Design of Parallel Architecture For Correlator using FPGA

Major Project Report

Master of Technology

In

Electronics & Communication Engineering

(VLSI Design)

By

PATEL DHAVAL D
(11MECV09)

Department of Electrical Engineering

Electronics & Communication Programme

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2013

Design of Parallel Architecture For Correlator using FPGA

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

In

Electronics & Communication Engineering

(VLSI Design)

By

PATEL DHAVAL D
(11MECV09)

Under the guidance of

Prof. N. P. Gajjar

Department of Electrical Engineering

Electronics & Communication Programme

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2013

Declaration

This is to certify that

1. The report comprises my original work towards the Degree of Master of Technology (Electronics & Communication) in the field of VLSI Design at Nirma University and not been submitted elsewhere for a Degree.

2. Due acknowledgement has been made in the text to all other material used.

PATEL DHAVAL D.

Certificate

This is to certify that the Major Project entitled **”Design of Parallel Architecture For Correlator using FPGA”** submitted by **DHAVAL D PATEL**, towards the partial fulfilment of the requirements for the degree of Master of Technology in Communication Engineering of Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Guide: Program Coordinator:

(Prof. N P Gajjar)
Asst. Professor, EC

(Dr. N M Devashrayee)
Professor, EC

HOD: Director:

(Dr. P N Tekwani)
Electrical Dept

(Dr. K Kotecha)
Computer Dept.

Acknowledgements

First of all I thank almighty God for providing me good health during the whole project work. I would like to take the opportunity to extend my gratitude to the management of Institute of Technology, Nirma University for providing us healthy working environment throughout the project work. Prof. N. P. Gajjar is the main source of motivation for me in the project. He guided me at every step of the work I have carried out. I would like to thank Prof. N. P. Gajjar for his valuable guidance and support. I again would like to thank Prof. N. P. Gajjar for allowing me to use the resources of ISRO Respond Lab for carrying out the project. I would like to thank Dr. N. M. Devashrayee for his continuous support throughout the project. I would also like to extend my gratitude to all teaching and non teaching staff members and my student mates who were helpful to me in direct or indirect way. At last but not least, I thank my family for their support and love.

- Dhaval D Patel
11MECV09

Abstract

Modern high-performance digital signal processing (DSP) applications face constantly increasing performance requirements and are becoming increasingly challenging to develop and work with. In DSP paradigm, many researchers see potential in achieving algorithm speed-up by employing Field Programmable Gate Arrays (FPGAs) reconfigurable hardware with parallelism feature. However, developing applications for FPGAs incur particular challenges on the development flow.

As a similarity measure, cross-correlation has found application in a broad range of signal processing. A dedicated hardware implementation of cross correlation is crucial for the requirements of real-time high-speed tasks such as automatic target matching, recognition and tracking. One efficient parallel architectures for real-time implementation of correlator using field programmable gate array (FPGA) are proposed in this project. In these architectures, several novel efficient approaches are proposed to reduce logic resource usage and computation time. These architectures can be applied in different situations according to the practical available resource of the FPGA chip used. Design ,Simulation, implementation realization of Multiple correlator in parallel architecture using embedded platform of Xilinx FPGA and Software suite.

The outcomes of this work are a multi-channel correlator developed in a reconfigurable environment with new design methodology and I/O framework with software control application. The outcomes are used to demonstrate the potential of implementing DSP applications in a FPGA architecture and to discuss existing challenges and suggest possible solutions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Necessity, who is the mother of invention.
- Plato

This chapter provides an overview of a high-performance DSP applications field from its origins to its current state. Appropriate background of the area of investigation is introduced and respective research objectives are outlined. The chapter concludes with contributions and organisations of this thesis.

1.1 Background

High-performance digital signal processing is very challenging work in todays engineering fields. Many applications face increasing performance demands and constant additional functional requirements. With digital signal processing becoming an integral part of everyday life, the demand for high-performance processing means has expanded rapidly in recent years.

Originally, signals in devices were manipulated using analog techniques (continuous-time domain). However, nowadays most of them are implemented in digital form (discrete-time domain). The genesis of the digital signal processing techniques can be connected to the advances in mathematical fields: finite difference methods, numerical integration method and numerical interpolation methods dating back to the seventeenth century. Of course, one of the major developments of the DSP area started in the 1950s, as a part of the far broader and embryonic field of digital computers. From the late 1960s, digital signal processing moulded into a separate field by itself. Thus, in the late 1970s when LSI (large-scale integration) technology became developed enough the realisation of a single chip DSP became practical. In 1978 AMI announced a Signal Processing Peripheral and released S2811 a co-processor for a host micro. It was followed by Intels 2920 in 1979. The unique feature of the latter device was the on-chip analog-to-digital and digital-to-analog converters (ADC and DAC respectively), though it lacked a multiplier. The DSP industry continued to grow and progress and, in the early 1980s, the world saw a second generation of DSPs with realised features like concurrency, multiple buses and on-chip memory. These were added with on-chip floating point operations in the third generation of DSPs in the early 1990s. In the late 1990s multi-processing features, image and video processors and low-power DSPs were introduced.

Contemporary signal processors are able to demonstrate much greater performance in many aspects: wider data buses and throughputs, higher processing speeds of up to 24,000 of 16-bit million multiply accumulate operations (MMACs)(Texas Instruments Inc., 2008), compatibility with various modern interfaces and buses such as PCI, USB, Ethernet and many others.

The means of performing signal processing are not, of course, limited to digital signal processors the ever-growing field of signal processing invoked multiple solutions, architectures, technologies, tools and approaches: the major of which will be covered in the subsequent chapters of this work.

Many large-scale, high-performance DSP applications in such fields as radio astronomy, telecommunication, high-energy physics, and others involve computationally-intensive and therefore often time-consuming correlation of wideband signals. Correlation relies on the two most common types of composed DSP operations multiply and accumulate (MAC), and multiply and add (MULT-ADD) operations. These operations have been implemented in digital processing successfully and efficiently. However, the challenge lies in the number of these operations, i.e. the problem size the running time and/or space requirements of an algorithm. Many DSP applications employing correlation operation in their algorithms require real-time or near real-time processing, eg antenna aperture synthesis, medical applications, cellular and telecommunications applications (see 2.1 for details on these applications). Along with necessity to perform computations on-the-fly, correlation involves considerable execution time or time complexity for wideband correlation. For example, a multichannel antenna array operating with 128 MHz bandwidth on each channel will yield a sampling rate of 256 MS/s with 8-bit sampling.

For 8-channel correlation, this will produce 2 GB/s input data stream. Such correlation will generate 28 unique cross product outputs (the other 28 are just a mirror reflection of the first 28 (see 2.2.1)). An estimated number of operations required to perform a 32-lag correlation with these parameters is about 230 Giga-operations of real-time processing. Being a classical DSP problem, correlation itself does not usually constitute a stand-alone, full and final application, rather it is an integral part of many DSP applications.

Here and throughout this thesis channel and antenna are used interchangeably. While such notation is acceptable and coomon for engineering and DSP fields, it differs in radio -astronmy where a channel is understood as a quantum of radio frequency bandwidth.

Although the processing capacity of DSP tools grew along with the requirements of the signal processing, the latter always outstands the former by considerable and everlasting margins. Almost as soon as the gap between ever-growing applications requirements and capabilities of the DSP tools started shaping up (in the mid 1980s see Figure 1.1) the search for counter-measures to close this gap started. The mostprevalent and widely-used approach is extensive approach gradual and proactive increase of the processing power of the DSP tools by increasing the number of employed computational units and/or operational parameters (operating frequencies, response times, storage capacities, etc.). Such approach proved to be productively working for Central Processing Units (or commonly known as processors), Digital Signal Processors (DSPs) and other conventional processing means for several decades and then started depleting quickly. The cost of the extensive approach hit the inevitable limitations very soon: high power consumption, complexity of dealing with growing number of computational units maintenance cost, etc.

No surprise, that the research vector began deviating towards technologies and methods which could offer intensive ways of dealing with the problem as opposed to almost exhausted extensive approaches.

Currently an intensive approach is envisioned by many researchers in parallelismsimultaneous execution of several computational operations during one clock cycle. Moreover, parallelisation of applications is especially effective in the DSP field as long as many DSP algorithms possess intrinsic parallelism and therefore potentially sustain a large capacity for acceleration.

Application-Specific Integrated Circuits (ASICs) possess parallelism fea-

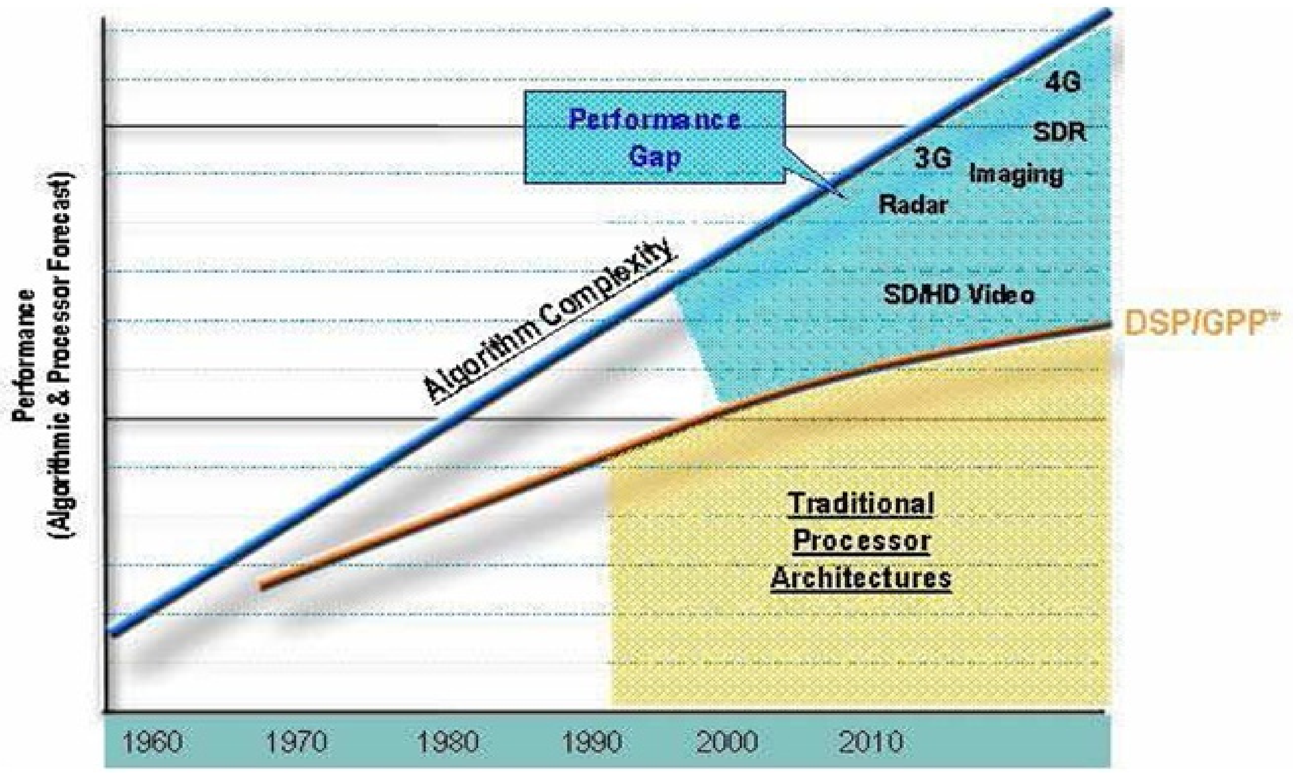


Figure 1.1: Performance Gap Between Traditional Processor Architectures and Growing Complexity of DSP Algorithms

tures and reach prominent efficiency of silicon utilisation for a specific operation determined during the manufacturing stage. Thus, they can be configured to meet the requirements of the particular application avoiding unnecessary generality. Reasonably, the performance efficiency achieved by ASICs for the targeted application is balanced by the impossibility of future modifications. Many ASICS applications do not require any updates, modifications or alterations at all (eg integrated circuits of cell phones).

ASICS counterparts Field Programmable Gate Arrays also possess parallelism features. Along with this, FPGAs are reconfigurable devices, i.e. they can be reprogrammed in the field. Adding a bug fix, a new feature or even updating a computational core of the application, significantly increases the flexibility of design. Due to this FPGAs have lower non-recurring engineering costs than ASICS. An FPGA is a semiconductor device consisting of programmable logic blocks and programmable interconnects. Along with parallelism, reconfigurability significantly expands the application scope of these devices. Armed with these two features over the years, FPGAs became one of the most promising technologies in digital electronics in general and in the DSP field in particular.

The interest towards FPGAs has risen even more radically in recent years with the growth of the chip capacities and the number of supported interfaces, which include, but are not limited to: PCI family interfaces, Ethernet family interfaces, support for memory interfaces like DDR/DDR2/DDR3/QDRII, USB, HyperTransport, RapidIO, VMEbus and many others. Moreover, FPGAs are particularly suitable for DSP applications due to: inherited parallelism in many DSP algorithms; high bandwidths to on-chip and external memories, which support multiple access ports thus allowing further exploitation of algorithms parallelism; streaming to-be-processed data directly to computational core implemented in FPGAs via available high-speed interfaces. Hence, these FPGAs features make them a very attractive option for applications acceleration or even a competitive alternative for traditional DSP techniques, attracting more and more attention from the DSP industry.

With widespread availability of commercially available FPGAs in the late 1980s,the term reconfigurable computing (RC) was introduced. A reconfigurable computing system is a system which is built from reconfigurable computing devices, eg FPGAs or FPGA-like devices. These systems have to be reprogrammable, permit orders of magnitude speed-ups versus traditional computational systems and support hardware-like levels of performance.

However, developing DSP applications for RC systems contain many more challenges and complexities than implementing applications in traditional software programming domain of DSPs, CPUs, etc. One of the main reasons is that FPGA design flow adheres to hardware development flow, which traditionally deals with low level hardware description languages and demands explicit configuration of available resources in FPGA. The following issues also impose substantial challenges when employing reconfigurable hardware in traditional DSP applications: hardware state ambiguity complicates design debugging; parallelism consideration and a run-at-aclock concept impose certain idiosyncrasies on algorithm implementation; explicit memory structure puts constraints on storing of design variables. Moreover,conventional processing methods (DSPs, CPU and related) have been employed in the signal processing field considerably longer than FPGAs and therefore have more advanced and powerful developing and debugging tools.

Therefore, employing FPGAs either as a computational accelerator or as a standalone DSP application platform can be beneficial and challenging at the same time. Nowadays the traditional approach of increasing the processing capacity of computational means diverge from the traditional approach of raising the number of employed semiconductors (Moores law) and operating frequency to a multicore and parallel execution approach. Many DSP algorithms possess intrinsic parallelism. FPGAs are a very attractive and potentially beneficial option to be employed in DSP paradigms for processing acceleration. Compelling reported speed-ups of 10X to 100X of equivalent software algorithms attract more and more attention from the DSP community. Another argument to employ FPGAs for DSP algorithms is that these devices follow the International Technology Roadmap for Semiconductors (ITRS) (http://www.itrs.net/) even more narrowly than modern microprocessors (eg in terms of contained SRAM memory or leading on the first fabrication lines). Many researchers agree on a high potential of simultaneous operation of conventional processing unit(s) such as a CPU of a PC and reconfigurable hardware such as FPGA. This architecture invokes previously unavailable possibilities and options in signal processing but it also brings new challenges in development flow.

