Design of Voice-Controlled Intelligent Robot

V.B.Saambhavi

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Master of Technology

भारतीय प्रौधोगिकी संस्थान हैदराबाद
Indian Institute of Technology
Hyderabad

Department of Electrical Engineering

May 2012

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

Saambhavi.V.B.

(Signature)

V.B.SAAMBHAVI
(Name)

EE10M009
(Roll No.)

Approval Sheet

This Thesis entitled Design of Voice-Controlled Intelligent Robot by V.B.Saambhavi is approved for the degree of Master of Technology from IIT Hyderabad

[illegible] 25/5

(Dr. Vamsi Boppana) Examiner
CTO
Xilinx Hyderabad

[illegible]

(Dr. Amit Acharyya) Examiner
Dept. of Electrical Eng
IITH

[illegible]

(Prof.S.S.S.P.Rao) Adviser
Chief Advisor and Mentor
CMC Limited

[illegible]

(Dr. P. Rajalakshmi) Co-Adviser
Dept. of Electrical Eng
IITH

[illegible]

(Dr. Krishna Mohan) Chairman
Dept. of Computer Science Eng
IITH

Acknowledgements

It gives me immense pleasure in acknowledging my gratitude to Prof.S.S.S.P.Rao for his continuous support, encouragement and inspiration. My sincere gratitude to Dr.P.Rajalakshmi who helped me with the implementations through the discussions she had with me and for her unwavering support. I would like to appreciate and thank Dr.Asudeb Dutta for being a member in the committee and for giving valuable feedbacks and ideas.
A work is not finished without expressing the gratefulness towards those who made the work possible. Accordingly, I would like to thank Prof.Kavi Arya for providing the Fire Bird V robot for the experimentation. I would like to express my gratitude to Prof.Patrick Lysaght and Ms.Veena Kumar for providing the Atlys Spartan-6 FPGA development board. I owe the Xilinx-Hyderabad engineers for helping me out with the software tool in the initial days and for providing excellent feedbacks about the project. I sincerely thank Dr.Shiv Govind Singh for his unflinching moral support and inspiration. I am obliged to convey my gratitude to Ms.Usha Priyadharshini and the Digilent support team who helped me to get started with the design on Atlys board. I would also like to thank Mr.Santhanam for helping out with the electrical design.
Thanks to my parents and sister Ms.Madhangi for their complete love, patience, encouragement and support.

Dedication

To my Parents and all my Teachers, because of whom I have grown into what I am, today.

Abstract

This project aims at developing a robot on a field programmable gate array, which can perform tasks by taking the user's voice commands. The robot has two wheels for locomotion and IR sensors. It can also autonomously direct itself according to the obstacles coming in its path with the help of IR sensors. The objective is to make the robot to recognize a limited set of commands for giving directions (typically 4), and move accordingly. Mel-frequency cepstral coefficients are used as the features that represent the input voice-commands and dynamic time warping algorithm is implemented for the recognition of the commands. The entire voice-command recognition module has both hardware and software components. The software runs on MicroBlaze, a 32-bit soft-core processor from Xilinx. A hardware-software co-design of a voice-command feature extraction circuit is implemented and validated on Spartan-6 FPGA. It is also compared against a complete software-based implementation and a complete hardware-based design.

Contents

Chapter 1

Introduction

1.1 Robot and Robot-control

Robotics is the technology which deals with the design, construction, operation and application of robots, which are machines capable of carrying out actions autonomously [1]. Today robots are pervasive everywhere and have become an integral part of life. Some examples include industrial robots, military robots, mining robots, entertainment robots, domestic robots (cleaning floors), health care robots (performing surgeries) and autonomous navigators. A robot has 3 main functions which are listed below.

- **sensing**: The sensing is achieved with the help of sensors which collect the analog information from its surrounding environment. This information is sent to the processor to decide and accordingly manipulate the output control signals to the actuators. Various ways of sensing, like vision, touch and hearing, gives the robot more ways to interact with its surrounding.
- **processing**: The ability of a robot to make decisions in various complex situations depending upon its sensor values involves complex processing.
- **actuation** The actuation is realized either with the help of manipulators like grippers or hand, or through locomotion with wheels or legs.

There are various robots designed and constructed for various applications. Education and research is one of the important fields of application. Fire Bird V, shown in Fig. 1.1, is a robot falling under this domain. It is designed and manufactured by Indian Institute of Technology, Bombay and Nex Robotics, and has been used widely for research in various institutes across India. Fire Bird V is used as a case study for this project.
The sensing, processing and actuation abilities of robots in general with particular emphasis on Fire Bird V are discussed in the following sections.

1.1.1 Sensing

Sensors are vital components in building an autonomous robot. Sensors provide the means for a robot to perceive its surroundings. Integrating the robot with a variety of sensors tends to increase its capacity to get further comprehensive understanding of the world around it. There are a wide

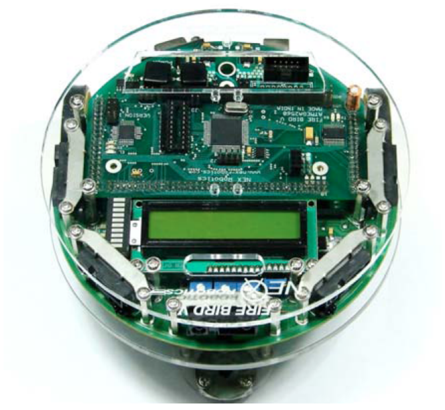


Figure 1.1: Fire Bird V

variety of sensors available which are capable of measuring distance, light, sound, temperature, speed, acceleration, etc. Some examples include infrared and ultrasonic sensors, proximity sensors, encoders, potentiometers, inclination sensors, gyroscopes, inertia measurement sensors, etc.
The robot Fire Bird V has three white line sensors for moving on traffic lanes, sharp GP2D12 IR range sensors for detecting objects between the range of distance of 10 cm to 80 cm, analog IR proximity sensors to detect objects that are less than 10 cm, position encoders to control the speed of motion, and battery voltage and current sensing for creating user alerts in case of insufficient battery voltage or current. These sensors (mentioned above) make the robot to sense different parameters and act according to that, making it totally autonomous.
There are also robots which are semi-autonomous, which can also act upon the commands from the users. When there is no command given, the robot behaves autonomously. For example, the robot Fire Bird V has wireless modules such as ZigBee communication and simplex infrared communication, and wired modules such as USB and RS232 serial communications through which the robot senses the instructions given to the robot.
Being able to give commands to the robot through one's voice is an interesting concept. For making the robot recognize the commands, a voice hearing/recording module should be included as one of the robot sensing capabilities. Incorporating these modules makes the robot resemble to human beings. Especially commands given through speech are a natural way of communication for a man. This creates the major objective for this project, which is to create a robot which can recognize and act accordingly to the voice-commands. Many other ways of commanding like touching and gesturing also exist.

1.1.2 Processing

Microcontroller, ASICs, Digital signal processors (DSPs) and Field programmable gate arrays (FPGA) are the major technologies used when designing a digital application. In the case of digital processing needed in robotics, microcontrollers are widely used as in the case of Fire Bird V, where Atmega2560 is used as the main controller. Microcontrollers (MCU) are general-purpose integrated chips that can be used for processing and control. They can be adapted to a wide variety of applications as the application running on the MCU is entirely software-based. Application development is restricted to software development and validation [2]. However when more complex signal processing algorithms need to be incorporated, it will not satisfy the needs as they are sequential and with limited resources. Hence, one of the other three technologies needs to be chosen.

Application Specific Integrated Circuit (ASIC)

An ASIC is custom-designed integrated circuit for a particular application. It might have one or more MCU or DSP cores, memory blocks such as ROM, RAM, EEPROM and flash depending upon the design requirements. An ASIC has optimized number of transistors and clock cycles. This optimizes the unit cost and power consumption. The development time is generally the highest compared to the other design platforms. Also the non-recurring engineering (NRE) cost of an ASIC is very high.

Digital Signal Processors (DSP)

DSP is a specialized microprocessor which hard-wires the basic functions of signal-processing algorithms for speed. Huge number of mathematical operations needs to be carried out rapidly and repetitively on a set of data in digital signal processing algorithms. A DSP provides lower-cost solution for better timing performance. In many a case, a DSP is a most favorable solution for only some signal processing applications however not all of the functions required of a broader application. This is because digital signal processing is hardly ever the only function required of a system. Many FPGAs include DSP slices, which enables for doing signal processing without the need for a dedicated DSP.

Field Programmable Gate Array (FPGA)

FPGAs are semiconductor circuits that can be configured any number of times for the desired functionalities after manufacturing. The field of FPGAs developed from programmable read-only memories and programmable logic devices. The PROMs and PLDs had the ability of being programmable after installation in a field. The first FPGA, XC2064 that was commercially established was invented by the Xilinx co-founders Ross Freeman and Bernard Vonderschmitt. XC2064 had configurable logic blocks and 3-input lookup tables. FPGAs after further development entered many applications such as in telecommunications, networking, consumer and automotive related areas. The NRE costs of a FPGA are negligible. The re-programmability of FPGAs provides a measure of flexibility that forms the major advantage over ASICs [3]. FPGAs have lesser time to market and long product life-cycle.

By developing functional robotic prototypes with FPGAs, ideas and algorithms with various I/O combinations can be effectively tested. Also the parallel nature of an FPGA design makes it an ideal candidate for the type of architecture in robotics where a variety of I/O components need be brought into and out of the device independently [4], whereas in a MCU all the I/O components fight for same resources. Xilinx is a leading provider of PLDs. Spartan is one of its major product families meant for low-power, low-cost and high-volume applications. The Spartan-6 is the latest in the family with the XC6SLX45 device having 43,661 slices and 58 DSP slices [5]. This FPGA is chosen as the implementation platform for this project.

1.1.3 Actuation

Actuators in robots convert stored energy into movement. The most popular actuators are electric motors that spin a wheel or gear, and linear actuators that control industrial robots in factories. There are also some recent advances in alternative types of actuators, powered by electricity, chemicals, or compressed air.
Fire Bird V has two DC geared motors in differential drive configuration and a castor wheel at the front for support. It also has two position encoders present at the wheels to sense the speed of motion of the robot.

1.2 Problem Statement

Increasing the capabilities of Fire Bird V being the major aim of project, we decided to incorporate an additional sensing module for the robot. Speech is an inherent way of communication for man. Hence, a speech recognition module to recognize its user's voice-commands is proposed to be incorporated in the robot for this project. Also the Atmega2560 microcontroller in Fire Bird V is removed and instead of that, hardware control circuits for the peripherals are built individually on the FPGA to bring about parallelism in the robot's actions.
The implementation of complex speech-signal processing algorithms on FPGAs is a better choice when compared to DSPs and ASICs. ASICs are very costly for the initial prototyping of robots. DSPs have dedicated multiplier blocks and it might be efficient for implementation of signal processing algorithms. However for the implementation of other functions of the robot, it is not very optimal. For example, many peripherals will have to compete for the same resources of the processor in a robot implemented on a DSP. In contrast, implementing on a FPGA creates parallelism with which peripherals can operate independently. Also the hardware DSP slices can be made use of for implementing the signal processing algorithms on FPGA.
Spartan-6 FPGA of Xilinx is chosen as the design platform because it is suitable for high-volume logic designs with DSP algorithms. Notably Spartan-6 XC6SLX45 has 43,661 logic cells and 58 DSP48A1 slices among other major features. The Atlys development board manufactured by Digilent Inc. has AC-97 codec with port available for connecting microphone, a 16Mbyte flash, 128Mbyte DDR2 with 16-bit data, USB ports for programming and expansion connectors for connecting a varied range of peripherals. The MicroBlaze, the 32-bit soft-core processor of Xilinx is chosen for the central controlling unit.
For the recognition of the voice-commands, a speech recognition module needs to be designed with a limited set of four words as the vocabulary. The speech recognition module is designed to be speaker-dependent that is the robot will be able to have maximum recognition rate with only one user.
Also some of the peripherals of the Fire Bird, namely locomotion units and IR sensors are to be incorporated with the robot on the FPGA with its embedded processor.

1.2.1 Objectives

The main objectives of the project are summarized below.

- To enhance the capabilities of Fire Bird V by implementing its controller architecture on a FPGA. This is accomplished by incorporating parallelism in the peripheral control circuits.
- To increase the sensing capability of the robot. This is done by including a voice-command recognition module on the FPGA. These voice-commands are processed and accordingly appropriate actions are taken by the robot.

1.3 Thesis Organization

The whole report is organized as follows: Chapter 2 presents an overview of the system architecture. Chapter 3 discusses about the design of the system in detail. Chapter 4 discusses the implementation results of the design and also testing results of the final system. Chapter 5 concludes the report with a brief listing of future work.

Chapter 2

Overview of the System Architecture

2.1 Review of Speech Recognition

Speech recognition is the technology by which a machine converts the speech signals into commands by recognition. The speech recognition systems in the literature can be classified based on the factors such as the vocabulary size, speaker population, speaking conditions, etc. The vocabulary size of the systems may vary from a small set of words (as few as around ten words) to thousands of words. The systems are made either speaker-dependent (which means that the speaker population is one) or speaker-independent (which means that the system can recognize the speech of any individual). Some systems will be designed to work under noisy environments and some for closed room conditions. A different speaking condition for a speech-recognition system is whether the user is required to leave distinct gaps between the words. A system constructed with this condition is termed as an *isolated-word recognition system.* A system which is designed to recognize the words with natural word spacing is termed as *continuous recognition system.*
Some applications with a limited vocabulary and that offers speaker-independence are in telecommunications. Here a person can dial a number by just speaking the numbers distinctly. Another example is in automatic operator-assisted calls for banking, airlines, etc. Example of systems with speaker-dependence recognition is in IBM Personal Dictation System, where the computer applications can be controlled by talking to them in natural and continuous speech. Further examples include the speech-recognition applications in automotive speech recognition, home automation, hands-free computing, robotics, video games and many others.

2.1.1 Algorithms of Speech Recognition

The general procedure of speech recognition is first to extract the specific features of the speech and to apply recognition methods on these features to find the most probable word. The features can either be in the frequency or time domain. Some of the time domain features are based on the number of zero-crossings, average energy per frame and the like. The features based on frequency domain consider the frequency components present in a frame as the measure. The recognition methods can mainly be classified into two types, namely, pattern recognition and statistical modeling. The pattern recognition approach uses a set of known speech-feature templates to identify the word that was spoken. This approach is preferred for applications where only a limited set of words needs to be recognized. The statistical modeling uses the probability and mathematical functions to determine the most likely outcome from the training data. This approach is preferred for applications where the system needs to recognize a large number of words with speaker-independence.

