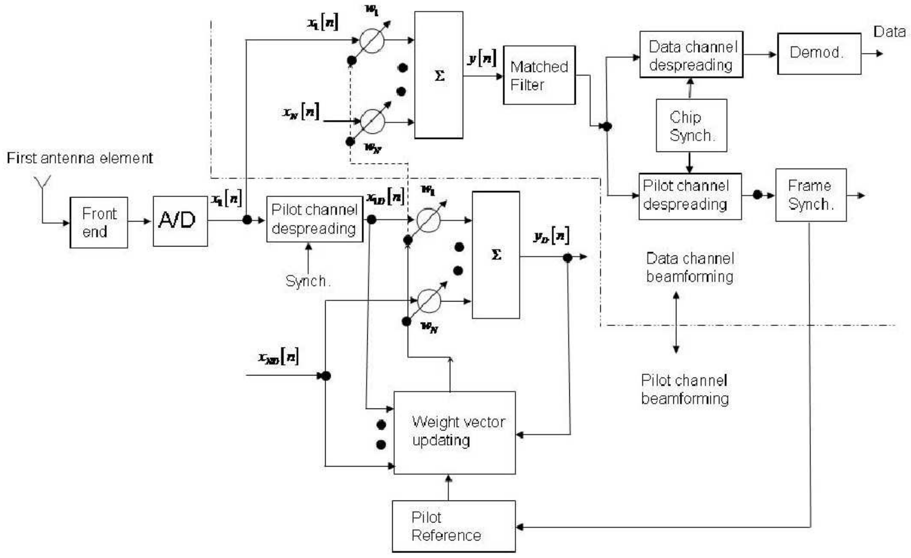


Fig. 3.7 System architecture diagram when Temporal Reference Beamforming (TRB) is performed after despreading the pilot signal.

After analog-to-digital conversion, the signal stream is splitted into two branches. One feeds the DBF for the communication system and the other feeds the module that calculates the DBF weights (in which we focus our attention). Taking into account both the spreading sequence of the pilot channel and the synchronization parameters to be provided by the receiver, the pilot signal is despreaded. Notice that despreading involves a high reduction of the data blocks to be dealt with and data rate can be much lower after this processing. There are two options when implementing the despreading block: whether to use a Matched Filter or not. A Matched Filter provides the optimum signal-to-noise ratio and should be implemented, in each branch of the array, with a digital filter of $8N_{sc}$ coefficients, being N_{sc} the number of samples per chip considered.

We have seen from simulations that a reduced number of Pilot Symbols must be processed under this architecture for the DBF to properly operate. This is on the order of 10 pilot symbols for optimum performance, that is, optimum beamforming. This performance is degraded less than 1 dB with 3 pilot symbol and less than 2 db with only 2 pilot symbol. A more detailed description is provided in the trade-off study of this Chapter. Denoting as N_{ps} the number of pilot symbols used, the information needed to be gathered for beamforming purposes is N_{ps} PRN sequences (2048-chips long or equivalently, 32-bits long) sampled with N_{sc} samples per chip. Thus, we have to deal with $N_{ps} \cdot 2048 \cdot N_{sc}$ samples and the time needed for recording data can go from $\sim$5% to $\sim$20% of a frame, i.e. from 0.6 to 2.5 ms. After despreading, the number of samples to be processed, for weight vector update, is reduced to $N_{ps} \cdot 32 \cdot N_{sc}$ samples, which is considerably lower.

Finally, notice that in the case of despreading before DBF, the reference signal must be also the despreaded pilot symbols. Pilot symbols are a sequence of 32-bit length continuing run of data 1 [1]. Thus, the reference signal is and all-1 sequence of length $N_{ps}{\cdot}32{\cdot}N_{sc}$. Then, the correlation between the received signal and the reference (needed in the computation of the TRB weights) can be done by adding the samples of the received signals, corresponding to the PS, at each antenna element and forming a vector $\hat{\mathbf{p}}$. The only requeriment is that we have to know when the PS starts, i.e. synchronism must be provided.

3.3.1.2 DBF Before signal Despreading

The block diagram of the DBF architecture, when it is performed with the spreaded signals, is depicted in Figure 3.8. The output of the beamformer is despreade and data is recovered after demodulation. These tasks must be done by the receiver.

One of the main drawbacks of this architecture is that the rate at which the beamforming weights must be computed is higher that in the case of the DBF after despreding architecture, because in this case spreaded signals at chip rate must be dealt with. However, despreading is not performed.

When this architecture is considered, the correlation between the reference and the incoming signals must be performed as exposed in equation (3.32) because the simplification done in equation (3.33) does not hold. In addition, this correlation is done at a higher rate.

As it is presented in the trade-off comparision of the two architectures, the beamforming before despreading needs more than ~50% of signal frame to achieve performances close to the optimum beamforming. However, with only 10 pilot symbols, the degradation is only 1 dB. With only 1 pilot symbol the degradation is on the order of 6 dB, approximately 4 dB more than the beamforming after despreading architecture.

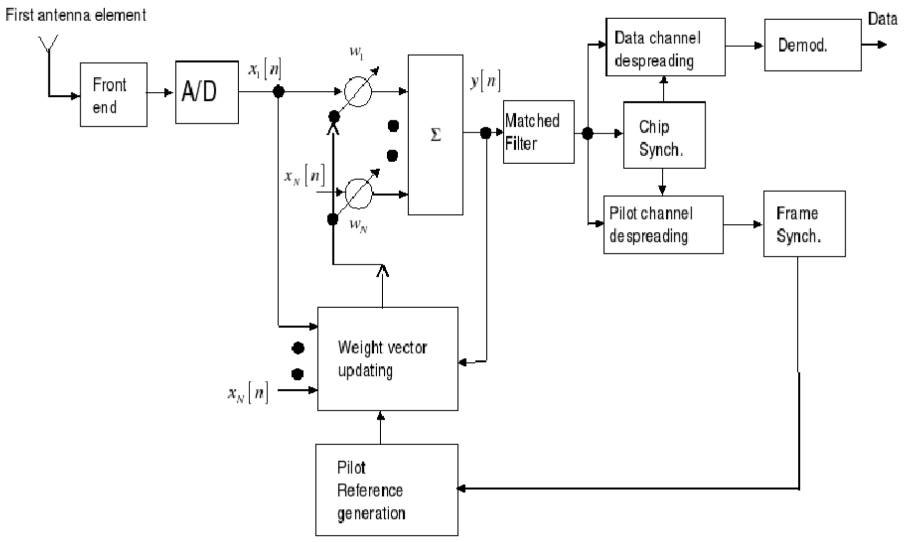


Fig. 3.8 System architecture diagram when Temporal Reference Beamforming (TRB) is performed before despreading the pilot signal.

3.3.2 Simulation results for DBF After and Before signal Despreading

First of all we are going to present some results of the optimum beamforming, which get the best performance in any signal scenario. Optimum beamforming assumes that the correlation matrix and the cross-correlation between the snapshot and temporal reference ($\hat{\mathbf{p}}$ vector) are perfectly known. Optimum beamforming performs quite similar for both architectures, so the result depicted below can be considered the same for both beamformings.

Next figures present the radiation pattern obtained with optimum beamforming using the array geometry described in 3.1 and using the antenna element provided by TTI. The signal scenario consists in the desired signal and two interferences. The desired signal is composed by 40 data channels, in fact, only one of these channels is the desired signal. The others are rather interferences as it will be below and the corresponding Pilot Channel. The signal-to-noise ratios are -16 dBs for the wanted signal and 10 dBs for the interferences. The elevation angle for the three signals is 20°. Three azimuths for the desired signal are considered and the interferences are 30° separated in azimuths from the desired signal.

Multipath is no present.

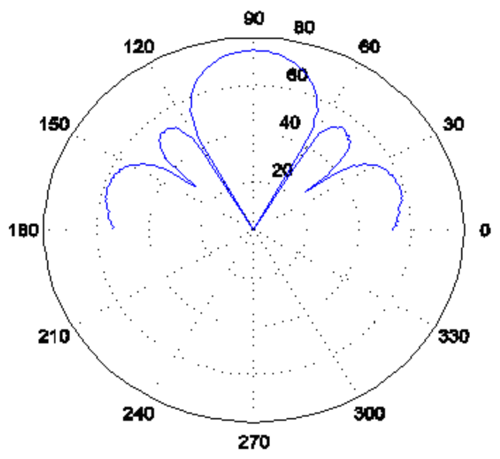


Fig. 3.9 Optimum Beamforming with conformal array, desired signal azimuth of 90°.

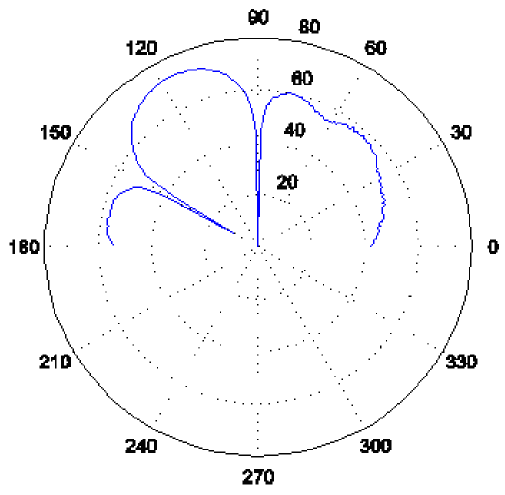


Fig. 3.10 Optimum Beamforming with conformal array, desired signal azimuth of 120°.

As it can be observed, the interferences are completely rejected. The gains with respect to the noise are 26.2, 25.1 and 20.2 dBs, respectively. The degradation form the broadside is due to that far from it, some antenna elements does not work properly. It is important to keep the array pointing to azimuths close to the array. However, it is important to remark that there is a wide range of azimuths where the array performs well.

Figure 3.12 present the pattern when one of the interference is only 8 degrees away from the wanted signal. The gain is 18.9 dBs, i.e. about dBs less than the case where two interferences are separated 30°. Any case, the gain is very high if we think that there are other 18 dBs more from the despread of the signal.

Comparing optimum beamforming and beamforming before and after despreading,

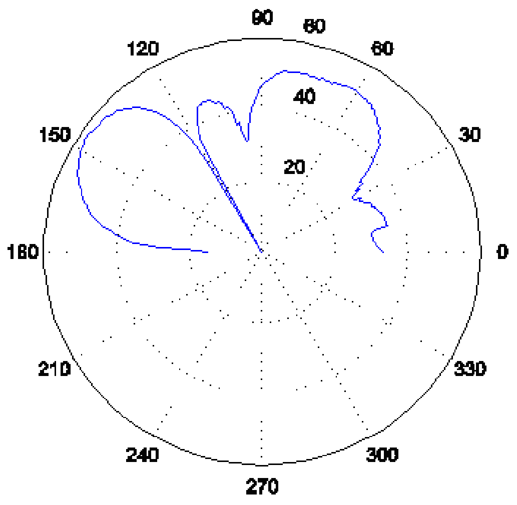


Fig. 3.11 Optimum Beamforming with conformal array, desired signal azimuth of 150°.

with 51 pilot symbols used in the two latter cases, the gains obtained are, respectively, 26.2, 26 and 26.2 dBs. There is no practical difference among the three beamformings. In Figures 3.13 and 3.14, the non-optimum beamformings are computed with only 1 pilot symbol. The gains are 20.3 and 24 dBs. As it was mentioned before, the beamforming after the signal despreading perform better than the one before signal despreading, with a few pilot symbols.

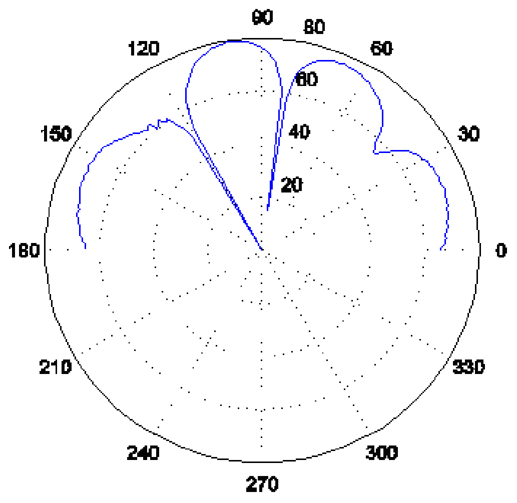


Fig. 3.12 Optimum Beamforming with conformal array, desired signal azimuth of 90°. One interference at 30° and the other at 8°.