In this thesis a new design methodology for developing applications in a FPGA environment is applied. Using a mixture of traditional hardware development tools and conventional software development tools, a multichannel wideband cross-correlation for DSP application on a FPGA architecture will be implemented. The prime objective of this thesis is to investigate the capabilities and challenges of this reconfigurable, hybrid architecture in the DSP field.

1.2 Research Objectives

In this work, we will investigate the implementation of a classical DSP problem wideband multi-channel cross-correlation in a hybrid environment of a commercial FPGA. There are a number of contributions contained within this thesis.

First, the given work addresses the problem of computational deficiency in DSP field. By implementing a classical DSP problem in a FPGA architecture, its abilities of achieving speed-ups for applications from DSP fields are argued.

The second contribution is the platform and the workflow for developing high performance DSP applications in a FPGA environment. The development work flow applied in this work is different from a traditional hardware design methodology. Rather than using low-level HDLs for hardware design implementation,the given work utilises high-level languages (HLLs) for hardware configuration. The potential of using HLLs for FPGA designs is evaluated and discussed. The given work delivers valuable outcomes for any DSP engineer developing applications in reconfigurable hardware with the aid of high-level programming (HLP) languages.

The third contribution is that this work tackles one of the most crucial issues of DSP applications, which becomes especially challenging and difficult in the FPGA domain input/output interfaces. The I/O framework developed in this work features original method of interfacing to FPGA via onboard SDRAM simultaneously with high speed communication with Fast simplex link(FSL) interfaces. The developed method can be beneficial to many applications requiring extensive data exchange. Particularly, it can be useful for applications targeting Xilinx virtex5 ML505 or to any xilinx development devices featuring PCIe and DDR/DDR2 SDRAM interfaces.

The given work also introduces the possible evolution of the proposed platform. The number of available interfaces on the exploited FPGA board (PCIe, Ethernet, SFP,HSMC, etc) and simple connectivity options of the conventional PC box provide a considerable degree of architectural possibilities. A highly scalable platform for high performance signal processing is proposed as a potential future development of the created design. In addition, one of the advantages of the suggested platform is the affordable cost as compared to proprietary DSP solutions: the cost of the prospective system is composed merely from FPGA board.

1.3 Thesis Layout

The thesis is organised into five chapters.

Chapter 2 briefly introduces the background of the investigated problem. Computationally intensive DSP applications employing correlation of signals are discussed. Correlation theory is discussed, which is followed by a discussion on DSP implementation technologies.

Chapter 3 briefly introduces the background of the Linear correlator architecture and both the auto correlator and cross correlator. Auto correlator and cross correlator is implementation on MATLAB and compare a golden output and also implement a 32-bit correlator design in Xilinx ISE 13.1 and simulated in ISE Simulator. All implementation is observer a simulation level.

Chapter 4 briefly introduces the multi channel correlator architecture and discussed implementation on a FPGA device. Also discussed in Microblaze processor and device utilization on virtex 5 and Spartan 3E starter broad. Multichannel correlator architecture is discussed to implementation on Xilinx EDK flow and observed a result on HYPER terminal.

Chapter 5 briefly introduces on final calculation and future scope on this project.

Chapter 2

Theory Background and Related Work

We live in a moment of history where change is so speeded up that we begin to see the present only when it is already disappearing.
- R. D. Laing

This chapter will discuss the most common digital signal processing applications in a high-performance domain. First, a brief outline of generic digital signal processing algorithm will be given. Then, the next section will highlight the most common high performance DSP applications, which will be followed by the discussion on the cross correlation problem as integral and often one of the most computationally intensive parts of these applications. The remaining section will present and consider contemporary DSP implementation technologies.

The term digital signal processing implies converting an analog signal into a form of numbers (digital form), the processing of the resultant sequences to either obtain information or to synthesise signals with desirable properties and possibly convert the output into analog form again. The overall scheme of the generic DSP algorithm is shown in Figure 2.1:

The high-performance DSP applications feature a considerable amount of computations in the Digital processor stage. Several typical high-performance DSP applications are considered in the following section.

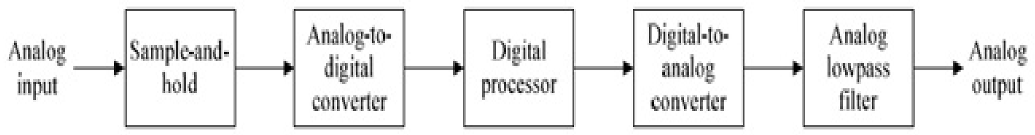


Figure 2.1: Generic Digital Processing Scheme

2.1 Typical High-Performance Signal Processing Applications

The number of signal processing applications in todays life is truly enormous. Nevertheless, not every DSP application is suitable for reconfigurable computing. A number of studies exist which investigate efficiency criteria of an application to be employed in FPGAs. The applications performance implemented in FPGA depends on :

1. Data parallelism available in the applications algorithm;
2. Data element size and arithmetic complexity;
3. Amenability to pipelining, and simple control requirements.

The following sections highlight several high-performance DSP applications,which have one common and integral operation cross-correlation of signals. The potential of implementing these applications in or with the help of reconfigurable hardware will be considered. The applications below will be considered in retrospect.

2.1.1 Radio Astronomy

Throughout human history, man has been always mysteriously attracted to the sky. With the discovery and subsequent invasion of new technologies, traditional methods of visual investigation of the sky, Ie methods of optical astronomy, were joined by radio astronomy techniques. Many astronomical bodies emit radio waves, which after certain processing can tell valuable and previously inaccessible information about their origin.Thus, in the last half of the 20th century the prominent advances in radio astronomy led to a number of foremost discoveries like masers, pulsars, radio galaxies, the Cosmic Microwave Background Radiation, etc.

With radio astronomy, scientists can study astronomical phenomena which are invisible to the human eye. While in optical observation the useful information is extracted from the spatial distribution of light across an object, ie image, radio astronomy uses a different principle. RF waves emitted by a certain phenomenon can be received and directly sampled in a time domain, thus the tools used for detecting and measuring this interaction are considerably different from optical telescopes.

To produce a radio image of a celestial phenomenon a principle of interferometry is used, which entails the superposition technique - interference (adding or overlaying) of signals from two or more antennas. This technique is also known as antenna aperture synthesis when multiple antennas are used to work as one using interferometry principle.

The core idea of antenna aperture synthesis is again to superimpose the signal waves from a number of radio telescopes and, while doing so, inphase waves will add up, while antiphase waves will cancel each other out. This creates a combined telescope with the size of the furthest observing telescopes apart. The image quality produced by such a composed antenna depends on the number of the projected separations between any two telescopes as seen from the radio source (number of baselines). With each radio telescope producing a data stream the processing and computational task can be extremely intensive. Besides, the processing is complicated by low signal-to-noise ratios which are common for radio astronomical observations.

The backbone operation of antenna aperture synthesis is correlation or finding the amount of similarity in the signal between two given antennas in an antenna array. The term correlation and underlying theory will be discussed more deeply in 2.2.1. Even for a medium-size antenna array, computation of correlation between all the elements of the array can be a very challenging task due to the number of involved mathematical calculations. The example considered in 1.1 with an 8-element antenna array requires 230×10^9 operations per second. An experienced reader will estimate that the problem size of the given example as average to below average. Nevertheless, such a system might require a performance power of not less than 230 GFlops (depending on implementation). In real-life, large-scale systems that correlate signal pairs of multielement arrays may contain millions of correlator circuits in order to accommodate all the required antennas and spectral channels. Hence, with an increase of any of the above parameters the computational complexity of aperture synthesis grows drastically. That is why antenna aperture synthesis and radio astronomy have been established as one of the most major and demanding consumers of DSP technologies.

2.1.2 RADAR Applications

RADAR or Synthetic Aperture Radar (SAR) applications are based on the principle of the scattering of electromagnetic waves. Originally, RADAR meant RAdio Detection And Ranging, however later the term became used as a standard word. The most common RADAR system consists of a transmitter and a receiver EM waves radiated by the transmitter are reflected (scattered) by a target, which are then collected by the receiver for further analysis. Any change in the dielectric constants of the target and a media surrounding it will be conveyed in the scattered waves. The basic principle of RADAR theory is illustrated in Figure 2.2.

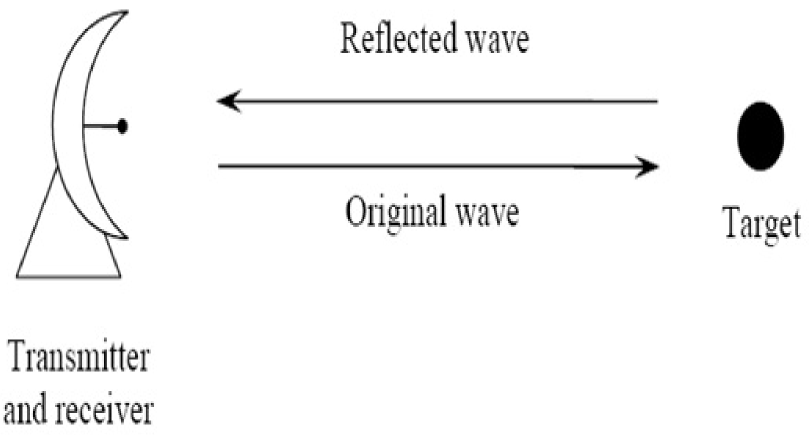


Figure 2.2: Basic RADAR Principle

The gathered data can include the objects position, movement or its particular features and attributes. The range of the applications, where RADAR technique is used, is wide: weather prediction, air traffic control, threat detection systems, military missile guidance and reconnaissance radars, etc.

Antenna aperture technique as mentioned in 2.1.1 is also used in phased array radars. In such arrays, comprised of a number of similar properly displaced antenna elements, the scanning beam is controlled by operating a phase of the signal of each individual transmitting antenna. Thus, the overall transmitted signal is maximised in a desired direction and suppressed in undesired directions.

In modern RADAR applications DSP techniques are used extensively: generation and forming of the transmission pulses, controlling the antenna beams pattern and direction, filtering of clutter, and beamforming (S. Bhaktavatsala, 2002). Cross correlation is one of the central operations in RADAR applications: it is used to find the relation or similarity between the original and reflected waves. When applied on a largescale for multiple signals and performed in a real-time fashion, such correlation becomes a challenging task.

Substantial utilisation of RADAR techniques in military area lays particular requirements on the DSP technologies in RADAR applications, eg a common trend is the need for smaller energy-efficient systems with high processing capabilities. Furthermore, typical operational signals in RADAR are very weak and with the recent tendency of radars being operated in a dense urban environment, the task of processing such signals becomes a major challenge. This issue can be mitigated by overlaying data from multiple sensors and known terrain features. In addition, newly-emerging digital beam-forming technologies based on a high-speed digital systems work with an ever-increasing number of scanning beams. The latter two issues increase RADAR system processing requirements considerably.

2.1.3 Medical Applications

Nonetheless, signal-processing technologies are not solely used for cognitive purposes.Perceptive, non-intrusive analytical capabilities of radio imaging make it an excellent option for diagnoses in medical areas. In the context of digital signal processing, the most interesting category amongst all the categories comprising the medical imaging is the ultra-wideband (UWB) imaging, which in turn is used primarily for early-stage breast cancer detection.

One of the most crucial factors in successful breast-cancer treating is detecting it at the earliest stage possible. Contemporary diagnosis methods like X-ray imaging, Magnetic Resonance Imaging (MRI), and ultrasound are capable of reducing malignant tissue. However, problems with a relatively high rate of false-negative diagnosis along with many unnecessary biopsies due to the low positive predictive rate make the use of X-ray mammography difficult and ineffective. Other methods like MRI and ultrasound are somewhat more effective in cancer lesions detection, yet do not always provide the necessary level of sensitivity, can be too operator specific and are very expensive.

Many of the existing drawbacks in early-stage breast-cancer detection can be alleviated with ultra-wideband imaging technology. The UWB imaging method employs the radar technique which was described in 2.1.2. A transmitting antenna (or a set of antennas) radiates a scanning burst of microwave energy. This electromagnetic energy penetrates through the target under investigation, scatters on the target, and further is collected by a receiving antenna or with an array of antennas. Then, the processing takes place with the primary goal to identify the presence and location of the considerable backscattered energy an indication of the dielectric difference between malignant and healthy tissue. Thus, the post-processing of the received signals has to be very sensitive to filter out the necessary information from the antennas noise, clutter due to heterogeneity in the breast tissue etc. Moreover, it has to be precise image resolution on the order of millimetres is desirable. Similar to RADAR applications, during the post-processing stage correlation is applied to find similarities or discrepancies in tissue readings provided by transmitted and reflected waves. Therefore, modern approaches of the existing radar application have to be adapted and improved, according to the requirements of UWB medical imaging. Now the medical diagnosis tools are still expect the DSP instruments to deliver an efficient and reliable method of breast-cancer testing.

2.1.4 Telecommunication

Communication technologies are one of the most actively developing areas today. The emergence of new wireless services along with the high growth of data rates in existing services, indicates an ever-growing demand for telecommunication capacity. With worldwide deployment of 3G networks, releasing beyond-3G and 4G standards and specifications, the challenges for the DSP area keep accumulating: high data throughputs (up to 1 Gbps), multimedia communications, seamless global roaming, maintaining high user capacity, and supporting migration and the compatibility between existing previous-generation and upcoming next-generation networks, etc . Consequently, the research community is focusing on different advanced signal processing issues to achieve substantial improvements in communication systems.

To demonstrate the computation requirements that lay in the telecommunication area, an example of Code Division Multiple Access (CDMA) standards can be used. CDMA based standards (CDMA2000, W-CDMA, etc.) have become increasingly popular during the emergence of the third generation networks due to their objective to maintain the ever-growing data throughputs and efficient spectrum utilisation. In brief, the idea of CDMA implies that a number of users share the same bandwidth of frequencies and are distinguished by the individual code (pseudorandom code). Such an approach has a much higher data bandwidth than traditional Time and Frequency Division Multiple Accesses (TDMA and FDMA respectively). However, these benefits are balanced with certain difficulties. For instance, the choice and assignment of a pseudorandom code to user is not a very simple routine in highly populated large-scale mobile networks. This problem can be computationally-intensive so certain solutions were proposed to address this issue. Similarly, an analogous problem arises on the receiving side to decode signal from multiple users in the most efficient and fastest way. It has been indicated that this problem also has significant computational needs.

Furthermore, the underlying complexity of the CDMA algorithm implies a challenging and complicated processing mission itself: as long as in CDMA the users share the same bandwidth the multiple access interference (MAI) has to be considered and alleviated. Prevention of this interference is exacerbated by the intersymbol interference (ISI) and multipath signal propagation which is natural to all urban mobile networks. For this purpose sophisticated channel estimation algorithms are applied. The computational complexity of such algorithms is considerable and furthermore they have to be implemented in real-time fashion. Therefore, the research community has turned to elaborate DSP techniques like real-time DSPs and FPGAs to respond to these challenges.