2.1.2 Feature Extraction

This is the process of taking out utilizable linguistic information from an uttered speech signal. Short sections of the speech signal are isolated and given for processing. This processing is repeated for the entire duration of the waveform. The result of this operation is a new sequence of features along the time axis, representing the speech signal [6]. For speech recognition, the most frequently used features are Mel-scale frequency cepstral coefficient (MFCC). MFCC takes observation sensitivity of human ear at different frequencies, and hence, is appropriate for speech recognition. This voice-command recognition module implemented for this project uses the MFCC to represent the features of the commands. The step-by-step calculation of MFCC is shown as a block diagram in Fig. 2.1 and is explained as follows.

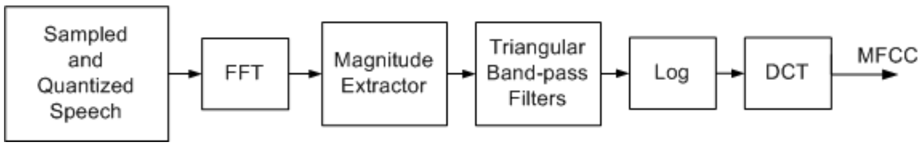


Figure 2.1: Feature Extraction - Block Diagram

- The speech signal is sampled and quantized.
- Pre-emphasis - The speech samples are sent through a high-pass filter to amplify the frequencies above 1 KHz in the spectrum because hearing is more perceptive in this region [7]
- Frame blocking: The speech samples are blocked into frames of N samples (amounting to a time period of 10-30 ms) with an overlap of some samples between frames. This amount of overlap captures the parameters that change from frame to frame.
- Fast Fourier Transform (FFT): FFT is performed on each of the frames to obtain the magnitude values of the frequency response.
- Triangular Band-pass Filters: The magnitude frequency response is multiplied by a set of triangular band-pass filters to get the log energy value of each filter. The positions of these filters are evenly spaced along the Mel-frequency scale. Mel-frequency is related to the linear frequency f by the Eq.(2.1): [8]

$$Mel(f) = 1125 * ln(1 + f/700) \tag{2.1}$$

- Discrete cosine transform or DCT: The DCT is applied on the logarithm of the energy obtained from the triangular band pass filters. The result gives the Mel-scale cepstral coefficients.

DCT transforms the frequency domain into a time-like domain called quefrency domain. The obtained features are similar to cepstrum, thus it is referred to as the Mel-scale frequency cepstral coefficients, or MFCC.

2.1.3 Recognition

Hidden Markov Modeling (HMM) is one of the commonly used statistical modeling methods for recognizing speech. For every word that needs to be recognized, a HMM is built with the help of a number of training sequences. For recognizing an unknown word, initially the features are calculated. It is followed by calculation of likelihoods corresponding to all the HMM word models. The model which has the highest likelihood is considered to be the recognized word [9].
One of the pattern recognition algorithms used for speech recognition is dynamic time warping. Dynamic time warping algorithm is used to contrast two time series provided the two sequences are sampled at the same rate. The algorithm gives the similarity measure between the two sequences, by minimizing the effects of shifting and distortion in time by allowing flexible adaptation of time series in order to detect similar shapes [10].
The simple form of the algorithm is outlined as follows.

- Initially a distance matrix is constructed, which has values that characterize the pair-wise distance between each point in the two sequences.
- The algorithm employs this distance matrix to find a best possible warping path, which goes through the least values in the distance matrix.
- A cost function which is the sum of all the values along the warping path is used to compute the resemblance measure. This value is used to evaluate the similarity between the two sequences under examination.

The dynamic time warping algorithm is chosen for this project due to the following reasons:

- Its architecture is easy to design
- The algorithm is faster

For small vocabulary systems, this algorithm is better suited compared to other recognition algorithms.

2.2 Speech Recognition in Robotics

Speech recognition has been applied to a multitude of applications like command-control operations, hands-free operation of mobile devices, entertainment games like robot-soccer, and smart-home appliances. A brief review of similar works in literature is given below.

- Reference [11] presents a robot is designed for speech-control application. Two computers are used to control the robot, one within the robot, and another used as the operation platform. The two computers communicate through WLAN. A microphone is attached with the operation platform used for speech-command acquisition. The speech signal is processed to find the uttered word, and a control signal is transferred wirelessly to the computer in robot for appropriate action. The robot is also equipped with ultrasonic sensors to avoid obstacles. They are also used to figure the structure of the objects for manipulating them. Mel-frequency cepstral coefficients are used as the features of the speech command and dynamic time warping algorithm is used for the recognition algorithm. The algorithm works with 100% recognition rate in normal environment, and 85% (average) recognition rate in noisy environment.

- [12] presents an isolated-word recognition system is designed on a DSP platform employing the dynamic time warping algorithm for recognition. The authors have also proposed an improved version of dynamic time warping algorithm. It obtains higher recognition rate with less storage space requirement and faster processing speed compared with the conventional DTW. TI floating point DSP TMS320C6713 DSK has been used for the implementation of this work. The vocabulary size used is 10 words and 12-dimensional MFCC features are used for the representation of speech. A recognition rate of 97.3% is obtained with the improved DTW with a recognition time of 0.823ms.

- In [13], a graphical user interface is designed using MATLAB for recognizing English digits when spoken. This work uses MFCC and HMM techniques to get recognition rate of 79.5-99.5% in clean environment and 67-88% in noisy environment for isolated word recognition system.

This present work stands out from the rest of the literature due to the following reasons and its contributions:

- The present implementations of speech recognition algorithm are in pure software running on high performance processors. However, a high performance processor is expensive and hence, is not suitable for implementations in embedded appliances. Pure hardware designs have been presented in literatures that perform the speech recognition at high speeds. However, the amount of resources needed is high. An optimum design is presented in this work which uses moderate amount of resources and gives a tolerable amount of delay.

- Three types of design implementations of speech recognition algorithm are performed and compared against each other in this work. The advantages and disadvantages of each method are also presented.

- The work uses dynamic time warping algorithm for the recognition part. The speech recognition applications in literature aim for a large vocabulary of words, for which the HMM based recognition algorithm works well. However, the recognition accuracy depends upon how well the training is done. For most of the systems, the recognition percentage does not reach 100%. On the other hand, with respect to the DTW based recognition, although only a few words can be modeled for a particular speaker, there is a possibility of obtaining almost 100% recognition rate. Algorithms with higher recognition rate are favorable for embedded devices, since the user will not require the system to recognize the commands wrongly resulting in malfunction. Furthermore, embedded devices will have only a limited set of functions for which the DTW algorithm is appropriate. Although the HMM based recognition systems are shown to be performing better than DTW based systems in the literature, the complexity of the design for HMM based system is greater than the DTW based system [14]. Another point to be noted is, as the number of words decreases for the DTW based system its recognition rate increases. This means that for smaller vocabulary systems, the DTW algorithm gives better recognition rate than HMM based system. Hence, for command and control applications which need about 20 words in the vocabulary, the DTW based systems give better recognition rate and good real-time factor.

- The design platforms used for the speech recognition applications are mostly DSPs and computer. Implementing on a computer uses redundant gates and demands more power. Implementing on DSP is not efficient because the design is limited in performance by clock rate and is not appropriate for non-signal processing functions of robot. Hence, the FPGA is a suitable platform for implementing the prototypes for a robot involving many I/O peripherals that needs to be controlled in parallel.
- The robot is also designed to be flexible for adapting to any user's voice. Before deployment of the robot, the robot can be trained to take in the user's voice commands and generate the reference templates according to the particular user. This maintains the recognition rate at a higher percentage for a varied number of users.

2.3 Architecture of the Embedded Robotic Platform

The block-level architecture of the entire robotic system implemented in this project is shown in Fig. 2.2. This is the platform containing all the peripherals on which the robotic system is developed. The peripherals connected to the robot are as follows.

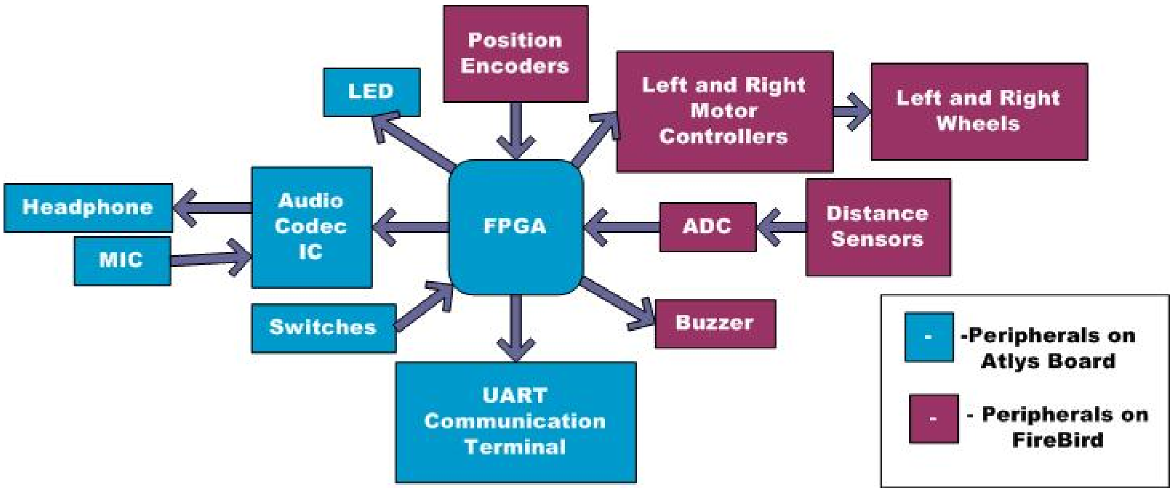

Position Encoders
LED
Left and Right Motor Controllers
Left and Right Wheels
Headphone
Audio Codec IC
FPGA
ADC
Distance Sensors
MIC
Switches
Buzzer
UART Communication Terminal
-Peripherals on Atlys Board
- Peripherals on FireBird

Figure 2.2: Block-level Architecture of the System

- **Microphone** to record the voice-commands of the user
- **Headphone** to hear the recorded sound for debugging purpose
- **LEDs and Buzzer** for indication purposes
- **Left and Right wheels** for locomotion

- **Position encoders** to determine the speed of motion
- **Switches** to take subjective inputs from the user
- **UART terminal** for decoding the system implemented on FPGA
- **Distance sensors** to avoid obstacles

As shown in Fig. 2.2, audio codec, LEDs, switches and UART communication terminal are connected to dedicated FPGA pins on the Atlys board. The remaining peripherals are connected through the VHDC connector available on the Atlys board.

2.4 The Project Design Flow

The project has two major parts. The first part is the design of the robot platform incorporating the locomotion unit and the sensors, and the second part is the design of the voice-command recognition system on the robot.
In the first part, the controller circuits are designed for the locomotion and sensing capabilities of the robot. The major aim in designing the controller circuits is to bring out parallelism in the operations of the peripherals. Hence, each controller circuit has inbuilt decision making capabilities that are independent of the other controller circuits. For example, the obstacle detection module functions in parallel with the locomotion control unit. So the robot keeps observing the environment for obstacle intrusion while it changes the direction of motion.
The issue of designing the voice-command recognition module in the second part was initially approached with the aim of implementing a working model of the algorithm on FPGA in complete software approach. This makes it possible to make any modifications algorithmically and analyze the results efficiently. Furthermore, the deployment of the system also becomes faster. Hence, a soft-core processor was chosen for this application and an embedded processor platform was created for implementing the software-based voice-command recognition module. However the time taken by the software routines to do the complex digital speech signal processing is high. Such high computation time makes recognition of voice-command in real-time difficult, unless a high performance processor is used. Implementing the entire algorithm in hardware will increase the response rate. Even though this approach gives good timing performance, there are other disadvantages. They are

- The resources needed for implementation of the algorithm is high.
- Separate hardware designs must be made for different embedded applications as the vocabulary for every kind of application varies.
- If further additional capabilities are to be added, the dedicated architecture cannot be modified to include them as they are designed only for speech recognition application.

The design methodology followed in this work, which included the hardware design approach evolved as depicted below. It was deliberated to improve the timing performance of the system. Hence, some of the computationally intensive parts of the software are translated into custom designed hardware blocks implemented on the FPGA fabric. Inclusion of these hardware blocks were found to speed up the execution of the algorithm. These blocks are incorporated into the embedded platform as

peripherals which can be accessed by the processor.
An optimization between the pure hardware-based and pure software-based approaches is to include both the hardware and software in the system and partition the tasks such that the resources are optimized and the delay is not compromised a lot. This type of design is called as hardware-software co-design. This has the combined advantage of being adaptable and fast enough. Also the area needed would be lesser than the pure hardware implementation.
Very few works in the literature discuss the co-designs of the speech recognition algorithm. For example, in [15], the feature extraction is implemented using software run on MicroBlaze with a hardware multiplier incorporated to avoid the multiplication through software routines. The work claims that the MicroBlaze with the custom hardware multiplier block helps to accelerate the speed performance of the system. The design works at 100 MHz and a total computation time of 1307.183us is reported for computing the features for one frame. However the works do not analyze the design factors enough to prove that a optimal design is implemented.
This project discusses the three types of implementations of the feature extraction circuit and brings out the comparison between the pure software-based implementation, the co-design implementation with both hardware and software and the pure-hardware based design.

Chapter 3

Complete Design of the System

3.1 Design of the Hardware Platform

The hardware platform of an embedded system constitutes the processor and the peripheral controller circuits attached to the processor through its processor bus. MicroBlaze, a 32-bit soft-core processor from Xilinx with its versatile architecture is chosen as the central controller unit for the robotic system.

3.1.1 MicroBlaze and its Bus Structures

MicroBlaze

The MicroBlaze has Reduced Instruction Set Computing (RISC) type architecture. The processor is flexibly configured with the custom-designed user peripherals, memory and its interfaces. It is implemented in the Xilinx's FPGA fabric. The MicroBlaze is integrated with a local memory (implemented with Block RAM) where the program and data are stored. This interface between the local memory and the processor is through Instruction Local Memory Bus (ILMB), Data Local Memory Bus (DLMB) and Local Memory Bus (LMB) interface controllers.

CoreConnect

CoreConnect is a bus system used to link the processors with peripherals and memory. It has 3 levels of bus structures, namely,

- processor local bus (PLB)
- on-chip peripheral bus (OPB)
- device control register (DCR)

The PLB is used for connecting the processor to peripherals that need high-performance operations. Slow speed peripherals are connected through OPB. DCR acts as a control bus and connects all controllers and bridges. The main bus in the system is the PLB linking the processor with DDR2 memory controller, on-chip memory, and the user-designed controller circuits for the peripherals.

3.1.2 Design Flow of Hardware Platform

The hardware platform design is created with MicroBlaze on the Atlys Spartan-6 development board, employing its Base system builder (BSB) of the Xilinx Platform Studio (XPS). A processor clock frequency of 66.67 MHz and 8 KB of local memory (program and data combined) is set. The peripheral interfaces for UART communication and DDR2 memory interfacing are incorporated in the system using the PLB. After the configuration, XPS generates the embedded platform, which contains the processor interfaced with the local memory through the LMB bus and with UART and DDR2 memory controller through the PLB bus. The block diagram of the system so generated is shown in Fig. 3.1 . This structure forms the hardware part of the embedded system platform.

