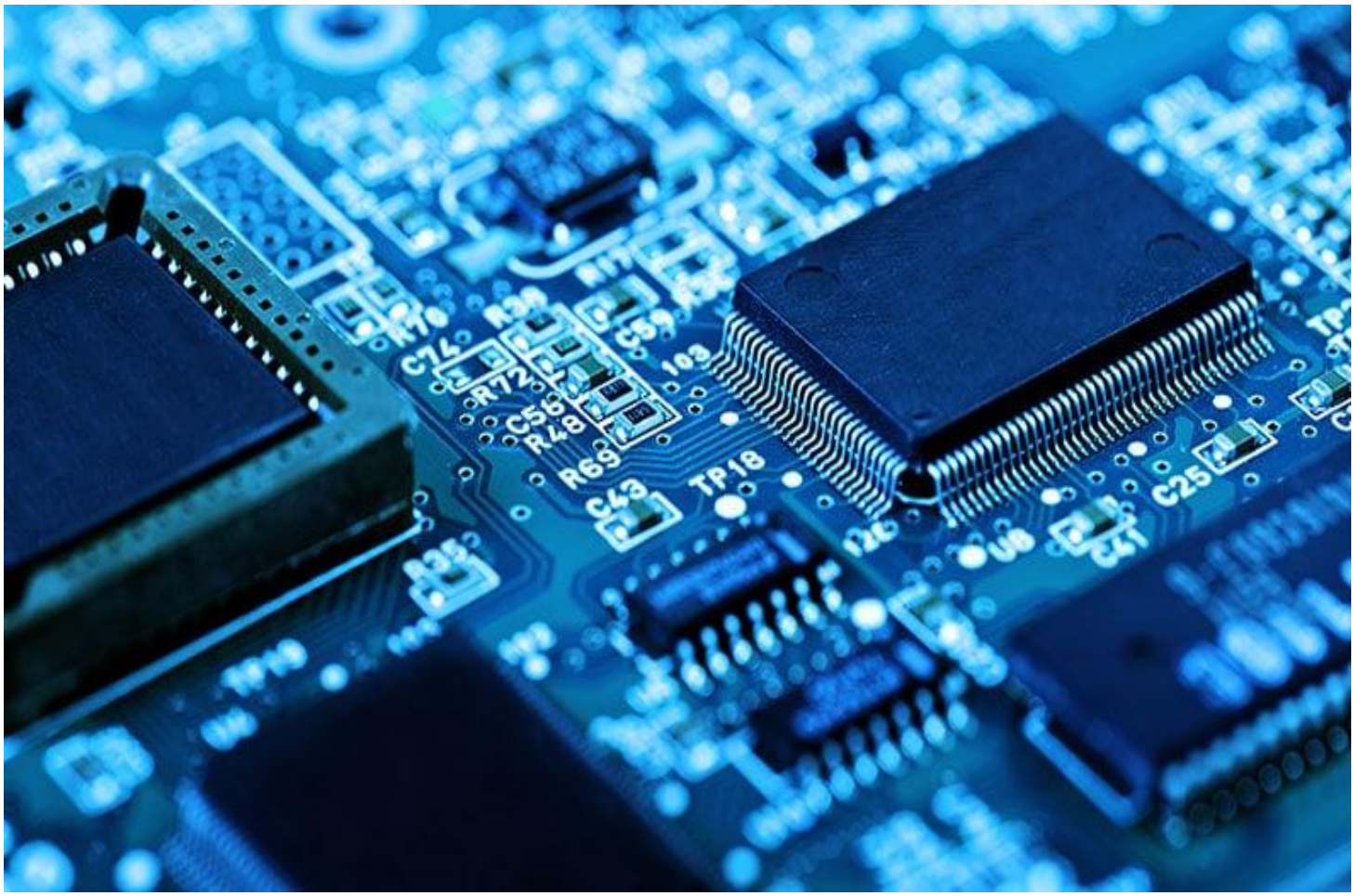
Technical University of Crete

School of Electrical and Computer Engineering

Diploma Thesis

‘Analysis of Compressive Existing Sensing Schemes and Systems and Design of FPGA-based Compressive Sensing Nodes’



Author: **Petsas G. Dimitrios**

Supervisor: **Professor Yannis Papaefstathiou**

Committee: **Professor Dollas Apostolos**

Professor Pnevmatikatos Dionysios

ABSTRACT

Nowadays, our world is related with the Information Technology Systems, the Internet and general the Communications. All these are coupled with recent engineering advances and are paving the way for a new generation of inexpensive sensors and actuators, capable of achieving a high order of spatial and temporal resolution and accuracy.

The Wireless Sensor Networks (WSNs) are very important networks and help in the implementation of many crucial Information Systems related to the Data Management. The technology for sensing and control includes sensor arrays, electric and magnetic field sensors, seismic sensors, radio-wave frequency sensors, electro optic and infrared sensors, laser radars, and location and navigation sensors.

This undergraduate thesis will try to analyze the Compressive existing sensing schemes in the Wireless Sensor Networks (WSNs), and design the network using FPGA-based compressive sensing nodes. It is targeted to communications developers, managers, and practitioners who seek to understand the benefits of this new technology and plan for its use and deployment.

ACKNOWLEDGEMENTS

I would first like to thank my diploma supervisor Professor Yannis Papaefstathiou of the School of Electrical and Computer Engineering at Technical University of Crete, for his careful guidance and patience. I would like to thank him for his trust, his precious advices and for dedicating a lot of time to me while discussing my work. This work could not have been written without him.

I would also like to acknowledge Professor Dollas Apostolos and Professor Pnevmatikatos Dionysios as members of the committee, and I am gratefully indebted to them for their approval and very valuable comments on this thesis.

I have to say thanks to my friends who supported me during my university studies. Showing respect and understanding of the particularities of each one of us, we have passed beautiful moments that I will remember in the rest of my life with nostalgia and tenderness. They supported me in easy and difficult circumstances of my life with discretion and courtesy and they helped me develop my personality. They encouraged me and gave me the strength to continue and succeed in my goals.

Finally, I must express my very profound gratitude to my parents Maria and Giannoulis for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching for and writing this thesis. Despite of the difficulties that have arisen during my studies and the preparation of this Diploma Thesis, they supported me with understanding and overwhelming love and admiration, aiming at my evolution and the formation of a right man and an ardent and good scientist. This accomplishment would not have been possible without them. I dedicate to them this diploma thesis and by extension, with the completion of my studies, my diploma. Thank you.

Dimitrios Petsas

Contents

1. Introduction

The Wireless Sensor Networks (WSNs) are very important networks that are utilized in the implementation of many crucial Information Systems related to the Data Management. The technology for sensing and control includes sensor arrays, electric and magnetic field sensors, seismic sensors, radio-wave frequency sensors, electro optic and infrared sensors, laser radars, and location and navigation sensors.

Firstly, we provide an introduction to the fundamental aspects of Wireless Sensor Networks (WSNs), related to the sensor network technology, the applications, the communication techniques, the networking protocols, the security and the system management.

Secondly, we will try to analyze the Compressive existing sensing schemes in the Wireless Sensor Networks (WSNs).

Thirdly, we design the network using FPGA-based compressive sensing nodes. We analyze the sensor network architecture and the benefits of this new technology and plan for its use and deployment.

This thesis is very important for the communications developers, managers, and practitioners and generally the focus is the Wireless Sensor Networks (WSNs) development.

2. Literature Review

The WSNs have many applications in the business area with useful services and operations.
We will describe in this part the literature review for this undergraduate thesis . The scope is the reference and analysis of the WSN networks using compressive sensing mechanisms and the communication with the FPGA hardware for the transmitting and management of the data in the Information Systems.

Procedure #01

We provide an introduction of the fundamental aspects of Wireless Sensor Networks (WSN), related to the sensor network technology, the applications, the communication techniques, the networking protocols, the security and the system management.

Procedure #02

We analyze the Compressive Sensing procedures, which are very important for the energy and in general for the performance of the sensor network.

Procedure #03

We analyze and design the network using FPGA-based compressive sensing nodes. We analyze the sensor network architecture and the benefits of this new technology and plan for its use and deployment.

Regarding Sensing, we will see the Data Compression over the Sensor Network, the background of compressive sensing algorithm, the simulation of compressive sensing algorithm and the power consumption.

Theoretical Part

3. Wireless Sensor Network (WSN) and FPGA Technology

3.1. What is Wireless Sensor Networks (WSN)?

The Wireless Sensor Networks (WSN) is an infrastructure composed of sensing, computing, and communication elements. The administrator of the Wireless Sensor Networks (WSN) is typically a civil, governmental, commercial, or industrial entity. This network gives the administrator the ability to instrument, observe, and react to events and phenomena in a specified environment.

Generally, the environment of the Wireless Sensor Networks (WSN) will focus on different actions; in global Information Systems related to many activities in the Business Domain, the biological system or an information technology (IT) framework.

An important action for these networks is that observers find them as an important technology that will experience major deployment in the next few years for an assortment of applications. Typical applications include, but are not limited to, data collection, monitoring, surveillance, and medical telemetry. In addition to sensing, one is often also interested in control and activation.

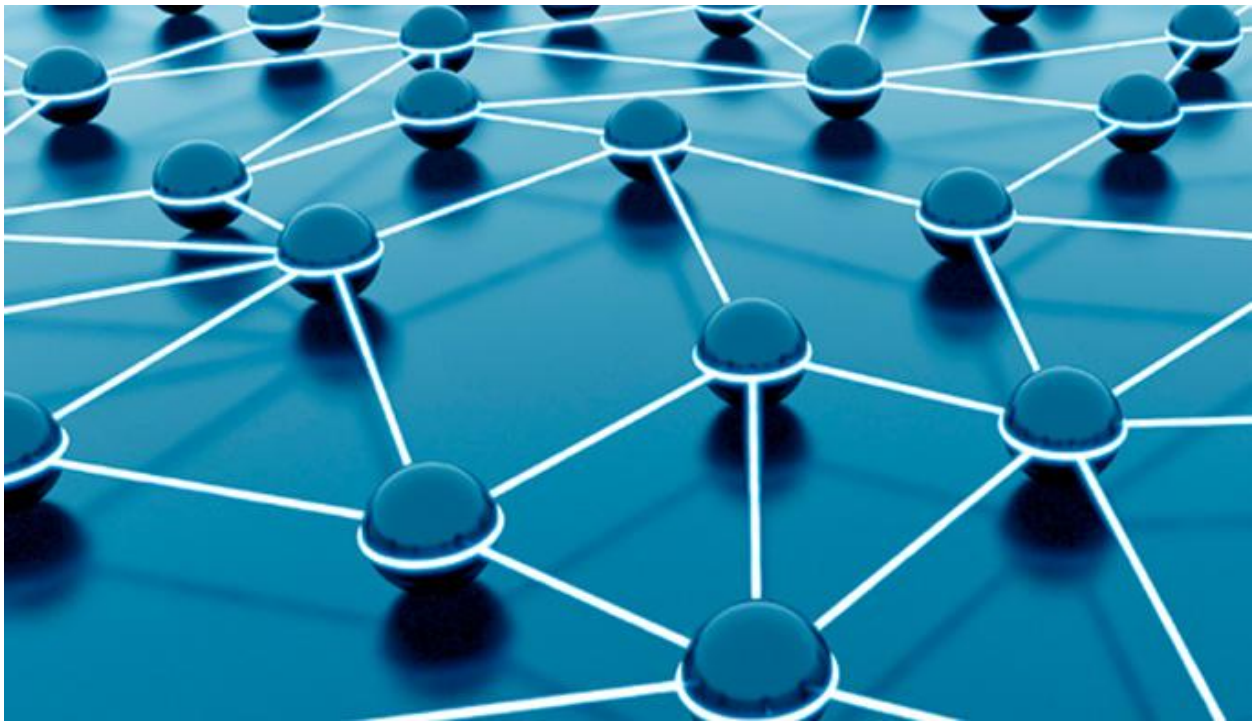
The basic architecture of the Wireless Sensor Networks (WSN) is comprised of four basic components:

- an assembly of distributed or localized sensors
- an interconnecting network
- a central point of information clustering and
- a set of computing resources at the central point to handle data correlation, event trending, status querying, and data mining.

Concerning the Operational Components of the Sensor Network, some crucial factors are the following:

- sensing and computation nodes are considered part of the sensor network
- potentially large quantity of data collected, in this case the algorithmic methods for data management, play an important role in sensor networks
- computation and communication infrastructure associated with sensor networks is often specific to this environment and rooted in the device and application based in nature of these networks
- the energy of node is very important for the network operation, the functionality and the reliability of the Information System

[01], [02]



3.2. Applications of Wireless Sensor Networks

The basic architecture of WSNs as we saw more analytically in the Chapter 3.1 is the component "Node". Generally WSNs are collections of compact-size and relatively inexpensive computational nodes. These Nodes use procedures with respect to environmental conditions or other parameters, retrieve crucial information and forward this to a central point for appropriate processing. Also, WSNs nodes can sense the environment, communicate with neighboring nodes, and can, in many cases, perform basic computations on the data which is being collected. WSNs can support a wide range of useful applications relating to Information Systems.

The Wireless Sensor Networks (WSN) will be classified into two basic categories:

Category 1 WSNs (C1 WSNs):

In this category, the invariably mesh-based systems with multi-hop radio connectivity among WSNs, utilize dynamic routing in both the wireless and wire line portions of the network. Military systems typically belong to this category.

Category 2 WSNs (C2 WSNs):

In this category, point-to-point or multipoint-to-point systems generally with single-hop radio connectivity to WSNs, utilize the static routing over the wireless network. In this case, there is only one route from the WSNs to the companion terrestrial/wire line forwarding node. In this procedure the WSNs are pendent nodes. Residential control systems belong to this category.

Category 1 WSNs **(Image 1)**, are Wireless Sensor Networks in which the End devices (sensors) are permitted to be more than one radio hop away from a routing or forwarding node. The forwarding node is a wireless router that supports dynamic routing (i.e. it has a mechanism that is used to find the best route to the destination out of a possible set of more than one routes); wireless routers are often connected over wireless links.

This Sensor Network Category has many important features such as:

- sensor nodes can support communications on behalf of other sensor nodes. It depends on their acting as repeaters
- the forwarding node supports dynamic routing and more than one physical link to the rest of the network is physically and logically present
- the radio links are measured in thousands of meters
- the forwarding node can support data processing or the reduction on behalf of the sensor nodes.

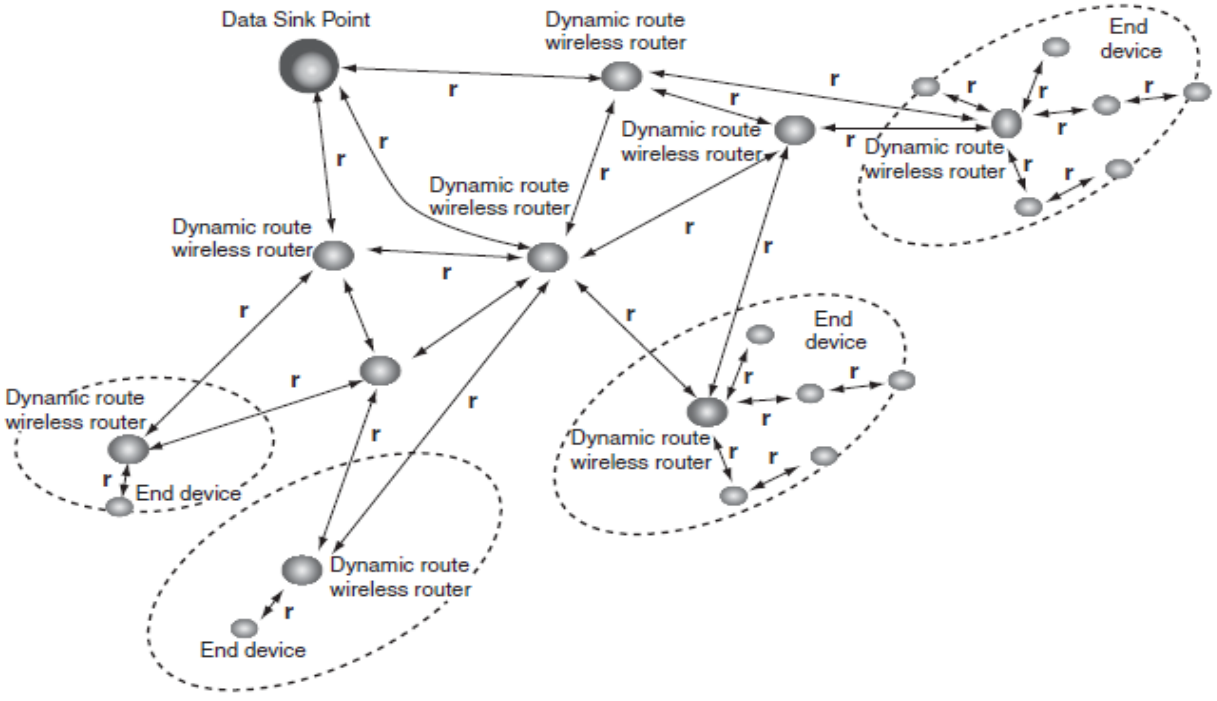


Image 1. Category 1 WSNs: multipoint-to-point, multihop systems utilizing dynamic routing.

Some examples of WSN applications in this category are:

1. Military sensor networks to detect and gain as much information as possible about enemy movements. Also interesting actions are explosions and other phenomena related to the Military.
2. Law enforcement and national security applications for inimical agent tracking or nefarious substance monitoring.
3. Sensor networks to detect and characterize chemical, biological, radiological, nuclear, and explosive (CBRNE) attacks and material.
4. Sensor networks to detect and monitor the environmental changes related the plains, forests and oceans.
5. Wireless traffic sensor networks to monitor vehicle traffic on highways or in congested parts of a city.
6. Wireless surveillance sensor networks for providing security in shopping malls, parking garages, and other facilities.
7. Borders monitoring with sensors and satellite uplinks.

Category 2 WSNs **(Image 2)** are Wireless Sensor Networks (WSN) in which the End devices (sensors) are one radio hop away from a terrestrially homed forwarding node. The operation of the connection is that the forwarding node is connected to the terrestrial network via either a landline or a point-to-point wireless link.

This Sensor Network Category has many important features such as:

- sensor nodes do not support communications on behalf of any other sensor nodes
- the forwarding node supports only static routing to the terrestrial network
- the radio link is measured in hundreds of meters
- the forwarding node does not support data processing or the reduction on behalf of the sensor nodes.

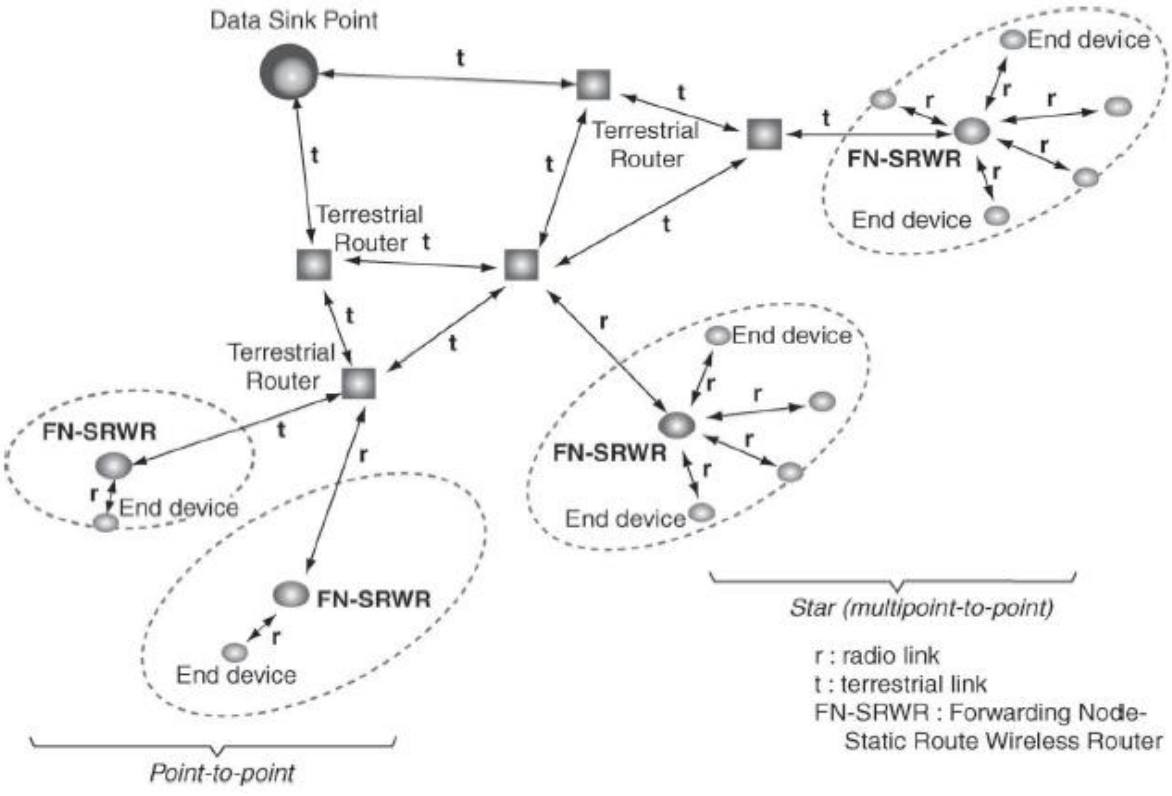


Image 2. Category 2 WSNs: point-to-point, generally single-hop systems utilizing static routing.