Tackling the processing difficulties is not of course the feature of only the CDMA standard. As it was mentioned above, with the rapidly-growing rates, throughputs, capacities, etc. the industry is facing expanding requirements throughout its applications. One example is 3G and 4G mobile standards. These standards offer high data throughputs to the end-users even comparable to office LANs in 4G networks. To supply such high speeds, a number of advanced and complex techniques are employed in these standards. One such technique is smart antennas. To maintain high data rates in complex urban environments these antennas use adaptive beamforming and direction-of-arrival (DOA) estimation algorithms. In turn, these algorithms employ cross-correlation operation for estimation which signals arriving from which directions to suppress and which to maximise. Such calculations have to be performed with complex numbers and most importantly should be done in real-time. Thus, it is evidently seen that the necessity for high-performance signal-processing utilities spans across the whole communication industry, leaving researchers in unrelenting pursuit for an adequate response.

2.2 Correlation as a Typical DSP Application Problem

In many of the aforementioned applications an integral and common part can be singled out all of them are dealing with combined sources of information providing a synergistic combination of knowledge about the investigated object. In other words, whenever a system is dealing with a number of input data streams collaboratively reducing the entropy of a studied phenomenon, the term multi-sensor data fusion is applied. The integral part of this fusion is to express the joint result of analysis of two or more originally different sources. For that reason a correlation operation is applied, which in turn is regarded no less as a backbone of the whole DSP area.

Thus, the prevailing number of high-end DSP applications such as antenna aperture synthesis, radioimaging, RADAR, radio astronomy, high-energy physics and many others, has a common and very computationally-intensive part the multichannel wideband correlation of signals. Correlation or, more generally speaking, finding a relation between a set of signals, is a computational core for the majority of signal processing operations and is considerably critical for computation performance.The result of the cross-correlation function is a measure of similarity between a pair of energy signals .

As it was noted before in 2.1.1, one of the applications where correlation is applied is radio astronomy. For example, it is used in the radio-astronomical technique known as Very Long Baseline Interferometry (VLBI). In turn, the antenna aperture synthesis is used in VLBI. The latter technique implies that the correlated product of signals from two radio telescopes gives visibility frequencies of celestial object. The frequency information is obtained by averaging additional multiplications by a lagged signal and finally the data is transferred to the frequency domain by applying Fourier transform.

2.2.1 Correlator Theory

A measure of similarity between a pair of signals, x[k] and y[k], is given by the crosscorrelation Rxy[k] sequence:

$$Rxx[n] = \sum_{k} x[k]y[k-n] \tag{2.1}$$

where the lag index n $[N/2, N/21]$, k is the time index, N is a number of lags and typically is a power of two. The lag term denotes the time-shift between the pair of signals with negative (n 0) and positive (n $\geq$ 0) lags being distinguished. Basically, the number of lags defines how many points or output values the correlation produces. In real life applications, where for example the correlation function is used together with Fourier transform, the number of lags can be referred as the resolution of correlation. A device which performs correlation of a set of signals is called a correlator. The number of lags is an important characteristic of a correlator along with the number of channels, ie number of supported input signals. When a signal is correlated with itself, such an operation is called autocorrelation and is often used in filtering and other processes.

One should note the incurred execution time or time complexity for wideband correlation. For a wideband signal according to the Nyquist condition, the processing involves computation of a greater amount of samples, hence the processing duration increases. In addition, the results of correlation computation abide to the following law:

$$N = \frac{Ns(Ns-1)}{2} \tag{2.2}$$

where N is the total number of the cross-products and NS is the total number of antennas (sources) to be correlated. Hence, wideband multi-channel correlation embraces a considerable amount of computations.

Strictly speaking, Equation (2.2) gives the number of unique correlation results or half of the total correlation results the remaining half can be obtained by simply reversing the results from the first half. The latter issue is caused by the following property of correlation: correlation of x[k] with y[k] is not the same as correlation of y[k] and x[k]. So, putting down mathematical notation of correlation of y[k] with x[k]:

$$Ryx[n] = \sum_{k=-n}^{n} x[k]y[k-n] = \sum_{l=-n}^{n} y[l+n]x[l] = Rxy[-n] \tag{2.3}$$

Thus, Ryx[n] is obtained by time-reversing sequence Rxy[-n].

2.2.2 Digital Correlators

As mentioned above, the number of lags is an important feature defining the resolution capabilities of a correlator. The higher the number of lags, the better a correlator can tell how similar two signals are to each other. In reality, the number of lags is set by the applications requirements and defines the number of multiply-and-accumulate and multiply-and-add computations. The latter statement is true for digital correlators, ie correlators that work with a stream of digitised samples x[n] from an analog output x(t). Two general types of digital correlators are distinguished:

1. Lag or XF Correlator

2. FX Correlator

In the lag or XF correlator Fourier transform to the frequency domain is performed after cross multiplication of signals. The number of channels in such correlators is an integral power of two with the signals' bandwidths also divisible by two to be compatible with digital computing techniques .

Whereas in the FX correlator Fourier transform is performed before cross multiplication of signals. Therefore, the total number of operations on the FX correlator is proportional to the number of antennas or more correctly signals coming from these antennas, whereas in the XF correlator the amount of computation is proportional to the number of antenna (signal) pairs. Hence, the FX correlators are more economical in terms of hardware requirements especially for a considerable number of signals.

2.2.3 Implementations of Correlators

Correlators can be implemented in hardware or software. Normally, hardware correlators are designed and manufactured for a certain and specific application and are implemented in Very Large-Scale Integrated (VLSI) circuits. The CABB Hardware Correlator is an example of a hardware correlator. This correlator has a complex and very large-scale architecture comprising a number of VLSIs, multiplexers, accumulators, filter banks and other devices. It is utilized in Australia Telescope Compact Array to process signals from six 22 m. antennas of Australia Telescope Compact Array. In addition, FPGAs are used in this correlator as well to produce different configurations of filter banks.

As for software correlators, they are implemented as a set of libraries or computer programs to perform the designated task: correlation of a given set of signals. Amongst known and acknowledged software correlators the following need to be mentioned:

K5 Software Correlator is probably one of the most famous correlators implemented in software. Currently the K5 correlator is involved in the VERA project in Japan and furthermore in collaborative work of Korea and Japan in the project East Asian Correlator in Seoul. K5 is an FX correlator.

Swinburne University of Technology has another software-based correlator.Initially this correlator was XF-type but it was considered slow and the recently new FX correlator DiFX has been implemented and tested. Both correlators have been implemented on the Linux parallel high-performance parallel cluster utilizing the Message Passing Interface (MPI) standard for process-to-process communications. There was a reported intention to explore hybrid architecture (ie comprising FPGA and Swinburne cluster) within this FX correlator.

The Jet Propulsion Laboratory of California Institute of Technology designed Softc software correlator . Launched as one of the many test programs to replace an outdated hardware correlator Block I in the Delta-Differenced One-way Range (DeltaDOR) spacecraft navigation system, Softc underwent a lot of changes and finally was employed in Mars Odyssey, Mars Exploration Rover, Deep Space 1 and other missions. It has significant processing accuracy (not less than 10-13); it can correlate 1, 2, 4, and 8-bit sampled data, upper, lower, or double sideband data and data using one of either two encoding schemes.

Range (DeltaDOR) spacecraft navigation system, Softc underwent a lot of changes and finally was employed in Mars Odyssey, Mars Exploration Rover, Deep Space 1 and other missions. It has significant processing accuracy (not less than 10-13); it can correlate 1, 2, 4, and 8-bit sampled data, upper, lower, or double sideband data and data using one of either two encoding schemes.

The international TOP500 list encompasses the 500 fastest and most powerful computing systems around the world (www.top500.org). As of November 2007, the top supercomputer is the Department of Energy's IBM BlueGene/L system in USA with a performance of nearly 500 TFlops. Another BlueGene/L computer located at the University of Groningen performs correlation tasks in a Low Frequency ARay (LOFAR) project. It consists of 12,288 700 MHz dual PowerPC 440 cores yielding 34.4 TFlop/s of correlation performance.

Nevertheless, such performance comes at a price development time and maintenance cost balance this substantial computational power. With BlueGene/Ls power consumption of 27.6 kW per rack (IBM Corporation, 2006) the LOFARs sixrack supercomputer consumes 165.6 kW per hour. Besides, the estimated development time is one man-year . One of the reported issues with the LOFARs correlators is the lack of the high bandwidth in BlueGene crucial for streaming DSP applications and overall necessity of faster intercommunication between the cores. Moreover, software correlators require substantial debugging and testing of the code: eliminating of processing errors and inaccuracies was one of the greatest hurdles in Softc correlator implementation. Developing, debugging and testing can be generalised as one of the greatest hurdles for all high-performance DSP systems.

Along with the considerable complexity of developing high-performance DSP systems go their relevant energy requirements. At the Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'08), Pete Beckman from Argonne National Laboratory in his keynote speech made a strong point about the power consumption requirements of contemporary and future supercomputers. In particular, it was predicted that within years the power consumption of a computational system would become the most determinative haracteristic. In time, increasing the processing capabilities by increasing the operational frequency and adding additional transistors (Moores law), depleted itself and gradually diverted to multi-core and parallel execution of the algorithms, where currently most of the research and development work is carried out. In turn, the same is envisioned for parallel operation parallelisation of the applications and algorithms will eventually exhaust with Flops per Watt ratio becoming the systems performance measuring unit. Therefore, with power requirements becoming one of the most significant factors additional constraints are laid upon the development of high performance DSP systems and, most importantly, on the technologies applied in these systems.

2.3 DSP Technologies

2.3.1 The Performance Requirements of the DSP Applications

The number of mathematical calculations involved in the aforementioned high-end DSP applications is extremely high. For instance, to perform only a 1,024-point FFT yields 10,240 complex multiplications and additions per operational cycle. Moreover, to provide trustworthy data, a radio telescope observing a celestial phenomenon has to employ FFT with even higher resolution as well as a number of other operations, eg correlation of wideband radio-frequency signals, thus yielding even higher number of computations. On top of that, any DSP application, whether it is an image processing routine or telecommunication operation, demands these computations be executed in a rapid manner.

Besides, relentlessly expanding requirements of todays electronic systems keep pushing the resources contemporary DSP instrumentation towards and over the verge of depletion. Figure 2.3 illustrates the performance gap that has emerged in the communication industry between increasing algorithm complexity originated from recent standards revolution and existing processing architectures.

So, how can one approach the ever-growing demands of the DSP field? The most universal approach to meet the substantial and constantly growing requirements of highend DSP applications is to increase the processing power of computational units (CPUs, DSPs, ASICs, etc).

This has a number of limitations and drawbacks such as:

1. High power consumption, which in turn leads to necessity of efficient power dissipation;

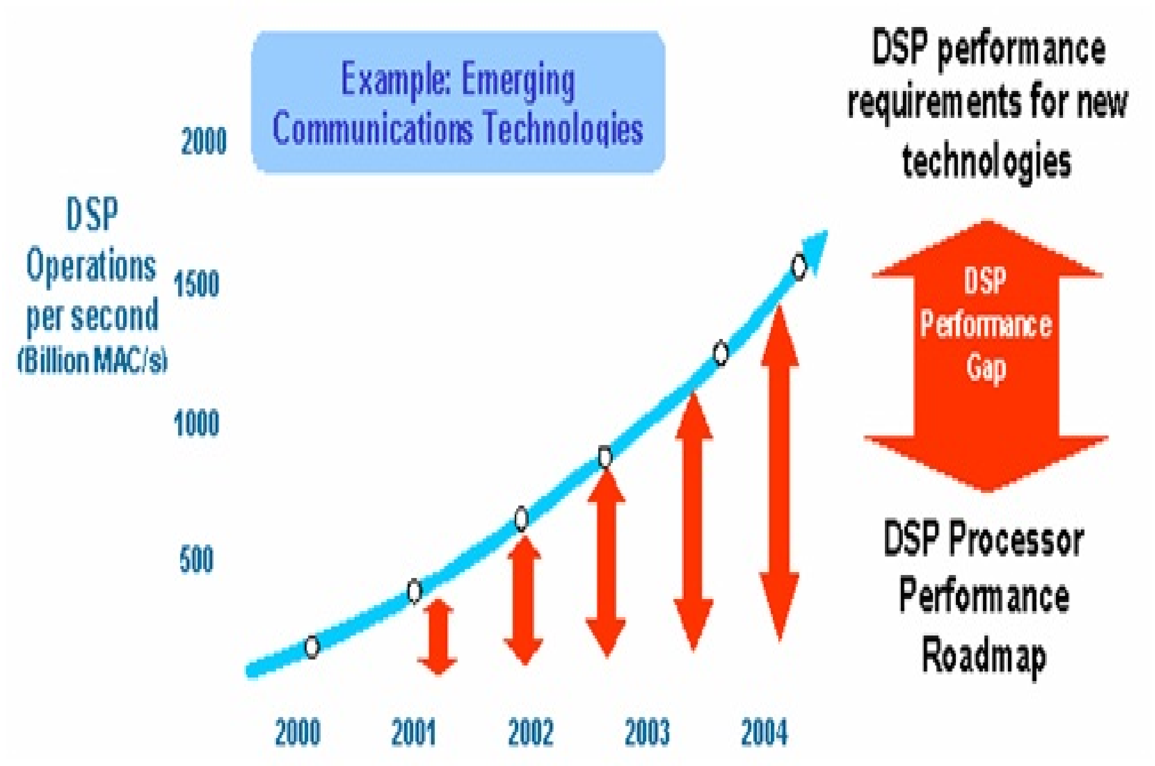


Figure 2.3: The DSP Performance Gap in Communications Industry

2. Complexity of accomodating a large number of transistors in a single chip, which are growing with each year according to Moores Law;
3. High market costs.

Hence, this is not always feasible to cover the requirements of a certain high-end DSP application by simply involving more computational power (units) due to the hardware constraints in contemporary tools. Therefore, the search focus has to be shifted towards renovating or enhancing the existing apparatuses or creating new ones. The next sections cover the most common tools available in the DSP field.

2.3.2 Digital Signal Processors (DSPs)

Currently there is a number of tools in the DSP area. One of the major tools for DSP applications are Digital Signal Processors. DSPs were first created in the late 1970s S2811 and Intels 2920. Although Intels device did not have a multiplier, it already had on-chip ADC and DAC a feature still present in the modern DSPs. The 1980s saw a second generation of DSPs with supported concurrency, multiple buses and on-chip memory. These were added with on-chip floating point operations in the third generation of DSPs in the early 1990s. In the late 1990s multi-processing features, image and video processors and lowpower DSPs were introduced.

Today DSPs are produced by semiconductor vendors such as Texas Instruments,Analog Devices, Motorola and others. Contemporary top-level DSPs are capable of achieving substantial speeds for example the high-performance multi-core TMS320C6474 from Texas Instruments can achieve up to 24,000 million instructions per second (MIPS) or 24,000 16-bit MMACs per cycle (Texas Instruments Inc., 2008). This DSP is also equipped with a 16/32-bit DDR2-667 Memory Controller, EDMA3 Controller, 1000 Mbps Ethernet MAC interface, two 1x Serial RapidIO Links and many other peripheries.