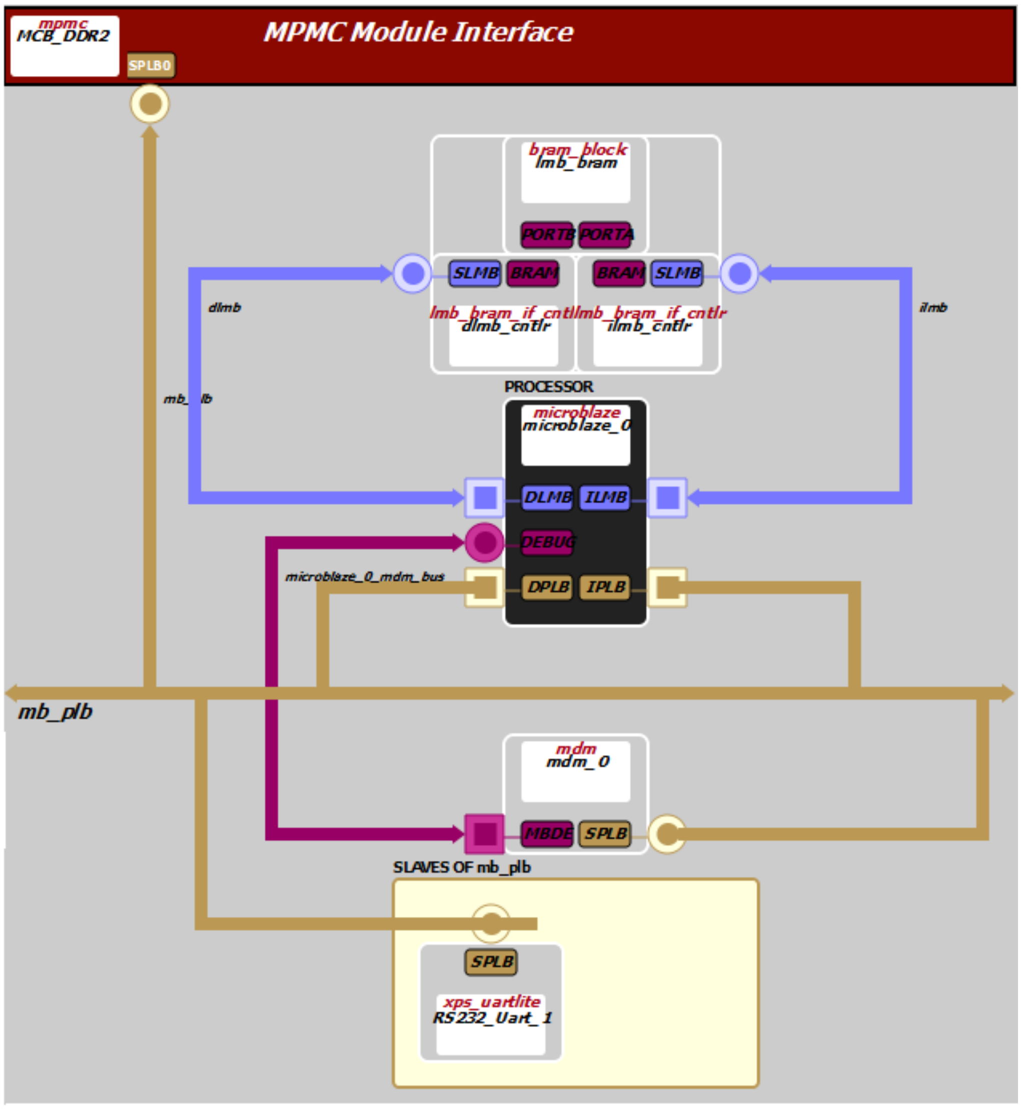


Figure 3.1: Block Diagram of the Embedded Platform Created by XPS

The 'Create and Import Peripheral Wizard' of XPS is used to create a general template of a peripheral module for the purpose of connecting the other controller circuits to the processor. This is generated as a slave, which is compliant with the PLB bus. Such templates with necessary characteristics are generated for each of the peripherals and modified later by the user to include their functionalities. These templates contain the interconnect structure, named IP Interconnect (IPIC). This helps in realizing the protocol for master/slave attachment with the PLB bus readily. The IPIC has an IP interface (IPIF) with which the user's logic is attached. The signals between the user's logic and the IPIF are made customizable. Memory structures like FIFO can be implemented along with the template and the functionality is taken care of by the IPIF module. Furthermore, this wizard helps in creating templates for the driver files to accomplish the software interfacing. After generation of these templates, the peripherals are customized and connected with the embedded platform through the PLB bus. All the peripherals so connected are memory-mapped and their addresses are generated. The final hardware platform created on the FPGA with the peripheral controller circuits is shown in Fig. 3.2 .

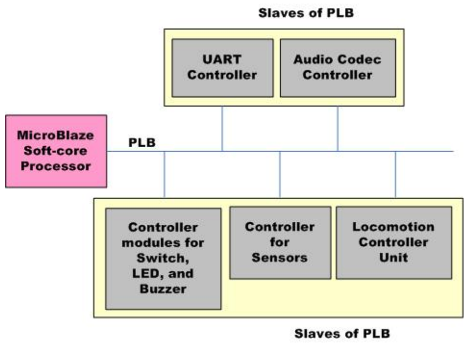

Slaves of PLB
UART Controller
Audio Codec Controller
MicroBlaze Soft-core Processor
PLB
Controller modules for Switch, LED, and Buzzer
Controller for Sensors
Locomotion Controller Unit
Slaves of PLB

Figure 3.2: Block Diagram of the Embedded Platform

The design methodology followed in the design of the controller circuits for the said peripherals are discussed in the subsequent sections.

3.1.3 Design of AC'97 Codec-Controller Interface

The LM4550 chip is interfaced with the system through a controller circuit built on the FPGA. The design of the interface is pictorially shown in Fig. 3.3. The controller and the LM4550 are connected by a digital interface consisting of 5 lines - a 12.288MHz bit-clock generated by the codec, two serial-data lines and a sync signal generated by the controller. The codec converts the analog speech signal into quantized digital samples, which are sent to the controller through the serial-data-in line, *SDI*. The controller takes in the serial-data and converts them into parallel data to be loaded into the user-registers. These user-registers are read by the processor to get the speech data. Every time a frame arrives, control signals are set by the control-circuit for the processor to read. Every frame contains a tag slot (containing flags showing the validity of the remaining slot fields of the frame) and twelve normal slots (containing the control bits and audio data). There are 256 bits sent or received per frame (16 bits in the Tag slot and 20 bits in each of the other 12 slots). LM4550 does not employ all the features of AC'97 codec and hence, not all the 12 slots are exploited. In LM4550, Slot 1 contains the address of the registers to be read/ written, Slot 2 contains the 16-bit data (MSBs) that is read / to be written into the register and Slot 3 and 4 contain the data from the left and right codec ADCs.

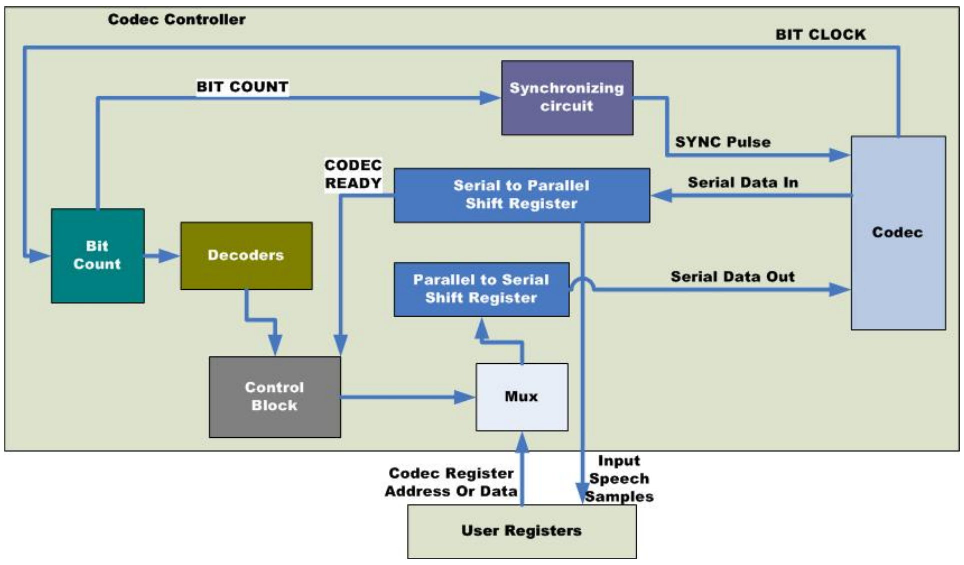


Figure 3.3: Block Diagram of the Codec-Controller Design

The codec has many registers, some of which are master volume register, line-in volume register, MIC-in volume register, PCM gain register, record source select register, etc. The sampling rate and the volume are set with the help of control information sent to the codec from the controller through the serial-data-out line *SDO*. Furthermore, the data that needs to be played through the headphone is sent through this line in the slots 3 and 4 of every frame. These control and headphone-out data are sent from the processor and are written into user-registers. These user-register data are taken in parallel and sent to the codec serially by the parallel-to-serial converter.
According to the bit-clock of the codec, the synchronization signal, *SYNC*, is generated by the controller circuit. This signal has a frequency of 48 KHz ($12,288MHz/256$). The signal is active high when the tag slot data is being sent or received and it is low in the other slot timings.

3.1.4 UART Communication

A USB-UART bridge on the Atlys board helps the PC to communicate with the system on the FPGA using a COM port. A 9600 baud rate communication with 8 data bits (LSB first), one stop bit and with no parity is established. The hardware control circuit and the software interface needed are available with the support library provided with the Atlys board.

3.1.5 Analog Sensors Interfacing

Three types of analog sensors are incorporated with the robot. They are IR proximity sensors, sharp GP2D12 IR range sensors, and two position encoders. The IR sensors are integrated for the robot to move autonomously by avoiding obstacles. The position encoders are for controlling the speed of motion of the robot. The sharp IR range sensor is equipped with an IR source and a CCD array. The IR light falls on the obstacle and returns back to the CCD array. The distance between the obstacle and the robot is determined from the analog output of the sensor, which is proportional to the angle of reflection of the light waves from the obstacle. For a range between 0 and 10 cm, the sensor's blind spot occurs, when the sensor gives erroneous readings. The distance and the sensor's output are non-linear. With the use of the formula in Eq. (3.1), the distance could be found out. [16]

$$distance = 10 * 2799.6 * (1/analogvalue^{1.1546}) \tag{3.1}$$

Also the sharp IR range sensors can be disabled by the processor when not in use. This saves the power. The IR proximity sensors are used to detect objects which are within the range of 0 to 10cm. When the light from the IR source falls on the object and gets reflected, the leakage current through the photo diode increases. This causes the diode voltage to drop. As the distance between the object and the robot increases, the leakage current gets lesser and lesser. When there is no object within 10cm, the output voltage across the diode will be approximately 5V. This module is also provided with an enabling circuit that can turn the module *on* and *off*.
The position encoders are used to control the velocity of motion of the robot. An IR LED and a photo transistor are mounted near the wheels facing each other. In between the setup, a slotted disc is placed which moves along with wheel. As the disc moves, it cuts the IR light falling on the photo transistor. This creates a series of square pulses as the output. A Schmitt trigger circuit cleans the output square waveform and feeds it into the FPGA pins. This waveform is analyzed for finding its frequency by a frequency counter circuit.

3.1.6 Locomotion Module

The robot is equipped with two DC geared motors connected with the FPGA pins through L293D motor drivers. It also has a caster wheel for support. Two position encoders are used to give feedback related to the position and velocity of the robot. Two modules are designed to control the locomotion of the robot. They are the movement control module and speed control module. The movement control module gives suitable logic levels according to the voice commands or the IR sensor outputs. The logic levels are fed to the motor drivers. There are 7 types of movement the robot can perform - forward, reverse, left, right, soft right, soft left and stop. For moving through a certain distance or angle, the outputs from the position encoders are used to calculate the distance that the wheels have taken. The calculation is done as follows [16].

wheel radius=2.55cm
number of slots in the position encoder disc=30
resolution of the position encoder=$2 * \pi * 2.55/30 = 0.534071 cm/pulse$

Using the above formula and by calculating the number of pulses obtained, the distance moved by the robot is obtained. For calculating the angle of motion, the following calculation is done. This calculation applies when the robot turns with one wheel moving clockwise and the other moving anticlockwise, or when the center of rotation is at the mid-point of the line connecting the wheels [16].

distance between the two wheels=15cm
radius of the circle traced when the robot rotates by 360°=7.5cm
circumference of the circle traced=47.1239cm
number of wheel rotations during the tracing=47.1239/16.0221=2.9412
resolution of the position encoder degrees=360/(30*2.9412)=4.08°/pulse

When the robot takes a soft turn (that is, one of the wheels turn with the other wheel at a stop), the calculation is done as follows [16].

radius of the circle traced when the robot rotates by 360°=15cm
circumference of the circle traced=94.2478cm
number of wheel rotations during the tracing=94.2478/16.0221=5.8824
resolution of the position encoder degrees=360/(30*5.8824)=2.04 °/pulse

The speed control module provides pulse width modulation (PWM) waveform. The power sent to the motor is directly proportional to the duty cycle of the PWM waveform. The duty cycle is the ratio of the period during which the waveform is logic 1 to the total period of the waveform. The PWM waveform generated has a period of 20ms. By varying the duty cycle, the power delivered to the motor is varied. The higher the duty cycle, more is the power delivered resulting in increased speed of motion.

3.1.7 LED, Switches and Buzzer

The Atlys FPGA board has 8 LEDs and 8 slide switches. The LEDs are used to indicate the status and the slide switches are used for controlling the operation of the robot manually. A buzzer (3 KHz piezo buzzer) is incorporated in the Fire Bird V robot which is used for beeping in case of an error or when attention of the user is needed by the robot. With a logic high voltage, the buzzer turns on.

3.2 Development of the Software Platform

Driver files are created for the MicroBlaze to interface with each peripheral. The templates for these files are automatically created by the tool, which had been customized later. The driver files for the codec-controller and the UART controller are taken from the library provided by Digilent along with the Atlys board. Apart from the driver files, a C library is created for performing the voice-command recognition module. Initially the entire voice-command recognition algorithm was implemented in software. The algorithm used in the program is discussed as follows.

3.2.1 Voice-Command Recognition

The general block diagram of the voice-command recognition module is shown in Fig. 3.4 and the steps followed during the design are detailed as given below.