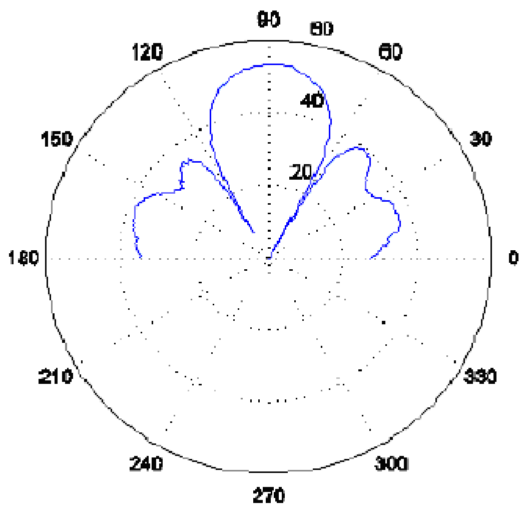


Fig. 3.13 Non-optimum Beamforming with conformal array, after despreading architecture, desired signal azimuth of 90^o. $N_{ps} = 1$

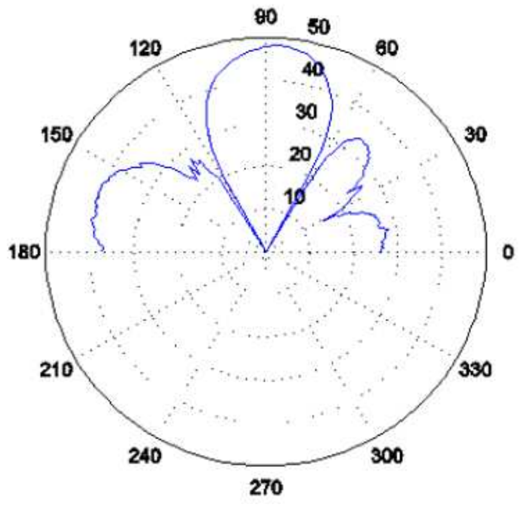


Fig. 3.14 Non-optimum Beamforming with conformal array, before despreading architecture, desired signal azimuth of 90^o. $N_{ps} = 1$

3.3.3 Trade-off between DBF After and Before signal Despreading

Simulation results have shown that both beamformings achieved optimum performance with enough length of sample. This performance is practically achieved in a frame duration, i.e., in 51 pilot symbols or equivalent in 12.75 ms. However, both attain near optimum beamforming with less pilot symbols. In particular, beamforming after signal despreading performs close optimum beamforming with about 10 pilot symbols ($\pm$2.5 ms). On the contrary, beamforming before despreading needs much more samples to get the same performance. This can be very important for tracking purposes and to deal with ESA requirements.

With respect to the computational burden, the following considerations can be taken into account:

(1) Beamforming after despreading has to despread signal. This operations consists in multiply every signal block, in every antenna element, with a duration of a pilot symbol by the spread pilot symbol, i.e., multiply 2048 received samples by a binary sequence of ± 1 corresponding to the spread pilot and then to sum up the products in groups of 64 samples. The result is a sequence of 32 samples. This spread pilot sequence is always the same and it does not need to be generated every block. Note that only the even blocks take part in the operation and that about 10 blocks are needed per each weight vector update. After the 10 blocks a set of 320 samples per antenna element are supplied for estimating the correlation matrix and the $\hat{\mathbf{p}}$ vector. The reference signal is a reference signal is a sequence of 32 ones per every block. Thus the $\hat{\mathbf{p}}$ vector is obtained just summing up the 320 samples in each antenna element.

(2) Beamforming before despreading does not despread the signals, it just gathers samples to estimate the correlation matrix and $\hat{\mathbf{p}}$ vector. Assuming about 25 pilot symbol block to achieved similar performance than beamforming after despreading, a total of about 50000 samples are needed to estimate the correlation matrix and $\hat{\mathbf{p}}$ vector. This can be very cumbersome as many operations are involved in the above estimates.

After estimating the correlation matrix and $\hat{\mathbf{p}}$ vector, the computational burden for both beamformings is the same.

From the above considerations it seems, at a first glance, that Beamforming after despreading is computationally more efficient, as it presents much better performance, with a few pilot symbols, which seems to lie in the fact that despreading by the orthogonal Walsh codes removes the multiple access interference, i.e., the signals belonging to the own system which comes from the same direction (which cannot be mitigated with an antenna array). As we mentioned before, in our simulations we have considered 29 signals leaving the same satellite.

3.3.4 Digital requirements

In this section, some issues regarding the practical implementation of the DBF Platform are discussed taking into account that the system architecture performs DBF before despreading the signal.

- The ideal case is when the **Matched Filter** is applied in every array arm, i.e. before beamforming. However, this is cumbersome and a high computational cost is required. In addition, the beamforming can operate without the Matched Filter, which can be applied to the output of the antenna array. For these reasons, the Matched Filter is introduced after DBF, as sketched in Figure 3.7.

- The **Number of Samples per Chip** (N_{sc}) is an important parameter which highly depends on the communication system of the target application, i.e. the required Bit Error Rate (BER) after demodulation of the signal. Nevertheless, it is accepted that $N_{sc} \geq 2$ samples per chip with interpolation. A general rule of thumb is to consider $N_{sc} = 4$ samples per chip.
- The **Number of Bits** considered in the ADCs has different values whether we are quantizing the signal (N_b) or the weight vector (N_{bw}). In what follows, we are considering 8 bits for the quantization of signal and weight vector.
- The **Basic Operations** that the DBF must be able to do are complex products, one for each antenna array element, and a sum after these products. Thus, the operations involved in the weighting of the array inputs are 4 real products and 3 real sums, for each N antenna element.

$$\begin{aligned} y[n] &= \mathbf{w}^H \mathbf{x}[n] = \sum_{l=1}^{N} w_l^* x_l[n] \\ &= \sum_{l=1}^{N} \left[w_l^R x_l^R[n] + w_l^I x_l^I[n] \right] + j \left[w_l^R x_l^I[n] + w_l^I x_l^R[n] \right] \end{aligned} \quad (3.29)$$

where $(.)^*$ denotes complex conjugate and $(.)^R$ and $(.)^L$ the real and imaginary parts of a complex number, respectively. Since we are considering $N_b = N_{bw} = 8$ bits for quantizing the signal and the weights, the main computational limitation is not due to the operations involved in equation (3.29) but to the stream of bits to be dealt with, i.e. the beamforming must keep the data flux constant before and after this operation.

- The **Weight Update Rate** depends on the scenario considered. Hence, in a mobile communications channel whose parameters vary fast the weight update must be performed at a higher rate than in a more static scenario. Indeed, this rate can be different of the data rate (and generally much more lower). Taking this into account, no matter the DBF technique considered, there are many alternatives to update the weight vector [6]. The fastest approach is to directly compute the inverse of the correlation matrix appearing in the weights equations, with a signal block of K samples. This is known in the literature as the Sample Matrix Inversion (SMI) technique, also found in the literature as the Direct Matrix Inversion (DMI). Thus, we recall that the equation for computing the optimum weights in the TRB is given by

$$\hat{\mathbf{w}}_{TRB} = \hat{\mathbf{R}}_{XX}^{-1} \hat{\mathbf{p}} \quad (3.30)$$

where α_d has been normalized. $\hat{\mathbf{R}}_{XX}$ is the estimation of the autocorrelation matrix, i.e. a $N \times N$ matrix estimated as

$$\begin{aligned}\hat{\mathbf{R}}_{XX} &= \frac{1}{K}\mathbf{X}\mathbf{X}^H = \frac{1}{K}\left[\mathbf{x}[1] \quad \mathbf{x}[2] \quad \ldots \quad \mathbf{x}[K]\right]\left[\mathbf{x}[1] \quad \mathbf{x}[2] \quad \ldots \quad \mathbf{x}[K]\right]^H \qquad (3.31)\\ &= \frac{1}{K}\begin{pmatrix} x_1[1] & x_1[2] & \ldots & x_1[K] \\ x_2[1] & x_2[2] & \ldots & x_2[K] \\ \vdots & \vdots & \ddots & \vdots \\ x_N[1] & x_N[2] & \ldots & x_N[K]\end{pmatrix}\begin{pmatrix} x_1[1] & x_1[2] & \ldots & x_1[K] \\ x_2[1] & x_2[2] & \ldots & x_2[K] \\ \vdots & \vdots & \ddots & \vdots \\ x_N[1] & x_N[2] & \ldots & x_N[K]\end{pmatrix}^H\end{aligned}$$

where $x_i[n]$ refers to the *n-th* snapshot of the *i-th* element of the array. Thus, the number of products required for the product of a $N \times K$ matrix by a $K \times N$ matrix is of $K \times N^2$, and the number of sums $(K-1) \times N^2$. The correlation between the received signal and the locally generated reference is expressed as $\hat{\mathbf{p}}$ and defined as

$$\begin{aligned}\hat{\mathbf{p}} &= \frac{1}{K}\mathbf{X}a_d^H = \frac{1}{K}\left[\mathbf{x}[1] \quad \mathbf{x}[2] \quad \ldots \quad \mathbf{x}[K]\right]\left[a_d[1] \quad a_d[2] \quad \ldots \quad a_d[K]\right]^H\\ &= \frac{1}{K}\begin{pmatrix} x_1[1] & x_1[2] & \ldots & x_1[K] \\ x_2[1] & x_2[2] & \ldots & x_2[K] \\ \vdots & \vdots & \ddots & \vdots \\ x_N[1] & x_N[2] & \ldots & x_N[K]\end{pmatrix}\begin{pmatrix} a_d^*[1] \\ a_d^*[2] \\ \vdots \\ a_d^*[K]\end{pmatrix}\\ &= \frac{1}{K}\begin{pmatrix} x_1[1]\cdot a_d^*[1] + x_1[2]\cdot a_d^*[2] + \ldots + x_1[K]\cdot a_d^*[K] \\ x_2[1]\cdot a_d^*[1] + x_2[2]\cdot a_d^*[2] + \ldots + x_2[K]\cdot a_d^*[K] \\ \vdots \\ x_N[1]\cdot a_d^*[1] + x_N[2]\cdot a_d^*[2] + \ldots + x_N[K]\cdot a_d^*[K]\end{pmatrix} \qquad (3.32)\end{aligned}$$

this $N \times 1$ vector needs of $K \times N$ products and $(K-1) \times N$ sums to be computed. The minimum length of the data sequence needed, i.e. K, is to be determined, though according to simulation results a minimum of 2048 snapshots are required for the architecture considering beamforming after despreading and higher for the before despreading architecture. Notice that these K snapshots can be taken at a lower rate than the sampling needed in the ADCs for demodulation purposes. Thus, the rate of the beamforming can be decreased, in the order of approx. 16 times. The DBF after despreading architecture is considered, vector $\hat{\mathbf{P}}$ can be computed in a more efficient way. Taking into account that the despreaded samples of the reference are all 1, equation (3.32) can be reduced to

$$\hat{\mathbf{p}} = \frac{1}{K}\begin{pmatrix} x_1[1] + x_1[2] + \ldots + x_1[K] \\ x_2[1] + x_2[2] + \ldots + x_2[K] \\ \vdots \\ x_N[1] + x_N[2] + \ldots + x_N[K]\end{pmatrix} \qquad (3.33)$$

which is quiet easier to implement, only $(K-1) \times N$ sums must be calculated. ESA requirements state that the DBF must be able to track angular velocities of

45^o/s. Assuming (to be specified) that the DBF can cope with maladjustments of 1^o between the scenario considered in the weight calculation and the scenario being dealt with, a realistic maximum update time is 22 ms, which is approx. equivalent to 2 frames.

- The **Reference Signal** is the pilot signal defined in [1]. In the specifications of the pilot signal, there are a priori known fields and some control bits which must be demodulated, i.e. unknown. The reference generation takes only into account the known part of the pilot channel, though the unknown data bits of this signal can also be used after demodulation introducing an extra delay in the weight update. Hence, the locally generated reference must have a sampling rate equal to the sampling rate of the input signal of the **Weight Vector Update**. The reference signal should be considered spreaded or not spreaded depending on the type of DBF architecture considered. A review of the pilot signal, used as the reference signal is provided in section 3.1.

4

DBF Platform prototyping and implementation

This chapter provides a detailed description of the hardware implementation of the DBF algorithm in the digital platform. Due to the large number of parallel instructions and functions to be implemented, a Field Programmable Gate Array (FPGA) device is considered as the digital platform. According to Chapter 3, the more suitable architecture to be implemented in Tracking mode, for the S-DMB system, is that of DBF after pilot signal despreading that is, only the pilot channel is despreaded and used for weight vector computation. The proposed DBF is composed of several blocks which are explained in greater detail during this Chapter. Basically, digitized signals of each radiating element are introduced in parallel in the FPGA. The designed system is composed of 2 operation modes: Acquisition and Tracking. On the one hand, Acquisition mode considers the acquisition of synchronism. On the other hand, Tracking mode starts when synchronism is locked and the TRB operates to fed the communications system with reliable data.