In general, DSPs are a specialized form of microprocessor designed specifically for digital signal processing. Nowadays DSPs have a well-developed tool set typically a high-level programming language as C++. DSPs perform real-time processing and have fixed hardware architecture with certain set of resources. Hence, DSPs have reconfigurability freedom only to the extent of the programming code running on them. Furthermore, the performance requirements of todays DSP applications have now exceeded the capabilities of even such powerful DSPs as Texas InstrumentsTMS320C6474.

Another common platform for performing DSP applications Application Specific Integrated Circuits (ASICs) possesses an alternative approach for performing signal processing applications.

2.3.3 Application-Specific Integrated Circuits (ASICs)

The inception of Application-Specific Integrated Circuits or more commonly ASICs started in 1980s when the now-defunct Ferranti Company released the first gate-array Uncommitted Logic Array (ULA). The first Uncommitted Logic Arrays contained only a few thousand gate circuits (transistors, logic gates, and other active devices) and they did not perform any specified function. A particular function of a ULA was configured by adding a final layer of metal interconnects to the ULA thus connecting the elements on the ULA in the desired, customised fashion. The later versions of these early developments became more complicated with a greater number of gates and in some cases included RAM elements.

Modern ASICs retain the same ideology they perform only limited sets of tasks laid in them during manufacturing stage. These devices are capable of performing their limited sets of functions faster than general-purpose DSPs. Due to application-specific circuitry ASICs are able to employ high-speed functions of the targeted algorithm in the optimized hardware (Kuo Lee, 2001). Most commonly ASICs are used for implementing well-tested and well-defined algorithms, eg Reed-Solomon coders in digital subscriber loop (xDSL) modems or stack functionalities of CDMA2000 standard in cell phones.

Depending on the grade of flexibility, three levels of ASICs are distinguished:

1. Gate Array is the least customisable. Transistors, gates and other devices are predefined but unconnected no metallization layers exist. A user specifies interconnection between the elements thus defining the function of the device. Today these devices are gradually replaced by structured ASICs where many features are predefined by the manufacturer: IP cores, power and clock sources, etc. This significantly reduces the design time, as a user has to specify much fewer design technicalities.

2. Standard cell methodology has a high degree of flexibility. It assumes that the ASICs design is defined by a user from the cell libraries created by the manufacturer and , therefore, has much less space for mistake than full custom design.

3. Full custom design is the most flexible and, therefore, the most expensive and time-consuming approach. It assumes developing an ASIC from transistor level.

Despite that ASICs can perform their specified application faster than generalpurpose DSPs, they do posses their own challenges and limitations. The most obvious limitation of ASICs originates from their most prominent strength: hardware optimised for performing dedicated applications means little or, most often, absolutely no degree of algorithm flexibility.

Another challenge with ASICs is that they are configured with hardware description languages (HDL) such as Verilog, VHDL and some other less popular options. These languages are low-level programming languages and differ significantly from high-level programming languages employed for programming conventional general-purpose DSPs. The challenges of hardware description languages are more broadly discussed in section 3.1.2.

Single DSP or ASIC can be employed as a platform for single or several signal processing applications. In the case of large-scale high-performance DSP applications, they may be employed as building blocks in sizeable computational systems such as supercomputers, computational clusters, grid computing, etc.

2.3.4 Hign-Performance Computing

For performing large-scale DSP applications, high-performance computing (HPC) systems can be used. HPC systems (supercomputers or computer clusters) comprising multiple computational processors communicate through versatile types of interconnect. The types of DSP applications employed on HPC systems are exceedingly large-scale and include but are not limited to: correlation of wideband RF signals involved in radio observation of celestial objects (eg CABB or DiFX), video-centric applications of new generation wireless telecommunications standards, such as wireless videoconferencing, real-time video streaming, etc. and many others.

Over the years, the HPC proved to be an effective and sophisticated tool for performing DSP applications. Technologies and tools applied in HPC have significantly developed over the past years density of transistors on processors (Moores law), communication speeds and throughputs, number of processors performing one task, uniform memory access with few or no caches, etc. In addition, modern HPC systems are practically linearly scalable.

Moreover, HPC systems have the potential to perform the assigned task in parallel, ie the task is split into several parts, each of which is performed by a separate computational unit in parallel. This is achieved by either using multiple computational processors within a single computer, ie a multiprocessor, or by multiple computers working on a single problem. The possibility to perform tasks in parallel becomes radically beneficial for DSP applications as most of them can be easily parallelised. More precisely the majority of DSP applications fall under the Single Instruction, Multiple Data streams (SIMD) category in taxonomy introduced by Michael J. Flynn. Figure 2.4

illustrates SIMD architecture.

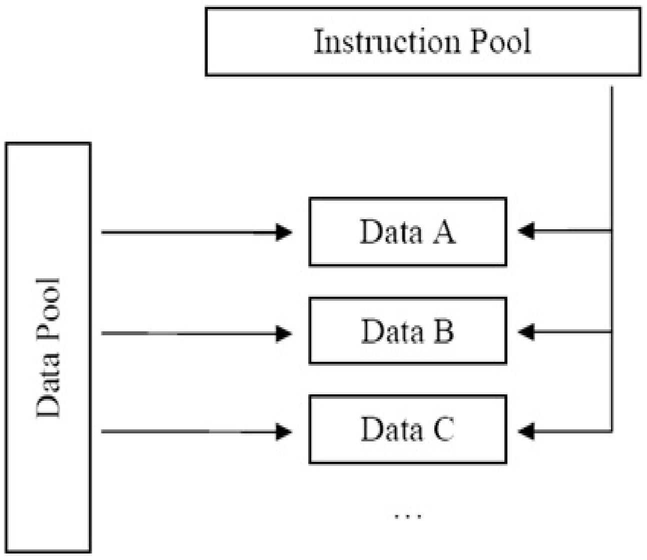


Figure 2.4: Single Instruction, Multiple Data (SIMD) model

In the SIMD model the same set of operations from the Instruction Pool is applied on different data streams Data A, Data B, etc from the Data Pool simultaneously and, therefore, this processing can be naturally parallelised.

A certain application can benefit from SIMD implementation if it involves a large number of the same repetitive operations applied on a large number of data bits. Many DSP applications satisfy this condition. For example, the correlation described by the Equation (2.1) consists of a number of simple mathematical operations, namely multiplication and addition which are applied to the same data set samples of input signals. Hence, computation of a single sample of a correlation function involves a precisely calculated number of calculation routines on a certain input sample. These routines can be successfully parallelised thus attaining a speed-up in the performance which consecutively leads to power conservation and increased throughput.

Real-life SIMD implementation examples include Intels MMX processors, their AMDs counterparts 3DNow! Processors, Graphics Processing Units of PC video cards, and many others. The SIMD model is applied in large-scale supercomputers as well.

2.3.5 FPGAs as a DSP Tool

Another prominent tool for parallelisation is a maturing field of FPGAs, which has drawn massive attention in recent years from leading electronics developing vendors and designers throughout the world. Recent profound advances in the Field Programmable Gate Array area demonstrate that signal, image and video processing applications which are typically implemented on FPGAs, comprise complicated calculations over a large amount of streaming data. These applications can gain substantial speed-up from available on-chip parallelism. The technological background of FPGAs is discussed in more details later.

An FPGA is a semiconductor device containing programmable logic blocks which can be interconnected and configured to meet the desired functionality specified by a certain application. Once an FPGA is programmed it operates as optimised hardware developed for a particular task. Designs incorporating FPGAs have at least two significant advantages in comparison with DSP devices and ASICs:

1. Parallelism the ability to perform several operations in parallel and therefore performs faster;
2. Reconfigurability or, in other words, the ability to be customised for a certain application.

FPGAs parallelism feature allows them to perform more operations at a single clock cycle than their conventional processing counterparts. Therefore, FPGAs operate at much lower frequencies than their conventional processing counterparts while achieving similar or even greater performance results. Lower operational frequencies lead in turn to lower power consumption, which has become one of the most crucial issues in recent years and is predicted to play an ever more dominant role in the foreseeable future of high-performance computational systems.

In the past years, the computational capabilities of commercially available FPGAs even overcame some commercially available CPUs in terms of achievable performance see Figure 2.5.

Figure 2.5 demonstrates millions of floating point operationsper second (MOPS) achievable by the FPGA representative (Xilinx FPGA) and the

CPU representative (Intel Xeon CPU) throughout their release dates. FPGA field is already renowned as a new computational paradigm by the research community.

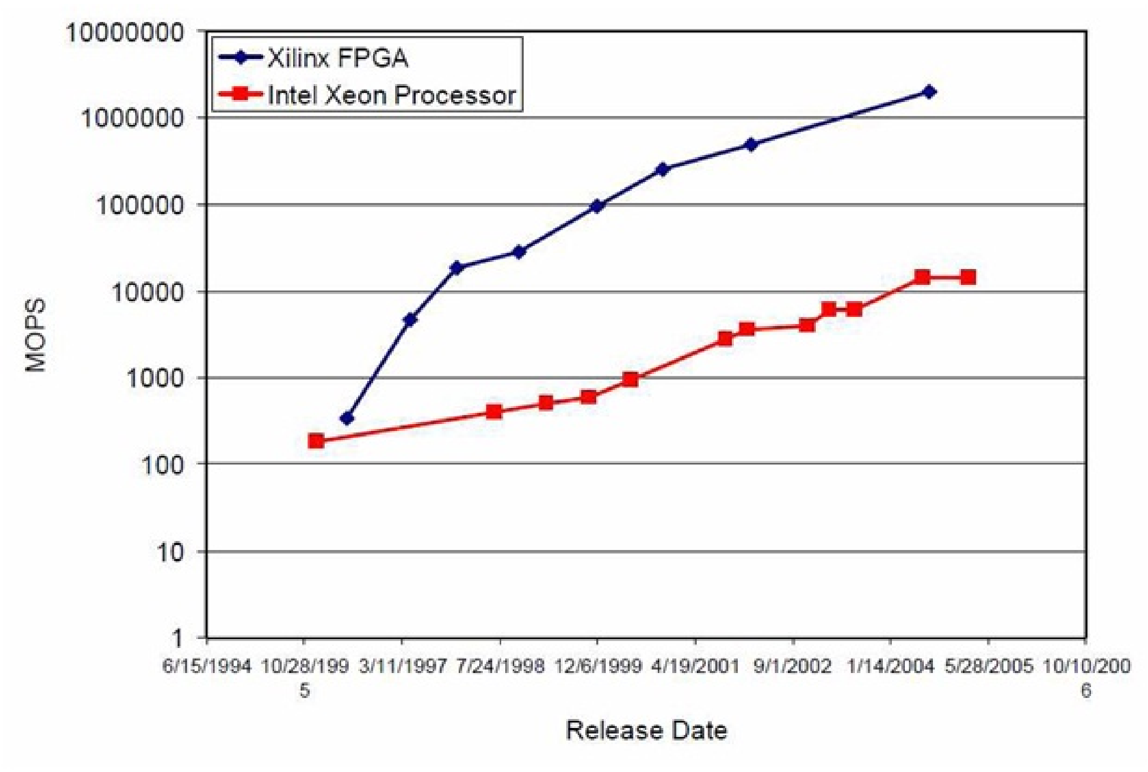


Figure 2.5: Moore's Law in the CPU and FPGA World

Nevertheless, along with prominent beneficial features FPGAs have certain challenges and drawbacks. First of all FPGAs are configured with low-level hardware programming languages which incur considerable programming and debugging efforts. Furthermore, all of the interfaces and features present on FPGAs have to be explicitly configured for each particular application/design. This and other challenges in FPGA programming discussed in later, complicate the utilisation of FPGAs for DSP applications.

2.4 Chapter Summary

This chapter introduces the background of the investigated problem. The following typical computationally-intensive DSP applications employing cross-correlation of signals are discussed: radio astronomy, RADAR applications, medical applications, and telecommunication.

The theory of the targeted cross-correlation problem with the focus on digital implementation of correlation is described. This is followed by a discussion on the computational requirements of the modern DSP applications. Traditional technologies (DSPs and ASICs) for performing DSP applications are considered along with methods of implementation of large-scale DSP applications (HPC). Their features and existing challenges are discussed.

Chapter 3

Linear Correlator Architecture

The future is always beginning now.
-Mark Strand

This chepter the investigation on a linare correlator architecture and highlights its potential advantages and challenges. Employment of this architecture for high performance DSP application is discussed along with the applicability of contemporary reconfigurable computing system for performing these application.

In statistics, dependence refers to any statistical relationship between two random variables or two sets of data. Correlation refers to any of a broad class of statistical relationships involving dependence.

Two types of correlation (I) Auto- Correlation (II) Cross-Correlation. Both in signal and Systems analysis, the concept of autocorrelation and crosscorrelation play an important role. In the following sections, we present some simple examples of how these two functions may be estimated in Matlab. The final part of this section provides some applications of autocorrelation and crosscorrelation functions in signal detection and time-delay estimations.

3.1 Auto Correlation

The autocorrelation function of a random signal describes the general dependence of the values of the samples at one time on the values of the samples at another time. Consider a random process x(t) (i.e. continuous-time), its autocorrelation function is written as:

$$Rxx(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)\ x(t+\tau)\, dx \tag{3.1}$$

Where T is the period of observation.

$Rxx(\tau)$ is always real-valued and an even function with a maximum value at $\tau = 0$. For sampled signal (i.e. sampled signal), the autocorrelation is defined as either biased or unbiased defined as follows:

$$Rxx(m) = \frac{1}{N- \mid m \mid} \sum_{n=1}^{N-m+1} x(n)x(n+m-1) Biased Autocorrelation \tag{3.2}$$

$$Rxx(m) = \frac{1}{N} \sum_{n=1}^{N-m+1} x(n)x(n+m-1) Uniased Autocorrelation \tag{3.3}$$

For m=1,2,..M+1. where M is the number of lags.

3.1.1 Properties of Auto Correlator Function

1. Maximum Value:

   The magnitude of the autocorrelation function of a wide sense stationary random process at lag m is upper bounded by its value at lag $m = 0$:

   $$\text{Rxx(0)}\geq Rxx(k) \qquad for \text{k}\neq 0$$

2. Periodicity:

   If the autocorrelation function of a WSS random process is such that: $Rxx(m_0) = Rxx(0)$ for some m_0, then Rxx(m) is periodic with period

m_0. Furthermore $E[x(n) - x(n-m)^2] = 0$ and x(n) is said to be mean-square periodic.