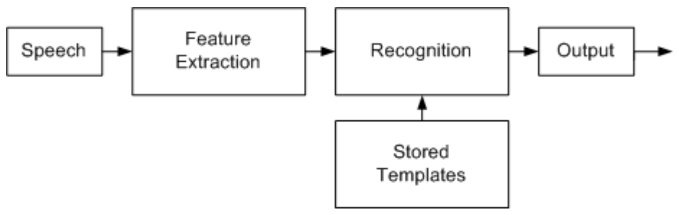


Figure 3.4: Voice-command Recognition Module - Block Diagram

Speech Acquisition

The speech is collected through the AC'97 codec with the help of the controller circuit. These speech samples are stored in the memory to be retrieved by the feature extraction circuitry. The steps involved in acquisition are as follows.

- The AC'97 codec is initialized - the codec and the controller are synchronized through the *SYNC* signal generated by the controller.
- Once they are synchronized, the microphone input is selected, and the volume and gain of the input channels are set by giving appropriate control information in the slots of the serial-data-out line *SDO* going to the codec.
- The sampling rate is set to 8000 Hz and the input coming through the microphone is recorded for a particular interval and stored in the DDR2 memory.

Feature Extraction

Spectral features are extracted from the speech in the feature extraction block. Mel-frequency Cepstral Coefficients are calculated by the steps detailed in Feature Extraction. All the steps of MFCC extraction are implemented in the software. To elaborate,

- A software function is written to compute the fast Fourier transform
- Triangular band-pass filtering involves finding the filter coefficients in Mel-frequency scale and repeated multiplication-accumulations with the absolute magnitude of the FFT values. The multiply-accumulate(MAC) function involved in the filtering is done using the software routines available for multiplication in the SDK library of the workspace created by the software.
- Logarithm calculation is implemented using the math.h library.
- The discrete cosine transform is also implemented using the math.h library and the multiplications are done through the software routines. The result of these steps gives the Mel-frequency cepstral coefficients.

Recognition

Dynamic time warping algorithm is chosen because of its efficiency and its simple architecture. The algorithm written for the recognition part is shown as a flowchart in Fig. 3.5.

- A and B are the two sequences of length m and n respectively that needs to be compared.
- Two matrices are defined - *dist* and *glob-dist* of size (m, n). The matrix *dist* contains the Euclidean distance between every elements of A and B. The matrix *glob-dist* is constructed to track the least distance path from the start to the end of the sequences.
- The final value *glob-dist(m,n)* gives the similarity measure between the two sequences.

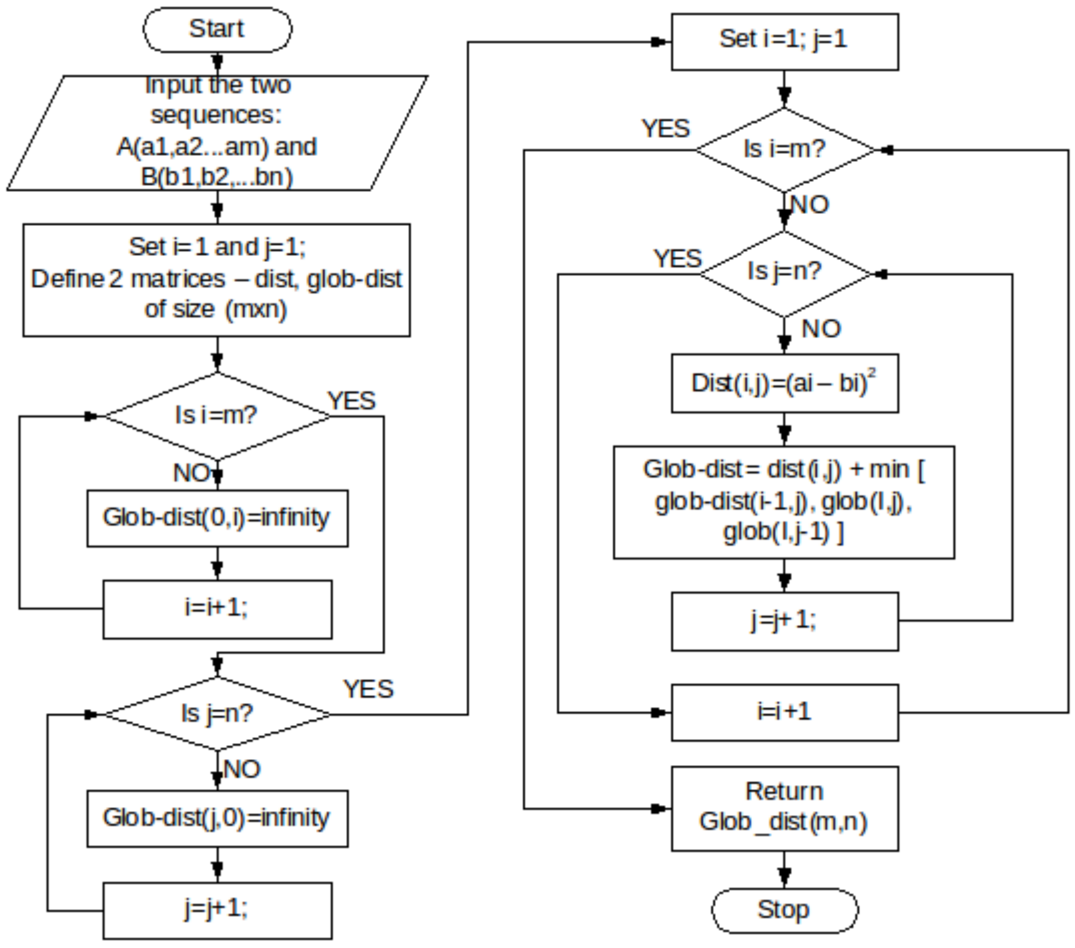

Start
Input the two sequences: A(a1,a2...am) and B(b1,b2,...bn)
Set i=1 and j=1; Define 2 matrices – dist, glob-dist of size (mxn)
Is i=m?
YES
NO
Glob-dist(0,i)=infinity
i=i+1;
Is j=n?
YES
NO
Glob-dist(j,0)=infinity
j=j+1;
Set i=1; j=1
YES
Is i=m?
NO
YES
Is j=n?
NO
Dist(i,j)=(ai – bi)²
Glob-dist= dist(i,j) + min [glob-dist(i-1,j), glob(I,j), glob(I,j-1)]
j=j+1;
i=i+1
Return Glob_dist(m,n)
Stop

Figure 3.5: Dynamic Time Warping - Flowchart of the Algorithm

This algorithm is used for the purpose of finding the uttered word. All the words that need to be recognized are processed for extracting the features. These features are stored in the memory. During recognition, the features of the uttered word are compared against the features of all the other words. The word in the template that gives the least cost is taken as the recognized word. The calculated cepstral coefficients are of 13-dimensions. Hence, the distance between any two series is calculated as the sum of square of the difference between the corresponding dimensions of the cepstral coefficients as shown in the Eq.(3.2).

$$distance = (rf_1 - tf_1)^2 + (rf_2 - tf_2)^2 + ... + (rf_{13} - tf_{13})^2 \tag{3.2}$$

where, rf_i and tf_i represent the i^{th} coefficient features of the reference and test templates respectively.

Output

The output block gives appropriate commands to the locomotion or the communication unit based on the control signals got from the recognition block. The algorithm aims to implement four commands - set forward, go reverse, turn left and drive right. The output block takes care of giving the suitable control signals for the locomotion unit.

3.2.2 The Main Software Execution

A main C program is written which is shown as a flowchart in Fig. 3.6.

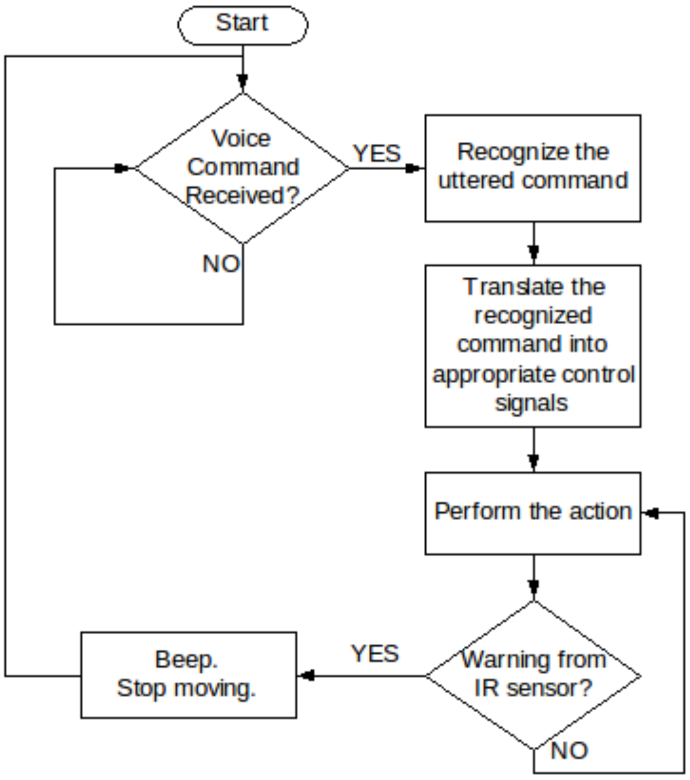


Figure 3.6: Flowchart depicting the Robot operation

As the system is switched on, the system starts waiting for the user's voice command. Once a word is uttered, the speech frames are sent for the system to recognize the word. As soon as the command is recognized, the control signals are generated by the processor (according to the program written) to the motor controller and the indicator modules. The robot performs the action according to the control signals until it senses an obstacle on its path, which is discovered with the help of the IR sensors. When the robot detects an obstacle, it stops in its path, warns the user through a beep and starts waiting for the next command. The behavior is programmed entirely in software. This creates a degree of flexibility in the design. The robot can also be made to perform the action for a particular interval after which it starts waiting for the next command. This software flow of this main C program can be altered to modify any functionality involving the peripherals.

3.2.3 The Performance of the Software-based Voice-command Recognition System

The execution timing of the program is shown in the Table. 3.1 for an input duration of 1 frame (1 frame consists of 160 samples amounting to a duration of 0.02 seconds). As seen in Table. 3.1, the time taken for processing one frame of input data is 18 seconds approximately. This implies that for an input of duration 2 seconds, the processing time will be more than 30 minutes practically. Although the recognition rate of the system is found out to be almost 100%, this system cannot be implemented for a real-time system.

Table 3.1: Performance of Software-based Voice-command Recognition System

Function	Execution time per frame (seconds)
FFT	11.831
Triangular Band-pass filtering	2.184
Discrete cosine transform	3.195
Recognition Algorithm	0.08879
Total Execution Time for one frame	17.2987

3.3 Design of Signal Processing Functions in Hardware (Co-design Implementation 1)

The major computations involved in the algorithm that take high execution times are

- FFT calculation
- Logarithm calculation
- Discrete cosine transform - finding the cosine values
- Repeated multiplications and accumulations involved in the Triangular band-pass filtering and discrete cosine transform
- Mapping the linear to Mel-scale frequency in the Triangular band-pass filtering

To improve the timing performance of the system, the extensive software routines for the above mentioned calculations are replaced by custom-designed peripherals which can incorporate hardware designs. Hence, the embedded platform shown in Fig. 3.2 is modified to incorporate a hardware FFT block, an unsigned multiply-accumulator, a LUT for storing logarithm values, a BRAM for storing the cosine values and Mel-frequency values, a signed multiply-accumulator and a cordic square-root calculator. The software is written appropriately to call these hardware units. This design is referred to as the *co-design implementation 1* in the thesis. The following sections discuss the design of the hardware of these blocks using FPGA.

3.3.1 Design of the Fast Fourier Transform

The fast Fourier transform is implemented with the help of the LogiCORE IP core FFT v8.0. The FFT core is configured to compute a 256-point forward DFT in pipelined streaming I/O architecture for every frame of the input. The IP core is interfaced with two first-in-first-out (FIFO) memory interfaces on either sides for collecting the input and output. This is shown in Fig. 3.7. The input frame is written into the FIFO named as the Write-FIFO by the processor. The FIFO streams all the data in the frame into the IP core if the core is ready. The IP core collects and processes the data and after a latency of 862 clock cycles, streams its output data. Also the IP is configured to output the samples in natural order. This output data is collected into a FIFO named as the Read-FIFO. The interface between the FIFOs and the FFT core is implemented as asynchronous handshakes. The FFT is made to be operating on fixed-point data. The output samples are scaled by a value of 256. The processor retrieves the FFT data from the Read-FIFO for further calculating the magnitude of the output complex numbers.

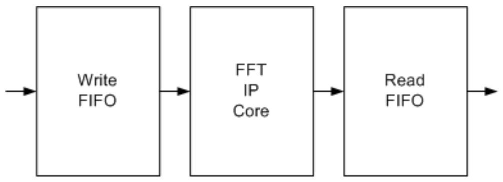


Figure 3.7: FFT - Block Diagram

Fig. 3.8 shows the simulation result of the FFT core. The system is simulated at a clock period of 10 ns.

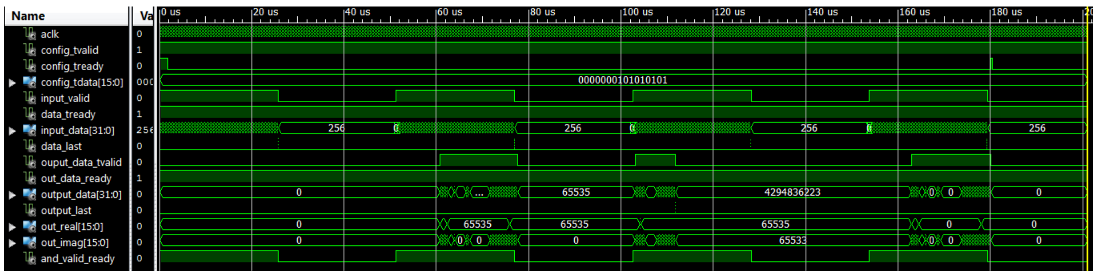


Figure 3.8: Simulation Result of FFT module

3.3.2 Design of the signed and unsigned MACs

With the help of the Xilinx core generator tool, a multiply accumulator unit is constructed and added as a slave peripheral to the PLB through the IP interface. Also software library is written for the MicroBlaze to interact with this core. Initially the two input values that need to be multiplied are written by the software into two registers in the custom-logic created within the IPIC module. Also the software sets the control bits required for indicating the validity of the inputs. The hardware unit performs the function and outputs the results into another register in the IPIC module which the processor can access through the driver files. Because of this implementation, extensive software routines needed for multiplications are avoided.

3.3.3 Design of the CORDIC square-root calculator

The LogiCORE IP - CORDIC is used to find the square root required in finding the magnitude of the complex FFT values. A slave peripheral with an IPIC is designed. User registers are defined for writing/reading the input/output values by the MicroBlaze processor.