Some examples of WSN applications in this category are:

Home Control

Sensing applications related to:

1. facilitate flexible management of lighting, heating, and the cooling systems in the home.
2. the automation of the control of multiple home systems to improve conservation, convenience, and safety.
3. capture the highly detailed electric, water, and gas utility usage data.
4. embed intelligence to optimize consumption of natural resources.
5. enable the installation, upgrading and networking of a control system without wires in the home.
6. enable, configure and run multiple systems using remote control.

Building Automation

Sensing applications related to:

1. integrate and centralize management of lighting, heating, cooling, and security.
2. automate control of multiple systems to improve conservation, flexibility, and security.
3. enable the rapid reconfiguring of lighting systems to create adaptable workspaces.
4. enable to network and integrate data from multiple access control points.
5. enable to deploy wireless monitoring networks for enhancing the perimeter protection.

Industrial Automation

Sensing applications related to:

1. extend existing manufacturing and process control systems
2. improve asset management by continuous monitoring of critical equipment.
3. reduce energy costs through optimized manufacturing processes.
4. help identify inefficient operation or poorly performing equipment.
5. help automate data acquisition from remote sensors to reduce user intervention.

6. provide detailed data to improve preventive maintenance programs.
7. help deploy networks to enhance employees to monitor the safety.
8. Warehouses, fleet management, factories and supermarkets.
9. Gas, water, and electric meters.
10. Smoke, CO, and H2O detectors.
11. Equipment management services and preventive maintenance.
12. Security services.
13. Lighting control.
14. Materials processing systems (heat, gas flow, cooling, chemical).
15. Gateway or field service links to sensors. This action is very important for monitoring and supporting of the preventive maintenance, the status of changes, the energy usability and the diagnostics.

Medical Applications

This area is very important and a crucial factor for the WSNs implementation. Number of hospitals and medical centers are exploring applications of WSN technology to a big range of services. Examples are the medical applications, including pre-hospital and in-hospital emergency care, disaster response, and stroke patient rehabilitation.

WSNs have the potential to affect many procedures regarding the medical data in many relevant services. The information is collected and integrated automatically into the patient care record. The information after the Data Collection procedures is used for real-time triage, correlation with hospital records, and long-term observation.

WSNs also permit collection of long-term medical information that populates databases of clinical data. This procedure enables the longitudinal studies across populations and allows physicians to study the effects of medical intervention programs. The WSNs permit home monitoring for chronic and elderly patients, facilitating long-term care and trend analysis.
[03], [04], [05], [06], [07], [08]

3.3. What is FPGA?

Field-programmable gate array (FPGA), generally such as term are reprogrammable silicon chips. The basic operation and functionality is that using prebuilt logic blocks and programmable routing resources, we can configure these chips to implement custom hardware functionality. We can develop digital computing tasks in software and compile them down to a configuration file. Also an alternative is to use bit stream that contains information on how the components should be wired together.

Image 3. Altera FPGA

In addition, the important advantage of FPGAs is that it is completely reconfigurable and instantly we have a new instance when we recompile a different configuration of circuitry. In the past years these technologies and generally these architectures were very difficult to be programmable and could be used only by engineers with a deep understanding of digital hardware design.

Nowadays, the new programming platforms include development IDEs and generally tools and technologies from the Software Vendors the operations have changed. The usability of high-level tools is changing the rules of FPGA programming. The new technologies convert graphical block diagrams or even C code into digital hardware circuitry.

FPGAs have many crucial advantages such as:

- FPGAs provide the hardware-timed speed and reliability
- FPGAs do not require high volumes to justify the large upfront expense of custom ASIC design
- FPGAs Reprogrammable silicon also has the same flexibility of software running on a processor-based system
- FPGAs not limited by the number of processing cores available
- FPGAs, unlike processors, are truly parallel in nature
- FPGAs have different processing operations do not have to compete for the same resources
- FPGAs' each independent processing task is assigned to a dedicated section of the chip, and can function autonomously without any influence from other logic blocks
- FPGAs' performance of one part of the application is not affected when you add more processing.

[09], [10]

3.4. FPGA Architecture

FPGAs, as illustrated in **Image 4**, consist of an array of programmable logic blocks of potentially different types. The basic building blocks are the general logic, memory and multiplier blocks which are surrounded by a programmable routing fabric that allows blocks to be programmable interconnected.

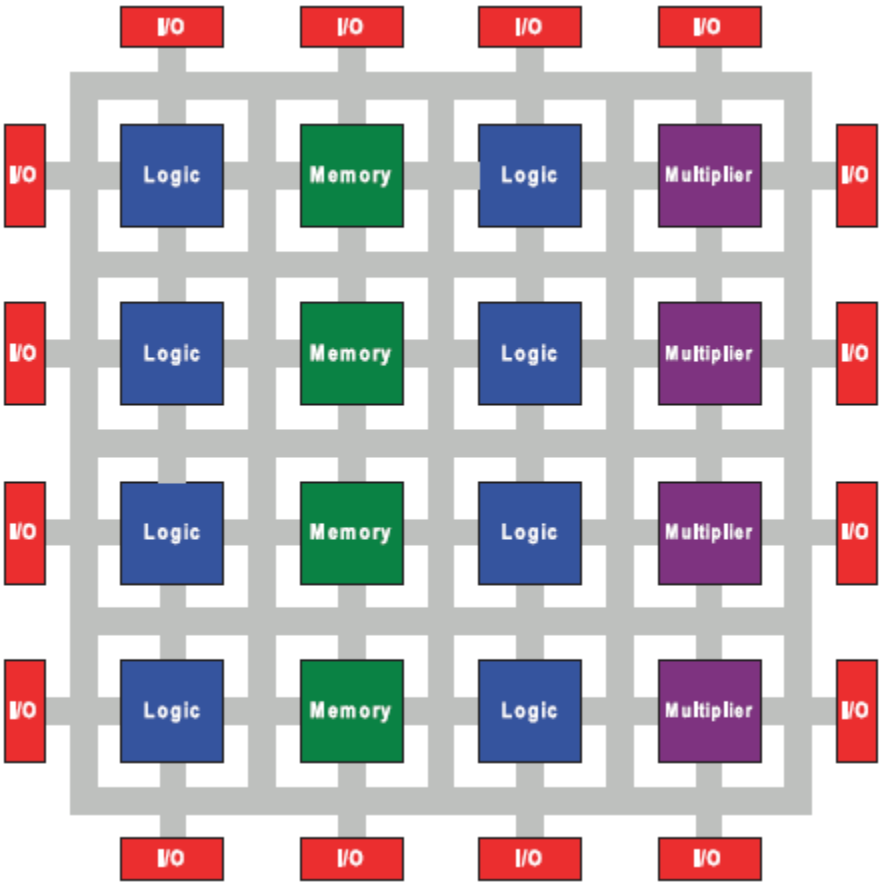


Image 4. FPGA Architecture

In the FPGAs architecture the basic connection was established using the programmable input/output blocks, labeled I/O in the **Image 4**. In this specific array we can see the architecture for the Logic, Memory and the Multipliers. The first programmable logic devices used very small fuses as the programming technology.

Historically, the originality of the contemporary Field-Programmable Gate Array was the integrated circuit in the early 1960s. The Cellular arrays consisted of a two-dimensional array which is consisted of logic cells with fixed point-to-point communication.

The next timing step in the history was by the mid-1960s, when field-programmability was achieved. In other words, the changing of the logic function of a chip after the fabrication process was implemented using “cutpoint” cellular arrays.

In the 1970s, we have many changes to the technology using the read-only memory (ROM)-based programmable devices. These devices were introduced and provided new practices related to the implementation of the logic functions.

Crucial points in this procedure were the Mask-Programmable ROMs and the Fuse Programmable ROMs (PROMs) using N Address inputs. The implementation can be with any N-input logic function. The first programmable Logic Arrays (PLAs) improved on this with two-level AND–OR Logic planes.

At the end, these architectures evolved further in the basic architecture. In 1977 the programmable array logic (PAL) devices were introduced by Monolithic Memories Incorporated (MMI), and the flexibility was provided by a programmable AND plane followed by a fixed OR plane.
[11]

3.5. FPGA Benefits

FPGA technology is very important. FPGA has many benefits from the usability related to the networks and the Information Systems.

Performance

FPGAs have many internal procedures related to the advantage of hardware parallelism. FPGAs exceed the computing power of digital signal processors (DSPs) by breaking the paradigm of sequential execution and accomplishing more per clock cycle. Controlling inputs and outputs (I/O) at the hardware level provides faster response times and specialized functionality to closely match application requirements.

Time to market

FPGA technology offers flexibility and rapid prototyping capabilities in the face of increased time-to-market concerns. A basic characteristic is that the Commercial off-the-shelf (COTS) hardware is available with different types of I/O already connected to a user-programmable FPGA chip.

Cost

The cost is a basic measure element for the technology. The nonrecurring engineering (NRE) expense of custom ASIC has the action to design, which far exceeds that of FPGA based hardware solutions. Also, the large initial investment in ASICs is easy to justify for OEMs shipping thousands of chips per year.

The users are a crucial factor. Many end users need custom hardware functionality systems in development. On the other hand, for the Information Systems (IS) the system requirements often change over time. The cost of making incremental changes to FPGA designs is negligible when compared to the large expense of the alternative ASICs.

Reliability

Reliability is an important issue for the Information Systems (IS). The FPGA circuitry is truly a real implementation of program execution. The programming environment is very interesting with many operations and functionality. Basic features are:

- The Processor-based systems often involve several layers of abstraction to help schedule tasks and share resources among multiple processes.
- The driver layer controls hardware resources and the OS manages memory and processor bandwidth.
- For any given processor core, only one instruction can be executed at a time, and processor-based systems are continually at risk of time-critical tasks preempting one another.
- FPGAs, which do not use OSs, minimize reliability concerns with true parallel execution and deterministic hardware dedicated to every task.

Long-Term Maintenance

The FPGA maintenance is very important for the Life-Cycle of the Information Systems. FPGA chips are field-upgradable and do not require the time and expense involved with ASIC redesign. FPGA are reconfigurable, chips can keep up with future modifications that might be necessary.

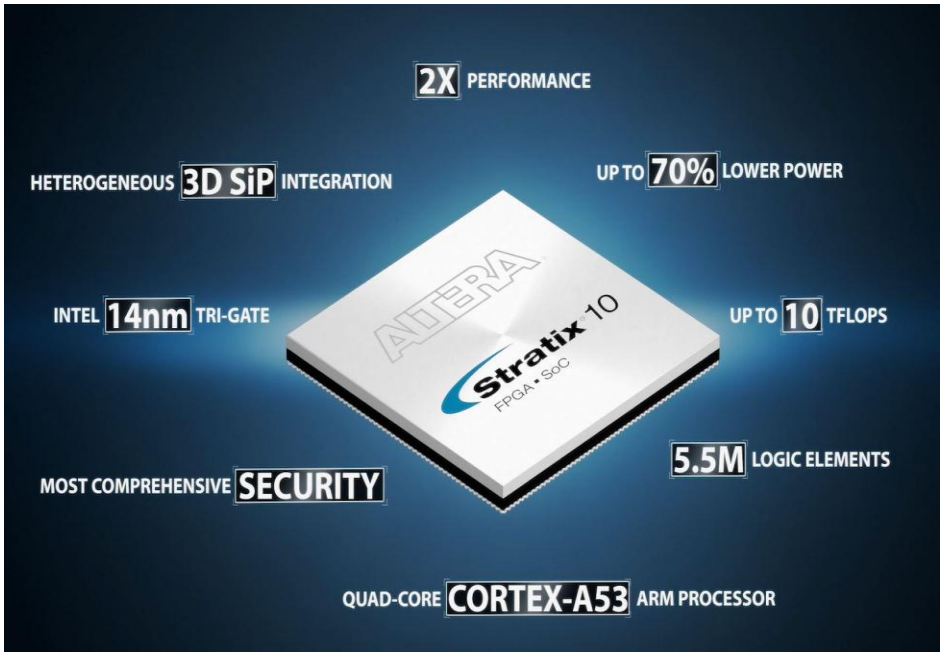


Image 5. Altera FPGA Advantages

4. Compressive Sensing (CS) for Wireless Sensor Networks (WSN)

4.1. What is Compressive Sensing (CS)?

The Compressive Sensing is related to the Compressed Sensing for Networked Data. Nowadays the networked data is a crucial factor for the Information Systems over the networks. Other basic factors are the network bandwidth and the Information System operations related to the traffic. So, the Compressive Sensing plays a basic role for establishing the system.

Initially, we have a basic scenario, in case we have a system with thousands or millions of independent components, all capable of generating and communicating data. The synchronous systems today consists of many components such as computers, cell phones, sensors, and actuators that are all linked to the Internet, and every wired or wireless device is capable of generating and prodigious volumes of data.

The basic task for these Information Systems with specific network architectures is the transmitting of information. The distributed sources of data and their storage, transmission, and retrieval are the building blocks for establishing the connection.

Especially, the transmitting information from one point to another is a very important component. The crucial point of efficiently transmitting or sharing information from and among a vast number of distributed nodes remains to have good techniques, architectures related to the distributed signal processing, communications, and information theory in large-scale networked systems.

In these systems the crucial factor is the Data Compression. In the phase of the architecture of the system we have many several decentralized compression strategies that could be utilized. The scope for this procedure is that the correlations between data at different nodes are known a priori. Then, the distributed source coding techniques can be used to design compression schemes without collaboration between nodes.

The architecture of the Information System is very important. Every system has different Business Specifications and philosophy. The Hardware infrastructure, the Application Software and generally the Business Operations are the building blocks. In the systems for applying distributed source coding techniques we may have specific patterns. This situation motivates collaborative, in-network processing and compression, where unknown correlations and dependencies between the networked data can be learned and exploited by exchanging information between network nodes.

The most important component for the system is the design stage. The design and implementation of effective collaborative processing algorithms is the crucial point. For this procedure we may have knowledge of the anticipated correlations and depend on somewhat sophisticated communications and node processing capabilities.

- The compressed sensing offers two highly desirable features for networked data analysis. Firstly, the method is decentralized meaning that distributed data can be encoded without a central controller. Secondly, the method is universal, meaning that sampling does not require a priori knowledge or assumptions about the data.

The basic purpose of sensing and sampling systems is to accurately capture the salient information in a signal of interest. The basic components are:

- Firstly, the system introduces some form of sensing diversity, which gives each physically distinct signal from a specified class of candidates, a distinct signature or fingerprint.
- Secondly, the diversified signal is sampled and recorded, and finally the system reconstructs the original signal from the sampled data. This procedure, which operates with the low-pass filtering, is a type of preconditioning that maps every signal having frequency less than the filter cutoff frequency to itself, while all higher frequency components are mapped to zero. This step is sufficient to ensure that the signal reconstructed from a set of uniform samples is unique and equal to the original signal.

The basic theory of compressed sensing (CS) extends the traditional sensing and sampling systems to a much broader class of signals. The basic pattern is that according to the CS theory, any sufficiently compressible signal can be accurately recovered from a small number of non-adaptive, randomized linear projection samples.

The sparse vectors are very compressible, since they can be completely described by the locations and amplitudes of the non-zero entries. Rather than sampling each element of x, CS directs us to first precondition the signal by operating on it with a diversifying matrix, yielding a signal whose entries are mixtures of the non-zero entries of the original signal. The resulting signal is then sampled k times to obtain a low-dimensional vector of observations.

The main results of CS theory have established that if the number of CS samples is a small integer multiple greater than the number of non-zero entries in x, then these samples sufficiently encode the salient information in the sparse signal and an accurate reconstruction from y is possible.

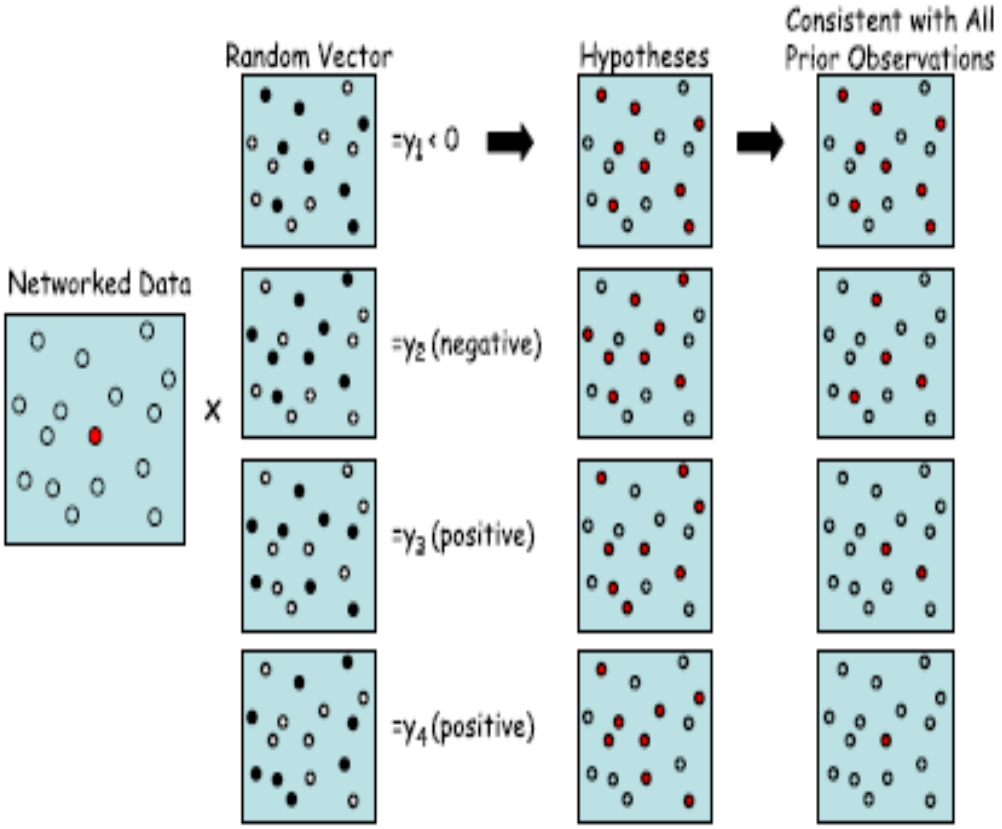


Image 6. A simple reconstruction example on a network of n = 16 nodes.

About the illustration of the CS random projection encoding, in this case we have a network of n sensors, only one of the sensors is observing some positive value, while the rest of the sensors observe zero. The scope is to identify which sensor is different using a minimum number of observations. We can make random projection observations of the data. Each observation is the projection of the sensor readings onto a random vector having entries ±1 each with probability 1/2. The value of each observation, along with knowledge of the random vector onto which the data was projected, can be used to identify a set of about n/2 hypothesis sensors that are consistent with that particular observation. The estimate of the anomalous sensor given k observations is simply the intersection of the k hypotheses sets defined by the observations.
[25], [26]

4.2. Compressive Sampling

The compressive Sampling is an important procedure. The Lossless compression algorithm is used for minimizing the power consumption of the wireless sensor network (WSN). We have a basic rule; the sensor and the receiver node consumes more power when the transmission of data is taking place.

Also, the huge data are processed in the network which also consumes energy. The basic scope is the battery life of the sensors by using compressive sensing to lessen the transmission activity and data processing.

Finally, the Compressive Sampling is a Data Compression technique that is used to represent the signals into sparse. This technique uses linear encoding and transformation matrix, and sparse recovery.

4.3. Compression Algorithms

In this chapter we will see the Data Compression over the Sensor Network, the background of compressive sensing algorithm and the simulation of the compressive sensing algorithm and the power consumption.

4.3.1. Compressive Sensing Algorithm Building Blocks

The Data Compression is an important procedure for the Sensor Network. The scope is to reduce the power consumption of the system. The basic architecture for the Wireless Sensor Network (WSN) is composed of different sensor nodes interconnected to each other and transmit the data to the base station. Analytically, about this procedure we have some assumptions related the Wireless Sensor Network.