3. Symmetry:

   The autocorrelation function of WSS process is a conjugate symmetric function of m:

   $$Rxx(m) = Rxx * (-m)$$

   For a real process, the autocorrelation function is symmetric:

   $$Rxx(m) = Rxx * (-m)$$

4. Mean Square Value: The autocorrelation function of a WSS process at lag, $m = 0$, is equal to the mean-square value of the process:

   $$Rxx(0) = E|x(n)|^2 \geq 0$$

5. If two random processes x(n) and y(n) are uncorrelated, then the autocorrelation of the sum $x(n) = s(n) + w(n)$ is equal to the sum of the auto -correlations of s(n) and w(n):

   $$Rxx(m) = Rss(m) + Rww(m)$$

6. The mean value: The mean or average value (or d.c.) value of a WSS process is given by:

   $$mean, \bar{x} = \sqrt{Rxx(\infty)}$$

3.1.2 Matlab Implementation on Auto Correlator Function

Matlab provides a function called xcorr.m which may be used to implement auto correlation function. Its use is indicated in the following examples.

when using the function xcorr, to estimate the autocorrelation sequence , it has double the number of samples as the signal x(n). An important point to remember when using the function xcorr is that the origin is in the middle of the figure (here it is at lag=1024).

Plot the autocorrelation sequence of a sinewave with frequency 1 Hz, sampling frequency of 200 Hz.

Note this version of estimating the autocorrelation function generates the same number of samples as the signal itself and that the maximum is now placed at the origin. (Rxx(1) is the origin).

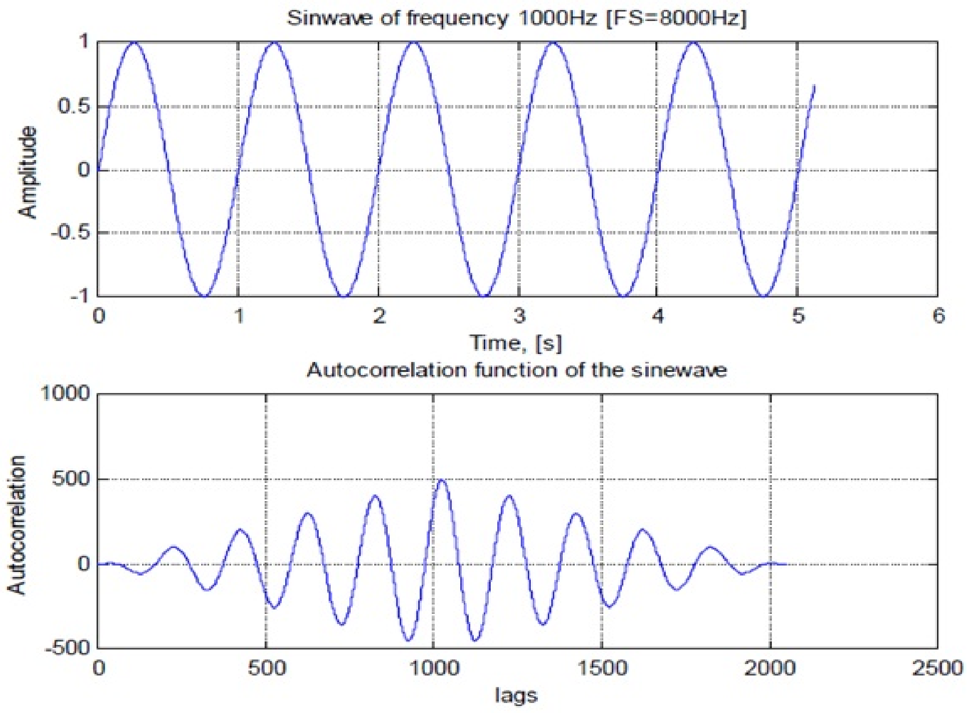


Figure 3.1: Matlab Simulation of Auto-Correlation

3.2 Cross Correlation

In signal processing, cross-correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them. This is also known as a sliding dot product or inner-product.

For two WSS(Wide Sense Stationary) processes x(t) and y(t) it is described by,

$$Rxy(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)\ x(t+\tau)\, dx \tag{3.4}$$

or

$$Ryx(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)\ x(t+\tau)\, dx \tag{3.5}$$

where T is the observation time.

For sampled signals, it is defined as:

$$Ryx(m) = \frac{1}{N} \sum_{n=1}^{N-m+1} y(n)x(n+m-1) \tag{3.6}$$

3.2.1 Properties of Cross Correlator Function

1. R$_{xy}$(m)is always a real valued function which may be positive or negative.
2. R$_{xy}$(m)may not necessarily have a maximum at m=0 nor R$_{xy}$(m)an even function.
3. R$_{xy}$(-m)= R$_{yx}$(m)
4. R$_{xy}(m)^2 \leq R_{xx}(0)R_{yy}(0)$
5. $|R_{xy}(m)| \leq [R_{xx}(0) + R_{yy}(0)]$
6. When R$_{xy}$(m)=0, x(n) and y(n) are Sid to be uncorrelated or they are said to be statistically independent (assuming they have zeros mean.)

3.2.2 Matlab Implementation on Auto Correlator Function

In the same frequency at the auto correlator function in sine wave and compare a pure noise signal, output of matlab is below :

3.3 Implementation of Linear Correlator Architecture

3.3.1 Theory of Linear Correlator Architecture

This so called 'linear' algorithm was used in the first commercially available correlator devices and represented the prior implementation form of digital correlators over several years. The parameter n denotes the detected photon count rate, i.e. the number of photon counts over one sampling time clock (STC) interval. The STC range, which represents the inverse of the system clock fsys, is defined as $\tau = t(i+1) - t(i)$, Considering a linear correlator structure, the dynamic range of the correlator is determined by the spread of discrete lag times $\theta_j : \theta_{min} < j.\tau < \theta_{max}$. Thus, the different

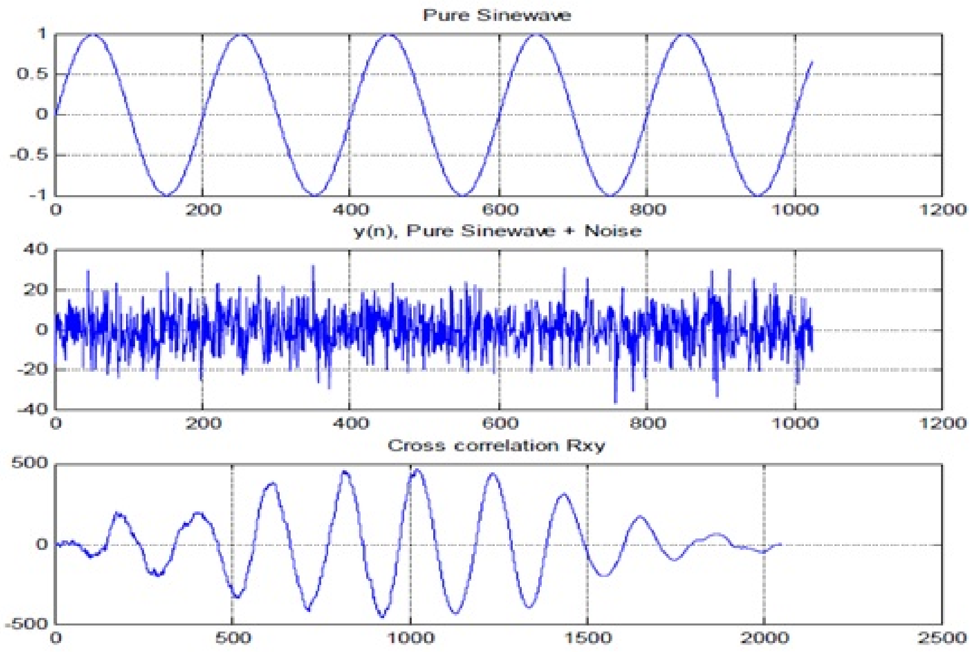


Figure 3.2: Matlab Simulation of Cross-Correlation

lag times in the linear correlator scheme are separated by one sampling time interval each. For every single lag time θ_j, an individual correlation channel $Corr(\theta_j)$ exists, that simply sums the products of the direct and delayed samples n(0) and $n(\theta_j)$. This multiplication/summation procedure is done for a certain number of different lag times θ_j in parallel to obtain a time discrete approximation of the full correlation function. A linear correlator can be easily implemented of simple multiplier-accumulator structures as illustrated in Figure . Apparently, the lag time range of a linear correlator is limited by the number of channels.

To achieve the covering of a dynamic range of 10^12, the number of correlation channels must be at least as high as the temporal range, i.e. 10^12. Since hardware resources are limited, the required dynamic range is in direct conflict with a fine timing resolution to accurately approximate the correlation time integral. One way to increase the temporal dynamic range of a correlator without increasing the number of correlation channels too much is to restrict the actual calculation to certain lag times only. An exponential

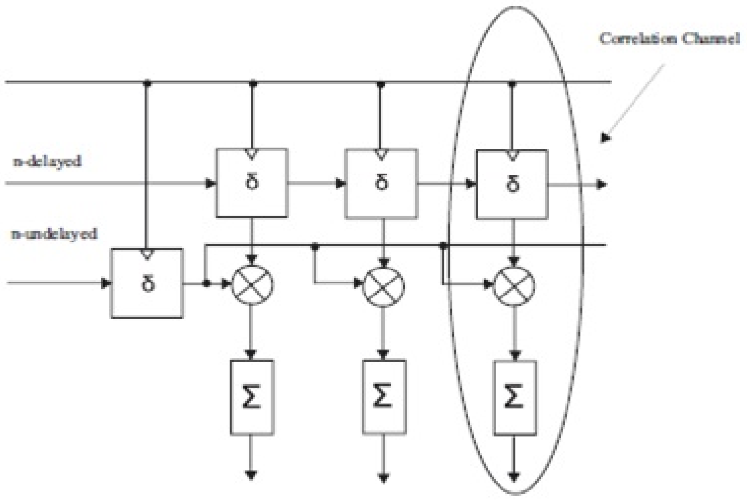


Figure 3.3: Linear Correlator Architecture

distribution may be used, so that the lag times becomes:

$$\theta_1 = \tau \text{ and } \theta_{i+1} = \tau$$

where k is the delay factor which remains constant for the complete lag time range. The number of correlation channels required to cover a given lag time range is now a function of the delay factor and may easily be computed as:

$$N = \ln(\frac{\theta_{max}}{\theta_m in}).\frac{1}{\ln(DelayFactor)} \tag{3.7}$$

which leads to values of about 240 correlation channels for a temporal dynamic range of $1 : 10^10$ and a delay factor of 1,1. However, for large lag times θ, the sampling of the correlation function reveals to be coarse. Simply more and more useful information is neglected as the lag time increases and the statistical properties of such a procedure, especially at large lag times is lost. A more promising algorithm, the Multiple-Tau Correlation Technique was published by K. Schaetzel. The major difference to the linear as well as to the exponential delay correlator structure is, that the sampling time is no longer held constant, but rather increased with the lag time. In fact, this leads to an implicit calculation of all possible products of direct and delayed samples. Blocks of eight or more correlation channels as illustrated in Figure with common sampling time are formed (in the following referred as sampling time blocks STB) and the sampling time is doubled from one block to another. Thus, the input signals for the different blocks are averaged over longer and longer periods. All events in the delayed and undelayed path are added up for each block over two sampling time periods and serve as input count rates for the next block, which operates again with half of the clock frequency. If a 16 channel sampling time block structure is applied, the first 8 of 16 correlation channels of every sampling time block STCn but STC0 are redundant. The lag-time range of these was already covered by the previous STC block, but even at a much higher temporal resolution due to the decreased sampling times. It would thus not make much sense to have all correlation channels for each of the individual STC blocks being computed from lag time index j = 0 on, but instead only those correlation channels should be computed which cover a not previously covered lag-time regime. The processing is thus reduces to $\frac{16}{8}$.

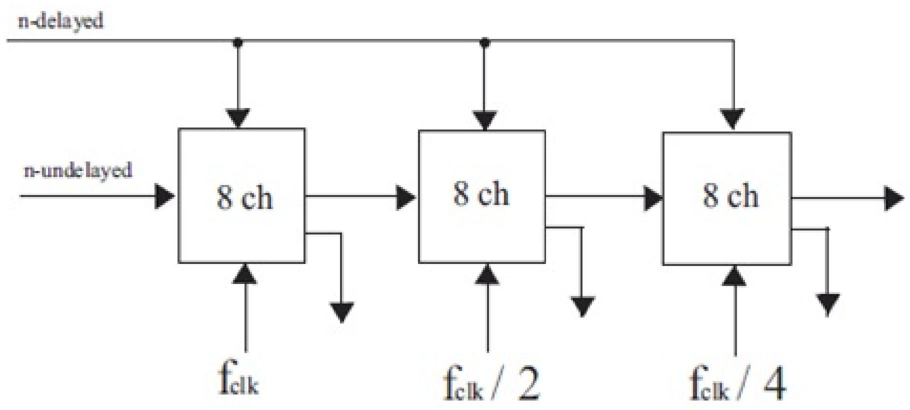


Figure 3.4: Multipe-Tau Sampling Block Structure

The Multiple-Tau correlation function estimation of the input count rates n and m can be summarized as follows:

$$Corr(n(i), m(j)) = \sum_{i=1}^{M} n(i)M(i+j), j = 0......15 \tag{3.8}$$

The additional l = 1..P sampling time blocks consisting of eight channels each similar of above equation.

The effective sampling time for every correlation channel block is determined by:

$$STC_l = STC_0.2^l$$

The effective lag-time for every correlation channel can now be calculated is :

$$\theta_j = STC_{l.j} = STC_0.2^l.j$$

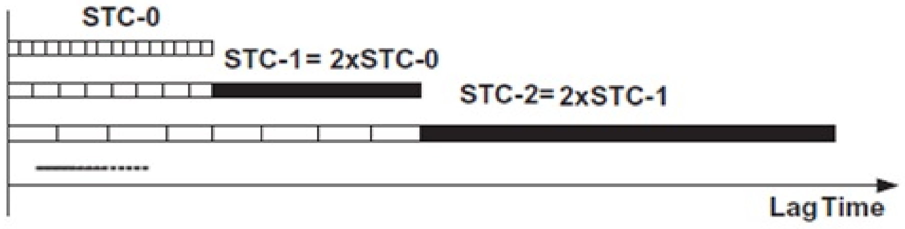


Figure 3.5: Multipe-Tau Sampling Time Block Scheme

This scheme is again graphically illustrated in Figure.Using this procedure, only 40 parallel sampling time blocks (or 328 correlation channels) are required to cover a dynamic range of nearly 10^13 with sampling times as small as 5ns for STB0 and nearly 1hr for the last STB block. Careful inspection of the Multiple Tau algorithm leads to the conclusion, that the required total processing speed needed has an upper bound with a value exactly twice as high as the processor speed needed to compute the first eight channels. The number of samples taken per given time is inversely proportional to the sampling time and as the sampling time is doubled for each block of eight channels this leads for the required processor speed F to

$$F = 8.(2.\frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ...) \rightarrow 24 \tag{3.9}$$

The first sampling time block is multiplied by two due to its 16 channel structure. Equation 11 states that Multiple Tau requires a fixed number of operations, completely independent from the temporal dynamic. Thus, for every instance in time (STC-Interval), only two different STBs must be processed.