Due to technological limitations, it is not possible to integrate the design of these two operation modes in a single FPGA device, being mandatory to split the design in at least 2 FPGA platforms. The first FPGA is in charge of performing I&Q separation, element selection, weight vector computation and digital beamforming. The second FPGA performs matched filtering and the acquisition of synchronism, at the output of the antenna array.

In what follows, we describe the operation of the two modes (Acquisition and Tracking). We will use the word "hardware" to talk about the blocks programmed in VHDL and we will use the word "software" to talk about the MicroBlaze.

The following notation has been considered in the diagrams shown along the Chapter:

- FPGA platform in black square.
- VHDL implementation within the FPGA in blue square.

- Microcontroller implementation within the FPGA in red square.
- N: number of radiating elements. We have an array of 40 antennas ($N = 40$).
- N_p: number of processed radiating elements. We select 12 antennas in order to perform the Beamforming ($N_p = 12$).
- N_b: number of quantization bits. We use 8 bits to quantify each sample ($N_b = 8$).
- N_{sc}: number of samples per chip. Every chip has 4 samples ($N_{sc} = 4$).
- N_c: number of chips in a PRN. The PRN sequence has 2048 chips ($N_c = 2048$).
- $c_{pc}[n]$: PRN of the Pilot Channel (spreading sequence)
- $\mathbf{y} = \mathbf{w}^H\mathbf{x}$: array output

4.1 Modes of operation

4.1.1 Acquisition mode of operation

Prior to the use of the Temporal Reference Beamforming (TRB) technique, an Acquisition stage must be considered. This block should deal with two major issues in the DBF platform:

1. I&Q splitting.
2. Provide a realiable data stream for coarse cold synchronism estimation.

In figure 4.1, a block diagram description of the proposed architecture is shown. It involves 2 FPGA platforms, whose main goals are:

(1) $FPGA_1$:

a. Obtain I&Q components from IF-sampled signal.

b. Select the *i-th* antenna element. All N elements will be sequentially scanned.

c. Compute the array output. In this mode the weight vector will be zero for all elements except for the selected element. Hence, for each selected element the output will be the data input stream of each element.

$$\mathbf{w}_i = \begin{bmatrix} 0_1 & \dots & 1_i & \dots & 0_{N_p} \end{bmatrix}^T \tag{4.1}$$

$$y = \mathbf{w}^H\mathbf{x} = x_i \tag{4.2}$$

(2) $FPGA_2$:

d. Apply the Matched Filter to I&Q components.

e. Correlate signal with the Pilot Channel spread spectrum code.

f. Estimate the synchronism for the current element.

g. Determine the synchronism among scanned elements, when all N elements have been processed.

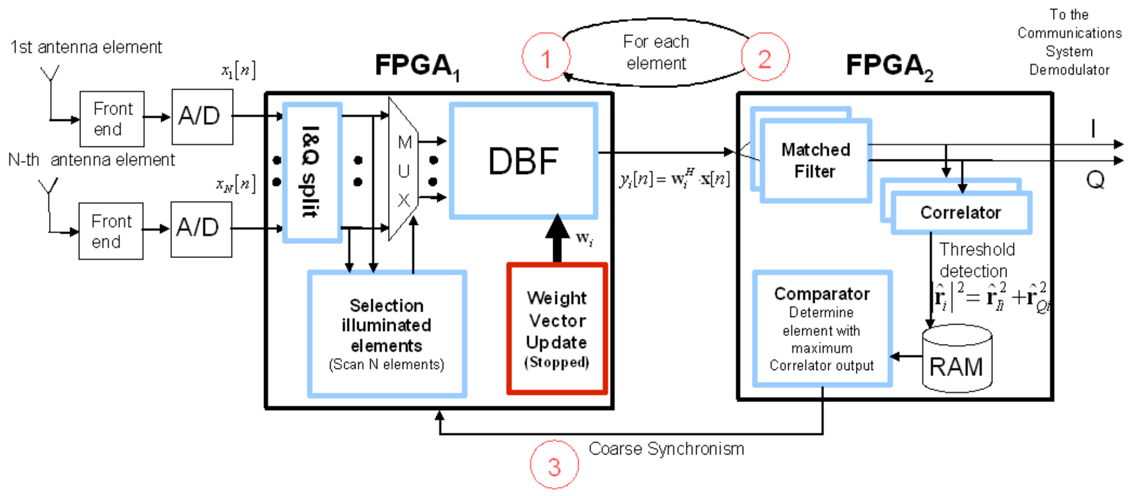

1st antenna element
N-th antenna element
Front end
A/D
$x_1[n]$
$x_N[n]$
FPGA$_1$
I&Q split
MUX
DBF
Selection illuminated elements (Scan N elements)
Weight Vector Update (Stopped)
w_i
$y_i[n] = w_i^H \cdot x[n]$
For each element
1
2
3
FPGA$_2$
Matched Filter
Correlator
Threshold detection
$|\hat{r}_i|^2 = \hat{r}_{Ii}^2 + \hat{r}_{Qi}^2$
RAM
Comparator Determine element with maximum Correlator output
To the Communications System Demodulator
I
Q
Coarse Synchronism

Fig. 4.1 Diagram flow of the Proposed Digital Platform desing (Acquisition mode).

The operation in this mode is as follows:

1. A element *i-th* is selected.
2. A coarse synchronism for the *i-th* element is estimated and stored.
3. After all elements have been processed, the best synchronism is chosen and Acquisition mode ends. Then, Tracking mode, whose operation is now sketched, is turned on.

4.1.2 Tracking mode of operation

When a synchronism is available, the Temporal Reference Beamforming (TRB) technique is ready to be used in the weight vector update block. This mode of operation is referred to as Tracking mode and is in charge of:

4. I&Q splitting.
5. Despreading the Pilot Channel with the proper synchronism.
6. Select the radiating elements to be processed.
7. Perform TRB to steer the radiation pattern antenna.
8. Tracking of the environment in terms of angular speed.

In Figure 4.2, a block diagram is shown. Notice that $FPGA_2$ is not used in this mode of operation.

(1) $FPGA_1$:

a. Obtain I&Q components from IF-sampled signal.

b. Despread Pilot Channel signal.

c. Compute the $\hat{\mathbf{p}}$ vector.

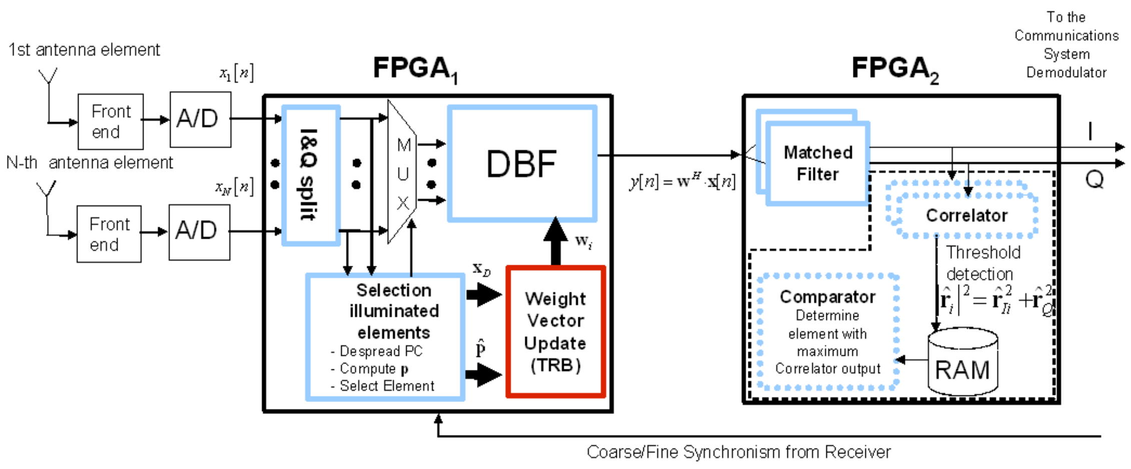
d. Select the illuminated elements to be processed.

e. Compute weights according to the Temporal Reference Beamforming (TRB) technique with the despreaded Pilot Channel signal of illuminated elements.

f. Compute the array output, $\mathbf{y} = \mathbf{w}^H \mathbf{x}$.

(2) $FPGA_2$:

g. Apply the Matched Filter to each I&Q component.



1st antenna element
N-th antenna element
Front end
A/D
$x_1[n]$
$x_N[n]$
FPGA$_1$
I&Q split
M U X
DBF
$\mathbf{w}_i$
$\mathbf{x}_D$
$\hat{\mathbf{p}}$
Selection illuminated elements
- Despread PC
- Compute $\mathbf{p}$
- Select Element
Weight Vector Update (TRB)
$y[n] = \mathbf{w}^H \cdot \mathbf{x}[n]$
FPGA$_2$
To the Communications System Demodulator
Matched Filter
I
Q
Correlator
Threshold detection
$|\hat{\mathbf{r}}_i|^2 = \hat{\mathbf{r}}_{Ii}^2 + \hat{\mathbf{r}}_Q^2$
Comparator
Determine element with maximum Correlator output
RAM
Coarse/Fine Synchronism from Receiver

Fig. 4.2 Diagram flow of the Proposed Digital Platform desing (Tracking mode).

The operation in this mode is as follows: after I&Q components splitting, the Pilot Channel is despreaded for all elements and vector $\hat{\mathbf{p}}$ is computed. With vector $\hat{\mathbf{p}}$ a selection of the illuminated elements is done and the weights (according to the TRB technique) are computed for these elements. The output is delivered to $FPGA_2$ directly.

The following section deals with the detailed specification of the blocks involved in the overall digital platform. These blocks have been introduced in Figures 4.1 and 4.2 and they are the following ones:

(1) $FPGA_1$:

a. I&Q splitting block (hardware).

b. Selection illuminated elements block (hardware).

c. Weight Vector Update block (software).

d. DBF block (hardware).

e. Communication between hardware-software blocks (hardware).

(2) $FPGA_2$:

a. Matched Filter block (hardware).

b. Correlator and Comparator blocks (hardware).

4.2 Detailed Block description of $FPGA_1$

A block diagram of the first FPGA is presented here. We have to differentiate the 2 modes of operation.

Firstly, in acquisition mode, an I&Q splitting is performed for each element and delivered to the $FPGA_2$. The Selection of illuminated elements block is scanning the N elements and the Weight Vector Update block is not operating in this mode. The DBF block multiplies the data input by the next weight vector: $\mathbf{w}_i = \begin{bmatrix} 0_1 & \dots & 1_i & \dots & 0_{N_p} \end{bmatrix}^T$, already commented. Figure 4.4 shows the diagram operation.

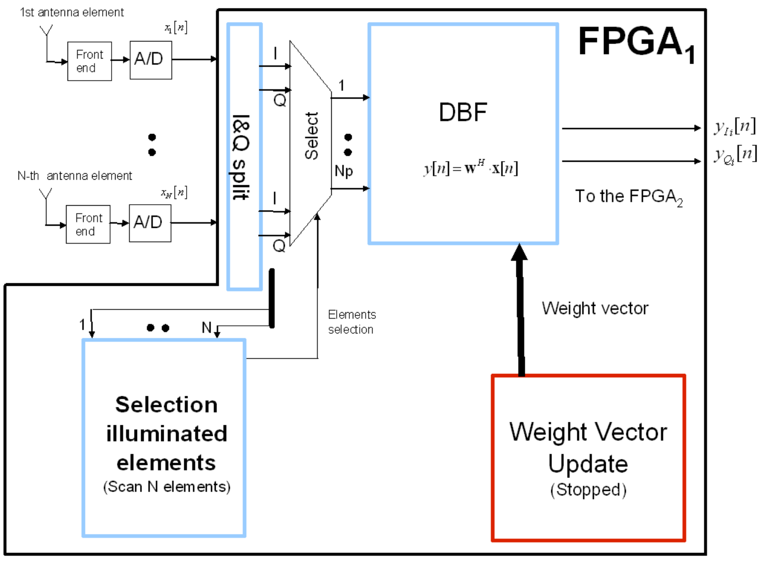


Fig. 4.3 Internal operation of $FPGA_1$ in the acquisition mode. The Selection illuminated elements block is in charge of select de element.

Secondly, when synchronism is available from $FPGA_2$, the operation of $FPGA_1$ operation is shown in Figure 4.4. The Tracking mode is activated and the TRB technique is used. However, prior to the weight computation, the Selection illuminated elements block is in charge of despreading the Pilot Channel signal, $\mathbf{X}_d$, computing the $\hat{\mathbf{p}}$ vector and selecting the illuminated elements.