3.3.4 Design of BRAMs

One of the ways of designing the hardware unit for logarithm calculation is through the look-up table (LUT) implementation, which is followed in this project. This method has been detailed in [17]. Any number, a, can be written as shown in Eq. (3.3), where p is an integer (called the power-value) and N is called the normalized value which is between 0.5 and 1.0.

$$a = 2^p * N \tag{3.3}$$

Taking log on both the sides, Eq.(3.3) can be written as

$$log_2 a = p + log_2 N \tag{3.4}$$

Also, the relation between logarithm to the base 2 and natural logarithm can be expressed as Eq.(3.5).

$$log_2 a = log_e a / log_e 2 \tag{3.5}$$

Using Eq.(3.4) and Eq.(3.5),

$$log_e a = (p + log_2 N) * log_e 2 \tag{3.6}$$

$log_e 2$ is a floating point constant. Once p and $log_2 N$ are calculated the final logarithm value can be calculated by adding them and later multiplying with the $log_e 2$ value. p can be found out by finding the position of the most-significant bit in the binary representation of the number a that is of logic 1. The value of $log_2 N$ is found out from the look-up table. The table contains 256 numbers of logarithms of values that are equally spaced with an interval of 0.5/256 = 0.001953 from 0.5 to 1.0. Only the logarithms are stored in the LUT. The indexing is accomplished by manipulating the input number. That is, if x is the input, then the index is calculated as $(a - 0.5)/0.001953$. Finally the scaled values (by 10000) of the Mel filter coefficients are corrected by subtracting the value 4.
All the values involved in the logarithm calculations and the values stored in the LUT are scaled by 10000 for fixed-point calculations. This scaling includes additional precision because of including the first four numbers after the decimal point. A hardware block including the LUT is created which can be used as a tailored instruction by the processor. The LUT is created as a single port ROM (using BRAMs). The index of the LUT is given by the address of the ROM.
Also two other BRAMs are generated to store the ready-made filter coefficients (calculated through MATLAB and loaded into the BRAM while designing as .coe files) needed for the triangular band-pass filtering and for calculating the cosine values.

3.3.5 Analysis of the System

The embedded platform designed incorporating hardware cores for computation intensive functions is shown in Fig. 3.9 . Table. 3.2 shows the execution time to process one input frame in the design. The processing time has improved by approximately 47 times compared to the complete software implementation.

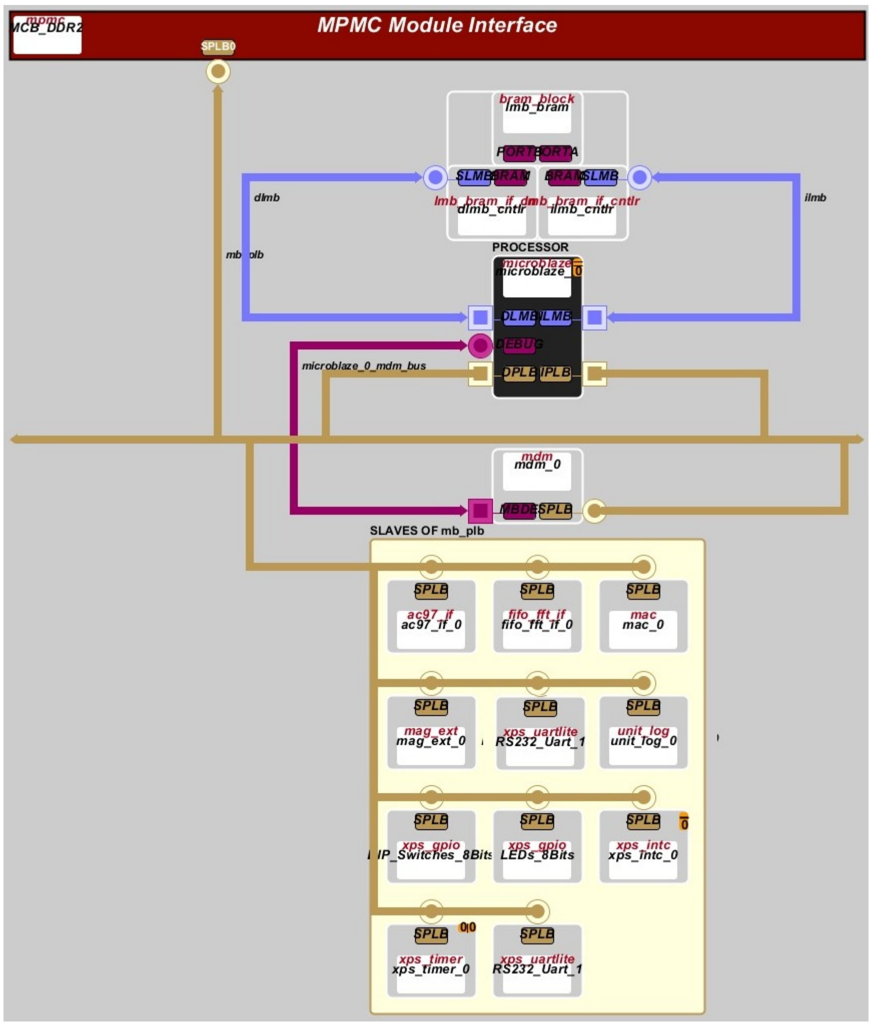


Figure 3.9: Embedded Platform with Hardware Cores for Computation Intensive Blocks

Table 3.2: Execution Timings for the Design with Hardware Cores for Computation Intensive Blocks

Function	Execution time per frame (seconds)
FFT	0.007635
Triangular Band-pass filtering	0.22
Discrete cosine transform	0.05675
Recognition Algorithm	0.08879
Total Time	0.3732

Although the timing performance has improved, for processing an input speech signal of duration 2 seconds, the time taken is approximately 45 seconds. The operating frequency of the processor is at 66.67MHz. The real-time recognition factor which is the ratio of the time taken to process the input speech to the duration of the input speech is 22.5. The lesser the value of the real time factor, the system responds better in real time. Hence, this design does not provide a real-time operating system. As seen from the Table , the triangular band-pass filtering and discrete cosine transform operations take the major portion of the execution time. To arrive at an optimized design with good area to delay ratio, the entire operations in triangular band-pass filtering and discrete cosine transform are moved to hardware, and the design of the embedded platform containing these hardware cores is discussed in the following section.

3.4 Hardware-Software Co-design with more Signal Processing Functions Implemented in Hardware (Co-design implementation 2)

3.4.1 Triangular Band-pass Filtering

Mel scale

Mel is a measuring unit of perceived pitch of a sound. The Mel scale was developed based on the hearing perception of human beings. It was built by picking the reference frequency as 1 KHz and assigning it as 1000 Mel. On an approximation, the scale is linear below 1 KHz and it is logarithmic after 1 KHz. This is because human ear can discern comparable pitch increments only with bigger and bigger intervals of frequency above 1 KHz [18]. There are several relations relating the Mel frequency to the linear frequency obtained through varied trial and analysis. The prevalently used relation is given by the Eq.(3.7): [18]

$$Mel(f) = 1127 * ln(1 + f/700) \tag{3.7}$$

Band-pass Filtering

Since the human ear can differentiate only frequencies according to Mel scale, different words of speech can be analyzed for containing frequency components that are logarithmically spaced. From this, it can be inferred that only certain frequencies of the spectrum of the speech signal is needed to characterize it effectively. Hence, the spectrum of each speech frame is passed through many band-pass filters to capture the important Mel-frequencies in the speech. This is called Mel-bank filtering. The result produces magnitudes of power at the frequencies that can be heard discretely by the human ear. Hence, these values are called ear-magnitudes.

Hardware Design

This whole block of calculating the ear-magnitudes is implemented in hardware. It is designed in such a way that it can be called as an instruction by the processor. Forty overlapping triangular band-pass filters centered on forty critical frequencies are constructed. All the triangular filters have unit area. Hence, the filters have decreasing magnitude and have larger frequency bandwidths at larger frequencies. This is to normalize the resulting magnitude of the power spectrum on the Mel-scale. The coefficients of the filters are calculated beforehand and are stored in the memory. Each frame of the speech signal is passed through the filters and the resulting power spectrum values on Mel frequency axis are stored back in the memory.

This filtering operation can be represented by the means of a multiplication between a matrix containing the filter coefficients and another matrix containing the speech frame [19]. This can also be inferred as a matrix multiplication between a sparse matrix and the speech matrix as shown in Eq.(3.8), where $f_{x,y}$, $a_{x,y}$, $emag_{x,y}$ represent the filter coefficients, absolute magnitude of the FFT values and ear-magnitudes respectively in the x^{th} row and y^{th} column of the respective matrices.

$$\begin{pmatrix} f_{1,1} & .. & .. & f_{1,256} \\ . & .. & .. & .. \\ . & .. & .. & .. \\ f_{40,1} & .. & .. & f_{40,256} \end{pmatrix} * \begin{pmatrix} a_{1,1} \\ . \\ . \\ a_{256,1} \end{pmatrix} = \begin{pmatrix} emag_{1,1} \\ . \\ . \\ emag_{40,1} \end{pmatrix} \tag{3.8}$$

The hardware implementation designed is as shown in Fig. 3.10.

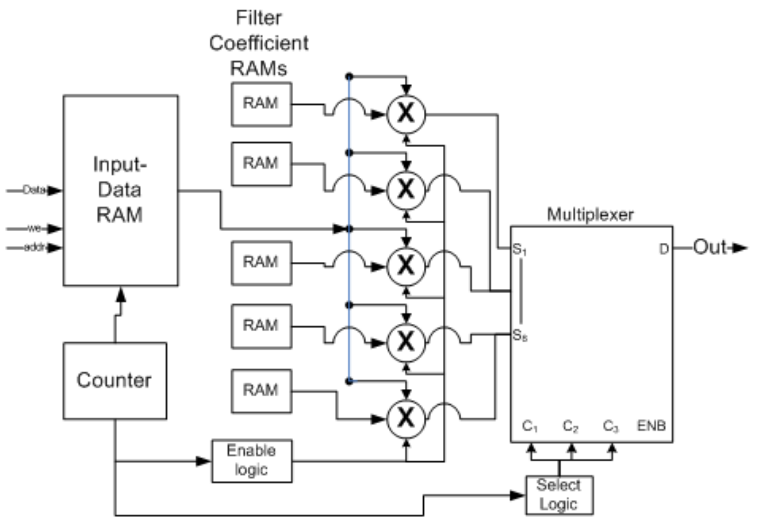


Figure 3.10: Design of the Ear Magnitude Extractor

The input data RAM represents the memory where the input speech frames are stored. There are 40 filter-coefficient RAMs which store the coefficients of each filter. In the filter coefficient matrix, there are 413 non-zero values out of the total of 10240 elements in the matrix. Thus there are 413 numbers of multiplications and several additions to be done per speech frame. After filtering, each frame produces 40 values, stored as a column in the emag matrix. Each element of the emag matrix, say emag$_{x,1}$ is produced by multiplying each element of the x^{th} row of filter-coefficient matrix with corresponding elements in the column matrix of the speech frame and accumulating the results of the same. There are five ways of implementing this sparse matrix multiplication.

1. Implementing the whole operation in software as done in section Development of the Software Platform. The processor executes all the operations sequentially from getting the data from the memory, multiplying and adding and writing data to the memory. This method although being simple has a huge amount of delay.

2. The subsequent multiplication and addition operations can be performed by a single hardware multiplier-accumulator (MAC) circuit. This can be designed as a custom instruction to be called by the processor. Hence, the processor will take the speech frame and multiply with

every row of the filter-coefficient matrix sequentially. This needs the speech frame to be accessed 40 times from the memory. Also since every element of the emag matrix is generated sequentially, the delay for processing one frame is high.

3. Since the filter-coefficients are fixed, every multiplier can be generated as a constant-multiplier. However since there are 413 total multiplications per speech frame, this will require a huge amount of resources. The input frame is accessed only once from the memory. However the design will not be ideal for resource usage optimization.

4. The number of outputs generated per speech frame is 40. Hence, 40 numbers of multiplier-accumulator circuits can be assigned for each filter. The filter-coefficient matrix is sparse. So when the 1^{st} element of the emag matrix is being calculated, 6^{th}-40^{th} MACs are ideal. Although the delay is decreased compared to the method 1 and the resources decreased compared to method 2, this method still uses redundant gates.

5. The 4^{th} method described above was implemented and analyzed. It is found that only 5 numbers of MACs are active at a time. Hence, the first 5 MACs can be reused for all the 40 MAC operations. That is, once the 1st MAC operation is done, it is relieved and reset to be used for the 6^{th} MAC operation. It is again used for the 11^{th}, 16^{th}, 21^{st}, 26^{th}, 31^{st} and 36^{th} MAC operations. Similarly the 2^{nd} MAC is used for the 7^{th}, then 12^{th} and so on. This reduces the resources needed with the delay remaining in the same order as the 4^{th} method. Reusing the same resources is done in this method.

The 5^{th} method of hardware design is used for implementing the Mel filter banking circuit. A counter circuit and additional control signals are designed in hardware on FPGA. These control circuits reset the MACs at regular intervals and are used for collecting the output values in order from the MAC units. Out of all the five methods, the 5^{th} way of implementation is optimized in terms of resources and better in terms of delay.
Also the implementation is done for fixed point numbers with appropriate scaling. The filter weights are scaled by a value of 10000 to include the first four numbers after the radix point. This scaling is approximated later by subtracting a value of 4 from the logarithm of these values. A LUT for the logarithm implementation is used and the hardware multiplier units are used for the data manipulation purposes.
Figure. 3.11 shows the top level hierarchy of the hardware module designed for computing the triangular band pass filtering. The chip uses only 5 MACs and some control logic for sequencing and resetting purposes.

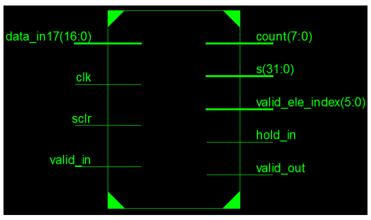


Figure 3.11: Ear-magnitude Extractor Module

Figure. 3.12 shows the RTL schematic implemented by the Xilinx synthesizer. Part of the figure is expanded and shown at the right of the main RTL figure, which depicts the use of the five MAC units.