3.3.2 8-bit Linear Correlator Design simulation

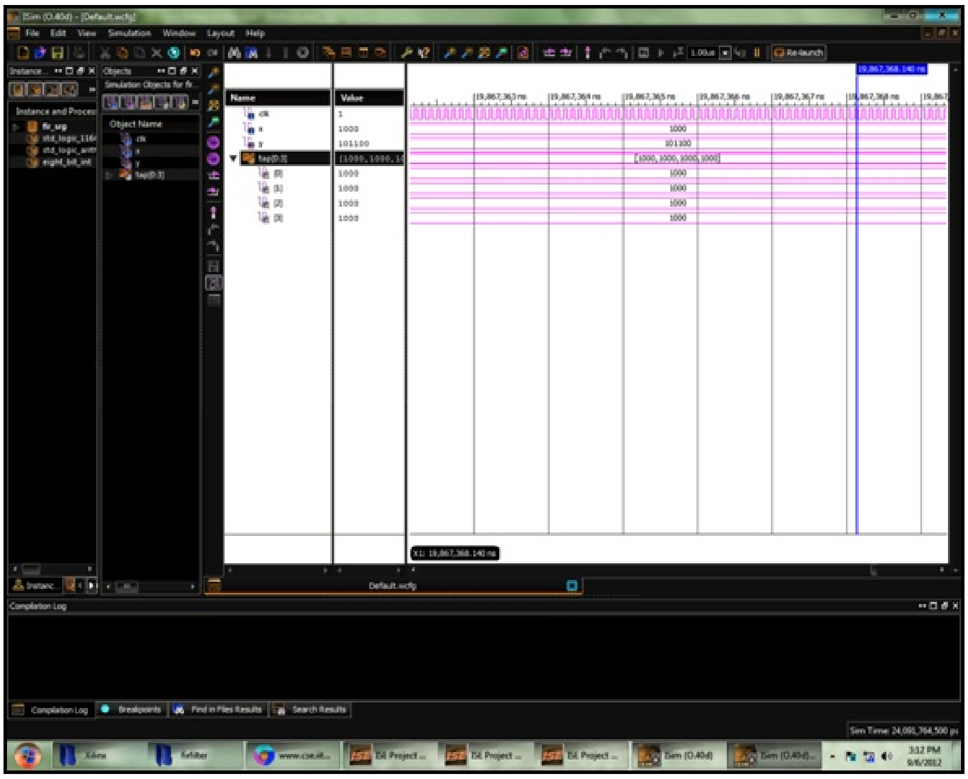


Figure 3.6: Xilinx Simulation on 8-bit Correlator

3.4 Chapter Summary

This chapter introduces the background of the Linear correlator architecture.Also discussed a properties of a autocorrelator and cross correlator. The following typical both the auto correlator and cross correlator are discussed and implementation in MATLAB. Both correlator are observed in simulation level.

The theory of the Linear correlator problem with the focus on digital implementation of correlation is described. This is followed by a discussion on the computational requirements of the modern DSP applications.Their features and existing challenges are discussed.

Chapter 4

MULTI CHANNEL CORRELATOR ARCHITECUTE

We can't solve problems by using the same kind of thinking we used when we created them.
-Albert Einstein

This chapter outlines the investigated FPGA architecture and highlights its potential advantages and challenges. Employment of this architecture for high performance DSP applications is discussed along with the applicability of contemporary hybrid reconfigurable computing systems for performing these applications.

There are a number of solutions on the market when FPGAs are deployed inside a computer system. This option is of particular interest in this work since such architecture has a computational power of a general-purpose processor (GPP), along with an FPGAs flexibility of reconfigurable hardware, and, therefore, the possibility for performance acceleration of DSP applications through parallelisation.

4.1 FPGA Architecture

The hybrid technology implies simultaneous work of an FPGA chip and a CPU of a commodity PC in one system. It might be particularly advantageous for such DSP applications such as antenna aperture synthesis, radio imaging, RADAR, radio astronomy, high-energy physics etc. A common and very computationally-intensive part in the above-mentioned applications is the multi-channel wideband correlation of signals. Such correlation can be implemented in a parallelised manner in an FPGA. Depending on the type of correlation (XF or FX) (Thompson et al., 2001b) both floating and fixed point numbers can be successfully and efficiently targeted to work on this architecture involving an FPGA as required.

0.5cm In addition, reasonably decreasing prices of FPGA devices and the off-the-shelf availability of hardware architecture, place the FPGA approach as a promising alternative to the large-scale and high-cost correlators such as CABB Correlator (Ferris, 2006) or various software correlators (A. S T. Deller et al., 2007; Kawaguchi et al.,2006; West, 2004). FPGA architecture can also be more convenient as the CPU can be utilised to work with unparallelisable tasks (fetching and streaming datasamples into an FPGA, acquisition of correlated data, etc.) whereas an FPGA can be utilised for actual correlations (multiplication and accumulation operations).

4.1.1 FPGA Technology

Inception of FPGAs dates back to 1960s when Gerald Estrins group at the University of California at Los Angeles did one of the first works on reconfigurable computing (Estrin, 1960, 2002). In 1984 Ross Freeman, co-founder of Xilinx Corporation invented a new type of semiconductor device which is now known as the Field Programmable Gate Array (Xilinx Inc., 1984).

FPGAs are historically connected to complex programmable logic devices (CPLDs). Figure 3.1 demonstrates that they belong to the same group called field programmable logic (FPL):

The structure of an FPGA is an evenly-spaced two-dimensional array tiled with logic blocks Configurable Logic Blocks (CLBs). Each CLB represents a simple memory used as a lookup table and flip-flops for buffering. CLBs communicate with other logic blocks via a programmable interconnection network see Figure 4.2.

The peripheral blocks of an FPGA are I/O blocks (IOB in Figure 4.2) dedicated for communication between internal logic blocks and the I/O pins. Modern FPGAs architecture features on-chip memory blocks as well as dedicated circuitry to perform DSP operations - DSP blocks.

The difference between FPGAs and said CPLDs lies in the granularity of a device, which designates the level of complexity of completing the routing between the blocks. Thus, FPGAs fall in the medium granularity devices group while CPLDs in the large granularity devices group. This distinction comes from the fact that CPLDs comprise simple programmable logic devices

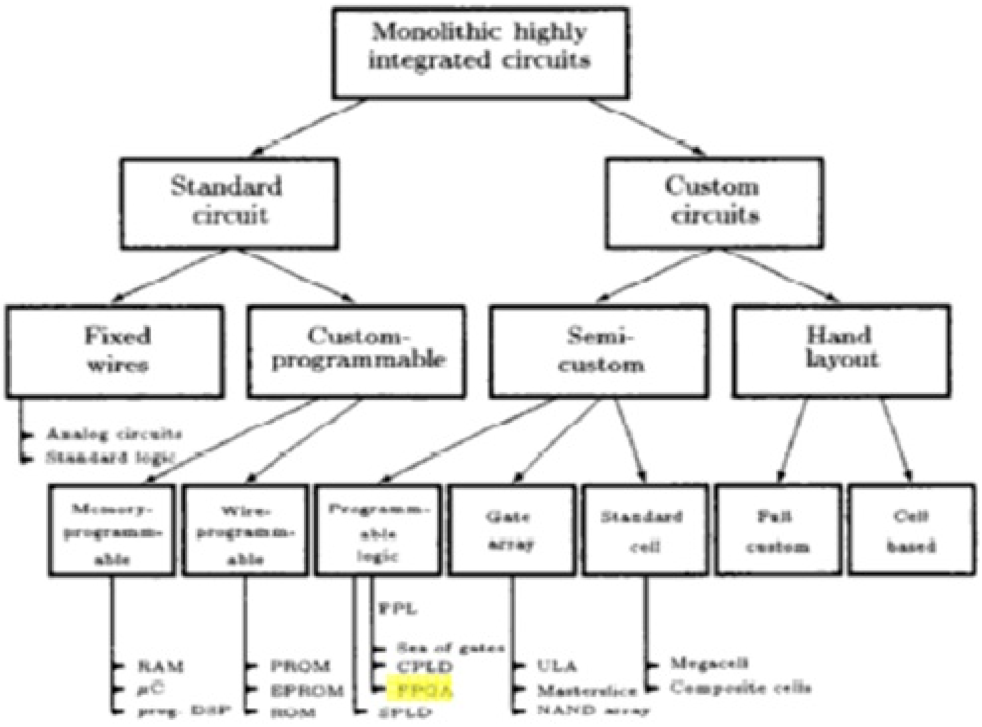


Figure 4.1: Classification of VLSI Circuits

(simple PLDs or SPLDs) with common densities of several thousand to tens of thousands of logic gates, whereas FPGAs normally contain tens of thousands to several millions of logic gates.

In order to define the behaviour of an FPGA it needs to be programmed with a configuration bit stream first. These bit streams are generated from structural register transfer level (RTL) specifications expressed by a user in the form of the HDL descriptions most commonly Verilog or VHDL. These HDL descriptions are createdby hardware designers and follow the design flow demonstrated in Figure 4.3 before the configuration stream is created.

4.1.2 Challenges in FPGA Programming

Nevertheless, the very flexibility that makes FPGAs so universal and beneficial at the same time imposes a considerable challenge on the whole RC design process. The following are the most prominent of the challenges that need mentioning:

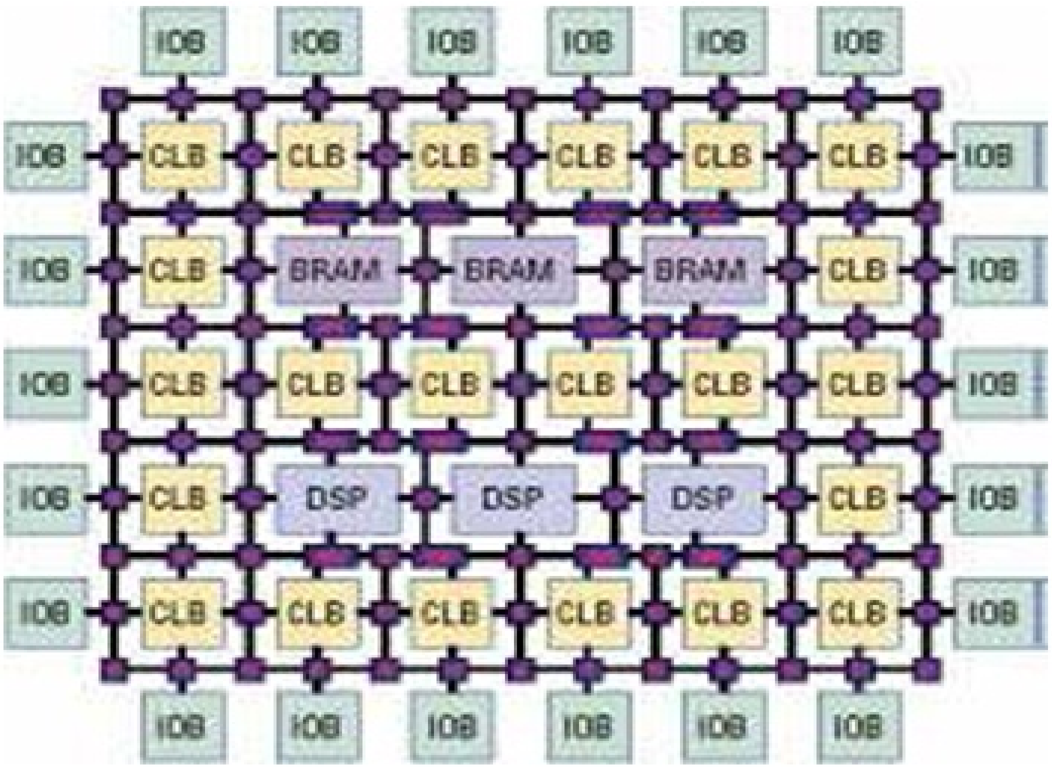


Figure 4.2: FPGA Internal Structure

1. A priori unawareness of FPGA about its I/Os. This issue implies that an FPGA initially knows nothing about how to communicate with external world. Any interface featured on an FPGA board has to be instantiated and configured in low-level specifications. To mitigate this FPGA vendors provide IP cores for most common interfaces. Robust, high-speed and lowlatency I/O interfaces are a crucial component in the DSP paradigm.

2. Compilation process and compilation time. Unlike conventional software programming where compilation normally takes seconds to minutes, hardware compilation is a complex task (see Figure 4.3) and may take hours to complete.

3. Storing variables in explicit memory hierarchy. In HDL each program variable has to be stored in the chosen memory type: external memory, onchip memory, logic blocks configured as memory or registers. Changing the type of the selected memory might cause changes throughout

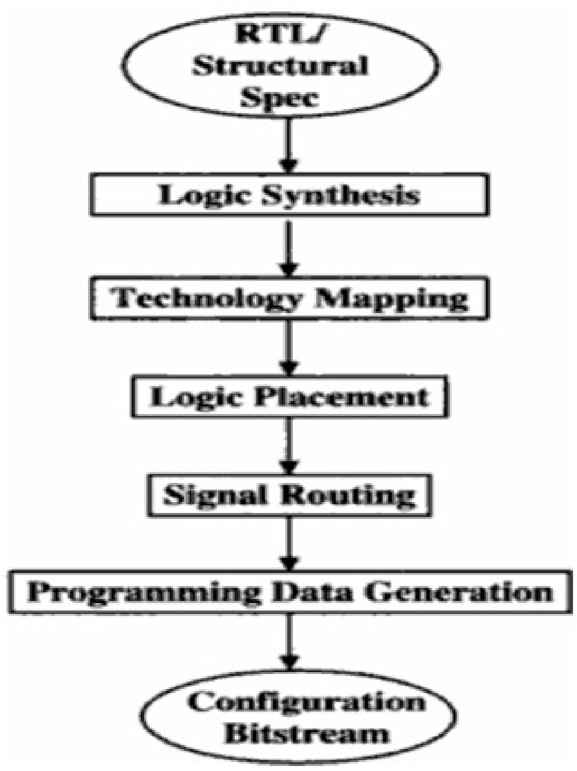


Figure 4.3: Low-level FPGA Design Flow

the whole design .

4. Implicit hardware state in FPGA and complicated debugging. Debugging of the hardware design has to be carried out at the granularity of nanoseconds which is complicated by the lack of transparency of the hardwares state on FPGA.

5. Significant difference in hardware design flow and conventional software design flow. This issue is discussed in more detail in the next section.

4.1.3 High-Level Programming for FPGA Architectures

The overall complexity of FPGA programming has been extensively studied in recent years and a number of solutions have been developed. Hardware and software design flows are considerably different. The general case of software and hardware design flows is depicted in Figure 4.4.