4.2.1 I&Q splitting block

The first component of $FPGA_1$ is the I&Q splitting device. It is a hardware block and hence, programmed in VHDL.

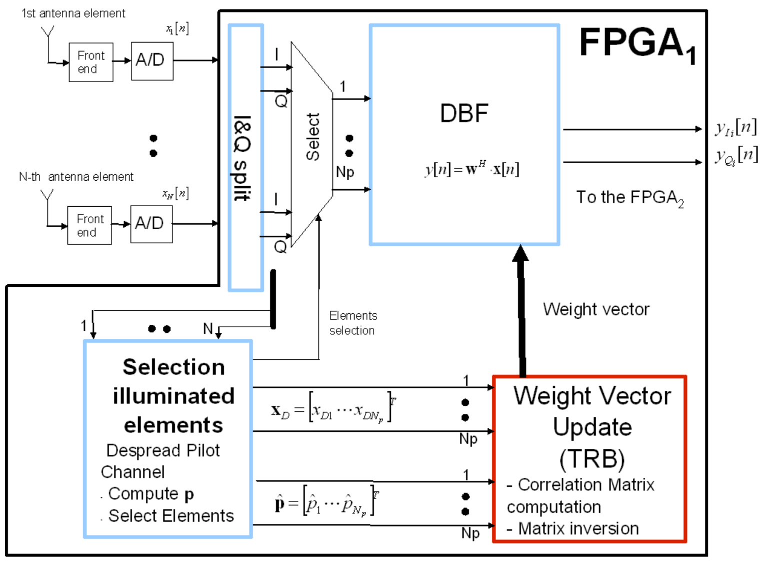


Fig. 4.4 Internal operation of $FPGA_1$ in the tracking mode. The Selection illuminated elements block is in charge of despreading the Pilot Channel, computing the $\hat{\mathbf{p}}$ vector and selecting the illuminated elements.

The A/D conversion can be performed whether in baseband or using IF-sampling. The first approach needs, at least, 2 samples per chip and two ADC are required for each array arm (one for each I&Q component). When using IF sampling, the minimum number of samples per chip is 4 but only one ADC is needed, at expenses of some extra digital processing which increases the computational cost of operation. In the proposed design, the IF-sampling philosophy is considered.

This technique has not the typical problems of the conventional analogical demodulator I&Q components, observe figure 4.5. We refer to the possible misalignment in the ideal 90° phase between I and Q arms in conventional A/D converters. Since IF-sampling performs I&Q separation in the digital domain, this problem is not present.

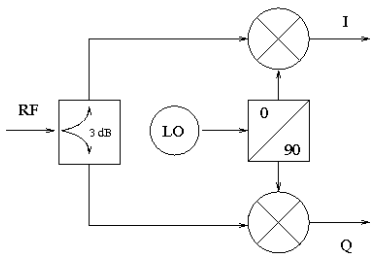


Fig. 4.5 Analogue demodulator I&Q components.

In order to prevent from aliasing, only the sampling frequencies corresponding to 4 and 8 samples per chip are suitable. So, for f_s = 65.536 MHz a proper intermediate frequency is f_{IF} = 49.152 MHz. Thus, the undersampling centres the signal bands, by aliasing, in f_c = 16.384 MHz which is the chip rate but also one forth of the sampling frequency ($f_c = f_s/4$). The relation among all those frequencies and signals spectra is depicted in Figure 4.6. After IF-sampling in 4.6.a, a digital frequency translation can be performed carrying the spectra to zero frequency. This is achieved in the digital domain multiplying the digitalized signal by $e^{\pm jn\pi/2}$. Figure 4.6.b shows the digital signal spectrum when multiplying by $e^{jn\pi/2}$. Note that the signal is cyclically multiplied by $\{1, j, -1, -j\}$ which consist in taking the even samples alternating the sign as the I component and the odd samples, also alternating the sign, as the Q component. The odd samples of the I component are null as well as the even samples of the Q components. Figure 4.7 illustrates the behavior of this block. The four blue squares (1,2,3,4) correspond to the four samples per chip.

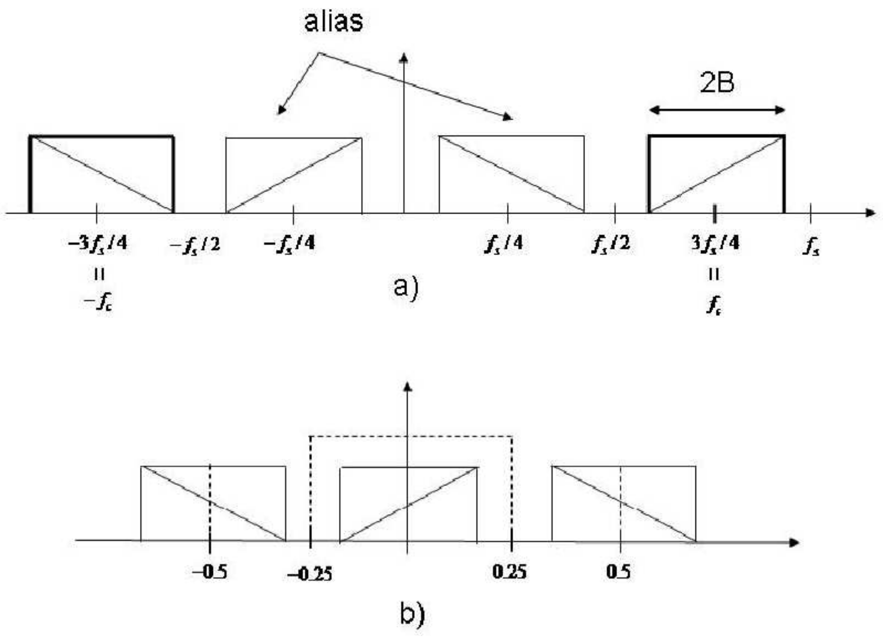


Fig. 4.6 IF sampling. (a) Analogue Spectrum and corresponding alias after undersampling. (b) Digital Spectrum after translation to the right.

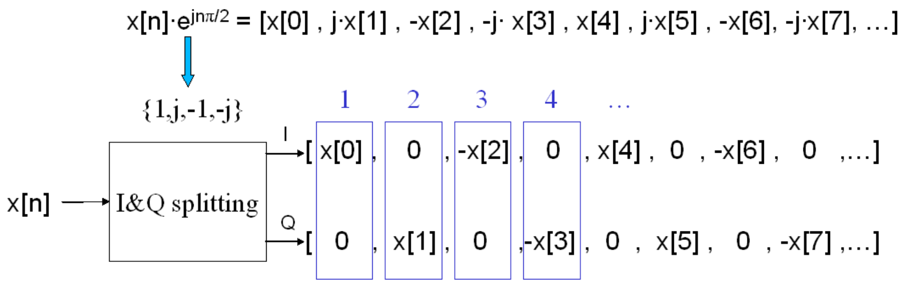

x[n]·e^{jnπ/2} = [x[0] , j·x[1] , -x[2] , -j· x[3] , x[4] , j·x[5] , -x[6], -j·x[7], ...]
{1,j,-1,-j}
1 2 3 4 ...
I
[x[0] , 0 , -x[2] , 0 , x[4] , 0 , -x[6] , 0 ,...]
x[n]
I&Q splitting
Q
[0 , x[1] , 0 , -x[3] , 0 , x[5] , 0 , -x[7] ,...]

Fig. 4.7 Graphical behavior of the I&Q splitting block.

Figure 4.8 shows the block programmed in VHDL. It is made up of four parallel blocks and each block corresponds to one sample of the chip. The blocks have the same input frequency $f_s = 65.536$ MHz but they operate at $f_c = 16.384$ MHz. It means that the outputs called *FLAGOUT_XyX*, which indicates that one sample have been processed, will be high one fourth of the time. Hence, although the input frequency is f_s, the rest of the system will work at f_c. The blocks *ONE_FOUR_CYCLE* and *THREE_FOUR_CYCLE* correspond to the phase component (I-component) and the blocks called *TWO_FOUR_CYCLE* and *FOUR_FOUR_CYCLE* correspond to the quadrature component (Q-component) in accordance to the system described in Figure 4.7 .

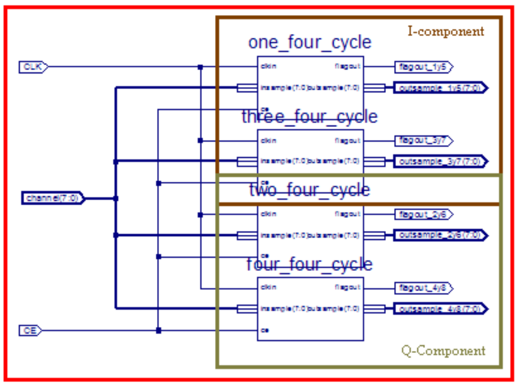


Fig. 4.8 Schematic description of the I&Q splitting block.

No simulation is showed due to the simplicity of the process.

4.2.2 Selection illuminated elements block

The input of this block is a data stream coming from I&Q splitting component of each element. Figure 4.9 shows the operation of the block with a general i stream of data. Basically, a correlation is performed with the known spreading sequence of the Pilot Channel, but in a sequential manner. Each sample of the stream is multiplied by the corresponding spreading code chip (since synchronism is available) and an accumulator is used to add all samples of the stream. When $64 \cdot N_{sc}$ samples have been added, the block outputs a despreaded bit, and when $2048 \cdot N_{sc}$ samples have been added, the block outputs the *i-th* element of the $\hat{\mathbf{p}}$ vector. In addition, when the $\hat{\mathbf{p}}$ vector is computed, the N_p major values correspond to the illuminated elements, which are then selected and used in the Weight Vector Update block.

The schematic implementation is composed of an accumulator of 64 samples (computting the spreading of each channel), an accumulator of 32 samples (computing the corresponding element of the $\hat{\mathbf{p}}$ vector) and a script which controls the chain of accumulators. The implementation design consumes a number of 1500 slices for each schematic shown in Figure 4.9. Thus, we will need N (40 antennas) similar blocks in parallel, one for each antenna being phase and quadrature considered in the schematic. Approximately, 60000 slices are needed for the overall design of the despreading plus $\hat{\mathbf{p}}$ vector computation.

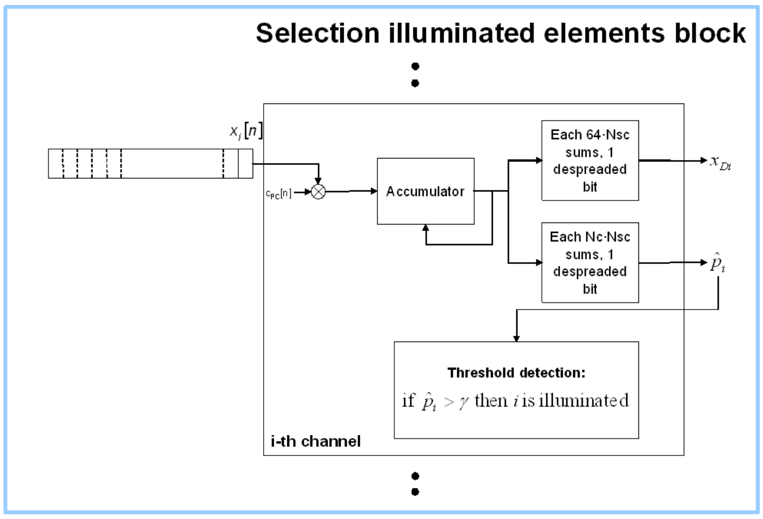

Selection illuminated elements block
$x_i[n]$
$c_{PC}[n]$
Accumulator
Each 64·Nsc sums, 1 despreaded bit
Each Nc·Nsc sums, 1 despreaded bit
x_{Di}
$\hat{p}_i$
Threshold detection:
if $\hat{p}_i > \gamma$ then i is illuminated
i-th channel

Fig. 4.9 Internal operation of the Selection illuminated elements block.

Figure 4.10 shows the schematic description of a single channel, as shown in Figure 4.9. It has been implemented the I&Q splitting, the PRN sequence product and the computation of the $\hat{\mathbf{p}}$ vector. The implementation is analogous to that of Figure 4.9, where the accumulator has been used. The outputs of this blocks are the corresponding despreaded Pilot Symbols and the corresponding element in the $\hat{\mathbf{p}}$ vector from each element. For the sake of clarity, here we present a schematic description of the block when there are 5 elements (notice that there will be 40 in the prototype) to be processed. Figure 4.11 shows how to use the blocks presented in Figure 4.10, after despreadings and $\hat{\mathbf{p}}$ vector are computed for each element, a multiplexor is in charge of feeding data to the $\hat{\mathbf{p}}$ vector modulus computation block and to a comparator, which sorts the elements in increasing order of highest $\hat{\mathbf{p}}$ vector modulus. This data is then provided to the embedded microcontroller.