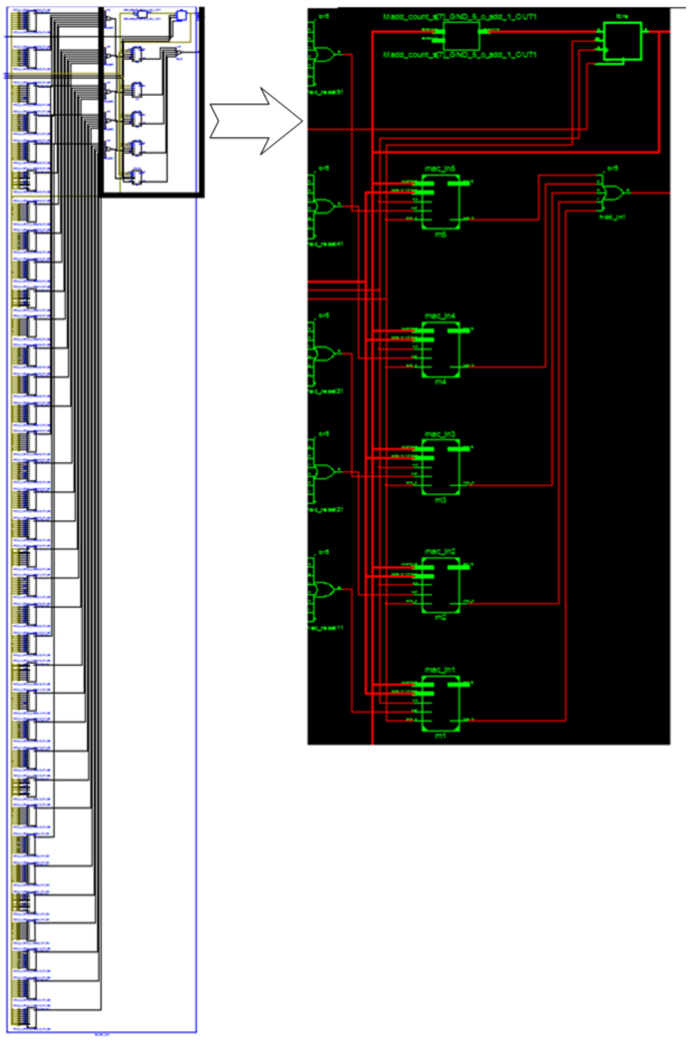


Figure 3.12: RTL Figure of Ear-magnitude Extractor

The logarithms of the filtered values are fed to the discrete cosine transform block for further computation. The design of the DCT block is discussed in the next section.

3.4.2 Design of Hardware module for Discrete Cosine Transform step

The logarithms of the Mel-scaled spectral coefficients are stored in the memory and accessed by the processor for the calculations of the final Mel frequency cepstral coefficients. This is done by taking the discrete cosine transform of the logarithm of the ear-magnitudes. It can be expressed by the Eq.(3.9):

$$cep_i = \frac{2}{40} * \sum_{1}^{40} (log_e(mag_j)) * \cos(\pi * 2 * 40 * (j-1) * i) \tag{3.9}$$

where i = 0 to (C-1). C in the above equation is an integer referring to the number of cepstral coefficients. In this implementation, C is chosen to be 13. The Eq.(3.9) can be rewritten as

$$cep_i = \frac{2}{40} * \sum_{1}^{40} (log_e(mag_j)) * m_{i,j} \tag{3.10}$$

where,

$$m_{i,j} = \cos(\pi * 2 * 40 * (j-1) * i) \tag{3.11}$$

where i = 0 to 12 and j= 1 to 40 . Eq.(3.11) can also be represented as the matrix shown in Eq.(3.12).

$$m_{i,j} = \begin{pmatrix} \cos(\pi * 2 * 40 * 0 * 0) & .. & .. & \cos(\pi * 2 * 40 * 0 * 39) \\ . & .. & .. & .. \\ . & .. & .. & .. \\ \cos(\pi * 2 * 40 * 0 * 12) & .. & .. & \cos(\pi * 2 * 40 * 39 * 12) \end{pmatrix} \tag{3.12}$$

Hence, the Eq.(3.10) can be represented as

$$\begin{pmatrix} cep_1 \\ cep_2 \\ cep_3 \\ . \\ . \\ cep_13 \end{pmatrix} = \frac{2}{40} * \begin{pmatrix} m_{i,j} \end{pmatrix}_{(13X40)} * \begin{pmatrix} emag_{0,0} \\ . \\ . \\ emag_{39,0} \end{pmatrix}_{(40X1)} \tag{3.13}$$

This is equivalent to a multiplication between two matrices. The first matrix is a constant matrix and the second is the frame matrix containing the logarithm of the ear-magnitudes. The result gives a column matrix with 13 cepstral coefficients. This operation can be implemented in three ways as described below.

- The whole operation can be implemented in software as discussed in the previous section. That is, the processor sequentially gets the required data from the memory, multiplies, adds and writes the data to the memory. The sequence of operations is defined in the program lines. Multiplication in software creates a huge amount of delay. Hence, this is not an ideal implementation.
- The multiplying and accumulating operation are performed in hardware with the help of a multiplier-accumulator circuit. This module can be used to replace the multiplication in software, thus reducing the delay compared to the method 1 explained above.
- The third method is to use an array of 13 multiplier-accumulators to compute each cepstral coefficient. This avoids the repeated memory access times required for computing each dimension of the MFC coefficients.

Out of the three methods, the 3^{rd} method has been chosen. The delay is reduced by a great amount in this method, since all the 13 MACs compute the coefficients in parallel. The MACs are built using the LogiCORE IP of Xilinx. The MAC design uses the DSP slices available in Spartan-6 and hence, it offers good speed. The cosine values in the equation are stored in memory. A control unit has

been built, which keeps track of the memory addresses needed to access the cosine values and the input ear-magnitudes. In this implementation, the ear-magnitudes are accessed from the memory only once, in contrast to the 2nd method, where the number of access times is 13. This reduces the delay although the number of slices used increases.
This implementation of the module in hardware is as shown in Fig. 3.13. The *Input Data RAM* contains the ear-magnitudes. The *Filter Coefficient RAMs* contains the cosine values. The result of the computation produces the Mel-frequency cepstral coefficients or MFCC. These are stored in memory for use in the recognition algorithm.

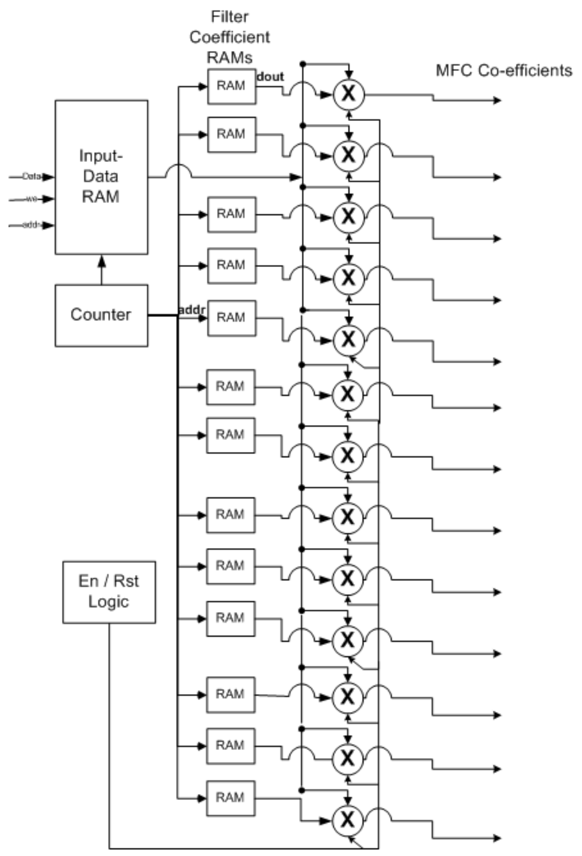


Figure 3.13: Design of the Cepstral Coefficient Extractor

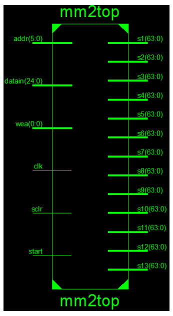


Figure 3.14: The Cepstral Coefficient Extractor Module

Figure. 3.14 shows the top level module of the hardware design implemented for the calculation of the discrete cosine transform calculation. The RTL schematic generated by the software is shown in Fig. 3.15. The blocks with the cross on top of it indicate the multiply-accumulate units.

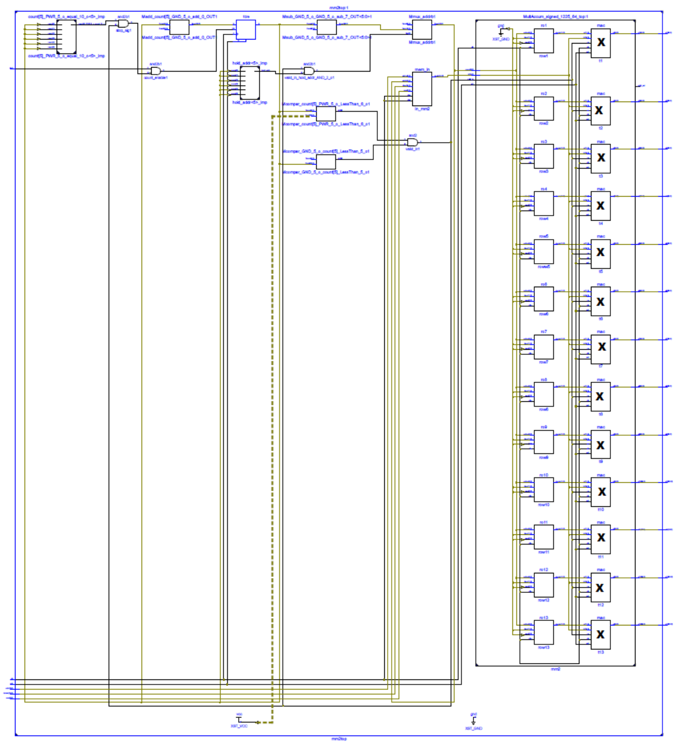
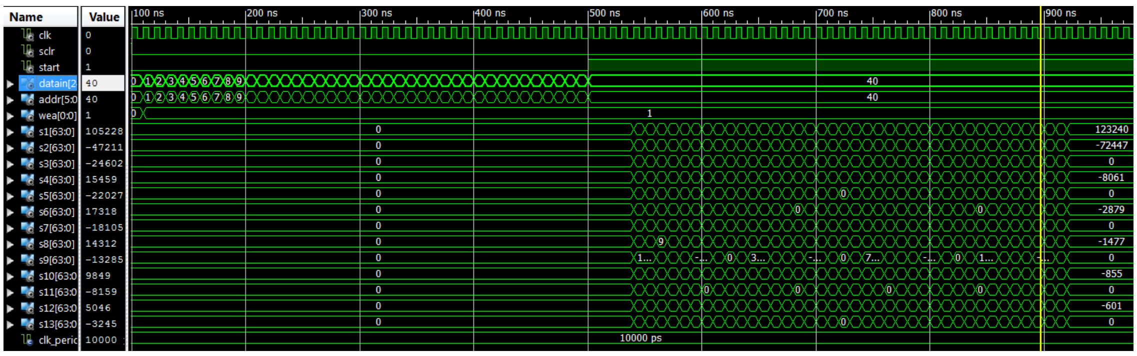


Figure 3.15: RTL of the Cepstral Coefficient Extractor

The 13 cepstral coefficients are drawn from this block as outputs simultaneously. These outputs are written in user registers incorporated in the custom user logic of the IP interface module. The processor is designed to read these outputs after the validity flag are set by the hardware showing that new output data has arrived. The processor keeps checking on the flag after giving a frame of ear-magnitudes as the input. Figure. 3.16 shows the simulation result of this block.



Name	Value
clk	0
sclr	0
start	1
datain[2	40
addr[5:0	40
wea[0:0]	1
s1[63:0]	105228
s2[63:0]	-47211
s3[63:0]	-24602
s4[63:0]	15459
s5[63:0]	-22027
s6[63:0]	17318
s7[63:0]	-18105
s8[63:0]	14312
s9[63:0]	-13285
s10[63:0	9849
s11[63:0	-8159
s12[63:0	5046
s13[63:0	-3245
clk_peric	10000

Figure 3.16: Simulation of the Cepstral Coefficient Extractor Design

3.4.3 Analysis of the Embedded Platform

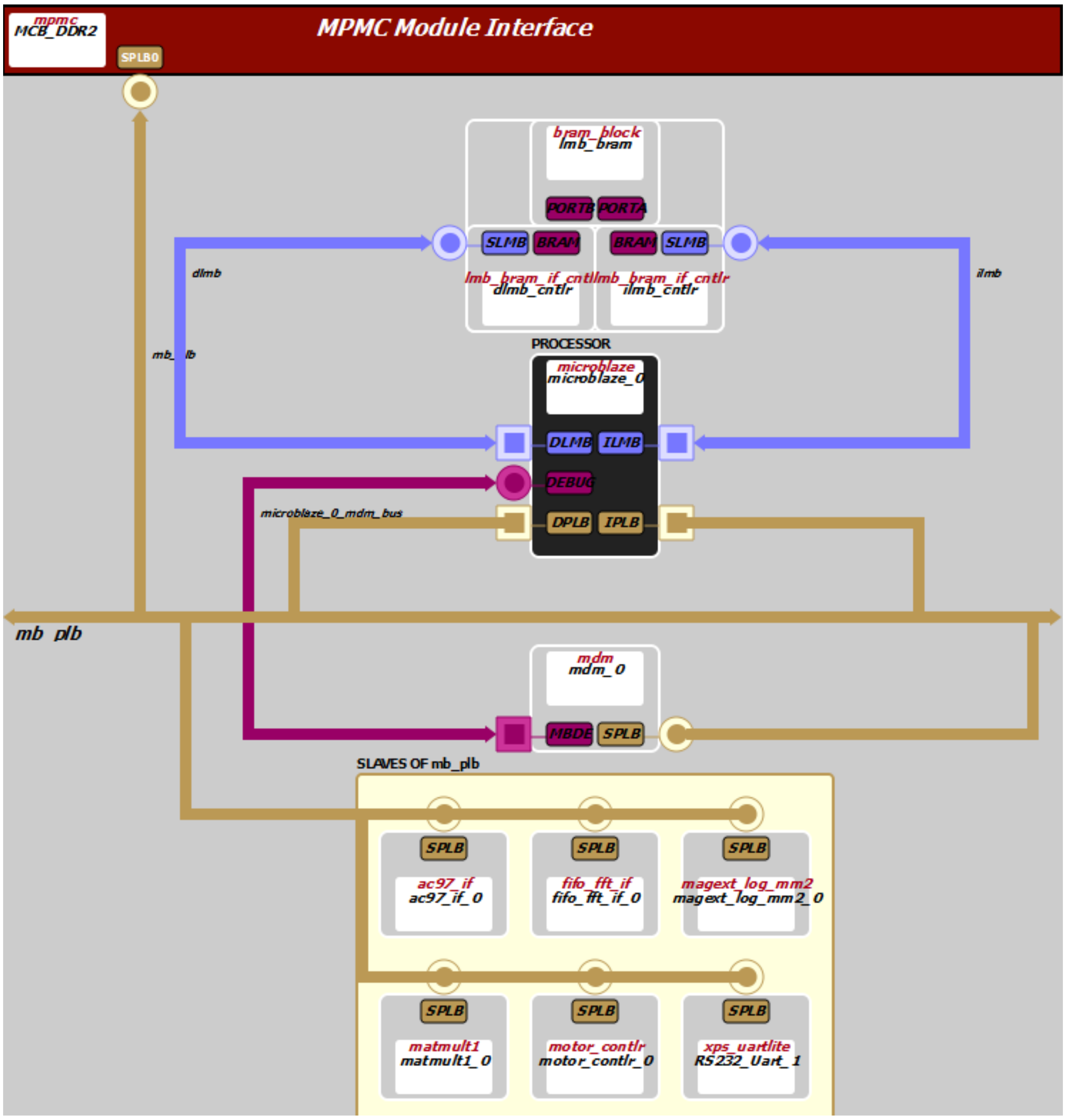


Figure 3.17: The Embedded Platform created with Custom-designed Hardware Signal Processing Blocks

Figure 3.17 shows the embedded platform created with all the hardware designs of the signal processing functions added as slave peripherals to the PLB of MicroBlaze processor. The software interface is achieved with the help of driver files created with the help of the EDK software. Through these driver files, the MicroBlaze has been made to read/write in the registers in the slave peripherals. For implementing the voice-command recognition algorithm, the same main software execution flow discussed in Chapter 2 has been followed. However, in place of the pure-software implementation

of the computation intensive blocks, separate functions in C are written. This in turn calls the hardware designs attached as peripherals of the processor.
Table 3.3 shows the timing performance of each of the signal processing step. The execution time is calculated for processing a speech signal possessing duration of 2 seconds. The processor operates at a frequency of 66.67 MHz and each frame contains 160 samples. Hence, 100 frames are to be processed for a speech signal of period 2 seconds. The real time factor of this design is 5.357 which is still higher than 1 (the real time performance is achieved when the factor is less than or equal to 1).