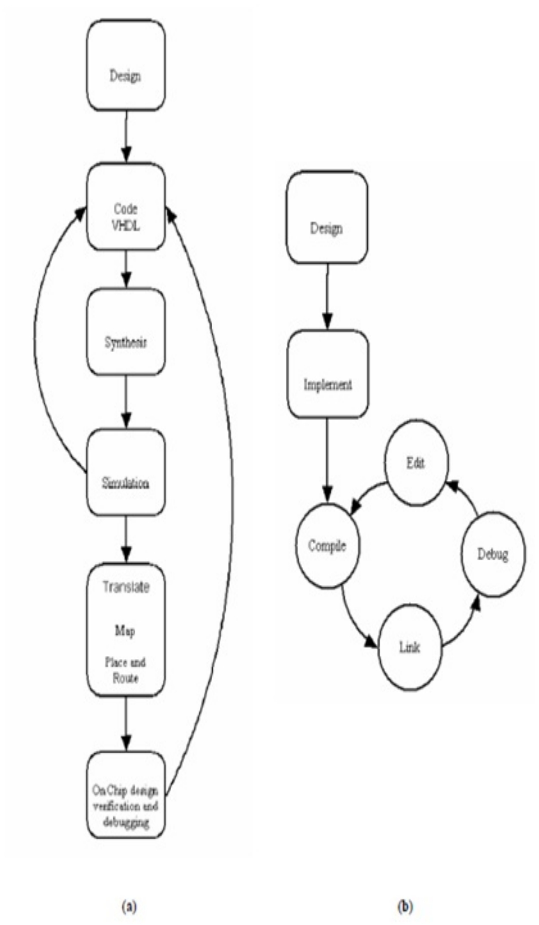


Figure 4.4: Hardware (a) and Software (b) Design Flows

4.2 Project Design Flow and Methodology

In order to evaluate and demonstrate the capabilities of the high-performance hybrid DSP system proposed in 3.2 the following project roadmap was established:

1. Feasibility study of cross-correlation implementation using a traditional hardware development environment for MicroBlaze -based embedded soft processor. Create a simple correlator model in HDL and evaluate development effort.

2. Implement software (in C code) multi-channel cross-correlation of a

model signal with added non-coherent noise. Define problem size (correlator lags and number of channels).

3. Using integrated interface between a hardware and software development kit to create FSL (Fast Simplex Link) IPCore in MicroBlaze soft-core processor

4. Using applicable FPGA development broad (virtex-5 xc5vlx110t fpga or spartan 3e starter board) to develop the high-performance hybrid DSP system discussed in 4.3.

5. In order to supply the input data into hardware design and maintain control and management functions, develop a relevant software control application. Measurement a execution time to performance on multi-channel correlator processor.

4.2.1 Development Hardware Platform

The MicroBlaze is a soft core processor designed for Xilinx FPGAs from Xilinx. As a soft-core processor, MicroBlaze is implemented entirely in the general-purpose memory and logic fabric of Xilinx FPGAs.

In terms of its instruction-set architecture, MicroBlaze is very similar to the RISC-based DLX architecture described in a popular computer architecture book by Patterson and Hennessy. With few exceptions, the MicroBlaze can issue a new instruction every cycle, maintaining single-cycle throughput under most circumstances.

The MicroBlaze has a versatile interconnect system to support a variety of embedded applications. MicroBlaze's primary I/O bus, the CoreConnect PLB bus, is a traditional system-memory mapped transaction bus with master/slave capability. A newer version of the MicroBlaze, supported in both Spartan-6 and Virtex-6 implementations, as well as the 7-Series, supports the AXI specification. The majority of vendor-supplied and third-party IP interface to PLB directly (or through an PLB to OPB bus bridge.) For access to local-memory (FPGA BRAM), MicroBlaze uses a dedicated LMB bus, which reduces loading on the other buses. User-defined coprocessors are supported through a dedicated FIFO-style connection called FSL (Fast Simplex Link). The coprocessor(s) interface can accelerate computationally intensive algorithms by offloading parts or the entirety of the computation to a user-designed hardware module.

Many aspects of the MicroBlaze can be user configured: cache size, pipeline depth (3-stage or 5-stage), embedded peripherals, memory management unit, and bus-interfaces can be customized. The area-optimized version

Table 4.1: Spartan 3E Starter Broad Device Utilization Summary

Device Utilization data-Spartan 3E Starter Broad			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	2,716	9,312	29%
Number of 4 input LUTs	3,183	9,312	34%
Number of occupied Slices	2,617	4,656	56%
Number of Slices containing only related logic	2,617	2,617	100%
Number of Slices containing unrelated logic	0	2,617	0%
Total Number of 4 input LUTs	3,307	9,312	35%
Number used as logic	2,644		
Number used as a route-thru	124		
Number used for Dual Port RAMs	320		
Number used as Shift registers	219		
Number of bonded IOBs	63	232	27%
IOB Flip Flops	31		
IOB Master Pads	1		
IOB Slave Pads	1		
Number of ODDR2s used	22		
Number of RAMB16s	9	20	45%
Number of BUFGMUXs	5	24	20%
Number of DCMs	2	4	50%
Number of BSCANs	1	1	100%
Number of MULT18X18SIOs	3	20	15%
Average Fanout of Non-Clock Nets	3.24		

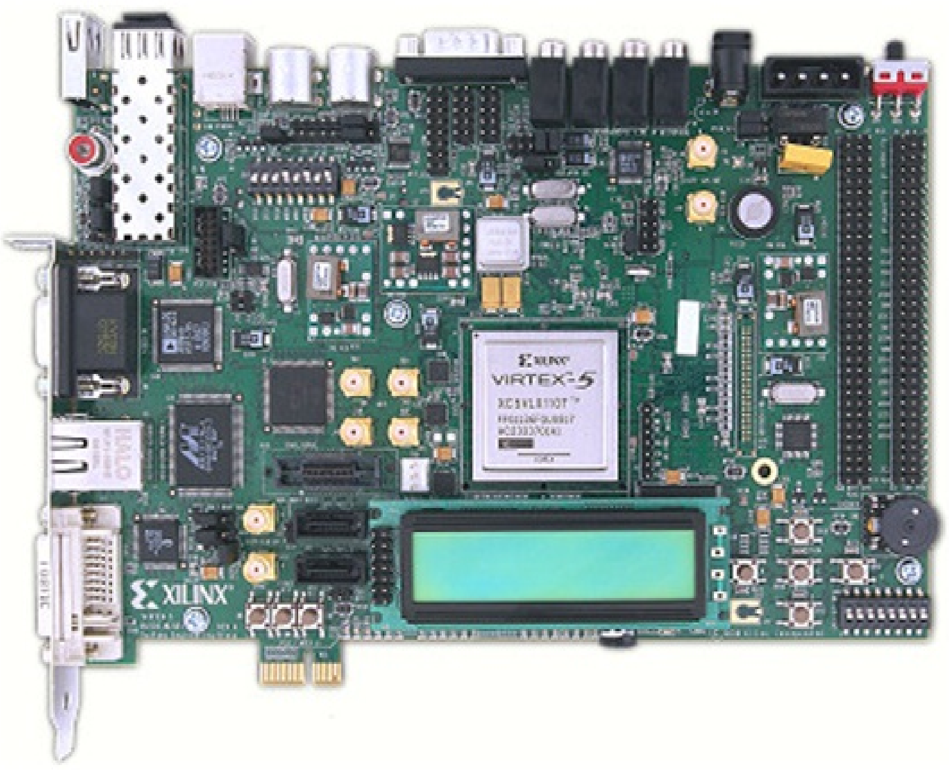


Figure 4.5: Virtex 5 FPGA Development Broad

of MicroBlaze, which uses a 3-stage pipeline, sacrifices clock-frequency for reduced logic-area. The performance-optimized version expands the execution-pipeline to 5-stages, allowing top speeds of 100MHz (on Virtex-5 FPGA family.) Also, key processor instructions which are rarely used but more expensive to implement in hardware can be selectively adder/removed (i.e. multiply, divide, and floating-point ops.) This customization enables a developer to make the appropriate design tradeoffs for a specific set of host hardware and application software requirements.

With the memory management unit, MicroBlaze is capable of hosting operating systems requiring hardware-based paging and protection, such as the Linux kernel. Otherwise it is limited to operating systems with a simplified protection and virtual memory-model: e.g. FreeRTOS or Linux without MMU support. MicroBlaze's overall throughput is substantially less than a comparable hardened CPU-core (such as the PowerPC440 in the Virtex-5.)

Table 4.2: Virtex 5 Broad Device Utilization Summary

Device Utilization data- Virtex 5 Broad			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	5,082	69,120	7%
Number used as Flip Flops	5,062		
Number used as Latch-thrus	20		
Number of Slice LUTs	4,041	69,120	5%
Number used as logic	3,851	69,120	5%
Number using O6 output only	3,653		
Number using O5 output only	73		
Number using O5 and O6	125		
Number used as Memory	178	17,920	1%
Number used as Dual Port RAM	64		
Number using O5 and O6	64		
Number used as Shift Register	114		
Number using O6 output only	113		
Number using O5 output only	1		
Number used as exclusive route-thru	12		
Number of route-thrus	91		
Number using O6 output only	82		
Number using O5 output only	7		
Number using O5 and O6	2		
Number of occupied Slices	2,344	17,280	13%
Number of LUT Flip Flop pairs used	6,610		
Number with an unused Flip Flop	1,528	6,610	23%
Number with an unused LUT	2,569	6,610	38%
Number of fully used LUT-FF pairs	2,513	6,610	38%
Number of unique control sets	460		
Number of slice register sites lost to control set restrictions	1,004	69,120	1%
Number of bonded IOBs	148	640	23%
Number of LOCed IOBs	148	148	100%
IOB Flip Flops	286		

Device Utilization data- Virtex 5 Broad			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of BlockRAM/FIFO	11	148	7%
Number using BlockRAM only	11		
Number of 36k BlockRAM used	11		
Total Memory used (KB)	396	5,328	7%
Number of BUFG/BUFGCTRLs	5	32	15%
Number used as BUFGs	5		
Number of IDELAYCTRLs	3	22	13%
Number of BSCANs	1	4	25%
Number of BUFIOs	8	80	10%
Number of DSP48Es	3	64	4%
Number of PLL_ADVs	1	6	16%
Average Fanout of Non-Clock Nets	3.31		

4.2.2 Development Software Tool

Once the hardware platforms for implementing the stages of the project design flow in 4.2.1 were selected, respective software development kits had to be selected. See fig-4.7.

In this figure 4.7 , Xilinx SDK offer a various design languages (C and C++). C-code for interface between a xps and sdk is below and upate a download bit file and download a fpga broad.

```
print(”– Entering main() –}r}n”);
int i,var;
Xuint32 arr[32];
for(i=0;i¡32;i=i+1)
{
putfsl(i,0);
xil_printf(”-̂-Sent Number–%d{n{r”,i);
}
for(i=0;i¡32;i=i++)
{
getfsl(val,0);
arr[i] = val;
xil_printf(”-̂-Received Number–% d got %8X{n{r”, i, val);
}
for(i=0;i¡32;i=i+1)
xil_printf(” ˚ 0x%x”,arr[i]);
```

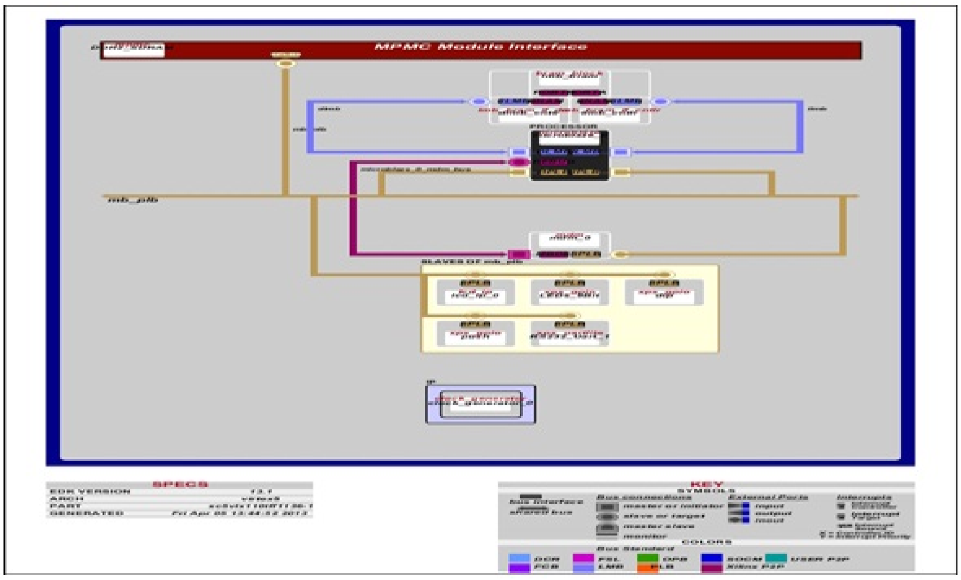
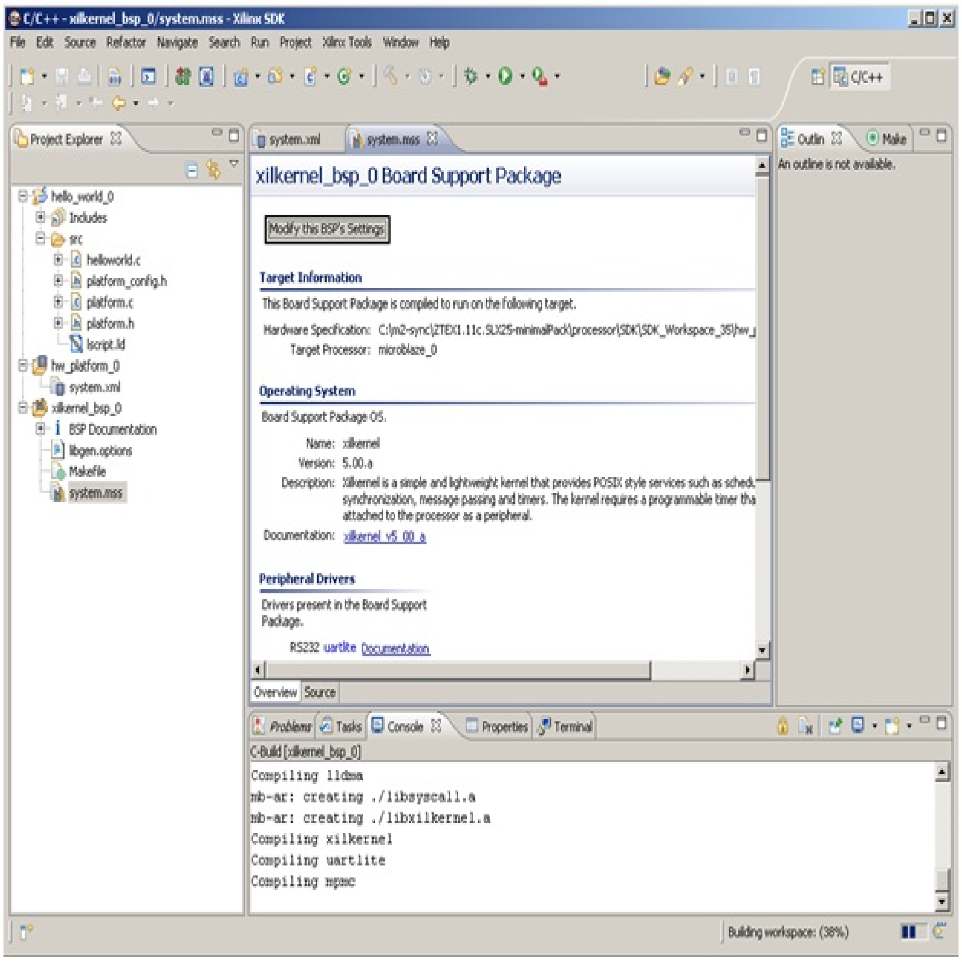


Figure 4.6: MicroBlaze Soft core processor Blockdigram

print("– Exiting main() –◦");

4.3 Result

Now a download a bit file in virtex 5 FPGA development broad through a JTAG cable. Output is observed in hyper terminal in discussed and calculated a execution time on processor to process the 32 bit data.



xilkernel_bsp_0 Board Support Package

Target Information

This Board Support Package is compiled to run on the following target.