Firstly, the sensor node measure has many physical conditions such as temperature, humidity, light intensity and carbon dioxide.

Secondly, it digitizes the inputs and transmits it to the base station which will be needed for monitoring and collecting the data. A typical sensor node is comprised of sensing unit or sensors, the processing unit, the communication unit and power supply unit.

Thirdly, the sensing unit consists of analog to digital converter and the commercially available sensors. The processing unit is comprised of the processor, the memory and input-output ports. The communication unit is the radio transceiver and receiver of the WSN.

Regarding this implementation, we have some crucial factors about the sensing:

- the sensor nodes deployed in different areas need continuous power supply to meet its purpose
- it processes huge amount of data and this is one of the main factors that it consumes more energy and resources
- what consumes more power in the activity of the WSN is the data transmission; the sensor node power is consumed while transmitting the data. Replacing the battery often may hinder accuracy and continuous monitoring of the system. Processing less data can contribute to extend the battery lifetime of the WSN.

WSN has a wide area of applications. It is used to monitor health conditions of the patients such as heart rate, blood pressure, temperature and other physiological signals.

In addition, WSN can also be installed in a greenhouse to detect the soil moisture of the plant. Low power consumption is very important in WSN to continuously monitor the conditions of the signals. In this paper, a compressive sensing algorithm is used to minimize the data to reduce the transmission of energy in the WSN.

Different to the traditional method of sampling all the signals with doubling frequency rate, the proposed method only sampled larger coefficients of the signals and disregarding the rest of the signals. It successfully reduced a huge part of power consumption in WSN.
[12], [13], [14]

Compressive Sensing (CS) algorithm is a procedure n signal processing particularly for data acquisition. In a normal situation, in a conventional sampling process known as the Shannon-Nyquist theorem, the procedure of sampling rate must be twice as much as the highest frequency of the signal being measured.

If the highest frequency of the signal is 30 KHz, the sampling rate must be greater than 60 KHz in order to recover the original signal. In compressive sensing algorithm, it requires a sampling rate lower than the Nyquist rate to recover the original signal. It will only process the large coefficients and disregard the zero coefficients.

This specific type of compression algorithm reduces the size of data sent and decreases the storage requirements of the system. This is very important because it will also lessen the transmission activity of the sensor node. The Compressive Sensing deals with acquiring the signal with a few numbers of measurements.

Also, it collects massive amount of data and compresses the data by neglecting most of the samples. Many researches in compressive sensing focus on image processing.

In the Compressive Sensing Algorithm there are three parts that need to be considered as illustrated in the following **Image 7**. Analytically the parts are:

- the first is the spare representation of the signal
- the second is the encoding and decoding of the measurement vectors and
- the third one is the recovery of the sparse signals.

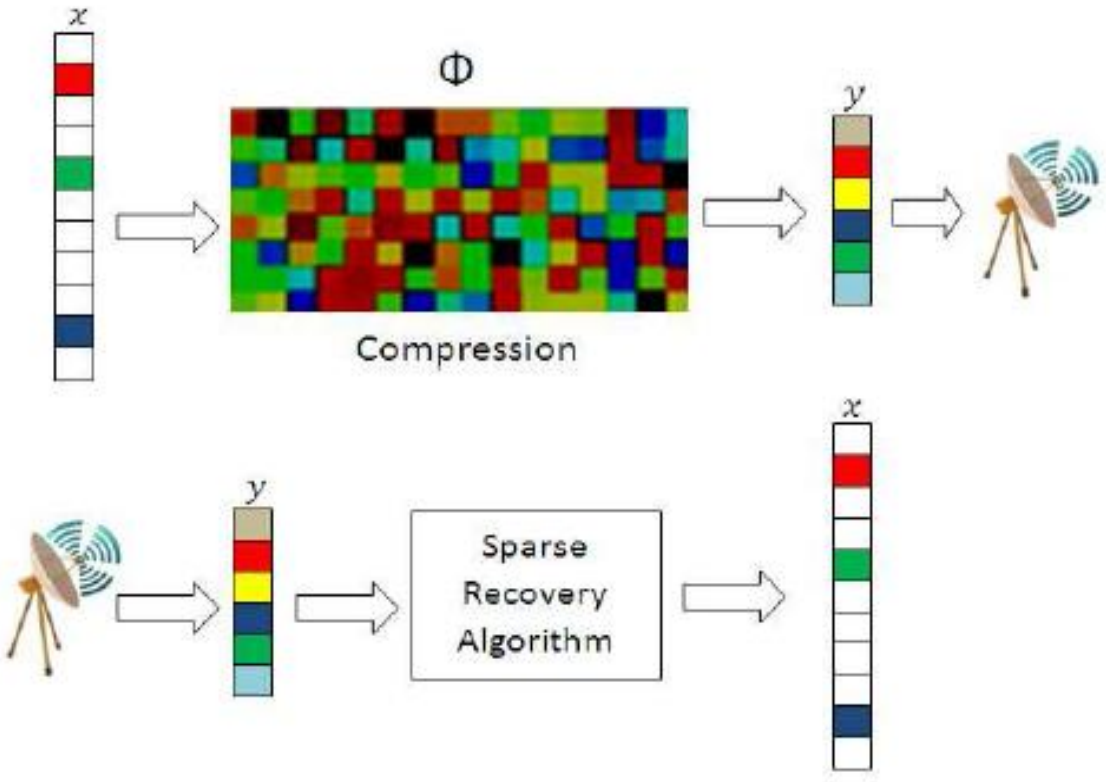


Image 7. Compressive sensing algorithm process

The basic building block of the first consideration in a compressive sensing algorithm is the sparsity of the signal. The sparsity is very important in order to recover the signal in its original form. In this case the signal should have a sparse representation. The representation of the compressible signal can be expressed as:

$$x = \sum_{i=1}^{N} s_i \psi_i$$

Image 8. Equation 1

where **s** is the coefficient sequence and **x** is the sparse representation of the signal,

and as follows….

$$\psi = \{\psi_1, \psi_2, \psi_3 \ldots . \psi_N\}$$

Image9. Equation 2

where **ψ** is in an orthonormal basis that we expand in .

This signal is said to be sparse if most of the coefficients are zero. Disregarding the zero coefficient will not lose much of the information of the signals. The compress signal can be represented in the form of equation.

Regarding this procedure, the architecture of this operation can be analyzed as follows:

- Compressive Sensing requires low coherent pairs. Coherence measures the largest correlation between any two elements
- If the sensing vector and the sparsifying matrix are more incoherent, less number of samples is necessary in order to accurately recover the original signal. The RIP can be connected to coherence property.
- The function and applicability of RIP in compressive sensing are presented in these functions. It is explained that RIP is suitable for uses as the sensing matrix property Φ.
- This gives an accurate result for the under sampled measurements. RIP on the other hand is the solution to noise and imperfect measurements that may arise while acquiring the signals. "A sensing matrix is said to satisfy the RIP of order K if there exists a constant δ ϵ (0, 1) of a matrix Φ such that

$$(1-\delta)||x||_2^2 \leq ||\Phi x||_2^2 \leq (1+\delta)||x||_2^2$$

Image10. Equation 1

where: **δ** is the isometry constant for all K-sparse vectors **x** .

[15], [16],[17],[18], [19]

4.3.2. Simulation of Compressive Sensing Algorithm

Simulation is an important procedure for the algorithm functionality. The Data transmission consumes most of the energy in WSN. In the Sensor Network the energy is the most important component for the operation part. The reduction of data transmission over the network is the primary importance.

The compressive sensing algorithm was simulated using MATLAB software to verify if the signal can be recovered to its original form. The sparsity of the signals is the non-zero coefficients. With this in mind the system will only focus on the large coefficients and zero coefficients will be neglected. Discrete Fourier transform was used to acquire the large coefficients of the signal. The sensing matrix

$$\phi = \{\phi_1, \phi_2, \phi_3, \ldots . \phi_N\}$$

is an M × N matrix. The sparse representation of the signals can be expressed using equation 1 **(Image 8)**. There are 512 samples with four different frequencies used in this simulation as shown **(Image 9)**. After running the program, the 512 samples signal can be represented with only six large coefficients as depicted in **(Image 9)**. Using L1 minimization, it recovered the original signal using only the six large coefficients as illustrated in **(Image 9)**.

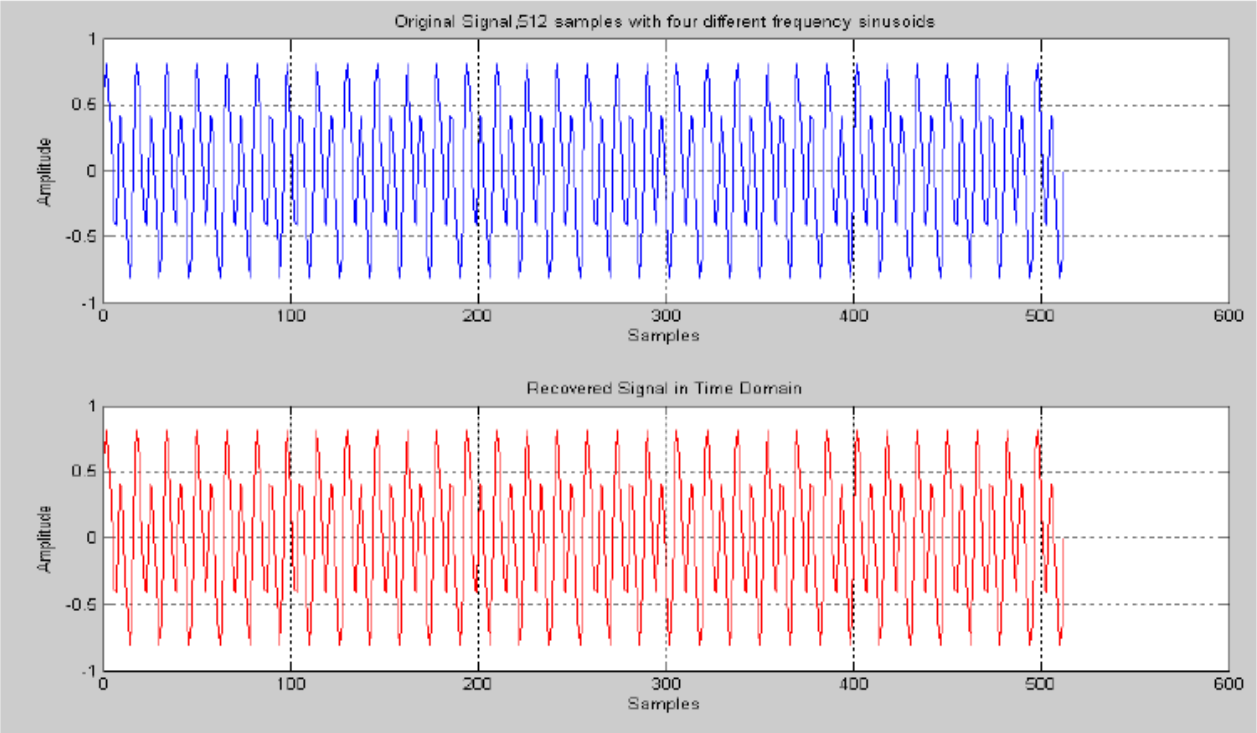


Image11. Original signal compared to the recovered signal

In short, the crucial factor for the performance of the network is the Data Transferring and the energy. For these operations using the compressive sensing algorithm reduced the amount of data that is being processed over the network. Limiting the number of measurements that are being processed will greatly reduce the power consumption of the system.

In addition, another characteristic is that the transmitting activity of the WSN is minimized. It prolonged the lifespan of the battery. For this operation we can use fewer sampling measurements, the original signal can be recovered using L1 minimization.
[20]

4.3.3. Message-passing algorithms for compressed sensing

The basic role of the Compressed Sensing is the sample certain high-dimensional signals yet accurately reconstruct them by exploiting signal characteristics. The reconstruction was established when the object to be recovered is sufficiently sparse.

Generally, the empirical measurements of the sparsity using the sampling tradeoff for the new algorithms agree with theoretical calculations. The state evolution formalism correctly derives the true sparsity under sampling tradeoff.

The Compressed Sensing implements techniques related to high-dimensional signals and yet recover them accurately. These techniques make measurements that traditional sampling theory demands such as the sampling proportional to frequency bandwidth and recover signals by applying simple linear reconstruction formulas.

A Message-passing iterative algorithm achieving the reconstruction performance in one important sense identical to LP. Especially, the vector y of n measurements is obtained from an unknown N-vector x 0 according to y = Ax 0, where A is the n × N measurement matrix n < N. Starting from an initial guess x 0 = 0, the first-order approximate message-passing (AMP) algorithm proceeds iteratively according to.

$$x^{t+1} = \eta_t(A^* z^t + x^t), \qquad [1]$$

$$z^t = y - Ax^t + \frac{1}{\delta} z^{t-1} \left\langle \eta'_{t-1}(A^* z^{t-1} + x^{t-1}) \right\rangle. \qquad [2]$$

Image 12. Message-passing Formula

In the following diagram (**Image 13**) we can see the phase transition lines for reconstructing sparse nonnegative vectors (problem +, red), sparse signed vectors (problem ±, blue) and vectors with entries in [−1,1] (problem, green).

Also the continuous lines refer to analytical predictions from combinatorial geometry or the SE formalisms. Dashed lines present data from experiments with the AMP algorithm, with signal length N = 1,000 and T = 1,000 iterations.

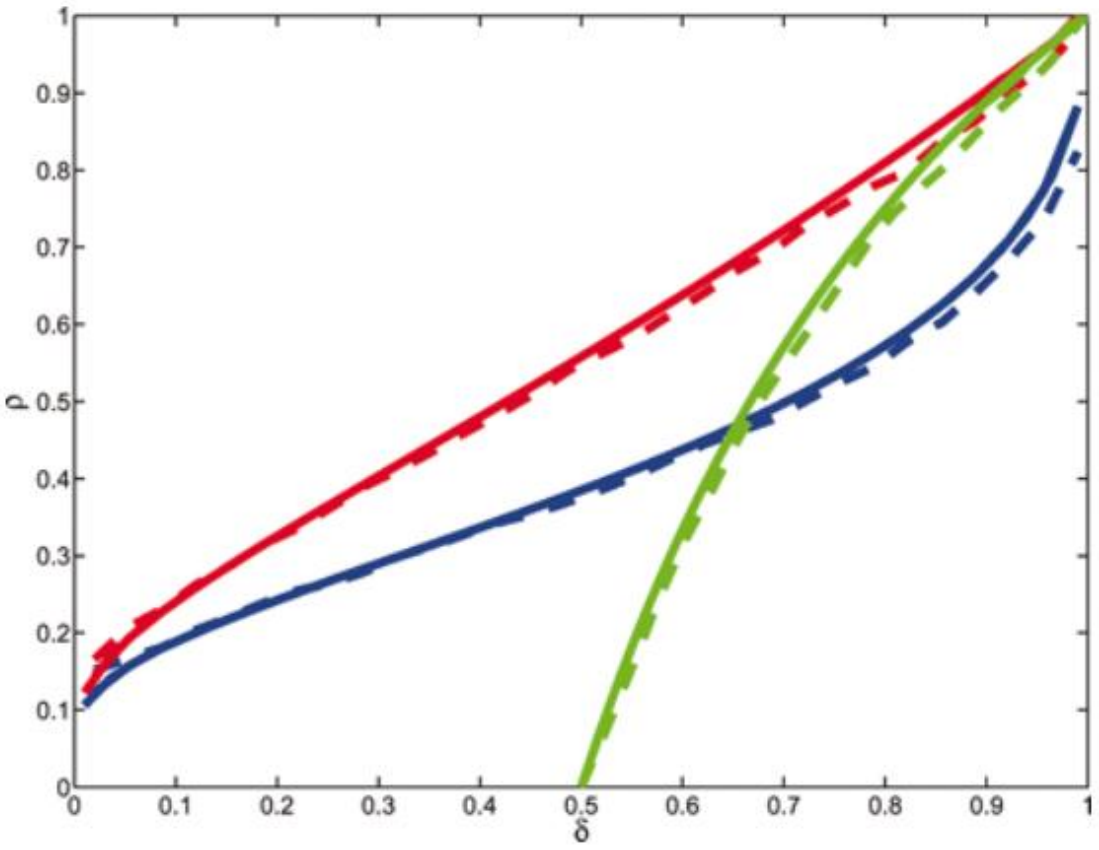


Image 13. Message-passing algorithms

[28], [29]

4.3.4. Compression Algorithms Review

The Wireless sensor networks (WSNs) have many resource constraints. The most important constraints are the limited power supply, the bandwidth for communication, the processing speed and the memory space.

The Data Compression on sensor data is very important. The Data Processing consumes much less power than transmitting data in wireless. This result is effective to apply data compression before transmitting data for reducing total power consumption by a sensor node.

In this section we will see some important compression algorithms, which have been specifically designed for WSNs. These important algorithms are: Coding by Ordering, Pipelined In-Network Compression, Low-Complexity Video Compression, and Distributed Compression.

Data Compression Techniques

The architecture of the Wireless Sensor Networks (WSN) is very powerful. The basic rule is that before sending data using compression mechanisms, we minimize the data size transmitting in network. About this implementation the crucial factor is the data compression algorithm.

The compression algorithms have many difficulties about the implementation. The point is the compression of Data. Data transmitting is the major procedure and the scope is to eliminate the size for better performance and quick Data Transmission.

About the difficulties, one reason is the size of algorithms regarding the instruction memory size of the sensor node. Another reason is the processor speed. The processing speed of sensor node is very small. So, the basic scope about the architecture is to design a low-complexity and small size data compression algorithms for sensor networks.

In the following paragraphs we will see some of the important data compression schemes for WSNs.

4.3.4.1. Coding by Ordering

The Coding by Ordering is part of Data Funneling Routing. The architecture and the operation of this scheme is the following:

1. Firstly, a data pass from sensor nodes in the interested region to a collector node is set up as shown in **(Image 14)**.
2. The crucial factor of this scheme is that in Data Funneling Routing, some of sensor nodes work as a data aggregation node. Aggregation is very important to aggregate, summarize the data and transmitting to the other nodes.
3. In the following scenario **(Image 14)** we have node A, B, and D are aggregation nodes. At an aggregation node, the sensing data collected by other nodes is combined, and the aggregated data is sent to its parent node.
4. At node D in the data collected by node E is combined with data collected by node D itself. Then, the aggregated data is transmitted to node B.
5. In the algorithm, when data is combined at an aggregation node, some data is dropped. To include the information of dropped data in the aggregated data, the order of data packet is utilized. For example, four nodes (N1, N2, N3, and N4) send the data to an aggregation node (Na).

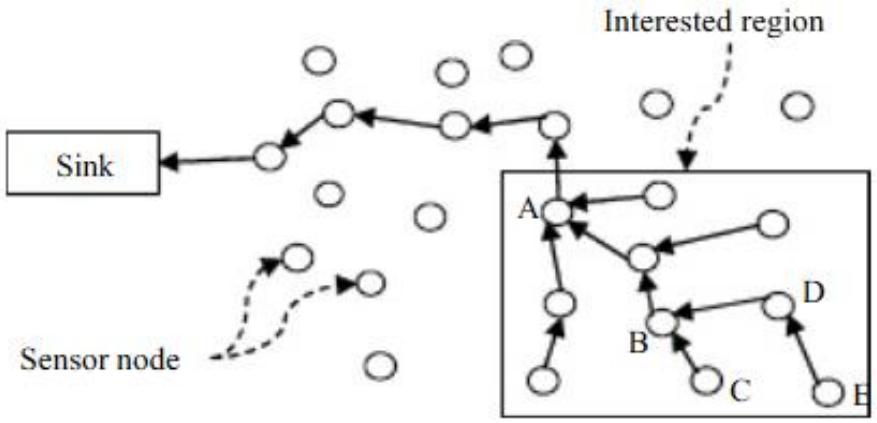


Image 14. Data Path

6. The data value of each node can be any integer ranging from 0 to 5. If we decide to drop the data from N4 and express the data from N4 by ordering packets from other 3 nodes (N1, N2, and N3), there are 3! = 6 possible ordering.

7. Therefore, by using permutation of three packets, the data value of N4 can be included in an aggregated packed without actually including the packet of N4. The possible combination of permutation and data value is presented in **(Image 15)**.