In Figures 4.12 to 4.16 a simulation screen is provided. There one can see a control flag that is used to depict when the processed data is from the Pilot Symbol (that is used as a reference signal for the TRB) or it is control data from the Pilot channel that is not used in the TRB (actually, this period of time is used to select the elements to be processed and to compute weight vector update). Notice that only one third of a frame is required for the selection of elements and weight computation, approximately.

Figure 4.12 describes the first simulation of the Selection illuminated elements block for 40 elements. For this system we have 3 inputs and 9 outputs. We now to explain the meaning and functions of the defined signals to understand the simulation figures.

(1) INPUTS:

1. CLK: input clock, 65.536 MHz.
2. CE: chip enable, always enabled.

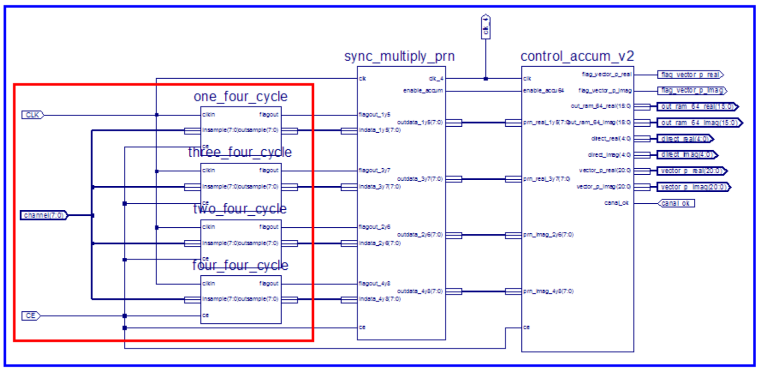


Fig. 4.10 Schematic description of the Selection illuminated elements block.

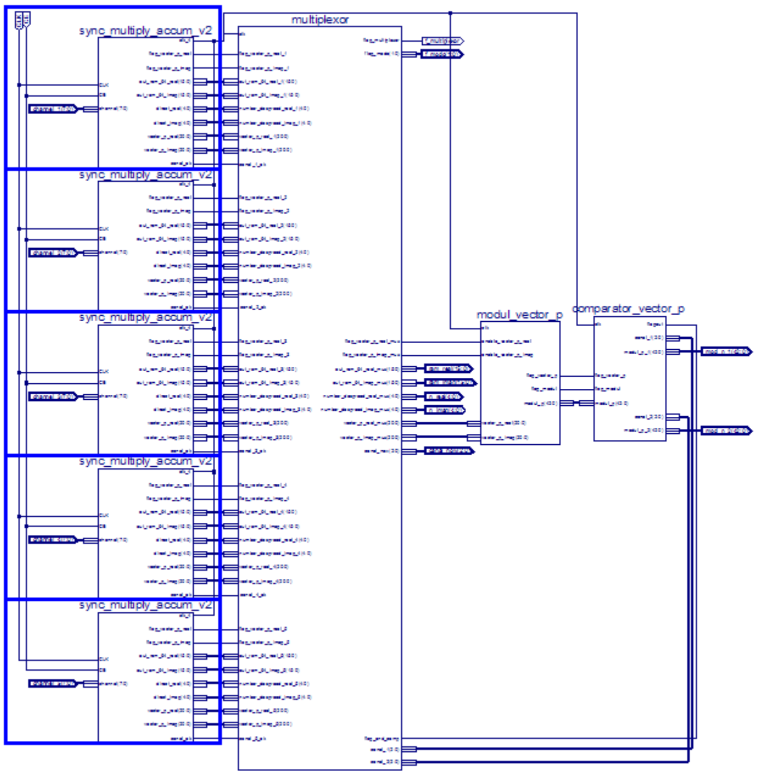


Fig. 4.11 Schematic description for the Selection illuminated elements block for 5 elements.

3. *DADES_40_CHANNELS*: data input spreaded. Not drawn in the simulation because are not relevant.

(2) OUTPUTS:

3. *FLAG_MULTIPLEXOR*: square signal to indicate the kind of data input. When it is low means that a Pilot Symbol is incoming and when is high means that a Control Data is incoming. In the latter, this signal controls to calculate the despreads, the $\hat{\mathbf{p}}$ vector and select the 12

channels.

4. *FLAG_MODE*: signal to indicate the mode operation of the system. We have 4 possible cases:
    (a) Case "0": Pilot Symbol incoming and selection module disabled.
    (b) Case "1": Control Data incoming. Computation of the $\hat{\mathbf{p}}$ vector for the 40 channels and selection of the 12 channels with the highest value.
    (c) Case "2": Control Data incoming. Communication with the MicroBlaze to pass the value of the computed despreads and $\hat{\mathbf{p}}$ vector.
    (d) Case "3": Control Data incoming. Nothing to do (waiting the computation of the weights).
5. *NUMBER_CHANNEL*: this signal shows the channel which is being processed.
6. *DESPREAD_REAL*: shows the value of the computed despread real.
7. *NUMBER_DESPREAD_REAL*: this signal shows the index of the computed despread real (a value in the interval [0,2047]).
8. *DESPREAD_IMAG*: shows the value of the computed despread imaginary.
9. *NUMBER_DESPREAD_IMAG*: this signal shows the index of the calculated despread imaginary (a value in the interval [0,2047]).
10. *P_VECTOR_REAL*: this signal shows the calculated real part of the $\hat{\mathbf{p}}$ vector.
11. *P_VECTOR_IMAG*: this signal shows the calculated imaginary part of the $\hat{\mathbf{p}}$ vector.

Figure 4.12 shows the simulation of 4 Pilot Signals and 4 Control Data. The *FLAG_MULTIPLEXOR* varies between "0" and "1" according to the incoming data. Also the *FLAG_MODE* signal changes according to the operation stage. When the *FLAG_MULTIPLEXOR* changes of "0" to "1" we observe that the system starts to compute the values. It spent more or less one third of the period of *FLAG_MULTIPLEXOR*. This is an important point because in hardware the selection is faster than in software, otherwise we would have to wait 1 Pilot Symbol more to pass the computed values. The next figures show the procedure to compute the parameters.

Figure 4.13 shows 1 period of the *FLAG_MULTIPLEXOR* signal. We can see with more detail the computation of the values and realize that the process only spent one third of the period.

Figure 4.14 shows the computation of each $\hat{\mathbf{p}}$ vector (real and imaginary) for the 40 channels. At the same time that is calculating the values, there is a module that sorts

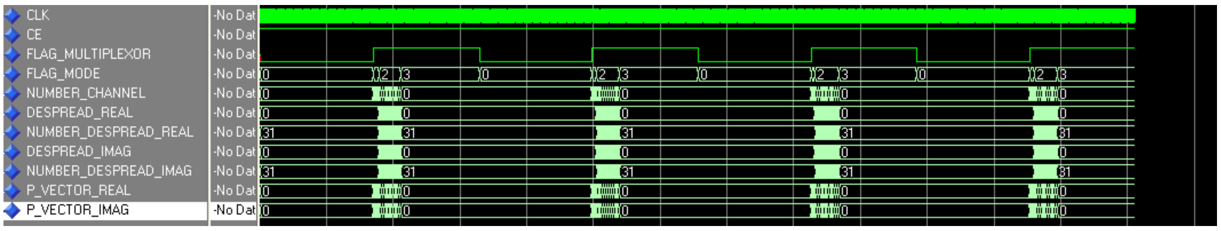


Fig. 4.12 Simulation results of the Selection illuminated elements block. This Figure shows all simulation, 4 Pilot Symbols and its corresponding selection.

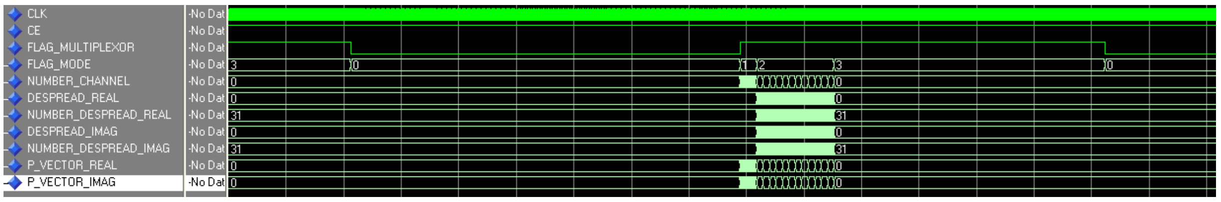


Fig. 4.13 Simulation results of the Selection illuminated elements block. This graph only shows 1 Pilot Symbol and its corresponding channel selection.

them in increasing modulus. At the end of the mode, the first twelve values will be the selected elements.

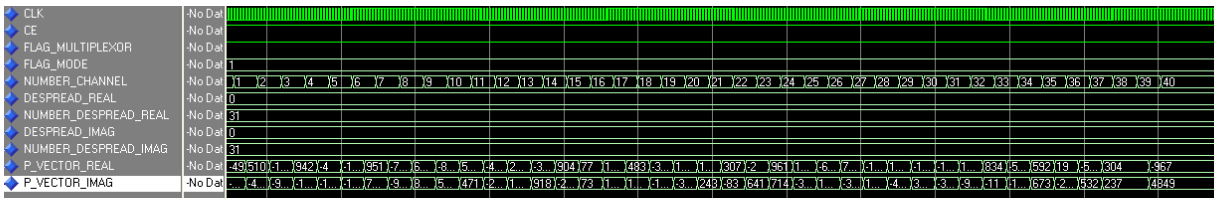


Fig. 4.14 Simulation results of the Selection illuminated elements block. This Figure depicts the computation of $\hat{\mathbf{p}}$ vector (real and imaginary) for the 40 channels.

After $\hat{\mathbf{p}}$ vector calculation for all channels, it is time to communicate with the MicroBlaze and to pass the calculated parameters of the selected elements. Figure 4.15 shows this process. According to the picture, the best channel is n.28 for the specific realization shown.

Figure 4.16 shows the 32 computed despreads (real and imaginary) for the n.28 channel.

4.2.3 Weight Vector Update block

The inputs for this block are the complex despreaded pilot signals of each illuminated element and its corresponding complex $\hat{\mathbf{p}}$ vector. Considering that in the S-DMB system specified in [1] the chip rate is 16.384 MHz and the processing gain is 64, the despreaded signal rate is 256 KHz. Despreaded data has a low rate, which alleviates the processing power needed to compute the correlation matrix and its inversion. However, the operations involved in the weight vector update are best suited to floating point arithmetic and to

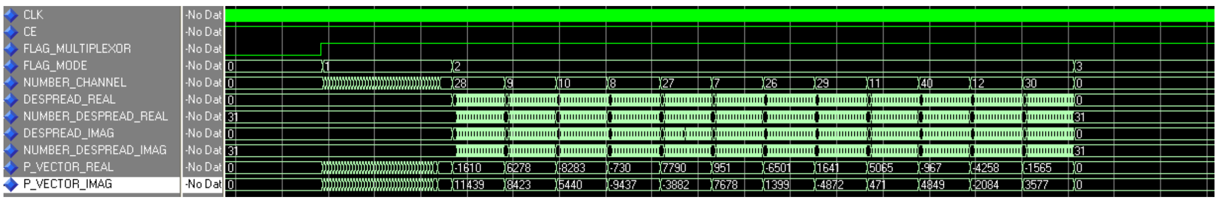


Fig. 4.15 Simulation results of the Selection illuminated elements block. This graph shows the selection of 12 channels with its corresponding computed $\hat{\mathbf{p}}$ vector and despreads.

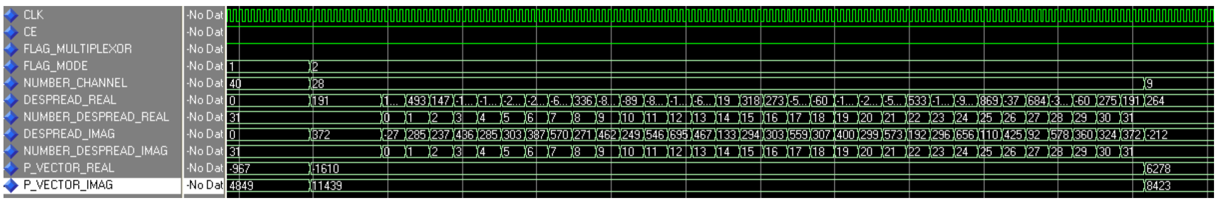


Fig. 4.16 Simulation results of the Selection illuminated elements block. This Figure shows the calculation of the 32 despreads (real and imaginary) for 1 channel.

processor-like set of operations. The architecture in Figures 4.1 and 4.2, proposes the use of an embedded Digital Signal Processing device within the FPGA. State-of-the-art Xilinx FPGAs provide the capability of including soft processors and hard processors:

1. Flexible MicroBlaze 32-bit and PicoBlaze 8-bit soft processors. This soft processors can be implemented in any Virtex family of Xilinx FPGA. The main drawback is that a soft processor consumes resources in the FPGA.
2. High performance PowerPC 32-bit hard processors. In order to save resources in the FPGA device, the use of the embedded hard processor is a good solution to deal with the floating point operations desired to compute arrays weights. The only family of Xilinx FPGAs that incorporate embedded hard processors is the Virtex-4 FX FPGA devices, which provide up to 2 on-chip PowerPCs, a 32-bit RISC processor that can operate at 450 MHz. We work with a Virtex-5 LX220 for this project, so we use the Microblaze option to do the Digital Signal Processing.