Table 3.3: Timing Performance of the Embedded System with both Hardware and Software Components

Function	No. of Processor Cycles	Total Execution Time (sec)	Execution time per frame (seconds)
FFT	36945540	0.554155	0.00554155
Ear-magnitude calculation	54172363	0.812544	0.00812544
Discrete cosine transform	31258530	0.468854	0.00468854
Recognition	592011084	8.879	0.08879
Total time	714387517	10.714	0.10714

3.5 Design of the Hardware-based Feature Extraction module (Feature Extraction Co-processor

To improve the real time factor of the system, a hardware co-processor was designed only for calculating the MFCC features. All the functions except the recognition are moved to hardware as separate sub-modules of the co-processor which are pipelined to increase the timing performance of the system. The following sections describe the designs of the sub-modules. Figure 3.18 shows the hardware designed for the co-processor module. The design of the FFT block described earlier is used. Similarly the designs of triangular band-pass filtering and discrete cosine transform blocks are detailed previously.

Magnitude Extractor

The magnitude extraction circuit is shown in Fig. 3.19. The real and imaginary parts of the complex numbers are sent sequentially to the squaring circuits. These squared values are added and its square-root value is calculated. The squaring circuit is implemented with 16x16 bit combinational multiplier and the square-root finder is constructed by configuring the LogiCORE IP Cordic v4.0. The Cordic circuit has a latency of 4 clock cycles. The FFT and magnitude extractor blocks are implemented as customized hardware units which can be called like an instruction by the processor. Incorporating such computation intensive blocks as hardware blocks increases the speed of processing.

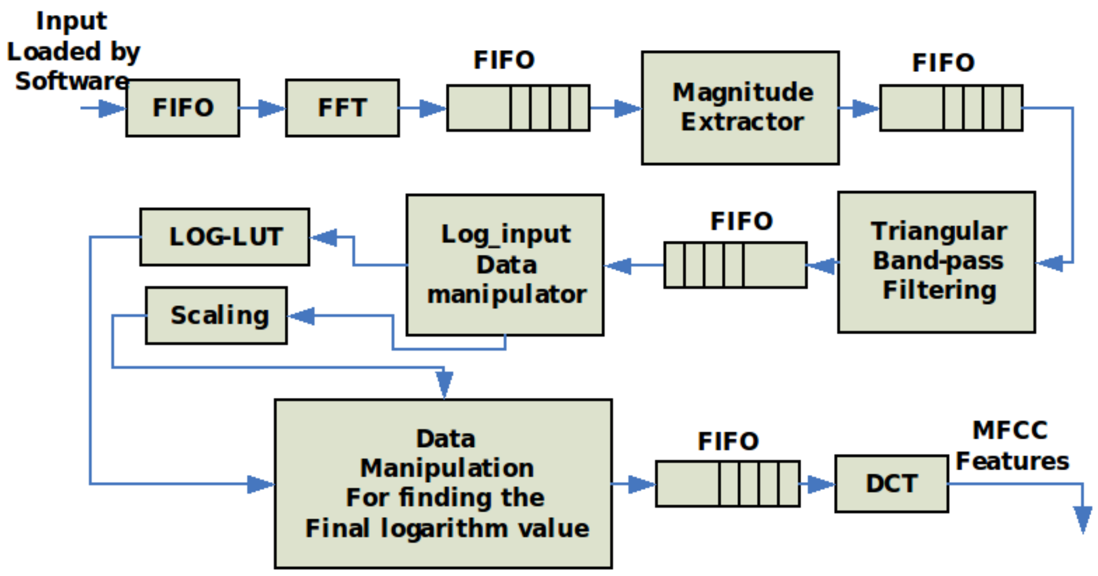

Input Loaded by Software
FIFO
FFT
FIFO
Magnitude Extractor
FIFO
Triangular Band-pass Filtering
FIFO
Log_input Data manipulator
LOG-LUT
Scaling
Data Manipulation For finding the Final logarithm value
FIFO
DCT
MFCC Features

Figure 3.18: Block Diagram of the Complete Hardware Design for MFCC feaure Calculation

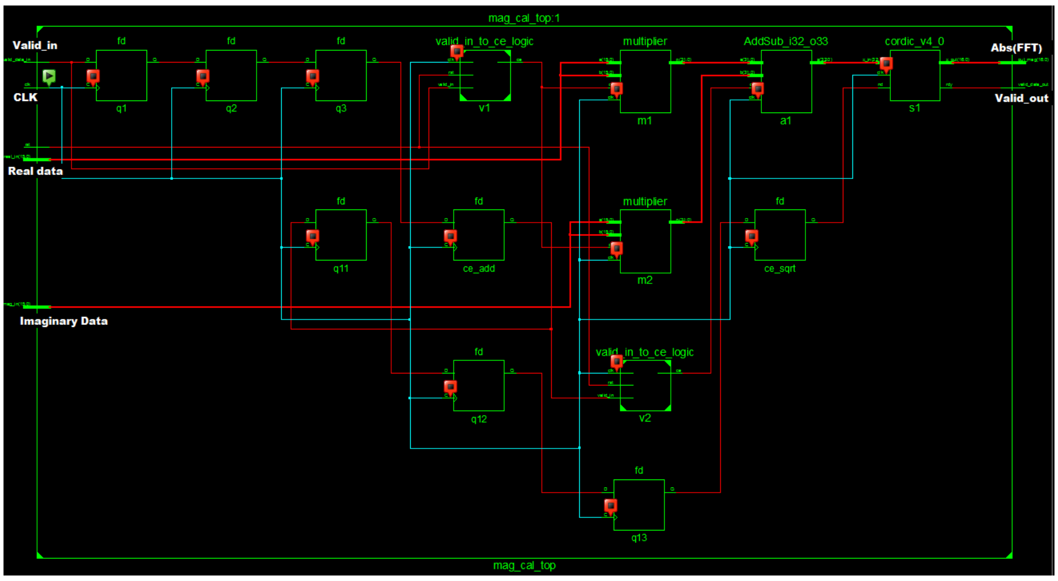


Figure 3.19: Design of the Magnitude Extraction block

Logarithm Calculation

Perceived pitch of a sound depends upon its frequency and also upon intensity level in its critical bands. Higher the intensity level, a more extreme pitch is perceived. Furthermore, the bandwidth of the critical bands varies with frequency. The bandwidth begins at 100 Hz for frequencies below 1 KHz and increases logarithmically above 1 KHz. Hence, logarithm of the total energy in critical bands is found to be a good representation of the critical frequency components. [18]. For this reason, all the ear-magnitude values are passed to a logarithm calculator circuit.
Figure 3.20 shows the top level module for calculating the logarithm of a number. The input to the

block is the number itself. Hardware blocks are dedicated for the data manipulation operations on the input data (described earlier). Figure 3.21 shows the RTL schematic generated by the software.

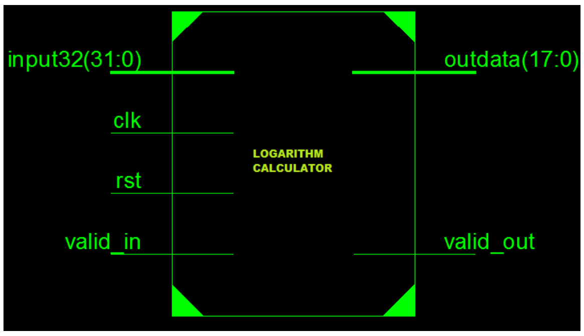


Figure 3.20: Hardware Module for Logarithm Calculation

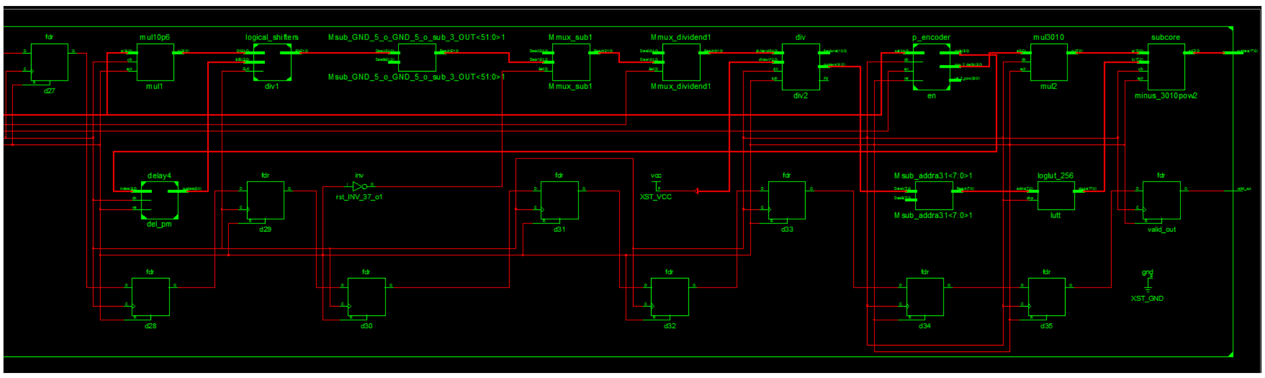


Figure 3.21: RTL Figure of the Logarithm Calculator

3.5.1 Analysis of the Design of Co-processor Module

The input speech samples are loaded into the FIFO for the FFT calculation (transform size=256) by the software. After a latency of 862 cycles, the FFT starts to output the 256 values sequentially. These values are of 32 bits each and are formatted such that the most significant 16 bits forms the imaginary part and the least significant 16 bits forms the real part of the complex number. These values are fed into the magnitude extractor block. The registered multipliers and the cordic square root calculator requires a latency of 18 clock cycles to perform. The output absolute values are fed into the triangular band-pass filtering module. This filtering module has been designed to have 5 multiply-accumulator units. This block is designed to have a latency of 85 clock cycles to complete its operation. The indices of the non-zero filter coefficient values in the matrix and their coefficients values are stored in a single port ROM, which incurs a latency of 1 clock cycle. This module is designed to operate at twice the clock frequency of the remaining blocks. This is due to the reason that the indices of the values in ROM are needed to be compared against the input data count value. The ear-magnitude values are fed into the logarithm calculator block, which in effect incurs a latency of 41.5 clock cycles. This perceived very high latency is due to the division of the input data by a value of 1953 to find the index of the logarithm LUT. This block is followed by the cepstral coefficient extractor module with a latency of 40 clock cycles. Thus, the total latency of the feature extraction block is found to be 1046.5 clock cycles. Table 3.4 summarizes the total latencies involved for all the sub-modules of the MFCC co-processor. And, the total time taken for the computation of one frame is 18.83μs while operating at a clock frequency of 66.67MHz.

Table 3.4: Latency of Sub-modules of the MFCC Feature Extracting Co-processor

Function	Latency in terms of no. of Clock Cycles
FFT	862
Magnitude Extractor	18
Triangular Band-pass filtering	85
Logarithm Calculation	41.5
Discrete cosine transform	40
Total Latency	1046.5 Cycles

Figure 3.22 shows the top level module of the pure-hardware based feature extractor module. The register transfer level diagram of the designed hardware block is shown in Fig. 3.23. The simulation output results of the feature extraction module is shown in Fig. 3.24. Here, the input is taken from the read FIFO of the FFT block and the output is the 13 cepstral coefficients obtained for one frame of input data. The first white line shows the resultant latency for the triangular band-pass filtering of the FFT magnitudes. The second line shows the latency needed for finding the logarithm of the values obtained from the band-pass filter. The third line shown is the latency incurred while calculating the discrete cosine transform.