Packet Permutaion	Integer Value
N1,N2,N3	0
N1,N3,N2	1
N2,N1,N3	2
N2,N3,N1	3
N3,N1,N2	4
N3,N2,N1	5

Image 15. Permutation and its Represented Integer Value

[30]

4.3.4.2. Pipelined In-Network Compression

The pipelined in-network compression scheme is very important with many characteristics. The basic architecture is related to the trading high data transmission latency for low transmission energy consumption. The most important characteristics are:

1. The aggregation node's buffer is the building block of this architecture.
2. The Collected sensor data is stored in an aggregation node's buffer for some specific duration of time.
3. During that time, the data packets are combined into one packet. The redundancies are in data packets. The result of this operation is that the data packets will be removed to minimize the data transmission.
4. In a specific scenario of the implementation in this scheme each data packet has the following form: <measured value, node ID, timestamp>. The compressed data packet has the following form :< shared prefix, suffix list, node ID list, timestamp list>.Also, the shared prefix is the most significant bits. This means that all measured values in combined data packets are common.

5. In this scheme the most important advantage is that the shared prefix system can be used for the node IDs and timestamps.
6. The crucial factor is that the efficiency of data compression depends on the length of shared prefix.
7. The compression ratios increases when set a long shared prefix and measured values have commonality.
8. The compression ratio decrease when set a long shared prefix.
9. Eventually, after a specific combination of a large amount of data packets, a large data buffer is required to temporary store those packets.

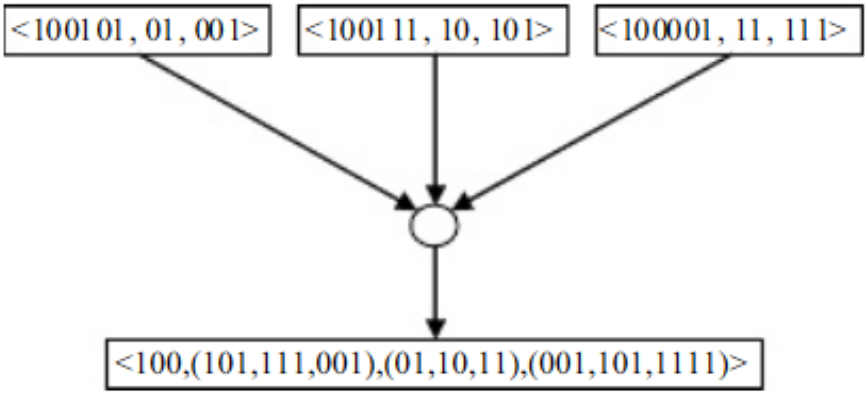


Image 16. In-network Compression

[31]

4.3.4.3. Low-Complexity Video Compression

Another important scheme is the Low-Complexity Video Compression scheme. The basic characteristic of the current video encoding technologies are mostly designed on utilizing the motion estimation and compensation. This means that these technologies will require a high computation power. The result is that the sensor nodes are not usually equipped with this architecture.

This specific scheme proposed method is based on block changing detection algorithm and JPEG data compression. In the **(Image 17)** we can see the block diagram of image data procession flow. This scheme is designed for the wireless videos Data Transmission.

The basic architecture of this scheme is that each video frame is divided into small blocks so that each block contains 64 pixels. The main problem related to the operation is to reduce the computational complexity. About resolving this, the main procedures are:

1. The checking procedure examines the set of white blocks in one frame. If a non-scanned block is adjacent to at least two active blocks, the non-scanned block is marked as active.
2. All marked blocks are encoded by JPEG and transmitted in wireless sensor network.
3. The result from the previous procedures (1 and 2) is that the quality of image processed by this algorithm is quite similar to the image processed by MPEG-2. The crucial point from this is that it achieves the certain amount of energy saving.

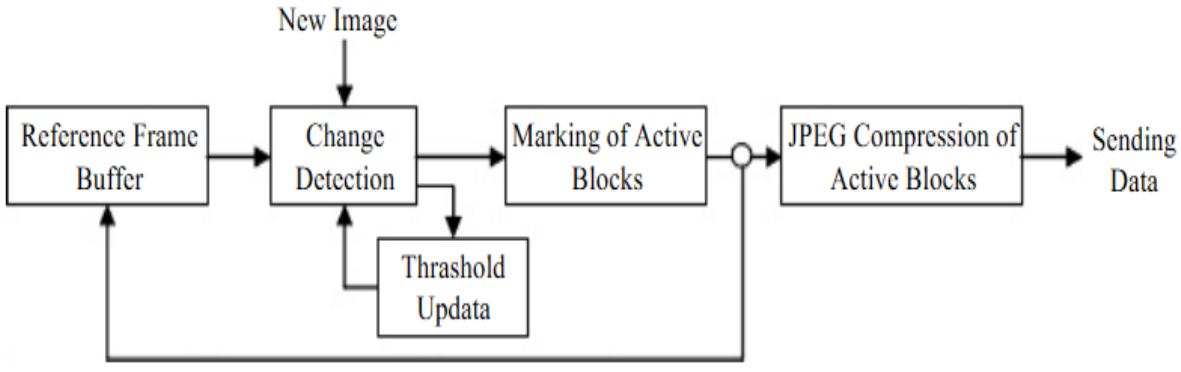


Image 17. Image Data Processing Flow

[32]

4.3.4.4. Distributed Compression

Another important scheme is the Distributed Compression. The basic architecture refers to separate compression and joint decompression of multiple correlated sources.

In the **(Image 18)** we can see the two sources (X and Y). The basic characteristics are:

1. The sources are correlated and discrete-alphabet independent are identically distributed.

2. In the sensor network, the sensor nodes will be densely populated in a sensor field. The result is that the correlation condition can be satisfied easily.

3. The source X can be compressed at the theoretical rate of its conditional entropy.

4. The Data Transmission has the following procedure: the first step is to transmit data to the decoder .The source X only sends an index value of coset, to which the code vector belongs. In the second step , the source Y sends a code vector as a side-information. In this case the decoder looks up the coset, which has the same index received from the source X.

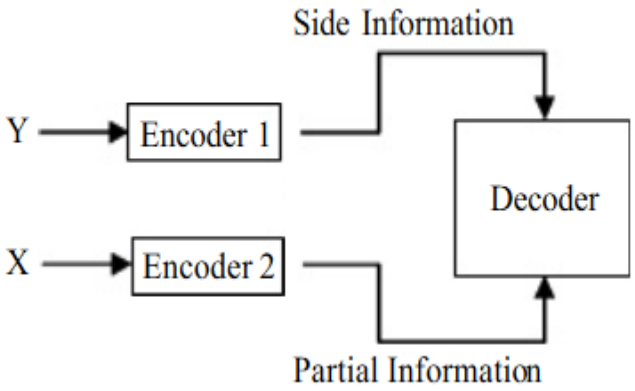


Image 18. Distributed Compression

In short, after the review of the Algorithms related to the compression we have many important points:

1. Nowadays, we have a wide range of application areas for WSNs. While the Hardware Infrastructure is developed every day, new services are available to the Users. New critical Information Systems will be implemented in this area. In the near future, as the technology progresses, the application areas of WSNs will become much broader than now.
2. The usability of these Information Systems through the WSNs has many obstacles to overcome for the practical use of sensor networks. One of the most important obstacles is the limited resource of wireless nodes.
3. In this review, five different types of data compressions schemes were presented. These basic schemes are: Coding by Ordering, Pipelined In-Network Compression, JPEG200, Low-Complexity Video Compression, and Distributed Compression.
4. These compression schemes are still under development. As the research and development advance the compression rate and power reduction manners are quite impressive.

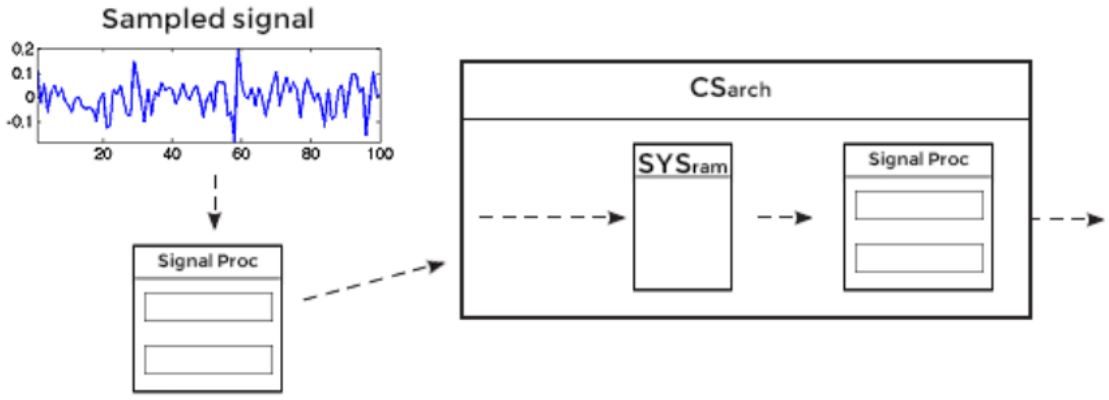
[33]

Designing Part

4.4. Hardware Approach

In this section, an abstract top-level hardware based architecture is proposed in terms of computation speed and resource utilization, in an eventual hypothetic $FPGA$ implementation.

There are two main computation models in the top-level design architecture diagram. The first model receives an input signal f and after n elements of the analog signal have been sampled, they are converted to n digital samples, which are represented as the main input data, $x \in \mathbb{R}^n$. As x have been sampled, it is send to the $\mathrm{CS}_{\mathrm{arch}}$ encoder model, which processes it in a way that will be explained later on the chapter. All n elements of the sampled data at the same time are transferred and stored at the RAM model $SYSR_{ram}$ of the primary $\mathrm{CS}_{\mathrm{arch}}$ model. The main goal of the architecture of $\mathrm{CS}_{\mathrm{arch}}$ is to obtain the high compressibility on the samples being taken, while the measurements are made of a sub-sampling of the values of the sampled signal. The result of the encoder is a compressed signal that has m samples, while $m < n$.



In order to examine the proposed architecture, we have to make a brief analysis of the main arithmetic operation in a Compressed Sensing encoder as presented in earlier chapter 4.3.1, is a matrix-vector multiplication, as formulated in the following equation:

$$y = \Phi x \ (1)$$

where the input signal has n samples, Φ is the sensing matrix with m rows and n columns and y is the compressed signal vector. Compressed sensing theory relies on the prerequisite that the input signal can be represented by a sparse signal under some basis, which can be formulated in the following equation:

$$x = \Psi s \ (2)$$

Recall at this point, that x is being sampled in an n dimensional basics $\{\psi_1, \psi_2, \dots, \psi_{n-1}, \psi_n\}$. Then we have as ordered in the following relationship:

$$x = \sum_{i=1}^{N} s_i \psi_i \ (3)$$

However, if some of the s_i coefficients are small, toss out the related $\psi_i^{'} s$ that means that there may be a subset of φ_i's that adds up to x.

For example, if the input signal is already sparse in the time domain, then Ψ which is a sparsifying matrix $n \times n$, it will be the identity matrix I. In case the input is sparse in the frequency domain, then Ψ will be a Fourier transform matrix. For example, a sine wave signal requires many coefficients to represent in time domain, while it requires only one $non - zero$ coefficient to be represented in the frequency domain. This is certainly a computationally intensive procedure, but on the other hand it contains a high level of regularity.

The following scheme represents, by simple means, the arithmetic operations that must be obtained on our architecture model to obtain the output compressed signal. The final equation is formed as:

$$y = \Phi x = \Phi \Psi s \ (4) \Leftrightarrow$$

$$\begin{bmatrix} \Phi_{(1,1)}x_1 & + & \Phi_{(1,2)}x_2 & + & \cdots & + & \Phi_{(1,n)}x_n \\ \Phi_{(2,1)}x_1 & + & \Phi_{(2,2)}x_2 & + & \cdots & + & \Phi_{(2,n)}x_n \\ & & & \vdots & & & \\ \Phi_{(m,1)}x_1 & + & \Phi_{(m,2)}x_2 & + & \cdots & + & \Phi_{(m,n)}x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

where Φ is a $m \times n$ sensing matrix and s_i values are the largest coefficients of the input samples. Besides sparseness we also require incoherence between the sensing matrix and the sparse representation matrix in order to minimize the number of measurements needed for the recovery. Another constraint that the sensing matrix and the sparse representation matrix must satisfy is the restricted isometry property to ensure stable recovery of the $k - sparse$ signal s. The coherence of the Φ and Ψ can be evaluated in by following expression:

$$(1 - \delta)\|s\|_2 \leq \|\Phi \Psi s\|_2 \leq (1 + \delta)\|s\|_2$$

As random sensing matrices are likely to obey the RIP above RIP property the Φ matrix can be populated with random variables in a various random distribution. We note at this point that sparsity is determined as $(1 - \frac{K}{N})$ where K is the number of significant values among the n input samples. Compressed sensing relies primarily on the signal of interest, so sparsity implies compressibility.

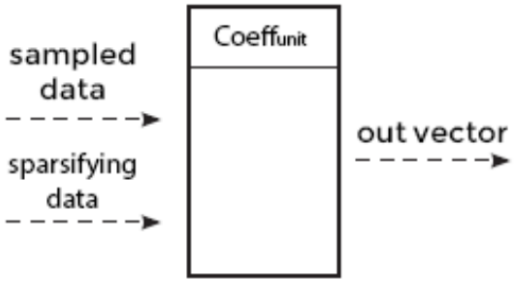
Back on the primary architecture, we can re-examine the CS_{arch} model and all the separate inner sub-models of the $\text{Signal Proc}_{\text{unit}}$. After the illustration of the main sub-models, we will examine the connection of them to achieve a reliable fast compression unit.

The $Coeff_{unit}$ model performs the computation of the weighting coefficients s:

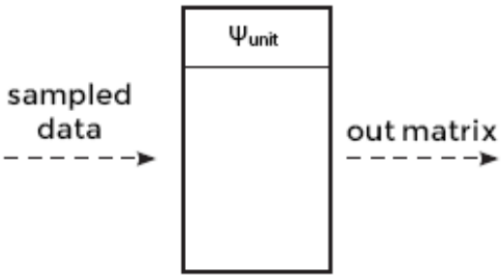
$$s_i = \langle x, \psi_i \rangle = \psi_i^T x$$

We recall at this point, that x is $K -$ sparse and a linear combination of k basic vectors. In this data acquisition system, the main coding process relies on the fact that all $n -$ samples of x have been acquired to compute the output coefficients. Recall that the model gets all input data from the RAM model of the primary design. The output n values can be stored back to the RAM

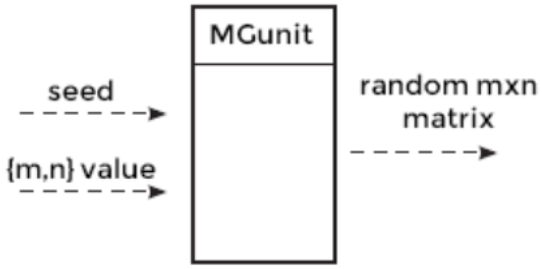
memory of the primary design and on the model's registers, if that approach can increase the terms of speed, especially in the case any parallel computations appear on the final design. In the following figure an abstract top-level scheme of the model is presented.



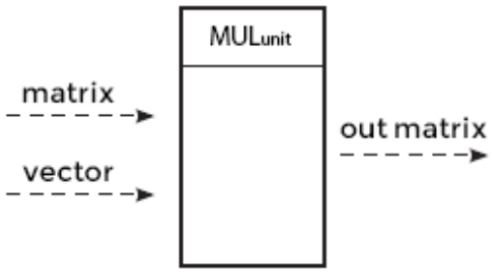
The $\Psi unit$ is the sparsifying matrix generation model, which is capable of generating the sparse representation matrix of the input sampled data in the order of the description that has previously been given on this chapter. The model can either work in parallel with the $Coeff_{unit}$ or in serial. As we propose a simple architecture scheme we assume that all models from this point work in serial, which means, that all the required input data of every model must have been computed at previous computation cycles. *Note that, we can try to parallelize the process by using some registers at every model to store intermediate data of computation and compute every output data in slices.* In the following figure, an abstract top-level scheme of the model is presented.



The $MGunit$ is the matrix generation model which is capable of generating random matrices with a possible given seed $value$. The model can also be configured with m and n values to configure output matrix dimensions. This model doesn't require any wait computation cycles as it can work without any previous computed data input. In the following figure, an abstract top-level scheme of the model is presented.



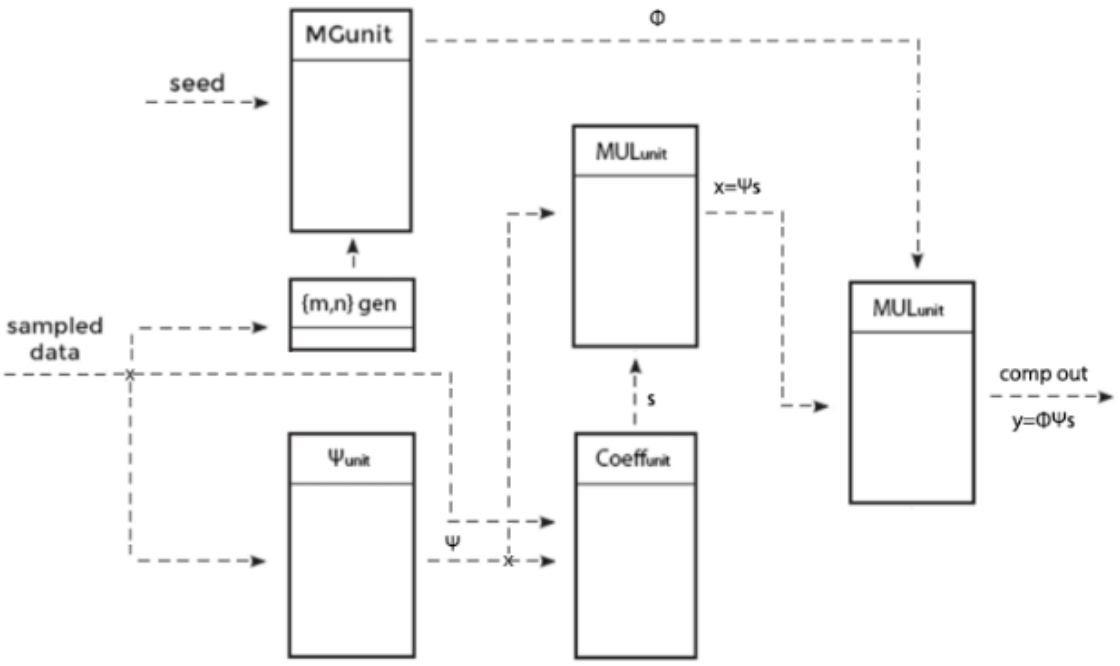
The MUL_{unit} model can multiply a matrix $A \in \mathbb{R}^{m \times n}$ with a vector $b \in \mathbb{R}^{n}$ and as result produce a vector $c \in \mathbb{R}^{n}$. All the input data of the model must have been computed before, to have correct results on the model's output. In the following figure, an abstract top-level scheme of the model is presented.



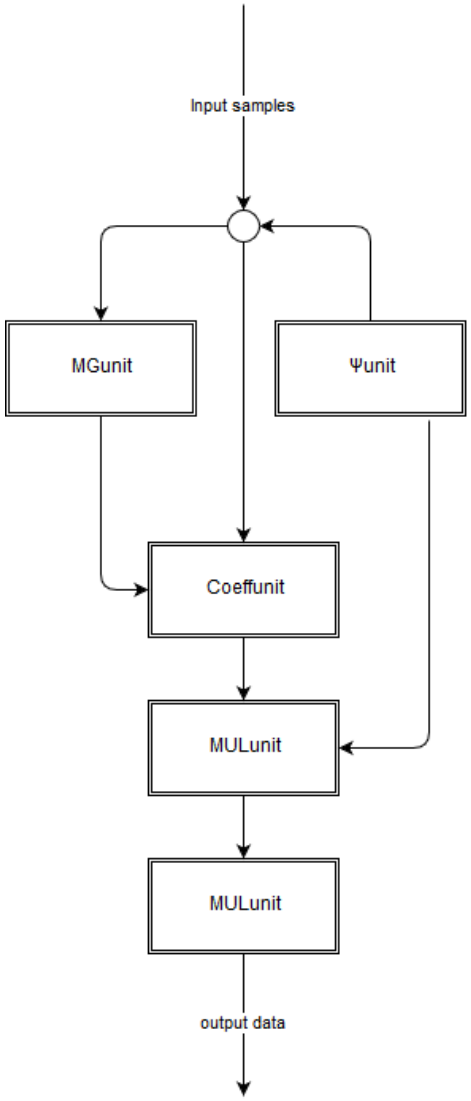
After the illustration of the above separate models, we can start combining them one by one, to achieve a compression signal unit. In abstraction terms, we divide the design in three levels of computation processes. On the first level, we receive the input n samples from the RAM model of the primary design. Input data can be driven to the following models:

- $\Psi unit.$
- An $\{m, n\}$ generator.
- $Coeff_{unit}$

The $\Psi unit$ is able to work in parallel with MG_{unit} as both models compute non-relevant data. Output data from both models must be also stored back in the RAM model which is derived on the next level models of the design. *For simplicity at the models' illustration we derive that output directly to the next levels models*. After the computation process of $\Psi unit$, the $Coeff_{unit}$ can start its own computation process and as soon as the process is over the output data is derived on the first MUL_{unit}. Output data from both models is also stored back in the RAM model, which is also derived on the next level models of the design. Finally, the output data of the first MUL_{unit} is derived with the stored data of MG_{unit} to the second MUL_{unit} in order to compute the final output. In the following figure, an abstract top-level final scheme of the compression model is presented.