We work with a Virtex-5 LX220 for this project, so we use the MicroBlaze option to do the Digital Signal Processing.

As said, there are 2 modes of operation. Depending on these modes, the block under study operates in different ways:

(1) Acquisition mode:
No software part is needed to perform this mode. Remember that this mode only consist in selecting one element of the antenna array and feeding the $FPGA_2$ to perform the acquistion of synchronism.

(2) Tracking mode:
Inputs: Despreaded Pilot Channel from illuminated elements and its corresponding $\mathbf{p}$ vector.
Outputs: Weight vector according to the TRB:

$$\hat{\mathbf{w}}_{TRB} = \hat{\mathbf{R}}_{xx}^{-1}\hat{\mathbf{p}} \tag{4.3}$$

The most computationally consuming operation is the generation of the autocorrelation matrix and its inversion. However, the computation is alleviated, since the incoming data is despreaded prior to the computation of weights, being the rate considerably lower in contrast to the spreaded signal case. In addition, the autocorrelation matrix inversion using QR decomposition also aims to alleviate the computational cost.

The procedure to solve systems of linear equations by the QR algorithm is based on the QR decomposition of matrices [10].

Considering matrix $\mathbf{A}$, its QR decomposition is:

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{R} \tag{4.4}$$

Where $\mathbf{R}$ is upper triangular, and $\mathbf{Q}$ is orthogonal, that is,

$$\mathbf{Q}^T \cdot \mathbf{Q} = \mathbf{1} \tag{4.5}$$

where $\mathbf{Q}^T$ is the transpose matrix of $\mathbf{Q}$. Although the decomposition exists for a general rectangular matrix, we shall restrict our treatment to the case when all the matrices are square. QR decomposition can be used to solve systems of linear equations. To solve

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \tag{4.6}$$

first form $\mathbf{Q}^T \cdot b$ and then solve

$$\mathbf{R} \cdot \mathbf{x} = \mathbf{Q}^T \cdot \mathbf{b} \tag{4.7}$$

by backsubstitution.

This algorithm shows that it is not necessary to perform matrix inversion to solve the system of linear equations.

The Xilinx's Platform Studio has been used to implement this part. The algorithm has been programmed in C and debugged before to download into the evaluation board. The communication between hardware-software will be explained in section 4.2.5.

Figure 4.17 shows the weights diagram of the first Pilot Symbol simulated. In the simulated scenario the desired signal comes at 0^o and the interference comes at 9^o. We observe that after the computation of the weights, the radiation pattern of the antenna array points to the desired signal while nulling the interference. We have considered all parameters discussed before, like the quantization of signal and weights with 8 bits.

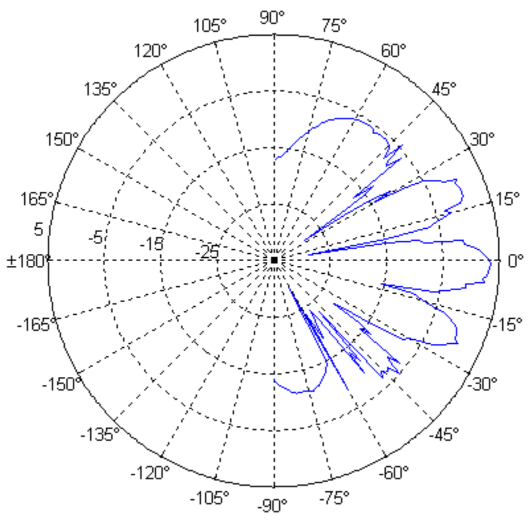


Fig. 4.17 Radiation pattern for the calculate weights.

4.2.4 DBF block

The output of the antenna array is the summation of the weighted inputs as exposed in Chapter 3. This block (see Figure 4.18 is in charge of this operation. The operation of this block is as follows:

(1) After A/D conversion, the digitized complex data from the N radiating elements is introduced in the FPGA. Notice that ADC must be implemented as an external device to the FPGA, since there are no state-of-the-art FPGAs that include ADC in its logic. As explained, there is a block in charge of detecting the illuminated elements. This information is delivered to the module prior to DBF, which consists in a bank of multiplexers that feed the DBF block with the corresponding N_p illuminated elements.

(2) N_p complex products between data and weights must be performed at a rate equal to the sampling frequency ($\sim$65 MHz), to ensure that the flux of data is conserved at the array output. Complex weights are delivered by the Weight Vector Update block, which operates at a lower rate.

(3) After programming and simulation, this block has been optimised to require $\sim$100 slices per illuminated processed element.

4.2.5 Communication between hardware-software blocks

In this section, we explain the communication between the "hardware" part done in VHDL and the "software" part corresponding to the microprocessor programmed in C. It is a critical point because requires synchronize both parts and control the data flow.

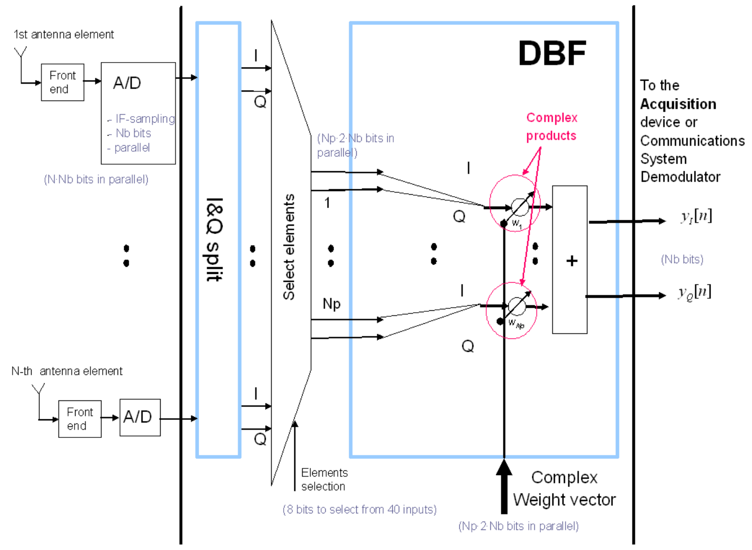


Fig. 4.18 DBF block: N_p parallel I&Q arms processed at the F_s rate.

We have also programmed this block in VHDL. Hence, it is a hardware implementation. We have chosen the Fast Simplex Link (FSL) bus to interconnect the devices because it offers a fast communication rate and it is easy to program.

The main features are:

- Implements a uni-directional point to point FIFO-based communication.
- Provides mechanism for unshared and non-arbitrated communication mechanism. This can be used for fast transfer of data words between master and slave implementing the FSL interface.

For this kind of communication, people usually call master to the device which control the communication with other devices and people usually call slaves to the devices which is controlled by other device. We can divide the process in two stages:

(1) The first stage corresponds to the case when the selection of channels is finished and we have to calculate the weight vector. In this case the hardware device will be the master and the microprocessor will be the slave.
(2) The second stage corresponds when the weight vector is update and we have to compute de Digital Beam Forming. In this case the microprocessor will be the master and the hardware part will be the slave.

Although it is a good choice to connect this kind of devices with this bus, it has a few restrictions. It is not possible to use the same bus to interconnect hardware with software and viceversa. In addition, we have to take into account that every channel has real and

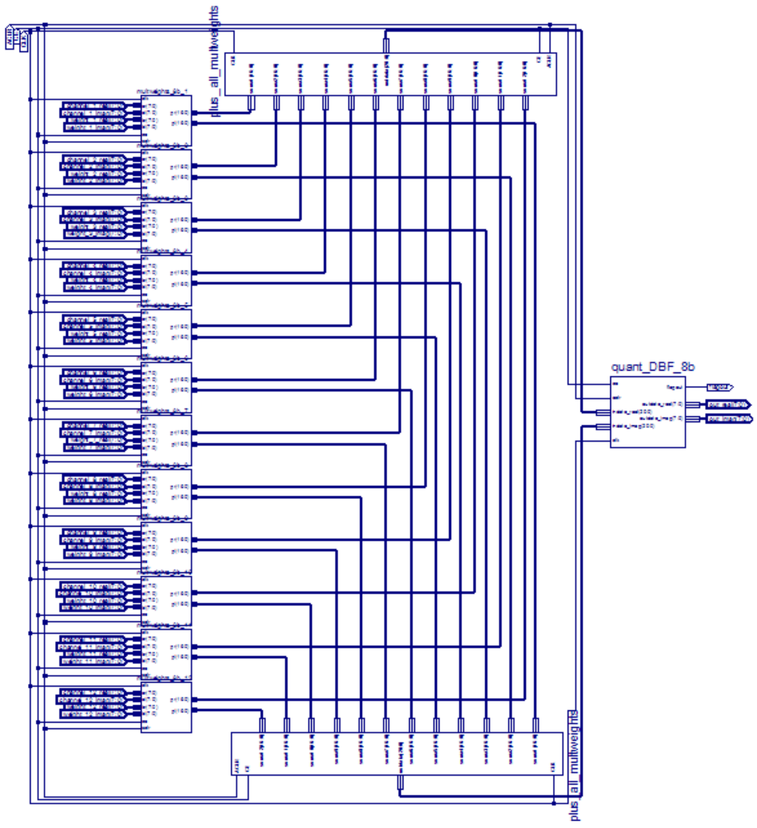


Fig. 4.19 Schematic description of the DBF block.

imaginary parts. However, we can use the same bus to pass the despreads and the $\hat{\mathbf{p}}$ vector for all selected channels. Hence, we need 2 FSL buses for the real part and 2 FSL buses more for the imaginary part. In total, 4 FSL buses. The maximum allowed links for one MicroBlaze microprocessor is 8, thus 4 FSL buses can be dealt with.

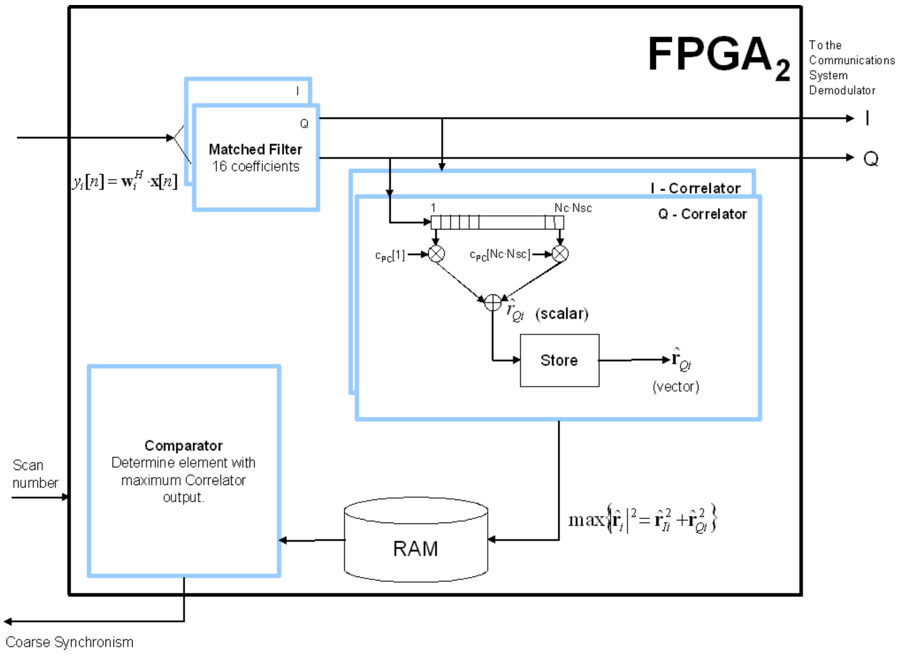
Once we have interconnected this two devices, the most complicated issue consists in synchronize them. In order to obtain it, we have used FIFO elements. We have to pass the despreads and the $\hat{\mathbf{p}}$ vectors (real and imaginary) of the 12 selected elements. So, we need 4 different FIFOs, 2 for the despreads parameters (real and imaginary) and 2 more for the $\hat{\mathbf{p}}$ vectors (real and imaginary). In principle, the capacity for the 2 firsts is 12×32 despreads and the capacity for the 2 lasts is 12×12. However, we have to multiply this initial capacity per 4 or 5 in order to assure that we don't lose any Pilot Channel.