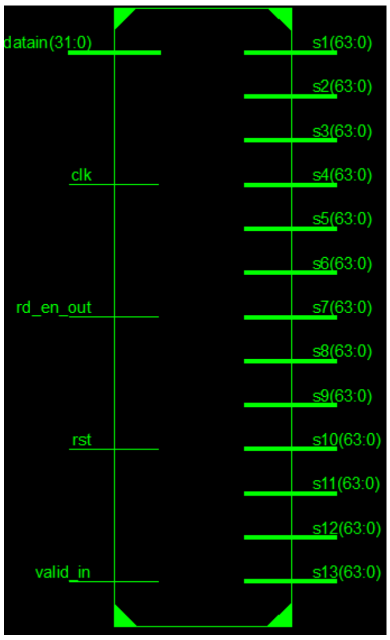


Figure 3.22: The Hardware Feature Extractor Module

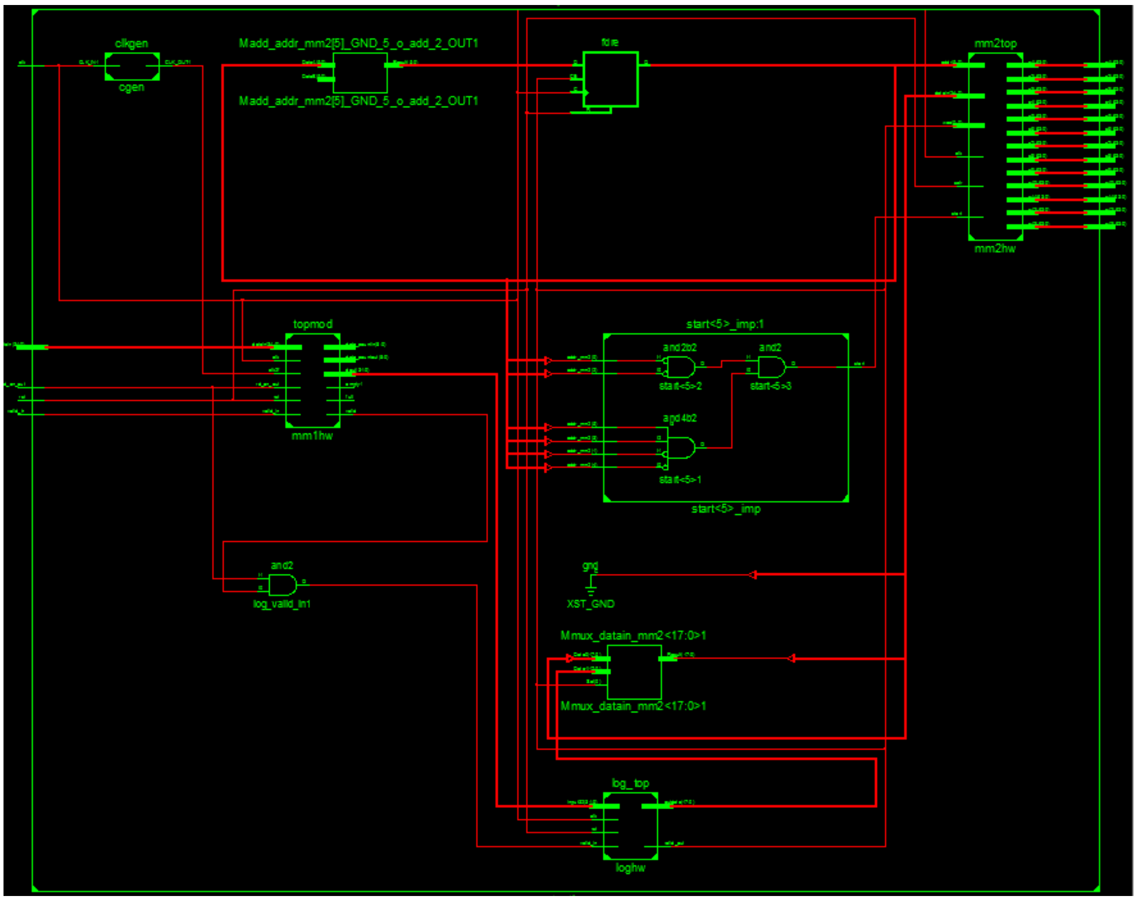


Figure 3.23: RTL Figure of the Feature Extractor Module

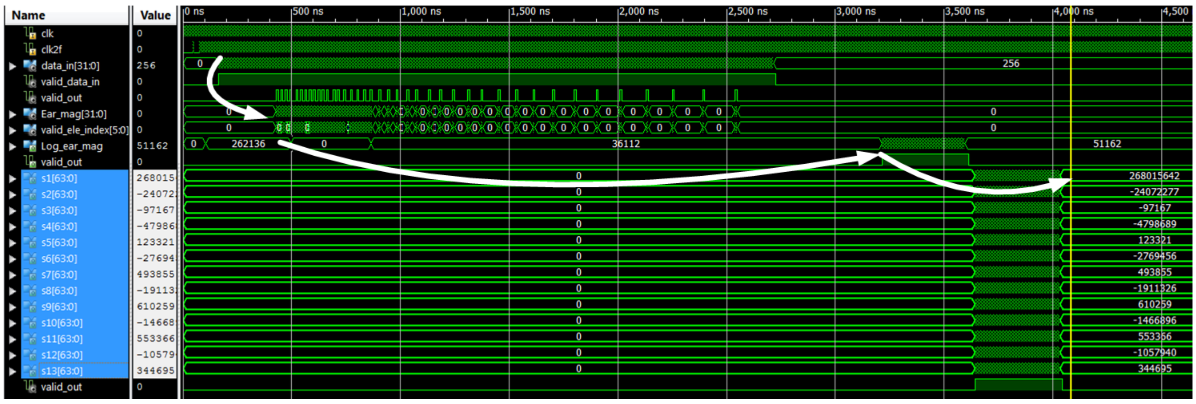


Figure 3.24: Output waveform of the Hardware Feature Extractor Module

3.6 Summary of the Design of Voice-Command Recognition Module

Four major variations of the design of voice-command recognition module are carried out in the project. They are as follows:

- Complete software-implementation of feature extraction and recognition
- Hardware/software co-design (Implementation 1) of feature extraction module with recognition module implemented in software. Hardware blocks include MAC, LUT for log and BRAMs for storing the filter coefficients and cosine values.
- Hardware/software co-design (Implementation 2) of feature extraction module with recognition module implemented through software design. Hardware blocks for the entire triangular band-pass filtering and discrete cosine transform operations are designed and included in the system.
- Complete hardware design of feature extractor and software implementation of recognition algorithm. The feature extraction module acts as a co-processor unit for the MicroBlaze processor.

The four methods have been found to possess their own advantages and limitations, which are analyzed in the next chapter.

Chapter 4

Results

4.1 MFCC Feature Extraction Block - Comparison Between the Software Implementation, Hardware-Software Co-designs and Hardware Design

As discussed in the chapter 3, the MFCC feature extraction block was designed in four ways. The area occupied by the designs and the execution time for the feature extractions from the speech are tabulated as shown in Table 4.1. As can be observed, the software implementation uses the least amount of resources, however takes a lot of time to finish the MFCC computation.

Table 4.1: Comparison of the Designs of Feature Extraction Module

Parameter	Software	Codesign 1	Codesign 2	Software with Feature Extraction Co-processor
Execution time (s)	17.2987	0.3732	0.10714	0.0888
No of slices registers	0	923	1010	2466
No of slice LUTs	0	1131	1788	1866
No of BRAMs	0	3	27	36
No of DSP slices	0	0	5	46
No of fully used LUT-FF pairs	0	501	998	3581

The hardware-software co-design (Implementation 1) using hardware blocks for MACs and BRAMs uses more area compared to software implementation however provides a better timing performance. It is to be pointed out that, this system does not provide real time operation capability.

The second variation of the hardware-software co-design (Implementation 2) with exclusive hardware blocks for computing the filtering and DCT operations has an acceptable real-time factor of approximately 5. However the major portion of the execution timing is due to the recognition algorithm which is implemented in software. More recognition architectures exist in literature that

provide variations of the DTW algorithm. Implementations of these algorithms might decrease the real-time factor making it suitable for real-time systems.
The pure hardware implementation of the feature extraction block has a real-time factor of 4.5 approximately. Again the bottleneck faced in the design is the execution time incurred by the recognition unit.
The execution timings for extracting the features are shown in Table 4.2. However the timings does not include the delay of the recognition unit. The hardware-software co-design (implementation 2)

Table 4.2: Feature Extraction Circuit - Execution Timing Comparisons

	Software	Hardware-Software co-design 1	Hardware-software co-design 2	Hardware
Execution Time for 1 frame	17.21	0.2844	0.0184	18.83μ

proves to be optimum with respect to both delay and area as seen from Fig. 4.1. In the graph, the area is the normalized measure of the total number of resources and the delay is in seconds. Hence, this implementation is used for the final design of the voice-command recognition module in the robot.

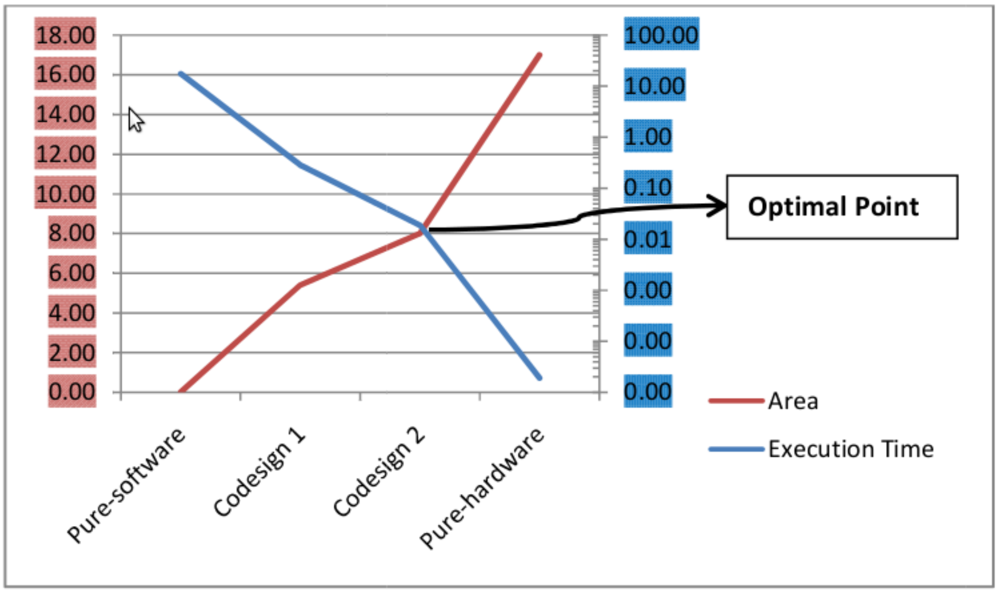


Figure 4.1: Graph showing the Design with Optimum area and delay

4.2 Testing

The system designed is tested for the voice-command recognition implementation. The algorithm was tested for its correctness by repeating the algorithm for 10 or more times. Table 4.3 shows the recognition rate of each command. For real time implementation, the system can be set to keep listening to the surrounding ambient noise. In this setup, the sound can be continuously recorded and analyzed to find whether any frame contains the uttered command. This can be done by

Table 4.3: Recognition Rate of the Design

Command	No. of times tested	No. of times the word was recognized correctly	Percentage of recognition
Set Forward	10	10	100%
Go Reverse	10	9	90%
Turn Left	25	20	80%
Drive Right	10	10	100%

incorporating a voice-activity detection module. A simple method of implementing this module is by evaluating the energy present in one frame and comparing with the threshold level. If the energy is greater, it would mean to indicate that the frame contains part of the voice command. Hence, all the frames which have acquired energy greater than the threshold level are sent to the feature extraction module and then to the recognition module. The voice-activity detection module based on energy calculation is implemented in this project. However since the efficiency of this module to detect the voice was not good, the total recognition rate was found to be less. Nevertheless novel algorithms for detecting the voice activity already exist in the literature. Such modules could be efficiently implemented in software with ease. Hence, the voice activity module was not included for the testing purpose of this project.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The Fire Bird V robot was built at IIT-Bombay which was made compatible with the following three microcontrollers as the main controller unit - Atmega2560, 8051 and ARM7 microcontrollers. An individual adaptor board for each of the above three microcontroller was designed. This adaptor board had been assembled on top of the base board containing the sensors and actuators of the robot. They wanted to design a FPGA-based adaptor board for the Fire Bird V base board. This problem was taken up and carried out by IIT-Hyderabad. The main sensors of the Fire Bird are integrated with the FPGA system. The system is designed to be flexible for the addition of the remaining sensors blocks and communication blocks of the Fire Bird. Apart from designing controllers for the present features of Fire Bird V, an additional feature of voice-command recognition unit is proposed and implemented on the Atlys Spartan 6 FPGA of Xilinx. The Xilinx ISE Design Suite 13.3 has been used in designing the system. An optimum hardware-software co-design for the feature extraction function has been developed on the FPGA platform. The implemented voice-command recognition system has almost 100% recognition rate when the input commands are spoken as per the reference words and by the same speaker. The bottleneck experienced in reducing the delay of the system is due to the software implementation of the dynamic time warping algorithm. Furthermore, the recognition time will increase with the number of reference templates. Due to this reason, the number of words was kept at only 4 for testing purpose. The feature extraction module has been efficiently designed optimizing both the area and the delay of the circuit.

5.2 Future Work

The project work can be extended as listed below.

- Improving the recognition algorithm to include additional words into the vocabulary
- Using the modified dynamic time warping algorithms which will help in decreasing the execution time of the algorithm
- Incorporating a wireless unit through which the voice commands can be sent even when the user is far from the robot

References

[1] http://oxforddictionaries.com/definition/robotics.

[2] EETimes. http://www.eetimes.com/design/industrial-control/4016917/A-tradeoff-between-microcontroller-DSP-FPGA-and-ASIC-technologies.

[3] EETimes. http://www.eetimes.com/design/embedded/4007108/Using-customizable-MCUs-to-bridge-the- gap-between-dedicated-SoC-ASSPs-ASICs-and-FPGAs-Part-1.

[4] N. Instruments. http://www.instrumentation.co.za/article.aspxpklarticleid=6575.

[5] XILINX. http://www.xilinx.com.

[6] P. Cosi, G. D. Poli, and G. Lauzzana. Auditory Modelling And Self-Organizing Neural Networks For Timbre Classification. *Journal of New Music Research* 23, (1994) 71–98.

[7] R. Vergin, D. OShaughnessy, and V. Gupta. Compensated mel frequency cepstrum coefficients. *1996 IEEE International Conference on Acoustics Speech and Signal Processing Conference Proceedings* 1, (1996) 323–326.

[8] R. Jang. Audio Signal Processing and Recognition.

[9] L.R.Rabiner and R.W.Schafer. Digital Processing of Speech Signals. Pearson Education, 2009.

[10] P. Senin. Dynamic Time Warping Algorithm Review. Ph.D. thesis, University of Hawaii, Manoa December 2008.

[11] X. Lv, M. Zhang, and H. Li. Robot control based on voice command. In Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on. 2008 2490 –2494.

[12] S. Li and H. Ren. An isolated word recognition system based on DSP and improved dynamic time warping algorithm. In Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on, volume 1. 2010 136 –139.

[13] A. Abushariah, T. Gunawan, O. Khalifa, and M. Abushariah. English digits speech recognition system based on Hidden Markov Models. In Computer and Communication Engineering (ICCCE), 2010 International Conference on. 2010 1 –5.

[14] H. Lee, S. Chang, D. Yook, and Y. Kim. A voice trigger system using keyword and speaker recognition for mobile devices. *Consumer Electronics, IEEE Transactions on* 55, (2009) 2377 –2384.

[15] H. Lim, K. You, and W. Sung. Design and Implementation of Speech Recognition on a Softcore Based Fpga. In Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on, volume 3. 2006 III.

[16] www.e-yantra.org/.

[17] R. Sharma. Log Approximation Fixed Point Arithmetic.

[18] T. Kamm, H. Hermansky, and A. G. Andreou. Learning the Mel-scale and Optimal VTN Mapping.

[19] D. Ning. Developing an Isolated Word Recognition System in MATLAB 2010.