We can also represent the architecture diagram on the following flow chart design, on which we can see in detail the separation of the different computation levels of the design.



As we described earlier on the chapter 4.3.3, the Approximate Message Parsing (AMP) has as basic variable, input messages, that are being associated to a directed edge in bipartite graph model over N nodes of signal and measurements. In order to compute these relations, we use an iterative algorithm whose main purpose is to achieve a reconstruction scheme capable of computing the reconstructed signal faster, compared to direct methods, as also to produce a valid reconstruction with the minimum residual error.

Recall that we made the assumption, that a vector y of n measurements is obtained from an unknown x_0 according to the relation $y = Ax_0$, where A is the measurement matrix. Starting from an initial guess $x^0 = 0$, the *first order*AMP algorithm proceeds iteratively according to:

$$x^{t+1} = \eta_t(A * z_t + x^t)$$

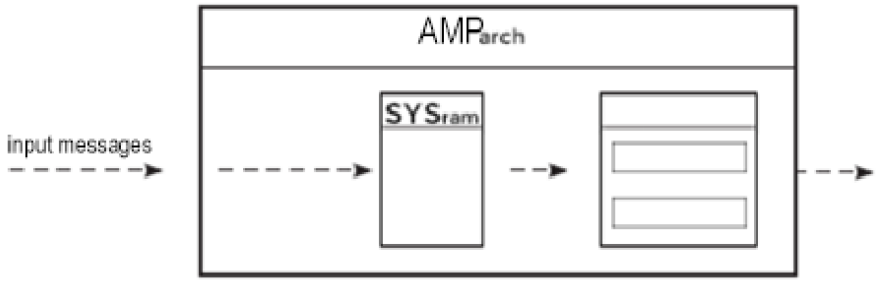
$$z_t = y - Ax^t + \frac{1}{\delta} z_{t-1} \langle \eta'_{t-1}(A^* z * z_{t-1} + x - x^{t-1}) \rangle$$

Here $\eta t(\cdot)$ are scalar threshold functions *(applied component wise)*, $x^t \in R^N$ is the current estimate of x_0, and $z^t \in R^n$ is the current residual. Also A^* denotes transpose of the matrix A. Later modifying the z_t equation to a simpler form of

$$z_t = y - Ax^t$$

we observe that we lack the initial crucial terms. We derive those terms from the theory of belief propagation in graphical models, to improve the sparsity-undersampling tradeoff. Extensive numerical and Monte Carlo work reported shows that AMP, defined by initial equation achieves a sparsity-undersampling tradeoff matching the theoretical tradeoff which has been proved for LP-based reconstruction schemes.

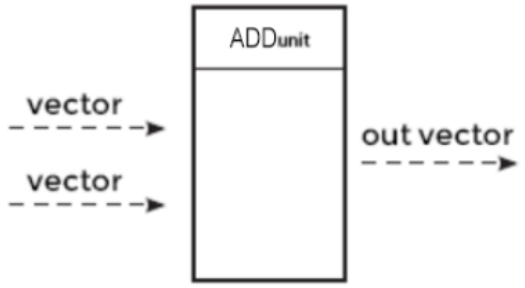
Back on the top-level design, we design a new scheme with two basic models. The main model, receives the compressed messages that have been send from the presented CS_{arch} model. All n of the message are elements that are transferred and stored at the RAM model $SYSR_{ram}$ of the AMP_{arch} model. The main goal of the architecture of AMP_{arch} is to obtain the high reconstruction speed with an accepted and bounded number of iteration points, while these values have a small residual error. The result of the AMP model is the reconstructed signal of the CS scheme, which is provided on the schemes' output.



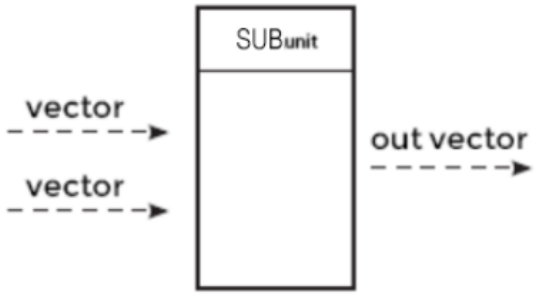
In order to analyze the AMP model's architecture, we can re-examine the iterative relation scheme in order to define all the separate inner sub-models that are essential to the final design. After the illustration of the main sub-models, we will examine the connection of them.

The $ADDunit$ is a vector addition model, which is capable of subjoin componentwise two given input vectors $a \in \mathbb{R}^n$ and $b \in \mathbb{R}^n$ and finally as result produce a vector $c \in \mathbb{R}^n$. Recall the model gets all input data from the RAM model of the primary design. The output value can be stored back to the RAM memory of the primary design and also on the model's registers, if that approach can increase the terms of speed especially in the case any parallel computations appear on the final design. As we propose a simple architecture scheme we assume that all models from this point, work in serial, which means, that all the required input data of every model must have been computed at previous computation cycles. *Note that, we can try to parallelize the process by using some registers at every model to store intermediate data of computation and compute every output data in slices.*

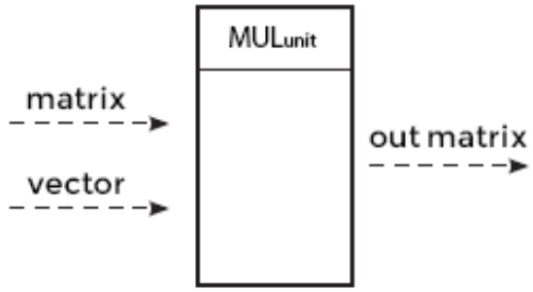
In the following figure an abstract top-level scheme of the model is presented.



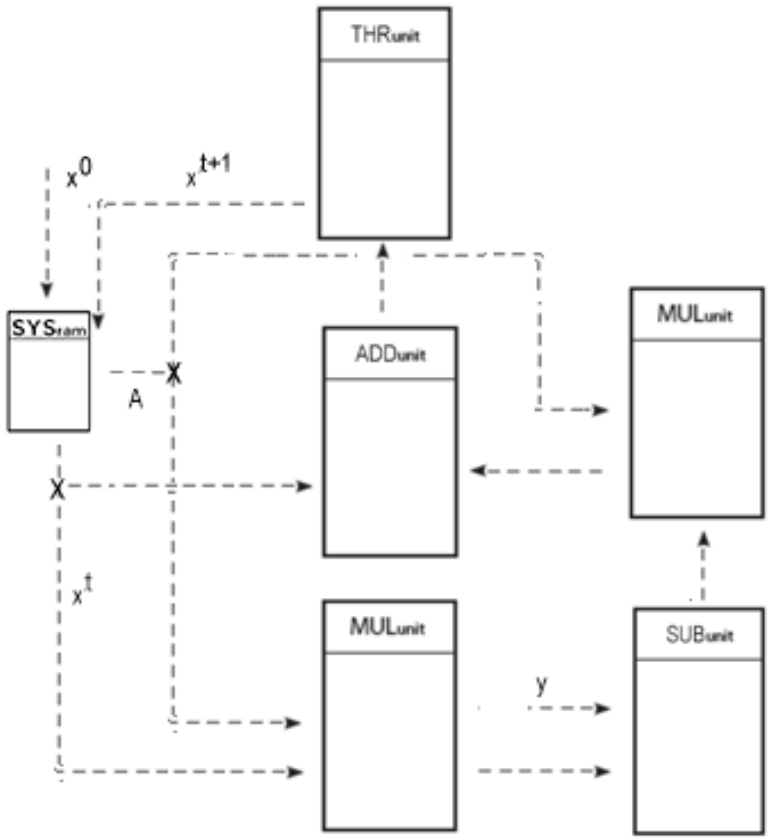
The $SUBunit$ is a vector substitution model, which is capable of subtracting componentwise two given input vectors $a \in \mathbb{R}^n$ and $b \in \mathbb{R}^n$ and as result produce a vector $c \in \mathbb{R}^n$.



Finally, the MUL_{unit} has been presented earlier $\mathrm{CS}_{\mathrm{arch}}$ model and as we recall, is a model capable of multiplying a matrix $A \in \mathbb{R}^{m \times n}$ with a vector $b \in \mathbb{R}^n$ and as result produce a vector $c \in \mathbb{R}^n$.



After the illustration of the above separate models, we can start combining them one by one, in order to achieve a valid AMP unit. Moreover, we divide the design in two levels of computation processes. On the first level, we compute the x^t estimation vector while receiving estimations from the RAM model of the primary design. Input vector is driven to the threshold model, while the output is stored back in the RAM model and also being derived at the residual calculation unit. The final estimation is driven to the next level of the design which is multiplicated with the measurement matrix and the final output reconstruction vector y is obtained at the output, while it is also being stored on the RAM model, too. In the following figure an abstract approach of the top-level final scheme of the compression model, is presented.



4.4.1. Core modules brief analysis

In the previous chapter we analyzed the main core modules of the compression and the signal reconstruction processes. Both modules include several sub modules, capable to be configured in *FPGA* platforms. Most of those modules implement linear algebra basic functions as addition, subtraction and multiplication. Those operations are four functional modes we described earlier at the presentation of the main submodules of our design. On all operations, we denote as input, the matrices A and B and C as the output matrix. Matrix dimensions can be configurable through three-dimension parameters M, N, and L.

The next set of parameters is the data width of individual matrices, A_{WIDTH}, B_{WIDTH} and C_{WIDTH}, representing precision of elements appear on the input and output matrices respectively.

As the final architecture design approach, is a real-time processing core, input data matrices must be fed on the modules in contiguous cycles of the input clock and this constraint must also be applied to both input matrices. The next batch of matrices can be fed in the following clock cycle making the core fully streaming capable. Similarly, the output matrix can be available at the output port contiguous cycles. The core can have the function to be paused and resumed dynamically with the addition of a clock enabling the signal at the input of each core module. This last feature can be used for flow control if needed. The basic linear algebra functions are represented on the following table and being analyzed on later paragraphs.

Function	Operations
Multiplication	$C_{MxN} = \sum_{K=1}^{N} A_{MxN} x B_{KXN}$
Addition	$C_{MxN} = A_{MxN} + B_{MxN}$
Subtraction	$C_{MxN} = A_{MxN} - B_{MxN}$

$\mathbf{SUB_{unit}}$ brief analysis

In this module all input and output matrices have the same dimension, *MxN(M is the number of rows and N is the number of columns)*, and also the same number of elements the data width parameters A_{WIDTH}, B_{WIDTH}, and C_{WIDTH} are denoted as A_w, B_w, and C_w respectively. The elements of the corresponding matrix in one clock cycle are denoted as NE_A, NE_B, and NE_C and the computation formula is the following one:

$$NE_{A,B,C} = ceil(\frac{MxN}{F})$$

which denotes the number of elements required to perform the operation. Last, the number of clock cycles required for a real-time processing operation is determined by the following formula:

$$C_{SUB} = ceil\left(\frac{MxN}{NE_{A,B,C}}\right) = ceil\left(\frac{MxN}{ceil(\frac{MxN}{F})}\right) \leq F$$

$\boldsymbol{ADD_{Unit}}$ brief analysis

In this module, all input and output matrices have the same dimension, *MxN (M is the number of rows and N is the number of columns)*, and the same number of elements as the addition is the same operation with the substation, being analyzed on the previous paragraph. The data width parameters A_{WIDTH}, B_{WIDTH}, and C_{WIDTH} are denoted as A_w, B_w, and C_w respectively in these cases follow also the previous scheme. The elements of the corresponding matrix in one clock cycle are denoted as NE_A, NE_B, and NE_C and the computation formula is the following one:

$$NE_{A,B,C} = ceil(\frac{MxN}{F})$$

which denotes the number of elements required to perform the operation. Last, the number of clock cycles required for a real-time processing operation is determined by the following formula:

$$C_{ADD} = ceil\left(\frac{MxN}{NE_{A,B,C}}\right) = ceil\left(\frac{MxN}{ceil(\frac{MxN}{F})}\right) \leq F$$

$\mathbf{MUL_{Unit}}$ brief analysis

In this module, unlike the addition and subtraction modules, the dimensions of the input and output matrices can be dissimilar and the factor F of the matrix scales rows only, while giving elements per cycle through the following formulas:

$$NE_A = ceil(\frac{M}{F})xK$$

$$NE_B = ceil(\frac{K}{F})xN$$

$$NE_C = ceil(\frac{M}{F})xN$$

The above variables denote the number of elements required to perform the operation. Next, the number of clock cycles required for a real-time processing operation is determined by the following formula:

$$C_{MUL_A} = ceil\left(\frac{MxK}{NE_A}\right) = ceil\left(\frac{M}{ceil(\frac{M}{F})}\right) \leq F$$

$$C_{MUL_A} = ceil\left(\frac{KxN}{N_B}\right) = ceil\left(\frac{K}{ceil(\frac{K}{F})}\right) \leq F$$

$$C_{MUL_A} = ceil\left(\frac{MxN}{NE_C}\right) = ceil\left(\frac{M}{ceil(\frac{M}{F})}\right) \leq F$$

Last, on the previous chapter we denoted some relevant to those processes core modules that do not necessary perform any linear algebra operation.

4.4.2. Core modules operation acceleration architecture analysis

The accelerator model consists of a linear list of multiplier/adder processing elements with local registers for buffering input operands data. Although those processing elements connection is possible, the linear list has the advantage of a much more regular structure which allows simpler routing between processing elements and produce a higher clock frequency. After the initial latency, a list of total N processing elements multiply two n element vectors in one clock cycle to the multiplication unit, or two square matrices of order n in n^2 clock cycles. In the addition/subtraction units n elements are computed in the relevant clock cycles. That initial latency consists of the time that is necessary to fill the input buffers and with the time that is necessary to fill the pipelines of the multipliers and the adders. All the subsequent loading of input data overlap with computation on previously loaded data, and all the pipelines of arithmetic units are constantly full. That linear list architecture can also extend to multiple FPGA target devices. In that case, the number of parallel arithmetic operations and the required communication bandwidth increase linearly with the number of that FPGA devices.

Let us again denote that A and B and C are matrices with dimensions $m \times n$ respectively. Both input matrices A consists of $i \times j$ blocks of order N, where

$$i = \left\lceil \frac{n}{N} \right\rceil$$

$$j = \left\lceil \frac{m}{n} \right\rceil$$

At this point we fill the right and bottom of the matrices with zeros if necessary, to have an integer number of blocks. It is possible to multiply the matrices A and B by performing matrix multiplication and addition operation only between the above blocks. The result matrix C has dimensions $m \times n$ and it consisted also of $i \times j$ blocks.

We refer to the blocks of matrices A, B and C as $[A - B - C]_{uv}$ respectively, where

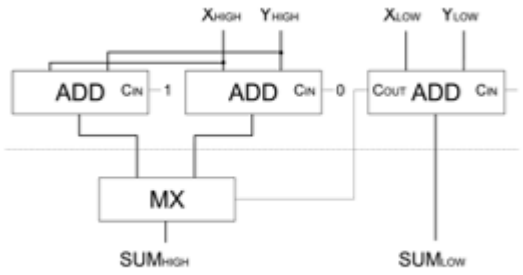
$$u \in \{1, \dots, i\}$$

$$v \in \{1, \dots, j\}$$

The architecture algorithm starts by receiving the blocks of inputs matrices. The results of each operation (addition/subtraction or multiplication) consecutive input blocks does not represent parts of the same result block. For that reason, the accelerator cannot add them together. These block additions take place simultaneously with the results reception and constitute only a small fraction of all arithmetic operations and therefore, for a sufficiently large i or j factors, this algorithm has almost equal bandwidth requirements in both input or output directions. Finally, those accelerator units with a clock frequency f perform n multiplications and $n-1$ additions in each clock cycle and therefore the total of fraction is $(2 \times n - 1) \times f$ floating-point operations per second (FLOPS).

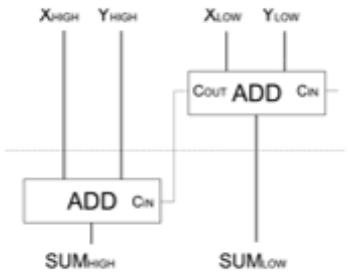
Addition/Subtraction

In the addition and subtraction units to perform a two-operand floating-point operation, it is necessary to assure that both operands have the same exponent. That can be achieved by right shifting the mantissa of the smaller one operation operand and increasing its exponent accordingly to that case, the sign of the result should be equal to the sign of the larger mantissa. It may then be necessary, to normalize the result by shifting it one place to the right *(in the case of effective addition)* or up to *n* places to the left *(in the case of effective subtraction)*, where *n* is the width of the mantissa. Finally, it is necessary to round the result according to the chosen rounding mode. The core element of the carry select approach used on the implementation of an adder is shown on the following figure.



The final floating-adder consists of up to ten pipelined stages $A_1 - A_{10}$. In the first stage, the mantissas of input operands are being compared and the exponent difference is calculated, which determines the number of places smaller mantissa should be shifted to the right. In the second stage, there are three levels of logic, each implementing a funnel shifter. The usual approach is to use a bunch of six shifters to perform a shift operation up to 32 bits. However, the six-input LUT

architecture of the target *FPGA* board may allow combining two shifts in the same LUT, thus reducing the number of required logic levels. In the third stage, mantissas are swapped if necessary, to assure that the first mantissa is the larger one. The addition/subtraction operation is performed with a ripple-carry adder which use one LUT per result bit and a dedicated fast carry-propagation chain. Although those adders are very efficient, their propagation delay increases linearly with operand size factor and thus it becomes a clock-limiting factor. Thus, as we may need a low logic count and high clock speed implementation than a low latency one, we may spread the adder through two additional pipeline stages, four and five as shown on the below figure

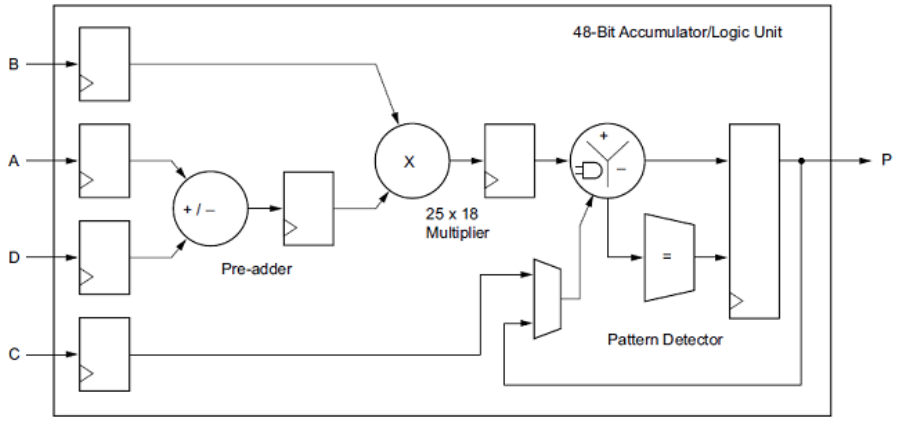


The next three stages contain the leading zero counter operation as also the normalization left shift operation. The calculated values represent the contiguous groups of 4 and 16 zeros in the result. By priority encoding them, it is possible to determine the number of leading zeros. The rounding addition takes place in the last two stages (with the usage of an adder split according to the above figure if necessary). The final adder supports operations with subnormal numbers. The resource utilization is $871\,LUTs$ and 1022 flip-flops and the achieved clock frequency is $509MHz$.

Multiplication

To perform a floating-point multiplication operation of two input operands, it is necessary to multiply their mantissas, add exponents, and calculate the result sign as the xor result of the input operand signs. The most complex part is the multiplication of the mantissas. In the case of IEEE double precision format, mantissas are 53 bits wide, and an efficient multiplication on FPGA target require the usage of multiple embedded multiplier-adder blocks. There are several ways to use the above blocks as basic core parts of a larger multiplication pipelined algorithm.