4.3 Detailed Block description of $FPGA_2$

A block diagram of the second $FPGA_2$ is presented here. We have to differentiate the 2 modes of operation again.

Firstly, in acquisition mode, a Matched Filter is considered for each I&Q component of the array output. Then each component is correlated and this correlation stored for each element of the array. After all elements have been processed, a decision on synchronism is made and the system enters in its Tracking mode.

Secondly, in tracking mode, a Matched Filter is considered for each I&Q component of the array output. Then this output is delivered to the demodulator system. Figure 4.20 shows the block diagram description of this FPGA.



FPGA$_2$
To the Communications System Demodulator
I
Q
I
Q
Matched Filter
16 coefficients
$y_i[n] = \mathbf{w}_i^H \cdot \mathbf{x}[n]$
I - Correlator
Q - Correlator
1
Nc·Nsc
$c_{PC}[1]$
$c_{PC}[Nc \cdot Nsc]$
$\hat{r}_{Qi}$ (scalar)
Store
$\hat{\mathbf{r}}_{Qi}$
(vector)
Comparator
Determine element with maximum Correlator output.
Scan number
RAM
$\max\{|\hat{\mathbf{r}}_i|^2 = \hat{\mathbf{r}}_{Ii}^2 + \hat{\mathbf{r}}_{Qi}^2\}$
Coarse Synchronism

Fig. 4.20 Internal operation of $FPGA_2$.

4.3.1 Matched Filter block

Figure 4.21 shows the schematic block description of the Matched Filter block. It is composed by 3 modules:

(1) The first block, called *specialfilter*, is in charge of multiply each sample by its corresponding number.
(2) The other 2 blocks, called *multbufferplus* are the acumulators, one for the real part and other for the imaginary part.

This scheme is the implementation of a typical FIR (Finite Impulse Response) filter. Although we can use filters done with CoreGenerator of Xilinx, we have preferred to program one in order to modify easily.

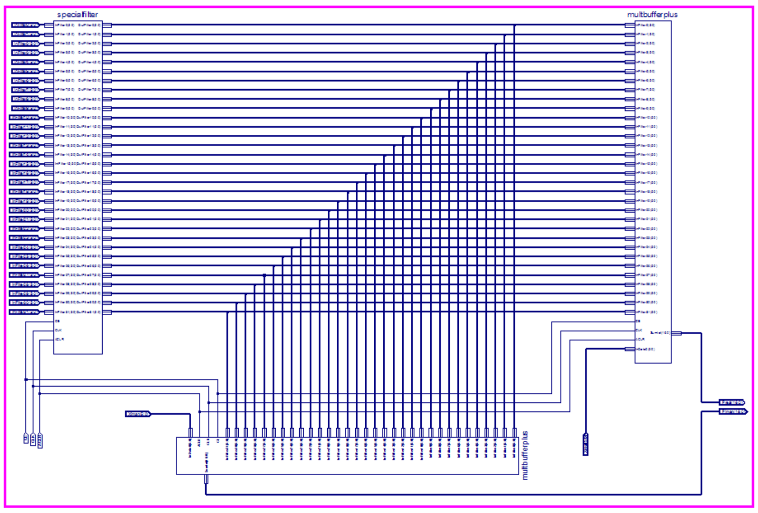


Fig. 4.21 Matched Filter of 16 coefficients.

4.3.2 Correlator and Comparator blocks

Figure 4.22 shows an schematic programmed in VHDL. It is composed of 3 stages:

(1) The first parallel blocks, inside the red square, are the I&Q module described in section 4.1.2.
(2) The second stage correlate the I&Q parts of the data stream with the spreading sequence of the pilot channel. Every block is composed of 2048 registers which store the correlation (1 register per sample), 1 ROM to store the PRN sequence, 1 multiplier and 1 adder.
(3) The last parallel blocks detect the highest value of the correlation for each sample. The architecture is very simple, 1 ROM to store the maximum value and 1 register to show the present value. Every block has four outputs: two outputs for the present value correlation and its corresponding index (a value in the interval [0,2047]) and two outputs for the highest value correlation and its index.

Figure 4.23 represent the final schematic downloaded into the $FPGA_2$. It is composed by the schematic explained previously and 3 new serial blocks:

(1) The first block, inside the green square, is the I&Q correlator module explained previosly.
(2) The next parallel blocks plus present outputs of the I&Q correlator for the real and imaginary parts.
(3) The third squares present outputs for real and imaginary parts.

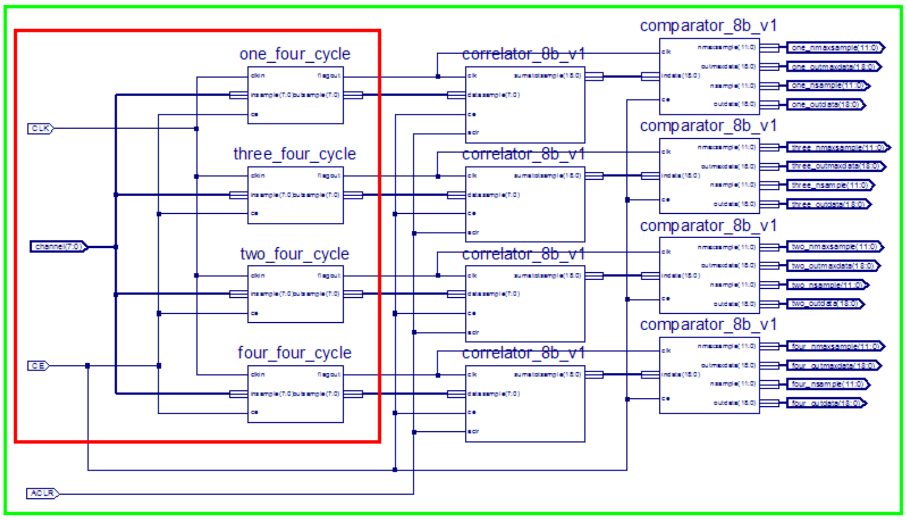


Fig. 4.22 Schematic description for the I&Q correlator.

(4) The four adds the squared real and imaginary parts. Being the output of this block the modulus of the correlation.

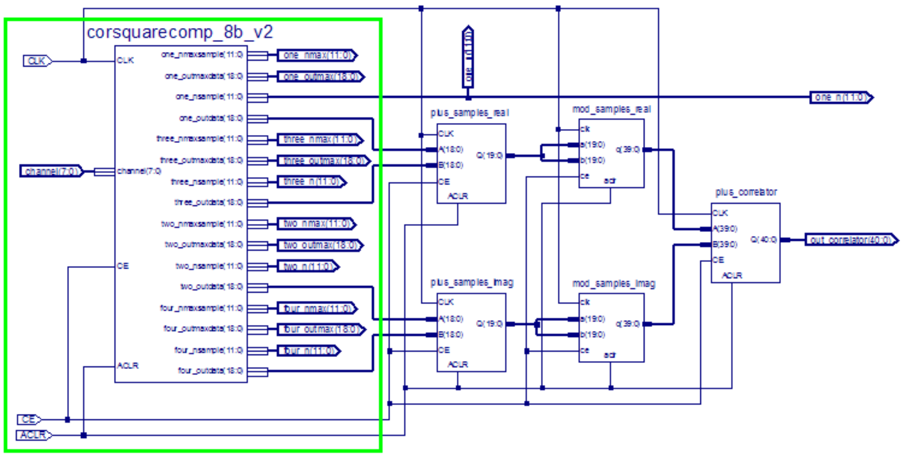


Fig. 4.23 Schematic description for the $FPGA_2$.

The number of slices for this schematic is on the order of 20000 slices for each sample. Since the presented schematic is N_{sc} samples per chip, the overall design of the Acquisition $FPGA_2$ is on the order of 80000 slices.

The correlation for one channel is showed in the Figures 4.24 and 4.25. The first picture has 5 subgraphs, the four first correspond to the correlation for each sample (1,2,3,4) and the last one the coherent sum of them. We observe that the even samples (2,4) have a high signal noise relation and consequently peaks more clears than the odd samples (1,3). According to the chapter 3, it is due to the modulation used: a BPSK.

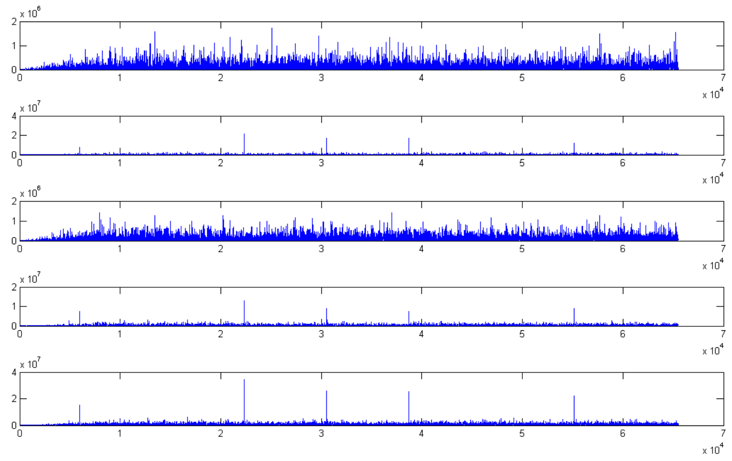


Fig. 4.24 Correlation of each sample and correlation modulus.

Figure 4.25 also shows the coherent sum of all samples. We look that the first and second peak are separated 16384 samples (250 μs $= \frac{16384 samples}{65.536 MHz}$) or 2048 chips. However, between the second and third peak there is only 8192 samples, it happens because the first symbol D_1 is formed by all ones as shown in Chapter 3. After the four peak the separation is 16384 samples and it stays constant.

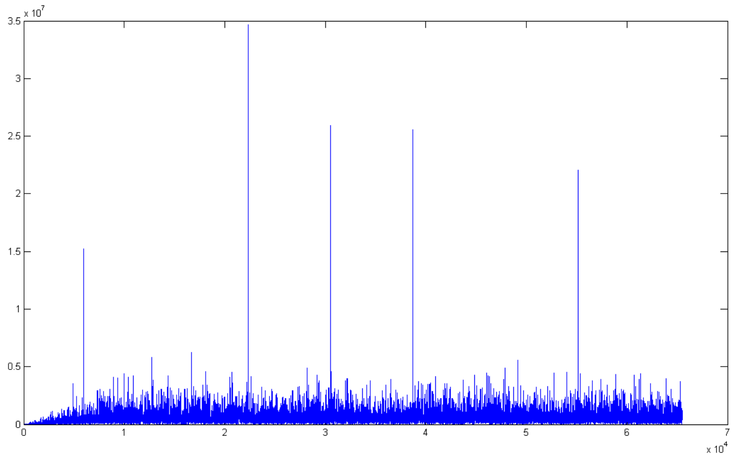


Fig. 4.25 Correlation modulus.

4.4 Hardware Requirements

Considering the designed blocks, the overall specifications for the two digital platforms are now considered. The following table shows the requirements for generic values and for the designed values.

4.4.1 Hardware Requirements of $FPGA_1$

Parameter	General value	Designed values
Input Pins	$N \cdot N_b$	~ 350
Output Pins	$2 \cdot N_b$	16
Input Clock	~ 65 MHz	~ 65 MHz
Multipliers	$4N_p$	48
Multiplexers (N:1)	$2N$	80
MicroBlaze processor blocks	1	1
Digital Clock Manager (DCM)	2	2

Fig. 4.26 Requeriments of $FPGA_1$.

A study on the slices required for a proper implementation of the whole $FPGA_1$ platform, shows that the following are required:

- I&Q splitting: $\sim$10 slices
- Selection illuminated channels: $\sim$60000 slices.
- DBF: $\sim$1000 slices.
- Weight Vector Update: $\sim$5000 slices.
- **Whole System: $\sim$66010 slices.**

Considering the requeriments of $FPGA_1$, we selected a Virtex 5 LX220 FPGA with up to 140000 slices approximately. This system requires about the 45% of the capacity of this FPGA.

4.4.2 Hardware Requirements of $FPGA_2$

Parameter	General value	Designed values
Input Pins	$2 \cdot N_b$	16
Output Pins	$2 \cdot N_b$	16
Input Clock	~ 65 MHz	~ 65 MHz
Multipliers	2×16	32
MicroBlaze processor blocks	0	0
Digital Clock Manager (DCM)	2	2

Fig. 4.27 Requeriments of $FPGA_2$.