Hardware Specification: C:\m2-sync\ZTEX1.11c.SLX25-minimalPack\processor\SDK\SDK_Workspace_35\hw_

Target Processor: microblaze_0

Operating System

Board Support Package OS.

Name:	xilkernel
Version:	5.00.a
Description:	Xilkernel is a simple and lightweight kernel that provides POSIX style services such as sched synchronization, message passing and timers. The kernel requires a programmable timer tha attached to the processor as a peripheral.
Documentation:	xilkernel_v5_00_a

Peripheral Drivers

Drivers present in the Board Support Package.

RS232 uartlite Documentation

C-Build [xilkernel_bsp_0]

```
Compiling lldma
mb-ar: creating ./libsyscall.a
mb-ar: creating ./libxilkernel.a
Compiling xilkernel
Compiling uartlite
Compiling mpmc
```

Figure 4.7: Software Development platform in xilinx EDK tool

4.3.1 HyperTerminal Output

4.3.2 Execution Time calculation

Now that frequency of processor 100 Mhz and sample per clock is 1053248535.

$$T = \frac{1}{f} \tag{4.1}$$

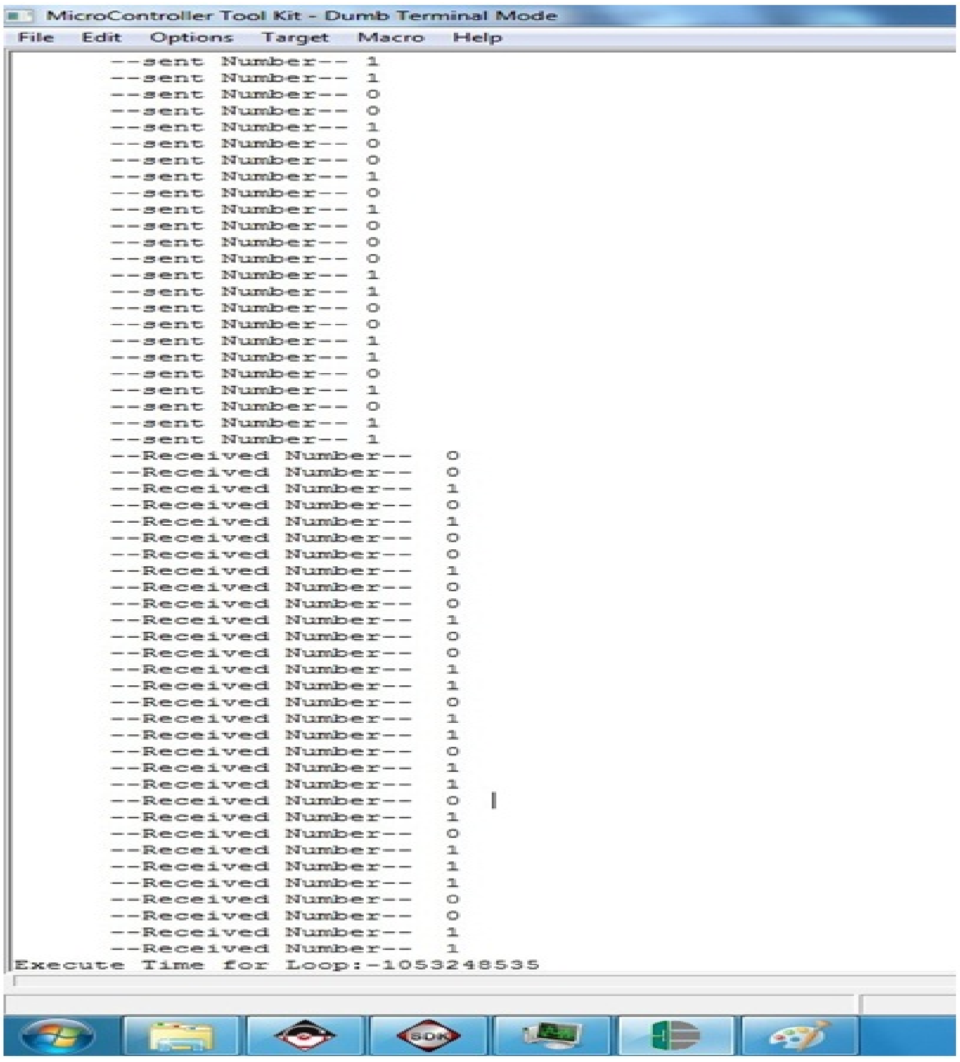
$$T = \frac{1}{100MHz} \tag{4.2}$$

T=10^{-8}
T=10 nsec
Execution time on processor
t= T $\times SampleperClock$
t= 10ns $\times$1053248535
t= 10.532 sec

4.4 Chapter Summary

This chapter introduces the background of the Multi channel correlator architecture. The following typical discussed and implementation in FPGA development broad . Multi channel correlator are observed in simulation level.

The theory of the Multi channel correlator problem with the focus on digital implementation of correlation is described. This is followed by a discussion on the computational requirements of the modern DSP applications.Their features and existing challenges are discussed. Calculation on execution time on processor are descried and discussed.


MicroController Tool Kit - Dumb Terminal Mode
File Edit Options Target Macro Help

```
          --sent Number-- 1
          --sent Number-- 1
          --sent Number-- 0
          --sent Number-- 0
          --sent Number-- 1
          --sent Number-- 0
          --sent Number-- 0
          --sent Number-- 1
          --sent Number-- 0
          --sent Number-- 1
          --sent Number-- 0
          --sent Number-- 0
          --sent Number-- 0
          --sent Number-- 1
          --sent Number-- 1
          --sent Number-- 0
          --sent Number-- 0
          --sent Number-- 1
          --sent Number-- 1
          --sent Number-- 0
          --sent Number-- 1
          --sent Number-- 0
          --sent Number-- 1
          --sent Number-- 1
          --Received Number--   0
          --Received Number--   0
          --Received Number--   1
          --Received Number--   0
          --Received Number--   1
          --Received Number--   0
          --Received Number--   0
          --Received Number--   1
          --Received Number--   0
          --Received Number--   0
          --Received Number--   1
          --Received Number--   0
          --Received Number--   0
          --Received Number--   1
          --Received Number--   1
          --Received Number--   0
          --Received Number--   1
          --Received Number--   1
          --Received Number--   0
          --Received Number--   1
          --Received Number--   1
          --Received Number--   0
          --Received Number--   1
          --Received Number--   0
          --Received Number--   1
          --Received Number--   1
          --Received Number--   1
          --Received Number--   0
          --Received Number--   0
          --Received Number--   1
          --Received Number--   1
Execute Time for Loop:-1053248535
```

Figure 4.8: HyperTerminal Output

Chapter 5

CONCLUSION AND FUTURE SCOPE

For everything you have missed, you have gained something else, and for everything you gain, you lose something else.
-Ralph Waldo Emerson

The structure of this chapter is as follows: first, discussion of the obtained results is given, which is followed by suggestions of alleviating known shortcomings and future developments. The chapter concludes with an overall summary.

5.1 Conclusion

This work delivers two main outcomes:

1. Linear Correlator architecture design using Xilinx and Matlab.

2. Multi-channel cross-correlator design working in a FPGA architecture and developed with new top-down design methodology.

All research objectives (see 1.2) are achieved, satisfied and are covered by outcomes of this project. The following sections group the discussions for these respective outcomes.

Linear Correlator architecture is design in Xilinx and Matlab (discussions in chapter 3) . Auto Correlator and Cross- correlator are implementation in Matlab in sampling frequency of 8000 Hz and simulation are observed. Also implement a 32-leg linear correlation in VHDL and observed a RTL view and simulation on a ISE simulator.

Multi-channel correlation design using high-level FPGA programming in Stage 2 are significantly less than using traditional FPGA development tools in EDK. Moreover, the correlator design of Stage 1 required more development time but has less functionally than the correlator developed in Stage 2, which is more flexible and scalable. The process of developing the correlator in Impulse C in SDK and required little to no HDL design techniques: clock and parallel execution had to be considered. However, no low-level debugging tools (signal analyzers) were used whatsoever.

The fact that Impulse C can actually generate accelerated HDL from C input demonstrates the potential of this tool. Admittedly, certain manipulations, uncommon for conventional programming, had to be performed with the C code to produce efficiently parallelised hardware design. For the targeted correlator design, these manipulations included array splitting, pipelining, and introducing stage delays. In fact, to generate HDL designs with reasonable speed-ups from high-level FPGA programming tools, a knowledge and practical understanding of hardware operation are still required. While there is yet no green button solution to generate final and complete hardware designs from entirely software algorithm implementations, consideration of hardware implementation execution and understanding of parallel and clock concepts are required.

The design of Parallel Architecture for Correlator is implemented in VHDL. Soft-core Embedded Processor development tool that mean EDK tool is study. The use of built in IP core reduces and resource requirement of the system and improves the overall performance in terms of speed and area. The design is carried out using Virtex 5 XC5VLS110t-1-1ff1136 with speed grade of -1. The result are obtained and observed at the simulation level. Calculated a execution time on processor.

5.2 Future Scope

For the correlator design, the primary goal was to achieve a speed-up in the computational core of the correlation algorithm on hardware platform. Therefore, in case of the further development of the correlator design a speed-up might be searched outside of the correlation computation loops more careful clock cycle considerations of input and output streaming interfaces, eliminating or merging variables manipulation stages, etc. Besides, as it was mentioned the implemented code of cross-correlation was not developed with the intention of performing cross-correlation with maximum efficiency. Therefore, the cross-correlation algorithm itself can be improved.

Additionally, in case the correlator design is applied for correlation calculation in practical applications (eg for radio-astronomical applications), an FX type of correlation might be considered. Due to the fact that in this correlation Fourier transform is applied before calculating cross-products, it offers some advantage in the required number of cross-products calculations, as opposed to XF correlators where Fourier transform is applied after cross-products are calculated.

The virtex 5 implementation of the correlator can be enhanced by employing shared memory interfaces between software and hardware processes. This will allow to overcome the shortcoming of the slow streaming communication of the FLS.However, implement the shared memory approach asynchronisation mechanism is required between the communicating processes. Currently, such a synchronisation utility is not feasible to implement with the supported tools. Support of such synchronisation tools will theoretically increase communication throughput between processes and, therefore, the overall virtex 5 performance.

The proposed approach of simultaneous involvement of the FPGA for correlation in this thesis can be expanded to other DSP applications, such as image processing, telecommunication, cryptography, provided that the data input-output interfaces are fast, well-tested and reliable. As stated before, FPGA has no knowledge about any I/O interfaces before it is configured, whereas for any DSP application input output throughput is one of the critical questions in achieving top performance. Thus, although FPGA-based computing can deliver a significant increase in performance for a number of applications, the current state of FPGA development requires a substantial amount of expertise and design efforts to achieve the respective speed-up.

Appendix A

Matlab Program for Auto-Correlation

```
N=1024;
f1=1;
fs=200;
n=0:N-1;
x=sin(2*pi*f1*n/fs);
t=[1:N]*(1/fs);
subplot(2,1,1);
plot(t,x);
title('sinwave of frequency 1000Hz [fs=8000Hz]');
xlabel('Time,[s]');
ylabel('Amplitude');
grid;
Rxx=xcorr(x);
subplot(2,1,2);
plot(Rxx);
grid;
title('autocorrelation function of the sinewave');
xlabel('lags');
ylabel('autocorrelation');
```

Appendix B

Matlab Program for Cross-Correlation

```
N=1024;
f=1;
fs=200;
n=0:N-1;
x=sin(2*pi*f*n/fs);
y=x+10*randn(1,N);
subplot(3,1,1);
plot(x);
title('pure sinewave');
grid;
subplot(3,1,2);
plot(y);
title('y(n),pure sinewave +noise');
grid;
Rxy = xcorr(x,y);
subplot(3,1,3);
plot(Rxy);
title('Cross correlation Rxy');
grid;
```

References

[1] Agarwal, R., B.V.R.Reddy, K.K.Aggarwal. (2006). A Switching Mechanism Detection to Reduce Complexity in Multiuser Detection for DS-CDMA Systems. Journal of Mathematics and Statistics, 2(2), 368-372.

[2] Altium Limited. (2008). Altium Designer. Retrieved August 13, 2008, from http://www.altium.com/products/altiumdesigner/

[3] Andrews, D., Niehaus, D., Jidin, R., Finley, M., Peck, W., Frisbie, M., et al. (2004).Programming models for hybrid FPGA-CPU computational components: a missing link. IEEE Micro, 24(4), 42-53.

[4] Baran, P., Bodenner, R., Hanson, J. (2004). Reduce Build Costs by Offloading DSP Functions to an FPGA. FPGA and Structured ASIC.

[5] Beckman, P. (2008). Looking toward Exascale Computing. On The Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'08) [Keynote Speaker]. Dunedin: University of Otago.

[6] Betz, V., Rose, J., Marquardt, A. (1999). Architecture and CAD for Deep-Submicron FPGAs. Norwell: Kluwer Academic Publishers.

[7] Buell, D., El-Ghazawi, T., Gaj, K., Kindratenko, V. (2007). Guest Editors' Introduction: High-Performance Reconfigurable Computing. Computer, 40(3), 23-27.

[8] Callahan, T. J., Hauser, J. R., Wawrzynek, J. (2000). The Garp architecture and C compiler. Computer, 33(4), 62-69.

[9] Carroll, B. W., Ostlie, D. A. (2007). 6.3 Radio Telescopes. In An introduction to modern astrophysics (2nd ed.). San Francisco: Pearson Addison-Wesley.

[10] Cavadini, M., Wosnitza, M., Troster, G. Multiprocessor system for high-resolution image correlation in real time. IEEE Trans. Very Large Scale Integr. VLSI Syst. 9(3), 439449 (2001)

[11] M. Wahab and D. Puckey, FPGA-based DSP Systems, Eds. W.R.Moore and W. Luk, Abindon EECS books, 1994.

[12] Altera Corporation, Stratix II Device Handbook, May 2007.

[13] Almudena Lindoso, Luis Entrena. High performance FPGA-based image correlation, J. Real-Time Image Proc. 2007, vol. 2, pp.223233

[14] Altera Corporation, Quartus II Version 8.0 Handbook, 2008.

[15] Analog Devices Inc., TigerSHARC Embedded Processor ADSPTS201S,2006.

[16] Xilinx Inc, Virtex 5 Handbook,May 2009.

[17] Xilinx Inc, EDK 13.1 Tool Handbook. May 2012

[18] C. Jakob, A. Th. Schwarzbacher, B. Hoppe, R. Peters, A FPGA Optimised Digital Real-Time Mutichannel Correlator Architecture, IEEE Computer Society 10th Euromicro Conference, 2007