Each block on this case, computes one partial product, and adds it together with a part of previously computed result. The schematic of that block is shown on the below figure.



The floating-point multiplication unit consists of total seventeen pipeline stages $M_1 - M_{17}$. The relatively large number of stages is necessary because multiple embedded multiplier-adder blocks require two clock cycles to execute multiplication and addition at full speed. For that purpose, they have internal pipeline registers after both the multiplier and the adder. In the first stage, the exponent and sign of the result is being calculated. Multiplications corresponding to blocks operands take place on the next twelve stages respectively.

The adder in each block accumulates the parts of the result which are located above and to the right from it and are calculated in previous stages. These additions take place in these stages too. In the fifteenth stage, the result mantissa is normalized as the rounding digit being computed. If we assume that input operands are normal, the multiplication result can be either normal or may require a single normalization left shift. That rounding process takes place in the final two stages using the proposed two-stage adder discussed on the previous section. The final resource utilization is 447 LUTs and 520 flip-flops and the achieved clock frequency is up to $492MHz$.

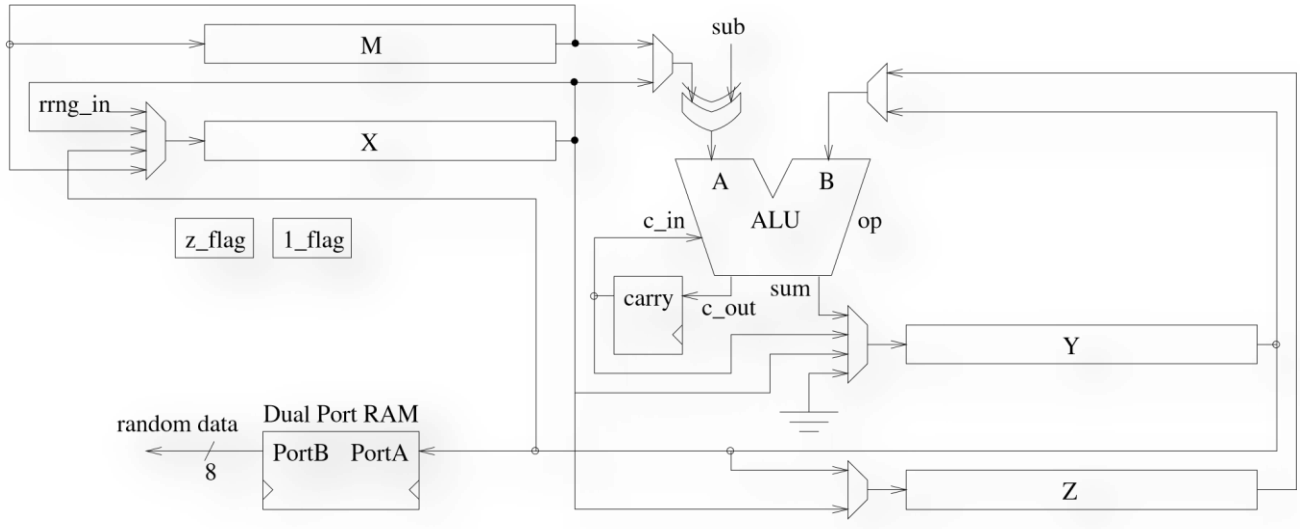
The above proposed multiplier unit, works only with normal input operands. It is relatively easy to add support for subnormal numbers by placing leading zero counters and left shifters on multiplier inputs *(to pre-normalize the mantissas)*. It is also necessary to add a right shifter in the last stage, to accommodate for a possible result mantissa conversion to subnormal format. The total cost of this additional logic is 727 LUTs and 635 flip-flops and the achieved clock frequency is derived lower than $420Mhz$.

Random Number Generator

First, we denote that there are two types of random number generators commonly used, the true random number generator which derives its output from a physical noise source whereas a pseudorandom number generator expands a relatively short key *(possibly from a true number generator)* into a long sequence of seemingly random bits based on a deterministic algorithm. In this architecture, a possible FPGA based implementation of a pseudo random generator is proposed. Apart from achieving a higher level of integration, keeping the critical random number generation operations internal to the device achieves better security since these data do not need to be passed to the FPGA via the pins.

There are many methods to generate pseudorandom sequences, and the classical software based methods, all of which can be implemented in the hardware model. In this work, we use the following equation to generate a *Blum Blum Shub* sequence $X_i$where i is a positive integer:

$X_{i+1} = X_i^2 mod\ M(1)$ where M is a product of two large prime numbers p and q, which both have a remainder of 3 when divided by 4. X_0 must be seeded and is co-prime with M. The *Blum Blum Shub* algorithm is appropriate for use also in cryptographic applications since it has a strong security proof which relates the quality of the generator to the difficulty of integer factorization. The implemented algorithm uses an M factor which is $1024 - bit$ length and the modulus *M* was hardwired in the design. The following circuit shows the data path of a generator unit.



There are four total $1024 - bit$ shift registers in the design: M, X, Y and Z. Register M stores the value of Mwhich will not be changed. Register X stores the value of each X_i and registers Y and Z can be combined to form a 2048 −bit register, register YZ ,to store the temporary results of each ALU operation. After the *mod* operation, the result, X_i^2 *mod M*, is stored in register Y. The final output is produced from 8 least significant bits of register Y which are shifted to the output buffer, forming the pseudorandom bit stream for the current iteration. To restore the registers' values for the next iteration, Y is then copied to X and Z. At the same time, zero is shifted into Y and the flag and carry registers are cleared. The above process requires a timing of $4.5n^2 + n$ clock cycles, where n is the size of the modules in bits, thus a frequency clock of 500 Mhz can be used, while each output is derived in a contagious cell of the random matrix.

4.4.3. Final Design

This work implemented a hardware architecture of a compressed sensing recovering algorithm as an educational compressed sensing theory exploration. As a result of this work generic FPGA blocks for matrix arithmetic operations and functions were developed.

Up to date, most of the work on Compressed Sensing was based on ideal operating conditions. However, in practice, various imperfections may occur, such as noise uncertainty, channel uncertainty, dynamic spectrum occupancy, and transceiver hardware imperfections like analog-to-digital converter errors, synchronization errors, etc. Therefore, it is a big challenge to further investigate the compressive sensing in the presence of practical imperfections. This work only focuses on current greedy algorithms and techniques and does not propose any further solution as only it uses these techniques to propose a theoretical functional core for Approximate Massage Parsing and Compressive Sensing techniques.

The brief analysis of Approximate Message Parsing shows that the recovery algorithms examined on this work are very well suited for hardware implementation. The implementation of architecture resulted from an Approximate Message Parsing algorithm high level design approach that is consisted of six different core modules on hardware level manners. The final design is capable of running on an 420 Mhz clock speed, while all core modules can separately run on higher clock speeds, the final complete architecture must remain on a safe clock speed that allows all modules to compute each result in safe and evaluated data. The final design is consisted of total $1.318 LUTs$ while using a total of 2.177 flip-flops. To compare the circuit sizes of hardware based implementation and the processor-based sensor node implementation, a Xilinx's soft-core CPU, *MicroBlaze*, is examined in FPGA as a reference design of processor-based sensor node. *MicroBlaze* is a 32-bit soft processor with RISC architecture.

The processor-based implementation includes a 32 Kbyte RAM. Since the FPGA logic capacity usage is directly related to its circuit size, it is fair to say that the circuit size of the hardware-based sensor node implementation is about four times larger than that of the *MicroBlaze* processor-based software implementation. It is a trade-off between performance and circuit size. The proposed event-driven architecture is not loaded into the *MicroBlaze* processor as our event-driven architecture is not processor compatible. IP core of *MicroBlaze* processor is loaded in our FPGA just to know how much FPGA resources it takes, as compared to our fully hardware based FPGA implementation of sensor nodes. The final core for our theoretical problem can run on this processor as an example with the following utilization.

Resource Type	**Total Resources Available**	**Hardware base design**	
Slices	13.312	*X*	*X*%
Slice flip-flops	26.624	2.177	8.18%
4 input LUTs	26.624	1.318	4.927%
bonded IOBs	221	*Up to 36*	16%
GCLKs	8	2	25%

The proposed architecture uses complex-valued multipliers to attain the required throughput on a small area, thus implementing fast parallel generation of the measurement matrix coefficients from LUTs with strongly reduced size of a design. Also, exploiting similar structures in the correlation update matrix allows to efficiently implement a coefficient generator with high throughput.

5. Conclusions and Discussion

Compressed Sensing is a technique that allows signal reconstruction without meeting the Shannon and Nyquist theorem. However, the apparent defiance of the sampling criterion in a rated sampling is possible only under two special requirements:

- The presence of a sophisticated and intelligent sampling
- The signal to be acquired must be a sparse signal. If the target signal is not sparse, it can be made sparse by compressing it in any suitable bases. For example, a signal is dense in time domain, but the signal is sparse in frequency domain. Thus, the second requirement can be solved by finding suitable bases.

That standard in Compress Sensing must be extended to include a much richer class of signals, as signals that have low-dimensional structure, not necessarily represented by sparsity, and signals that can have arbitrary dimensions, not only finite-dimensional vectors. Also, it is certain that this technique will revolutionize one of the areas that benefit most from data compression, such as telecommunications, soon, as new and fast-growing field of applied mathematics that addresses the shortcomings of conventional signal compression.

Given a signal with a few nonzero coordinates relative to its dimension, compressed sensing seeks to reconstruct the signal from few non-adaptive linear measurements. In the theoretical design approach, with the hardware it has been further proven that sub-Nyquist sampling of signals is possible without losing much of the recovery performance. In the future, theoretical research might enrich different algorithms and methods which might be considered, instead of Approximate Message Parsing, to increase noise robustness and spatial resolution. Moreover, while the Compress Sensing reconstruction power for signal sources in the transformed domain is well recognized, there is very little literature discussing how to optimize the measurement matrix in a tractable manner and the performance bound for the algorithms process. Additionally, with a fixed number of samples, there is an essential need to know how many samples should be allocated for each band of the input to achieve the reconstruction with the least error. For one thing, it is confirmed that the advantages of the structured measurement matrix can provide theoretically a recovery performance of various reconstruction algorithms for natural sources

with a small mean square error. The key insight is that with the optimized sample allocation, the reconstruction error decays at the same rate for both Compress Sensing and the best linear reconstruction. Thus, the reconstruction gain of the Compress Sensing algorithms over the linear techniques is fundamentally limited.

In the recent years, the area of Compress Sensing has branched out too many new fronts and has worked its way into several application areas. As Compress Sensing is an exciting, rapidly growing field that has attracted considerable attention in signal processing, statistics, and computer science, as well as the broader scientific community. Since its initial development, only a few years ago, thousands of papers have appeared in this area, and hundreds of conferences, workshops, and special sessions have been dedicated to this growing research field. In this chapter, we have reviewed some of the basics of the theory underlying Compressed Sensing. A significant part of the recent work on Compress Sensing from the signal processing community can be classified into two major contribution areas. The first group consists of theory and applications related to Compress Sensing matrices that are not completely random and often considerably structured. This largely follows from efforts to model the way the samples are acquired in practice, which leads to sensing matrices that inherent their structure from the real world. The second group includes signal representations that exhibit structure beyond sparsity and broader classes of signals, such as continuous-time signals with infinite-dimensional representations. For many types of signals, such structure allows for more signal compression when leveraged on top of sparsity. Additionally, infinite-dimensional signal representations provide an important example of richer structure which clearly cannot be described using standard sparsity.

6. Design FPGA-based Compressive Sensing Nodes

In this section, we will present a reference related to the designs and implementations of sensor nodes that rely on FPGAs. About these designs we have many cases such as standalone or as a combination of microcontroller and FPGA.

6.1. FPGAs Features

Generally, about the architecture of these systems, the researching rule is that the design patterns with focus on the combination of microcontrollers and FPGA is very important. Using this specific architecture for FPGA-based Compressive Sensing Nodes we have many advantages such as Data Processing, Cryptography and Data Compression.

FPGAs architecture is vital for the operational part and the functionality related to the implementation in the Information Systems. Some main features:

1. The logical blocks and look-up tables (LUTs) are the main building blocks in the FPGAs architecture. FPGAs have an interconnection of logical blocks in the form of a bi-dimensional array.
2. These logical blocks consist of look-up tables (LUTs) constructed over simple memories for storing the Boolean functions.
3. Each LUT consists of a number of inputs. Build sequential circuits using multiplexor and Flip-Flop Hardware resources.
4. FPGA systems using Hardware Description Languages (HDL) such as VHDL and Verilog for programmable reason and the development of the Software.

Finally, the Application Specific Integrated Circuit (ASIC) is an alternative base technology to the FPGAs. ASIC is an Integrated Circuit which is customized for particular use. ASIC does not have the advantages of the FPGAs.

The FPGAs have many advantages in comparison to ASIC regarding the reprogramming capabilities for the Hardware manipulation and the operations. Some important advantages:

1. The platform is updatable with new releases and firmware software.
2. FPGAs include high-speed multipliers and adders. These cooperate at the highest supported frequency.
3. Nowadays, the cost is very important for the final Hardware selection. The FPGAs costs are much lower in comparison to ASICs.
4. The FPGAs have very good energy consumption compared to ASICs.

6.2. FPGAs Dynamic Reconfiguration

A main feature of FPGAs is the Dynamic Reconfiguration. It is a very important building block of the FPGAs architecture. The Dynamic Reconfiguration modifying the system at run-time in order to change the content of the logic blocks. This feature is related to the interconnection configuration. The basic operation is to allow the reconfiguration of either portions or the whole content of the FPGA.

The advantages of usability of Dynamic Reconfiguration are:

1. Minimizing of the circuit area through the Dynamic Reconfiguration.
2. The power consumption is very important and especially its reduction.
3. The Dynamic Reconfiguration is carried out on the Hardware Platform of the Hardware Vendors because they have different interfaces.

[34]

6.3. FPGA Vendors

Generally, the architecture of FPGAs is related to the ASIC prototyping. The main building block in the progress of the architecture is the Hardware Vendors. The Hardware Vendors through researching methods and from the experience in the Information Systems in the Network Platform considers the Hardware Platforms.

Main Hardware Vendors have focused on reducing energy consumption using optimized HIPs. In this area Hardware Vendors such as Xilinx, Altera and Actel based their platforms on the combination of HIPs and small LUTs of 4 and 6 inputs. In the following modules we will see a brief description related to the Hardware Platforms of the FPGAs Vendors.

[35]

6.3.1. FPGA Vendor Xilinx

Xilinx is a crucial Hardware Vendor in the area of the FPGAs development. Xilinx have focused on the development of reconfigurable platforms for analog signal processing. The FPGAs are based on reduced energy consumption. The main features are 28 nm and of High-K Metal Gate (HKMG) fabrication process. Generally, classifies their FPGAs in three series:

1. Xilinx Virtex (high-end),
2. Artix (low-power)and
3. Kintex (low-cost).

These platforms are synchronous with many advantages related to the performance. Main components are the 6 input LUTs, the powerful embedded blocks like 28 Gbps serial transceiver, 25 and 18 bits multipliers and the 48-bit accumulators. Also, very important FPGAs versions are the old-fashioned such as the Spartan-3, Spartan-3E and Spartan-6 with the combination of the low-cost and the powerful embedded blocks.

[35]

6.3.2. FPGA Vendor Altera

Altera is another important Hardware Vendor in the area of the FPGAs development. Altera has the similar actions with Xilinx. Altera provides FPGAs based on 28 nm, specially 28 Gbps serial transceivers and 8 input LUTs. The main feature is the high-speed transceivers.

Generally, Altera classifies their FPGAs in three series:

1. Stratix (high-performance),
2. Arria (mid-range)
3. Cyclone (low-cost).

Also, Altera like Xilinx supports three soft-cores (Nios II, Cold Fire and ARM Cortex M1) in order to adapt their FPGAs to SoC-oriented platforms.

[35]

6.3.3. FPGA Vendor Actel

Actel is another important Hardware Vendor in the area of the FPGAs development. Actel has the main focus on low-power and mixed-signal FPGAs. Actel architecture is mainly on FLASH memory instead of SRAM. Some of the main features are the limited number of writings and have reduced power consumption.

[35]

6.4. FPGA Categories based Sensor Nodes

In this section, we will describe and analyze the basic Sensor Nodes Categories through FPGAs. These basic categories are:

1. Commercial Sensor Nodes,
2. FPGA Standalone Sensor Nodes,
3. Sensor Nodes Based on Microcontroller and FPGA and
4. FPGA Coprocessors

6.4.1. Commercial Sensor Nodes

As regards the Commercial Sensor Nodes, two basic features are worth noticing. Firstly, the construction details and secondly, the active power. The Commercial Sensor Nodes Platforms are based on 8-bit (Mica, Mica2Dot andMica2) and 16-bit microcontrollers (MSP430) running at low frequency. Also, for the connection to external nodes they use peripherals such as ADCs and a 802.15.4/ZigBee transceiver.

Generally, the point is the impact of using FPGA-based nodes coupled with FPGA coprocessors. This combination is vital for the architecture of these platforms. FPGA coprocessors are a dynamic building block about the operations and the functionality related the Sensor Nodes Systems.

Important features are the Mote, the specific model from the Hardware Vendor, the Communication, the Total Active Power (mW).

For example some basic construction details about the Commercial Sensor Nodes in the platform Microcontroller:

1. The Mote “Mica” with ATMega103 @ 4 MHz with communication TR1000, ASK the Total Active Power (mW) is 27.
2. The Mote “Mica2Dot” with ATMega128 @ 7.4 MHz with communication CC1000,FSK the Total Active Power (mW) is 44.
3. The Mote “Tmote Sky” with TiMSP430 @ 8 MHz with communication CC2420, OQPSK the Total Active Power (mW) is 32.
4. The Mote “Imote2” with Intel PXA271 @ 13–400 MHz with communication CC2420, OQPSK the Total Active Power (mW) is 86.8.

[36], [37], [38], [39]

6.4.2. FPGA Standalone Sensor Nodes

FPGA Standalone Sensor Nodes is a category of the sensors over FPGAs. Crucial points are the FPGAs processing of sensor, the communication with an external transceiver the microcontroller to interface sensors and external transceivers as the commercial sensor nodes and the networking standard in WSNs with low-power.

Generally, key features are the Platform, the specific model from the Hardware Vendor, the Communication, the Power Consumption (mW), the Frequency (MHz), the Sensors and the Applications.

For example, some basic construction details about the FPGA Standalone Sensor Nodes are:

1. In the Platform Cyclone I (Nios-II) in the model EP1C6 the Communication is Bluetooth (RS-232) with no Power Consumption (mW), Frequency (MHz) Sensors and Applications.
2. In the Platform Cyclone II (OpenRisc) in the model EP2C70 the Communication is ZigBee (RS-232) with 221 Power Consumption (mW),no Frequency (MHz), Temperature Sensors and General Applications.
3. In the Platform Spartan-3 in the model XC3S2000 the Communication is ZigBee (XEMICS) with 700– 1,100Power Consumption (mW), 11 Frequency (MHz), no Sensors and General Applications.

[36], [37], [38], [39]

6.4.3. Sensor Nodes Based on Microcontroller and FPGA

Sensor Nodes Based on Microcontroller and FPGA is a category about sensors over FPGAs. Basic features are the power reduction implementing an FFT and image processing, and power consumption with algorithms in both microcontroller and FPGA.

Generally, the features are the Microcontroller the FPGA Platform the specific FPGA Model from Hardware Vendor, the Communication the Standby Power Consumption (mW), the FPGA Execution Power Consumption, the Microcontroller Execution Power Consumption and the Application.

Some basic construction details about the Sensor nodes based on combination of microcontroller and FPGA coprocessor:

1. Microcontroller ADUC841, with FPGA Platform Spartan-3 with FPGA Model XC3S200 with Communication ZigBee (ETRX2) with Standby Power Consumption (mW) 60, with no FPGA Execution Power Consumption (mW) with no Microcontroller Execution Power Consumption (mW) and General Application.
2. Microcontroller ATMega, with FPGA Platform IGLOO with FPGA Model AGL600with no Communication with Standby Power Consumption (mW) 35.97, with FPGA Execution Power Consumption (mW) 12.36 with Microcontroller Execution Power Consumption (mW) 382.94 and Multimedia/FFT Application.
3. Microcontroller SENTIO32 (AVR32), with FPGA Platform IGLOO with no FPGA Model with Communication ZigBee (CC2520) with no Standby Power Consumption (mW), with
   FPGA Execution Power Consumption (mW) 5.93 with Microcontroller Execution Power Consumption (mW) 77.5 and Multimedia/Image Application.
4. Microcontroller MSP430, with FPGA Platform IGLOO with FPGA Model AGL125with Communication ZigBee (CC2420) with no Standby Power Consumption (mW), with
   FPGA Execution Power Consumption (mW) 5.00 with Microcontroller Execution Power Consumption (mW) 20.00 and Multimedia/CRC32 Application.