A study on the slices required for a proper implementation of the whole $FPGA_2$ platform, shows that the following are required:

- Matched Filter: ∼200 slices.
- Correlator and Comparator: ∼80000 slices.
- **Whole System: ∼80200 slices.**

Considering the requeriments of $FPGA_2$, we selected a Virtex 5 LX220 FPGA with up to 140000 slices approximately. This system requires about 60% of the capacity of this FPGA. Xilinx manufacturers recommends using up to the 80% of the recurses of the FPGAs. For this reason, we have used 2 FPGAs to implement the system.

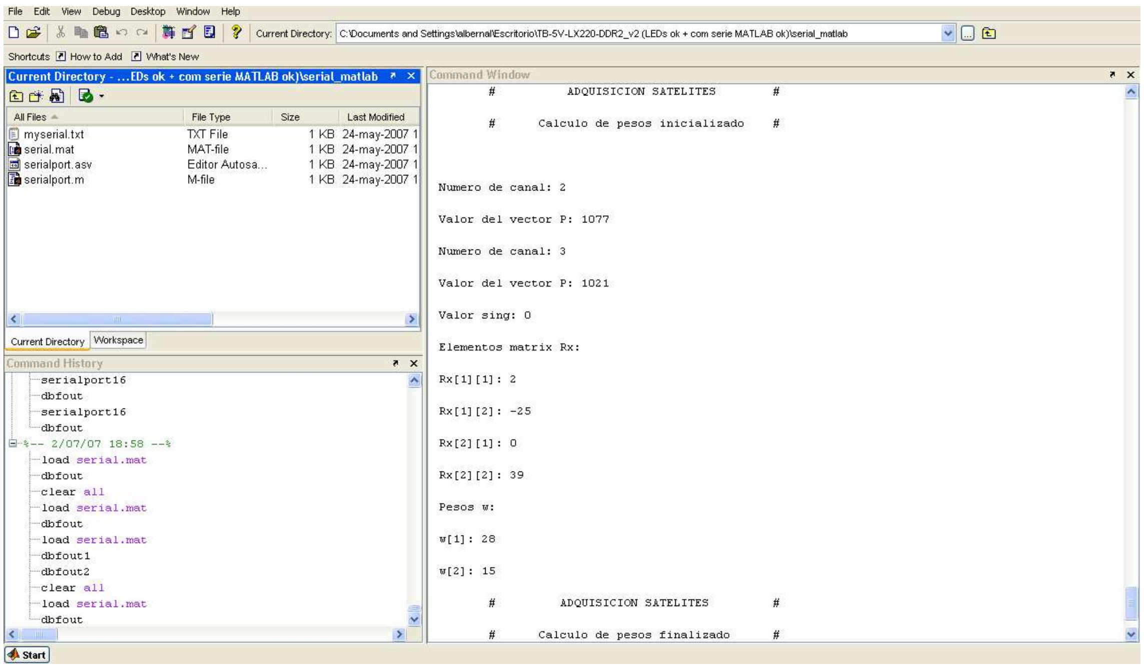
4.5 Simulating data on an FPGA

There are different ways to simulate data on an FPGA. Firstly, we have to differentiate between the simulations before and after downloading the program.

Before downloading the program, we can simulate the system or parts of them using Test Benchs or reading data from files. However, this simulations are not suitable for big systems because they spent a lot of time. The best technique consists in create internal ROMs, one for each channel, using the CoreGenerator of Xilinx. This technique has the advantage that it can be used for testing the system once we have downloaded the program on the FPGA. The simulations presented in subsection 4.2.2 corresponds with this technique. For our system, we have made 40 ROMs where we have loaded 1 Pilot Symbol and 1 Control Data (16384 samples per ROM). The different periods that we can look at the pictures always are the same. The limitation of the number of samples per ROM is due to the maximum number of block BRAMs that we can use. For this configuration, we use all block BRAMs. Another option that we have implemented is to reduce the system from 40 channels to 12 channels (the best 12 channels) and loaded 3 Pilot Symbols and 3 Control Data. So, we can observe how fast changes the weight diagram.

After downloading the program, we can use the technique explained previously or download the data by the Serial Port. MATLAB has the capability to send and receive string of bytes at a high baud rate (with up 115200). However, if we want to download 1 Pilot Symbol and 1 Control Data, we have to download 40 $\times$ 16384 samples and this process takes about 3 hours. It is not a practical approach in order to check the behavior of the system. Hence, we have simulated the system creating ROMs for each element.

Only we have used the MATLAB interface to send the computed weight vectors and print the radiation pattern. A screen capture of MATLAB interface is showed in Figure 4.28. For the sake of simplicity, we have simulated a simplified architecture of an antenna array composed of 5 elements selecting the 2 most illuminated. Without lost of generality this can be extended to generic N-element antenna array. In the screen capture, we can observe the selected elements: 2 and 3, with their corresponding $\hat{\mathbf{p}}$ vector. Also we can see the triangular matrix of the QR decomposition obtained from the 2 $\times$ 2 autocorrelation matrix. Finally, the computed weights via the TRB technique are printed in the MATLAB shell. All this data is stored in a .mat file for post-processing.


File Edit View Debug Desktop Window Help
Current Directory: C:\Documents and Settings\albernal\Escritorio\TB-5V-LX220-DDR2_v2 (LEDs ok + com serie MATLAB ok)\serial_matlab
Shortcuts How to Add What's New
Current Directory - ...EDs ok + com serie MATLAB ok)\serial_matlab
Current Directory
Workspace
Command History
Command Window
Start

All Files	File Type	Size	Last Modified
myserial.txt	TXT File	1 KB	24-may-2007 1
serial.mat	MAT-file	1 KB	24-may-2007 1
serialport.asv	Editor Autosa...	1 KB	24-may-2007 1
serialport.m	M-file	1 KB	24-may-2007 1

```
serialport16
dbfout
serialport16
dbfout
%-- 2/07/07 18:58 --%
load serial.mat
dbfout
clear all
load serial.mat
dbfout
load serial.mat
dbfout1
dbfout2
clear all
load serial.mat
dbfout
```

```
       #          ADQUISICION SATELITES        #

       #      Calculo de pesos inicializado    #



Numero de canal: 2

Valor del vector P: 1077

Numero de canal: 3

Valor del vector P: 1021

Valor sing: 0

Elementos matrix Rx:

Rx[1][1]: 2

Rx[1][2]: -25

Rx[2][1]: 0

Rx[2][2]: 39

Pesos w:

w[1]: 28

w[2]: 15

       #         ADQUISICION SATELITES         #

       #      Calculo de pesos finalizado      #
```

Fig. 4.28 Screen capture of the programmed MATLAB interface to obtain processed data from the FPGA.

5

Conclusions

The main scope of this project is to implement a Digital Beamforming for an antenna array receiver. To this aim, several studies, simulations and schematic blocks have been presented along the Chapters 3 and 4. In what follows, we detail the main contributions to the project:

- The simulation of the S-DMB standard.
- The study of Digital Beamforming techniques. Three different techniques have been studied: MVB, TRB and HB. Advantages and drawbacks to perform the Digital Beamforming after or before the pilot signal Despreading also have been studied. After calculations and simulations, we have concluded that the best technique was TRB after the pilot signal Despreading because of:
  - TRB maximizes the SINR with no need for array calibration.
  - DBF after pilot signal Despreading reduces the data rate to be processed in the MicroBlaze and provides a reference signal that has a higher SNR.
- Design of the system in VHDL/C, evaluation of the resources required and simulations. We have used 2 FPGAs to implement synchronize acquisition and DBF.
- All blocks have been programmed in VHDL taking advantage of its parallelization properties, except for the computation of weights, which has been programmed in C, using the MicroBlaze soft-processor, due to the obligation to carry out this operation in floating-point.
- System implementation
  - IF-Sampling. This technique has allowed to save resources since I&Q splitting is performed in the digital platform reducing the number of

    required I/O. This allowed to parallelize all following blocks for each sample.
  - DBF based in Temporal Reference Beamforming (TRB). A computation of weights according to this technique have been successfully implemented.
  - Selection of illuminated elements. A fast algorithm for the selection of illuminated elements have been presented and implemented, taking advantage of the estimation of $\hat{\mathbf{p}}$ vector.
  - Communication between hardware-software blocks with the FSL bus. Fast communication have been achieved with this kind of bus within the FPGA.
  - Acquisition of synchronism in CDMA. We have implemented a CDMA correlator operating at high frequency rates, obtaining sample accuracy synchronism.
- The communication between PC and FPGA using the RS232 port has been programmed in MATLAB in order to download data easily and within the usual programming and simulation environment. This interface allows us to receive the weight vector and print the radiation pattern.
- System simulation and debugging on the 2 FPGAs.
- The system has been implemented and tested correctly.

Future work:

- Implement a web server in the FPGA using the Ethernet port of the evaluation board in order to check the selected illuminated elements, the despreads and $\hat{\mathbf{p}}$ vectors computed.
- Implement the communication between PC and FPGA in real time in order to print the evolution of the radiation pattern in a moving scenario.
- Interconnect the Digital Platform with the other parts of the CORPA project to develop a joint test plan.
- Study the rejection level of the DBF against multipath an interferences in a real platform, in contrast to simulated based studies.
- Study the effect of weight vector quantization in the degradation of the SINR.
- Development of a general DBF platform to deal with several systems, such as the Global Navigation Satellite System (GNSS) or other satellite communication standards.
- Study the implementation of other DBF techniques and algorithms, focusing on adaptive algorithms such as LMS and RLS.

References

[1] "ITU-R BO.1130-4: Systems for digital satellite broadcasting to vehicular, portable and fixed receivers in the bands allocated to BSS (sound) in the frequency range 1 400–2700 MHz," International Telecommunications Union - Radiocommunications (ITU-R), Tech. Rep., 2001.

[2] S. Hirakawa, N. Sato, and H. Kikuchi, "Broadcasting Satellite Services for Mobile Reception," *Proc. IEEE*, vol. 94, no. 1, pp. 327–332, January 2006.

[3] J. G. Proakis and M. Salehi, *Communications systems engineering.* Prentice–Hall, 1994.

[4] R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays.* John Wiley & Sons, 1980.

[5] B. D. V. Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE Signal Processing Mag.*, vol. 5, no. 2, pp. 4–24, April 1988.

[6] H. L. V. Trees, *Optimum Array Processing. Detection, Estimation and Modulation Theory, Part IV.* Wiley Interscience, 2002.

[7] G. Seco, J. A. Fernández-Rubio, and C. Fernández-Prades, "ML estimator and Hybrid Beamformer for multipath and interference mitigation in GNSS receivers," *IEEE Trans. Signal Processing*, vol. 53, no. 3, pp. 1194–1208, March 2005.

[8] C. Fernández-Prades, "Advanced Signal Processing Techniques for GNSS Receivers," Ph.D. dissertation, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, May 2006, (available on-line at http://gps-tsc.upc.es/comm2/publications/T_2006_Fernandez.pdf).

[9] G. Seco, "Antenna Arrays for Multipath and Interference Mitigation in GNSS Receivers," Ph.D. dissertation, Dept. of Signal Theory and Communications, Universitat Politècnica de Catalunya, Barcelona, Spain, July 2000.

[10] W. T. Vettering, S. A. Teukolsky, W. H. Press, and B. P. Flannery, *Numerical Recipes in C.* Cambridge University, 1992.

[11] P. Closas and J. A. Fernández-Rubio, "CORPA Phase I: Final Report," Universitat Politècnica de Catalunya (UPC), Tech. Rep., December 2006.

[12] O. Lücke, A. Pellón, P. Closas, and J. Fernández-Rubio, "Cost-Optimised Active Receive Array Antenna for Mobile Satellite Terminals," in *IST'07 Mobile Communications Summit*, July 2007.

[13] I. Berkeley Design Technology, "Choosing a DSP Processor," 1996-2000, http://www.bdti.com/articles/choose_2000.pdf.

[14] H. Engineering, "Choosing FPGA or DSP for your application," 1997-2007, http://www.hunteng.co.uk/info/fpga-or-dsp.htm.

[15] I. Andraka Consulting Group, "DSP with FPGAs," 2007, http://www.andraka.com/dsp.htm.

[16] Mathworks and MATLAB, "Serial Port I/O," 2000, http://www.math.carleton.ca/˜help/ matlab/MathWorks_R13Doc/techdoc/matlab_external/ch_seria.html.

[17] Xilinx, “Fast Simplex Link (FSL) bus (v2.00a),” 2005, http://www.xilinx.com/bvdocs/ipcenter/data_sheet/FSL_V20.pdf.