[36], [37], [38], [39]

6.4.4. FPGA Coprocessors

FPGA Coprocessors is another category about sensors over FPGAs. Basic features are the low-cost FPGAs, decent power consumption and high frequency.

Its features are the Platform the specific FPGA Model from Hardware Vendor, the Frequency (MHz), the Power Consumption (mW) and the Application.

For example some basic construction details about the FPGA Coprocessors:

1. The Platform Spartan-3, the Model XC3S5O the Frequency (MHz) is 28, the Power Consumption (mW) is 38 and General / Routing Acceleration Application.
2. The Platform Virtex-5, the Model XC5VLX330T the Frequency (MHz) is 100, the Power Consumption (mW) is 364.85 and Multimedia / Loeffler DCT Application.
3. The Platform Cyclone I, the Model EP1C4F the Frequency (MHz) is 50, the Power Consumption (mW) is 98.92 and Security / AES Application.
4. The Platform IGLOO, the Model AGLN250V2 the Frequency (MHz) is 27, no Power Consumption (mW) and Security / ECC Application.

[36], [37], [38], [39]

7. Combination of Compressive Sensing Algorithm and FPGA

In this chapter we will analyze in depth the combination of the Compressive Sensing Algorithm and FPGA. The basic scenario is that the Information System produces online real time data and we need a mechanism that can receive uncompressed data and transmit compressed data at the output terminal.

In this specific scenario we have to describe the methods of creating dedicated hardware for the data transmission using a channel with compressed data.

7.1. Implementation Methods

About the Combination of Compressive Sensing Algorithm with the Hardware (FPGA) the basic issue is the data compression. The hardware should be capable of implementing the basic requirements for reprogramming. In the following paragraphs we will see the basic implementation methods.

From the following methods the second is very important as regards the Soft Core Processor on FPGA. The main advantage is that the end user and especially the Programmer and Developer have the opportunity of selecting the processor of the requirements using tools such as EDK (Embedded Development Kit) for the hardware description.

7.1.1. Using FPGA as a Hardware

The basic method is to implement the Hardware for the data compression algorithm using FPGA and algorithms both using HDL language. The HDL is the basic programming language to coding the FPGA. HDL is used to describe the structure and behavior of electronic circuits and especially digital logic circuits.
[40]

7.1.2. Using Soft Core Processor on FPGA

Another implementation schema is the development of soft-core processor hardware to implement the software algorithm on this processor. For example the Hardware Vendor Xilinx with the production of Programmable Devices such as FPGAs & 3D ICs has the EDK tool kit. This tool is very important because it allows us to change the software algorithm without changing the hardware description. The programming of the device has the additional opportunity to change the compression technique without making any difference to the existing hardware.
[45].

Also, a feature of great importance is that Xilinx offers a comprehensive multi-node portfolio. This portfolio is related to address the requirements across a wide set of applications. We can design a state-of-the art and high-performance networking application using the highest capacity and performance with low-cost. All Xilinx FPGAs and 3D ICs are Programmable and provide the system integration with optimization and performance.
[40]

7.1.3. Using an available hardcore processor

Another implementation schema is the already available processors like the ARM processor. In this category we must implement the algorithm using programming languages like C or C++ on the processor hardware. C programming language is the core of the programming languages, it consists of libraries and features for better management of the hardware. C++ is the Object Oriented version of the C programming language.

In this category the algorithm for data compression can be implemented on this processor by writing the source code in the C code or C++ programming language. The Developers have important tools such as the SDK (Software Development Kit) for better programming.
[40]

7.2. Soft Core Processor

From the previous methods, the Soft Core Processor on FPGA is the most important. The basic advantage is that the user has the opportunity of selecting the processor using tools such as EDK (Embedded Development Kit) for the hardware description.

FPGA is an integrated circuit. FPGA has the feature that is designed to be programmable and configured by the user-developer after the manufacturing. The user-developer may implement many customizations using the appropriate software for programming the FPGA. About the FPGA configuration, it is generally specified using a hardware description language (HDL). Using FPGA we can implement any logical procedure and algorithm. We can update the functionality of the hardware using the partial reconfiguration of a portion of the hardware. FPGAs have programmable logic components called logic blocks. These blocks are very important for operation reasons using hierarchy of reconfigurable interconnection.

The programming of the processor architecture and the behavior are programmable, using the hardware description language (soft core processor). We have the opportunity to compile any application specific Integrated Circuit, in our case the FPGA devices. Using the soft core processor we have many advantages:

1. Soft core processors can be customized.
2. About the programming of the FPGAs, we have an abstraction level; the architecture and behavior are described using an HDL.
3. Obsolescence mitigation.
4. Component and cost reduction.
5. Hardware acceleration.

Crucial factors are the Peripherals and memory controllers. To implement and embed applications and generally to code to the soft core processor on FPGA, we need extensive libraries of intellectual property (IP) in the form of peripherals and memory controllers.

About the Hardware Vendors and especially the Xilinx's family FPGA, an important product is the MicroBlaze. It is a microprocessor Programmable Gate Arrays with EDK software tool. This processor is a virtual microprocessor. The basic architecture is that of combining the blocks of code called cores inside a Xilinx FPGA. Also, architectural features are the 32-bit Harvard RISC; it is optimized for implementation in Xilinx FPGAs and the processor having separated 32-bit instructions and data buses.
[40]

7.3. Embedded Hardware Platform (EHP)

The basic advantage of Soft Core Processor on FPGA is that the user has the opportunity of selecting the processor using tools such as EDK (Embedded Development Kit) for the hardware description. So, we have the Embedded Hardware Platform (EHP), which is the basic environment for the development.

7.3.1. Embedded Development Kit (EDK)

The Hardware Vendor Xilinx provides the development suite called EDK. It is used for the design and development of processors on FPGA devices. This tool with the IDE for development, libraries and generally all the resources are used to create the hardware system. For example, the Microblaze soft core processor can be programmable with this suite.

7.3.2. Software Development Kit (SDK)

All the Software Vendors for the development have a Software Development Kit (SDK). The SDK is a part of EDK suit and we can use it for writing the Software. We have the Hardware and we want to write the appropriate software with configurations and operations, core parts and customizations to run on the processor. The SDK is the development suite for this purpose. In **(image 19)** we can see the hardware and software flow for developing the application to run to the processor of the FPGA.

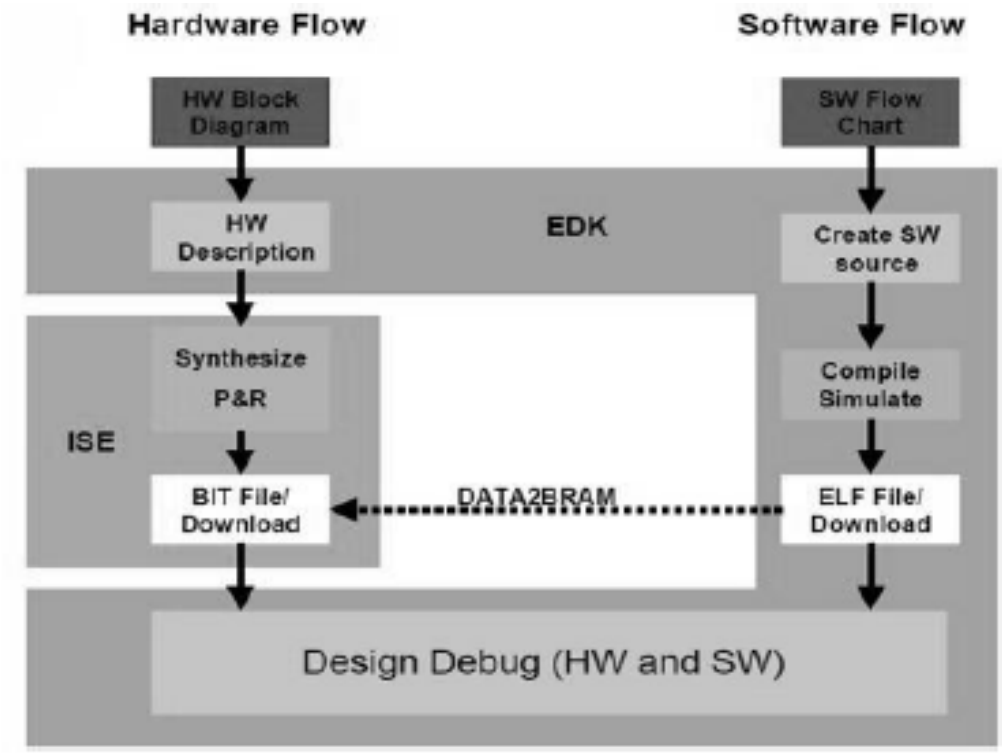


Image 19. Hardware/Software Flow for EDK/SDK

7.4. Final thoughts

The Soft Core Processor on FPGA is of significant value. We have the functionality of selecting the processor using tools such as EDK (Embedded Development Kit) for the development. Using the embedded systems for the development we have many opportunities and advantages for the applications especially for the data compression.

About the data compression, we require the implementation of various data compression algorithms. The best technique is to use FPGA (Field Programmable Gate Array) with softcore processor. The basic architecture is the classical; we show it in the **(Image 20)**. We have:

1. **The Input**, which is the data source transmits the data received by the Processor
2. **The Process**, which is the FPGA using data compression algorithms implemented in the processor and compresses the data
3. **The Output**, which is the compressed data received at the output port of the processor.

Business rules are:

1. The input for the data source may be any user transmitting bit stream data representing like text, audio, image or video etc.
2. The uncompressed data from the data source is received by the processor block. These uncompressed data are managed by the implemented data compression algorithms.
3. The compressed data are transmitted through the GPIO of the processor; an alternative is via the serial port.
4. Moreover, we can have other hardware interfaces for transmission such as Ethernet or Fiber Optics.
5. The data reception and transmission may also synchronous or asynchronous.

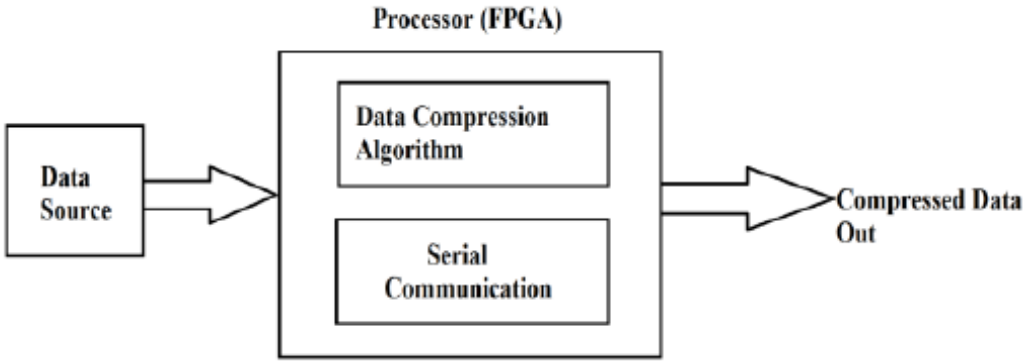


Image 20. Data Compression procedure

8. Compression Algorithms in reconfigurable part of FPGA

In this section we will see basic compression algorithms to implement in the reconfigurable part of the FPGAs.

Field programmable gate arrays (FPGAs) have an important advantage of dynamic reconfiguration. The crucial factor of dynamic reconfiguration is the ability to rapidly adapt to dynamic changes. Also another dynamic feature for FPGAs is that it needs better utilization of the programmable hardware resources for CROSS platform applications.

About the FPGAs configuration we have two basic parameters: First, we know that the multi-million gate equivalent FPGAs configuration time is increasing. Moreover, we know that the High reconfiguration cost can potentially wipe out any gains from dynamic reconfiguration.

In this environmental situation, a means of alleviating this problem is to exploit the high levels of redundancy in the configuration bit stream by compression. We need a configuration compression technique for exploiting the redundancies both within a configuration's bits stream. This configuration compression technique will be between bits streams of multiple configurations. By maximizing reuse, the proposed technique performs in high percentages better than the previously proposed techniques in the history of FPGAs.

8.1. FPGA-Based of the LZSS Compression Algorithm

8.1.1. Architecture and functionality

The LZSS compressor has a complex hardware architecture. It uses handshake interfaces for both input and output streams. This compressor consumes 32-bit word; the format has two values LSBF and MSBF and also produces D/L pairs which are used by a Huffman coder. The result is the production of a stream of packed 32-bit words.

The basic advantage is that the streaming usability and especially the stream interfaces allow connecting to high performance interfaces. Also, it permits and compresses real-time streaming data without separate buffering and compressing stages. In the (**Image 21)** we can see the overall structure.

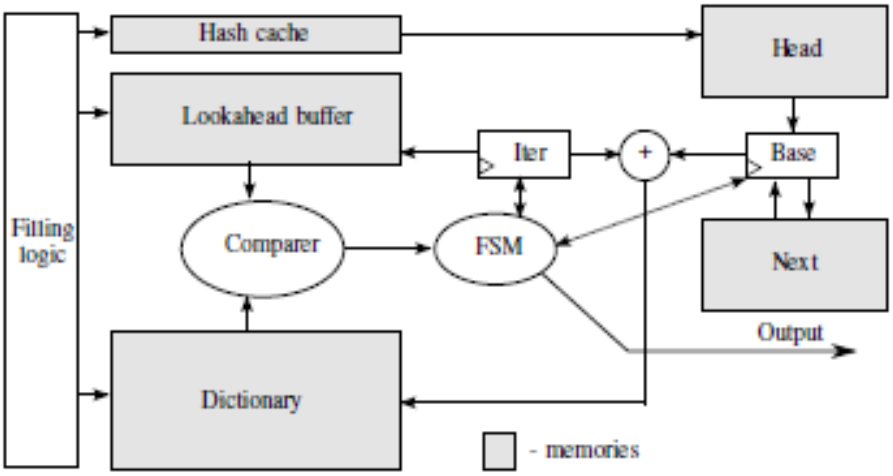


Image 21. Structure of the LZSS Compressor

We begin the analysis of the procedures with comparing the front of the uncompressed stream. This comparison is related to several offsets inside the dictionary to find the longest match. In order to speed up the comparison, we have two basic independently addressable ring buffers: the Look ahead buffer which contains the front of the input stream usually (up to 512 bytes) and the Dictionary which contains the last N bytes of the input stream that have just been processed.

Also, regarding the hash table structure, it is the same to the ZLib. In this case, we have independently addressable tables: firstly the Head table which contains the offset in the dictionary buffer of the last string and secondly the Next table which contains the relative offset of the previous string.

About the illustration of the high compression performance we have the following state flow of the main FSM:

1. The compressor waits until the look ahead buffer contains at least 262 bytes.
2. The value from the head table is used as the first string address, which means that it is routed to the next table to get the address of the next string. The hash value is the same.
3. In the next step we have the next clock cycle with the next table read in parallel.
4. After the previous steps we have the final production.

8.1.2. Results

In this paragraph we array some features and results from the evaluation for the LZSS compressor with focusing on the performance of the software implementation, providing the FPGA utilization information. About the tests, they are related with a 10MB and a 50MB fragments to factor out DMA setup time.

In the **(Image 22)** we can see the performance comparison related to the input and output streams.

Data sample	SW speed (MB/s)	HW sped (MB/s)	Speedup	Compression ratio
Wiki 50MB	3.15	47.57	15.06x	1.69
Wiki 10MB	3.11	44.05	14.15x	1.68
X2E 50MB	2.55	51.86	20.32x	1.7
X2E 10MB	2.54	47.84	18.78x	1.7

Image22. Performance Evaluation

In the **(Image 23)** we can see the FPGA utilization of lookup tables (LZSS + fixed-table Huffman). The result is that it remains insignificant.

Hash size	Dictionary size	LUTs	Registers
15 bits	32KB	2620	864
12 bits	8KB	2263	817
9 bits	4KB	2105	775
Available in XC5VFX70T FPGA		44800	44800

Image 23. FPGA Utilization

In the **(Image 24)** we can see that increasing the dictionary size improves the compression ratio.

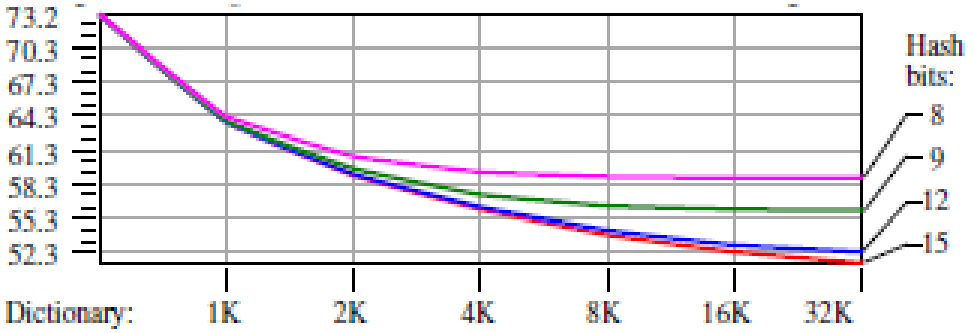


Image 24. Compressed size in MB of a 100MB

In the **(Image 25)** we can see that increasing the hash size reduces the amount of the matching iterations.

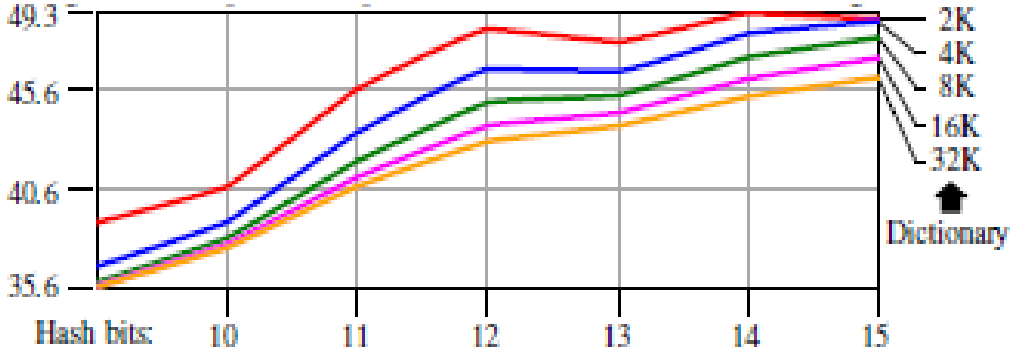


Image 25. Compression speed (MB/s) for a 100MB Wiki fragment

8.1.3. Thesis

In this sub-section we have presented the high-performance flexible implementation of the LZSS algorithm on a FPGA. Basic features are the independently addressable dual-port block RAMs, the performing of several specific FSM, and the data structure optimization. The result from this implementation is a 15- 20x performance increase comparing to the optimized software implementation.

8.2. Reconfigurable Computing Platforms

The Compression Algorithms in the reconfigurable part of FPGA is very important for the development of the computing platforms. Generally, the Reconfigurable computing has been an emerging technology. It has the possibility of developing such systems for space that becomes exploited. The basic idea of reconfigurable computing the future design of hardware of fixing a hardware bug or updating a hardware-implemented algorithm has never been available. Also, it's very important that we have the option of reconfiguring a system once launched that provides numerous advantages and a great deal of flexibility.

The basic scope of the computing engines is that they were designed with processor with software and a general architecture in order to achieve the desired function. The FPGAs have two important factors: the speed and the gate count have increased. Also, the computing engines such as peripherals can be designed directly into these devices.

Generally, for the developers, the process to reprogramming the devices and the notion of creating reconfigurable platforms for onboard processing are very important. The reprogramming of the FPGSs devices brings about many advantages and possibilities: reduced size and complexity, reduced design cycle times, resource sharing.

8.2.1. The advantages and flexibility in Reconfigurable Platforms

Relative to the development, the long design cycles and the high program risk are disadvantages. The Electronics plackets cannot be repaired, it fixes in many cases which are possible, but this action is more expensive than discovering the problem on the pre-launch stage.

An advantage of the usability of reconfigurable platforms is to have the opportunities to reduce the long design cycles and mitigate the risk by exploiting its flexibility.

Another advantage of reconfigurable platforms is that in the FPGAs we have performing upgrades to existing FPGA designs on-orbit. It is very important because it is more efficient FPGA design and the system can be upgraded with a newer version.

Furthermore, we have new orientations or structures being moved around, and the circuit requires a faster design for the critical application. In the past, these situations could never have been fixed, but nowadays using the reconfigurable computing platforms they can now be implemented with upgrades.

A final advantage is that the reconfigurable system is that the option exists to change the entire FPGAs contents. A classic example in this case is that an FPGA is designed to provide JPEG compression as part of its data transfer capabilities. In this case, it is possible to reconfigure only this module should the user wish to take advantage of the JPEG2000 compression algorithm at a later date.

8.2.2. Architecture in Reconfigurable Platforms

The basic architecture of a reconfigurable system has the core reprogrammable FPGA. The Radiation-Hardened FPGAs are used to perform the mission-critical functions and the interfacing by performing all the low-level verification and maintenance of the reprogrammable FPGA.

The FPGAs maintain and verify the memories within the system. Also, additional system-specific circuits are usually required and interface in the reprogrammable FPGA. The FPGAs have core functions which are highly critical for the system.

The reconfigurable FPGA is generally loaded with user code to run higher-level functions and interfaces. A basic characteristic is that related to the larger I/O capability, they usually interact with the off-chip circuits.

Also, regarding the architecture, the Fault Tolerance is a crucial factor for the operations and the functionality. The effects of radiation in space are a significant challenge in adapting suitable the ground-based designs for use in orbit.

Also, relatively to the architecture, the Triple Module Redundancy (TMR) is a vital factor for the operations and the functionality. The TMR is a method used to enhance reliability by performing a majority vote on three identical circuits. The basic scope of the TMR process is to perform the vote in order to eliminate a potentially erroneous output and provide a correct single value. Also, TMR can be selectively used on critical modules using techniques such as the reducing of the amount of triplication required.

Also, as to the architecture, the Scrubbing is crucial for the operations and the functionality. Scrubbing is a method by which the FPGA undergoes a partial reconfiguration. This reconfiguration is related to the constant interval in order to minimize the probability of observing the effects of a Single Event Upset (SEU). The SEU is a radiation-induced error that normally causes a bit flip in memory-based devices. Using the reprogramming with refreshing

the contents of the memory, the device is essentially getting a new start removing any bit flips that may have been present.

8.2.3. The future of Reconfigurable Platforms

Nowadays, the world is online with many Information Systems and online services. Also, the networks are highly considerable elements for the Information Systems development. FPGAs in hardware section are very important for many applications. Using FPGAs we have many solutions for business problems in a vast amount of different business sectors. Nevertheless, in the Information Systems, the parameters Performance / Tuning are remarkable for the operations and the functionality. In this direction, the Compression Algorithms in reconfigurable part of FPGA is the key for the performance in these cases. While reconfigurable platforms offer significant advantages, the challenges mentioned herein need to be overcome before reaping the benefits. The Reconfigurable Platforms is very important issue for the future of the Information Technology (IT).

9. Outcomes

1. **Wireless sensor networks (WSNs) Constraints**

   The Wireless sensor networks (WSNs) have many resource constraints. The most important ones are the limited power supply, the bandwidth for communication, the processing speed and the memory space.

2. **Data Compression (Compressive Sensing)**

   The Data Compression is a very important procedure for the Sensor Network. Its scope is to reduce the power consumption of the system. The Data Processing consumes much less power than transmitting data in wireless. This result is effective to apply data compression before transmitting data for reducing total power consumption by a sensor node.

3. **Compressive Sensing Crucial Factors**

   We have some crucial factors about the sensing:

   - the sensor nodes deployed in different areas need continuous power supply to meet its purpose
   - it processes huge amount of data and this is one of the main factors that consume more energy and resources
   - what consumes more power in the activity of the WSN is the data transmission; the sensor node power is consumed in transmitting the data.

4. **Compressive Sensing (Data Compression Algorithms)**

   Compressive Sensing has compression algorithms, which have been specifically designed for WSNs. These important algorithms are:

   - Coding by Ordering
   - Pipelined In-Network Compression
   - Low-Complexity Video Compression
   - Distributed Compression.

5. **Simulation of Compressive Sensing Algorithm**

Simulation is a considerable procedure for the algorithm functionality. The Data transmission consumes most of the energy in the WSNs. In the Sensor Networks the energy is the most important component for the operation part. The reduction of data transmission over the network is of primary importance.

6. **Dynamic Reconfiguration**

The future issue and the crucial factor for the FPGAs Platform is the Dynamic Reconfiguration. The basic role of Dynamic Reconfiguration is to modifying the system at run-time in order to change the content of the logic blocks. The advantages of usability of Dynamic Reconfiguration are very semantic for the architecture and the implementation in the Information Systems. This factor is very important as regards the construction of sensor nodes.

7. **Power Consumption and Cost**

Important parameters are the power consumption and the cost. Many proposal platforms, related to the Networks that are based to these architectures, have focused on these parameters. The good performance, the capabilities and possibilities are connected to the architecture, the power consumption and the cost.

8. **FPGAs**

Field-programmable gate arrays (FPGAs) are reprogrammable silicon chips. The basic operation and functionality is that using prebuilt logic blocks and programmable routing resources, we can configure these chips to implement custom hardware functionality. We can develop digital computing tasks in software and compile them down to a configuration file. Also, an alternative could be the use of a bit stream that contains information on how the components should be wired together.

9. FPGAs Advantages

FPGAs have many significant advantages, some of which are:

- FPGAs provide the hardware-timed speed and reliability
- FPGAs do not require high volumes to justify the large upfront expense of custom ASIC design
- FPGAs' Reprogrammable silicon also has the same flexibility of software running on a processor-based system
- FPGAs are not limited by the number of processing cores available
- FPGAs unlike processors, are truly parallel in nature
- FPGAs' performance of one part of the application is not affected when we add more processing.

10. Hardware Vendors

Main Hardware Vendors have focused on reducing energy consumption using optimized HIPs. In this area Hardware Vendors such as Xilinx, Altera and Actel have based their platforms to the combination of HIPs and small LUTs of 4 and 6 inputs.

11. FPGAs architecture

FPGAs architecture is very important for the operational part and the functionality related the implementation in the Information Systems. Main features are:

- The logical blocks and look-up tables (LUTs) are the main building blocks in the FPGAs architecture. FPGAs have an interconnection of logical blocks in the form of a bi-dimensional array.
- These logical blocks consist of look-up tables (LUTs) constructed over simple memories for storing the Boolean functions.
- Each LUT consist of a number of inputs. It builds sequential circuits by using multiplexor and Flip-Flop Hardware resources.
- FPGA systems use Hardware Description Languages (HDL) such as VHDL and Verilog for programmable reason and the development of the Software.

12. FPGA Categories based Sensor Nodes

The basic Sensor Nodes Categories through FPGAs are:

- Commercial Sensor Nodes,
- FPGA Standalone Sensor Nodes,
- Sensor Nodes Based on Microcontroller and FPGA and
- FPGA Coprocessors

13. Sensor Nodes Categories through FPGAs Comparison

The basic categories are the standalone, combinations of microcontrollers and FPGAs and the FPGA Coprocessors. A crucial point is that the execution of algorithms was faster on the FPGA in comparison to the microcontroller, and that the power consumption was also reduced. Generally, in the future the microcontrollers and FPGA co-processor is promising related to the parameter “low-power”.

14. Sensor Network Challenges

Generally, regarding the FPGAs based in the sensor networks, the technology has many challenges. Linking challenges in sensor networks with FPGAs in the Information Technology (IT) makes it highly possible to construct and initiate new Information Systems (IS) with high performance capabilities.

10. Open issues, Challenges and Future Research

Following the previous module on the Empirical Results from the FPGAs analysis, we discovered several interesting issues. We have seen and compared three basic categories of sensor nodes. These categories are the standalone, combinations of microcontrollers and FPGAs and the FPGA Coprocessors. Crucial points from these categories are:

- the standalone category consumed less in comparison to commercial sensor nodes
- the execution of algorithms was faster on the FPGA in comparison to the microcontroller
- the power consumption was also reduced.

Generally, in the future the microcontrollers and FPGA coprocessor are promising relative to the parameter “low-power”.

About the future research, the crucial factor for the FPGAs Platform is the Dynamic Reconfiguration. The basic role of Dynamic Reconfiguration is the modification of the system at run-time in order to change the content of the logic blocks. The advantages of the usability of Dynamic Reconfiguration are very semantic for the architecture and the implementation in the Information Systems. This factor is very important related to the construction of sensor nodes. A basic parameter is whether the costs and power consumption are reduced.

About the Open issues and the Challenges, the basic parameters are the power consumption and the cost. The area of researching nodes is in order to this issue. Many proposal platforms related to the Networks based to these architectures, have focused on these parameters. The good performance, the capabilities and possibilities are connected to the architecture, the power consumption and the cost. This is the basic area for research in the future.

11 Conclusion

In this Thesis the basic area was the Wireless Sensor Network (WSNs) and FPGA Technology. The WSNs are implemented on many crucial Information Systems related to Data Management. An issue of great significance was the Compressive Sensing (CS) for Wireless Sensor Networks (WSNs) related to the Data Compression and Data Transmission to the Network.

Generally, in this thesis we have the following:

1. **Wireless Sensor Network (WSN) and FPGA Technology**

   The technology for sensing and control includes sensor arrays, electric and magnetic field sensors, seismic sensors, radio-wave frequency sensors, electro optic and infrared sensors, laser radars, and location and navigation sensors.

   Field-programmable gate array (FPGA) are reprogrammable silicon chips. The basic operation and functionality is that using prebuilt logic blocks and programmable routing resources, we can configure these chips to implement custom hardware functionality.

2. **Compressive Sensing (CS) for Wireless Sensor Networks (WSN)**

   The Data Compression scope is to reduce the power consumption of the system.

3. **Compressive Sensing Algorithms**

   Compressive Sensing involves compression algorithms which have been specifically designed for WSNs. These important algorithms are:

   - Coding by Ordering
   - Pipelined In-Network Compression
   - Low-Complexity Video Compression
   - Distributed Compression.

4. FPGA Categories based Sensor Nodes

The basic Sensor Nodes Categories through FPGAs are:

1. Commercial Sensor Nodes
2. FPGA Standalone Sensor Nodes
3. Sensor Nodes Based on Microcontroller and FPGA
4. FPGA Coprocessors

5. Design FPGA-based Compressive Sensing Nodes

From the Analysis we have many categories. Main categories are the standalone, combinations of microcontrollers and FPGAs and the FPGA Coprocessors.

Crucial is that the execution of algorithms was faster on the FPGA in comparison to the microcontroller, and the power consumption was also reduced. Generally, in the future the microcontrollers and FPGA coprocessor is promising related the parameter “low-power”.

6. Empirical Field

From the analysis in the specific area we delved into many subjects : the Wireless Sensor Networks (WSN), Wireless sensor networks (WSNs) Constraints, Data Compression (Compressive Sensing), Compressive Sensing Crucial Factors, Compressive Sensing (Data Compression Algorithms), Simulation of Compressive Sensing Algorithm, Dynamic Reconfiguration, Power Consumption and Cost, FPGAs, FPGAs Advantages, Hardware Vendors, FPGAs architecture, FPGA Categories base Sensor Nodes, Sensor Nodes Categories through FPGAs Comparison and Sensor Network Challenges.

7. Open issues, Challenges and Future Research

Finally, we have many open issues that constitute serious challenges for the future research.

First of all, while spending time researching on the above technologies we discovered that the design approach of Compressive Seeing scheme even on the theoretical level had to determine first the current algorithms and approaches of relevant algorithms. While there are a lot of academic and theoretical work developed over these algorithms, most of them are algorithms that rely on the fact that if the signal's highest frequency is less than half of the sampling rate, then the signal can be reconstructed perfectly by means of sinc interpolation. After understanding the base relations that a compressive scheme must apply, we started designing an architecture of a theoretical scheme which can match the mathematical model. Applying a theoretical mathematical model scheme on a working architecture and approach came up with a lot of difficulties since we had to design a scheme using a top down technique. Design an encoder and a decoder comes up to the fact that all modules of each component should communicate in a way that it is possible for such a system to exist in real world.

Although we designed a theoretical architecture system we tried to estimate the fraction in terms of speed and clock configuration of that system. That part was the most crucial and it changed the design process of the whole architecture a lot of times in order to produce soft core parts that could be combined with relevant working speeds.

Regarding the future research, we should place extreme emphasis on the Dynamic Reconfiguration of the FPGAs Platform. The basic role of this Dynamic Reconfiguration is to modify the system at run-time in order to change the content of the logic blocks.

As for the Open issues and the Challenges, the basic parameters are the power consumption and the cost. The good performance, the capabilities and possibilities are connected to the architecture, the power consumption and the cost. This is definitely the fundamental step for research and development in the future, on account of the ongoing desire-need for improvement of the aforementioned technology.

12. References

- **Bibliography**

[01] C. S. Raghavendra, K. M. Sivalingam, T. Znati Eds.,Wireless Sensor Networks, Kluwer
Academic, New York, 2004.

[02] E. Cayirci, R. Govindan, T. Znati, M. Srivastava, Editorial: ‘‘Wireless Sensor Networks,’’ Computer Networks: International Journal of Computer and Telecommunications Networking, Vol. 43, No. 4, Nov. 2003.

[03] N. Bulusu, S. Jha, Eds., Wireless Sensor Networks: A Systems Perspective, Artech House, Norwood, MA, 2004.

[04] M. Hatler, Wireless Sensor Networks: Mass Market Opportunities, One World Inc., San Diego, California, February 22, 2004.

[05] J. Adams, ‘‘Designing with 802.15.4 and ZigBee,’’ Industrial Wireless Applications Summit, San Diego, CA, Mar. 9, 2004.

[06] M. Welsh, D. Malan, B. Duncan, T. Fulford-Jones, S. Moulton, ‘‘Wireless Sensor Networks for Emergency Medical Care,’’ presented at the GE Global Research Conference, Harvard University and Boston University School of Medicine, Boston, MA, Mar. 8, 2004.

[07] M. Welsh, D. Myung, M. Gaynor, S. Moulton, ‘‘Resuscitation Monitoring with a Wireless Sensor Network,’’ American Heart Association, Resuscitation Science Symposium, Supplement to Circulation: Journal of the American Heart Association, Oct. 28, 2003.

[08] K. Akkaya, M. Younis, ‘‘A Survey on Routing Protocols for Wireless Sensor Networks,’’ Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD, Aug. 18, 2003.

[09] The Linley Group, A Guide to FPGAs for Communications, First Edition (July 2009).

[10] FPGAs for DSP (BDTI Industry Report), 2nd ed. (Berkeley Design Technology Inc., 2006).

[11] S. E. Wahlstrom, “Programmable logic arrays — cheaper by the millions,” Electronics, vol. 40, pp. 90–95, December 1967.

[12]Srisooksai, T., Keamarungsi, K., Lamsrichan, P., & Arak, A. (2012). Practical data compression in wireless sensor networks: A survey. Journal of Network and Computer Applications.

[13] Ko, J. G., Lu, C., Srivastava, J., Terzis, A., & Welsh, M. (2010). Wireless sensor networks for healthcare. Proceedings of the IEEE, 98(11), 1947-1960.

[14] Aminian, M., & Naji, H. R. (2013). A hospital healthcare monitoring system using wireless sensor networks. Journal of Health and Medical Informatics.

[15] Candes, E. J. & Wakin, M. B. (2008). An introduction to compressive sampling. IEEE Signal Processing Magazine, 25, 21-30.

[16] Baranuik, R. G., Cevher, V., Duarte, M. F., & Hegde, C. (2010). Model-based compressive sensing. IEEE Transactions on Information Theory, 56, 1982-2001.

[17] Mallt, S., & Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. IEEE Trans. on Signal Processing, 42, 3397-3415.

[18] Duarte-Carvajalino, J. M., & Sapiro, G. (2009). Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. IEEE Trans. on Image Processing, 18, 1395-1408.

[19] Hugel, M., Rauhut, H., & Strohmer, T. (2014). Remote sensing via L1-minimization. From http://rauhut.ins.uni-bonn.de/RadarCS.pdf

[20] Measuring power consumption of CC2530 with Z-stack. For http://www.ti.com/lit/an/swra292/swra292.pdf

[21]A. Septimus and R. Steinberg, “Compressive sampling hardware reconstruction,” in Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2010.

[22]E. J. Cand`es and M. B. Wakin, “An introduction to compressive sampling,” Signal Processing Magazine, IEEE, vol. 25, no. 2, pp. 21–30, 2008.

[23]M. Akc¸akaya and V. Tarokh, “A frame construction and a universal distortion bound for sparse representations,” IEEE Transactions on Signal Processing, vol. 56, no. 6,2008.

[24]W. Z. S. Zhiguo, “Method for prediction of multi-scale time series with wdesn [j],” Journal of Electronic Measurement and Instrument, vol. 10, p. 012, 2010.

[25]P. Boufounos and R. Baraniuk, “Quantization of sparse representations,” DTIC Document, Tech. Rep., 2007.

[26] W. U. Bajwa, J. Haupt, A. M. Sayeed, and R. Nowak, “Compressive wireless sensing,”
in Proc. IPSN’06, Nashville, TN, Apr. 2006.

[27] W. Wang, M. Garofalakis, and K. Ramchandran, “Distributed sparse random projections for refinable approximation,” in Proc. IPSN’07, Cambridge, MA, Apri 2007.

[28]Maleki A, Donoho DL (2009) Optimally tuned iterative thresholding algorithms for compressed sensing. arXiv:0909.0777.

[29]Jafarpour S, Xu W, Hassibi B, Calderbank AR (2008) Efficient and robust compressed
sensing using high-quality expander graphs. Comput Res Reposit, abs/0806.3802.

[30]D. Petrovic, R. C. Shah, K. Ramchandran, and J.Rabaey, “Data Funneling: Routing withAggregation and Compression for WirelessSensor Networks,” In Proceedings of First IEEEInternational Workshop on Sensor NetworkProtocols and Applications, May 2003.

[31]T. Arici, B. Gedik, Y. Altunbasak, and L. Liu,“PINCO: a Pipelined In-Network CompressionScheme for Data Collection in Wireless Sensor Networks,” In Proceedings of 12th InternationalConference on Computer Communications and Networks, October 2003.

[32]E. Magli, M. Mancin, and L Merello,“Low-Complexity Video Compression for Wireless Sensor Networks,” In Proceedings of 2003 International Conference on Multimedia and Expo, July 2003.

[33]J. Kusuma, L. Doherty, and K. Ramchandran,“Distributed Compression for Sensor Networks,” In Proceedings of 2001 InternationalConference on Image Processing, October 2001.

[34]Kuon, I.; Tessier, R.; Rose, J. FPGA Architecture: Survey and Challenges. Found. Trends Electron. Des. Autom. 2007.

[35]Zhang, X.; Heys, H.; Li, C. FPGA Implementation of Two Involutional Block Ciphers Targeted to Wireless Sensor Networks. In Proceedings of the 2011 6th International ICST Conference on Communications and Networking in China (CHINACOM), Harbin, China, 2011.

[36]Potdar, V.; Sharif, A.; Chang, E. Wireless Sensor Networks: A Survey. In Proceedings of the International Conference on Advanced Information Networking and Applications Workshops, Bradford, UK, 26–29, 2009.

[37]Kwok, T.T.O.; Kwok, Y.K. Computation and Energy Efficient Image Processing in Wireless Sensor Networks Based on Reconfigurable Computing. In Proceedings of the 2006 International Conference on Parallel Processing Workshops, Columbus, OH, USA, 14–18, 2006.

[38]Nahapetian, A.; Lombardo, P.; Acquaviva, A.; Benini, L.; Sarrafzadeh, M. Dynamic Reconfiguration in Sensor Networks with Regenerative Energy Sources. In Proceedings of the Design, Automation Test in Europe Conference and Exhibition, Nice, France, 16–20 2007.

[39]Pham, D.M.; Aziz, S. FPGA Architecture for Object Extraction in Wireless Multimedia Sensor Network. In Proceedings of the 2011 Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Adelaide, Australia, 6–9, 2011.

[40] Ian D. L. Anderson and Mohammed A. S. Khalid Jason G. Tong, "Soft-Core Processors for Embedded Systems," in 18th International Conference on Microelectronics, 2006.

[41] Ž. Jovanović, V. Milutinović: FPGA Accelerator for Floating-Point Matrix Multiplication

[42] Tai-Chi Lee, Mark White, and Michael Gubody : Matrix Multiplication on FPGA-Based Platform

[43]https://www.xilinx.com/html_docs/xilinx2016_4/sdsoc_doc/index.html?q=/html_docs/ xilinx2016_4/sdsoc_doc/topics/devices/con-sdsoc-arch-dsp48.html

[44] K.H. Tsoi, K.H. Leung and P.H.W. Leong : Compact FPGA-based True and Pseudo Random Number Generators

[45] http://www.xilinx.com/products/silicon-devices/fpga.html