الجمـــهوريـــة الجـــزائريـــة الديـمقراطيـــــــة الشـــعبيـــــة
THE PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزارة التعـلـيـم الـعـالـــي و الـبـحـث الـعـلـمي
THE MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
AMAR TELIDJI UNIVERSITY OF LAGHOUAT

كـلـيـــــة الـتكنولوجيا
FACULTY OF TECHNOLOGY

قسم الالكترونيك

DEPARTMENT OF ElECTRONIC

Master's dissertation

Domain : Science and Technology

Field : Electronic

Option : Electronic Embedded System

By: *Cotte mahdi*

THEME

FPGA-based power monitoring System for power converters

Mme. Abouchabana nabil	**Pr.**	**President**
Mme. Oubati ibrahim	**M.C. A**	**Examinator**
Mme . belkairi mohamed	**Pr.**	**Supervisor**

Academic year 2023/2024

الجمـــهوريـــة الجـــزائريـــة الديـمقـراطيـــــــة الشـــعبيـــــة
THE PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزارة التعـلـيـم الـعـالـــي و الـبـحـث الـعـلـمـي
THE MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
AMAR TELIDJI UNIVERSITY OF LAGHOUAT

كـلـيـــــة الـتكنولوجيا
FACULTY OF TECHNOLOGY

قسم الالكترونيك

DEPARTMENT OF ElECTRONIC

Master's dissertation

Domain : Science and Technology

Field : Electronic

Option : instrumentation

By: ***Sahouane mohamed***

THEME

FPGA-based power monitoring System for power converters

M^me. Abouchabana nabil	**Pr.**	**President**
M^me. Oubati ibrahim	**M.C. A**	**Examinator**
M^me. belkairi Mohamed	**Pr.**	**Supervisor**

Academic year 2023/2024

Abstract

This study investigates the development and implementation of a power monitoring system for power converters using the DE0-Soc Nano board. The first section details the design of an Analog-to-Digital Converter (ADC) with an SPI protocol to capture two channels—one for current and one for voltage—providing a foundation for accurate signal acquisition.

The second section focuses on processing these channels through Fast Fourier Transform (FFT) blocks to convert the signals from the time domain to the frequency domain. This transformation allows for precise measurement of the current and voltage magnitudes, which are crucial for harmonic analysis.

The third section explores the application of the CORDIC algorithm to determine phase measurements, enabling a deeper analysis of the power system's dynamics.

Finally, the last section discusses the transmission of magnitude and phase harmonic data via UART to an embedded processor, which is then displayed on a graphical user interface (GUI). This project demonstrates the effectiveness of using FPGA-based solutions for real-time power monitoring and harmonic analysis in power converter systems.

الملخص

تتحرى هذه الدراسة تطوير وتنفيذ نظام لمراقبة الطاقة لمحولات الطاقة باستخدام لوحة DE0-Soc Nano. يوضح القسم الأول تصميم محول تناظري إلى رقمي (ADC) باستخدام بروتوكول SPI لالتقاط قناتين: إحداهما للتيار والأخرى للجهد، مما يوفر أساسًا لاكتساب الإشارات بدقة.

يركز القسم الثاني على معالجة هذه القنوات من خلال كتل تحويل فورييه السريع (FFT) لتحويل الإشارات من النطاق الزمني إلى النطاق الترددي. يتيح هذا التحويل قياسًا دقيقًا لحجم التيار والجهد، وهو أمر أساسي لتحليل التوافقيات.

يستكشف القسم الثالث تطبيق خوارزمية Cordic لتحديد قياسات الطور، مما يمكّن من تحليل أعمق لديناميات نظام الطاقة.

أخيرًا، يناقش القسم الأخير نقل بيانات التوافقيات الخاصة بالحجم والطور عبر واجهة UART. إلى معالج مدمج، ثم عرضها على واجهة مستخدم رسومية (GUI) يوضح هذا المشروع فعالية استخدام حلول FPGA في مراقبة الطاقة وتحليل التوافقيات في أنظمة محولات الطاقة في الوقت الفعلي.

Résumé

Cette étude examine le développement et la mise en œuvre d'un système de surveillance de l'énergie pour les convertisseurs de puissance en utilisant la carte DE0-Soc Nano. La première section détaille la conception d'un convertisseur analogique-numérique (ADC) avec un protocole SPI pour capturer deux canaux : un pour le courant et un pour la tension, fournissant ainsi une base pour l'acquisition précise des signaux.

La deuxième section se concentre sur le traitement de ces canaux à l'aide de blocs de transformation de Fourier rapide (FFT) afin de convertir les signaux du domaine temporel au domaine fréquentiel. Cette transformation permet une mesure précise des amplitudes de courant et de tension, cruciales pour l'analyse harmonique.

La troisième section explore l'application de l'algorithme CORDIC pour déterminer les mesures de phase, permettant une analyse plus approfondie de la dynamique du système d'alimentation.
Enfin, la dernière section aborde la transmission des données d'amplitude et de phase des harmoniques via UART à un processeur embarqué, qui les affiche ensuite sur une interface utilisateur graphique (GUI). Ce projet démontre l'efficacité de l'utilisation de solutions basées sur FPGA pour la surveillance en temps réel de l'énergie et l'analyse harmonique dans les systèmes de conversion de puissance.

Acknowledgments

We thank Almighty Allah for giving us the courage, will, health and patience to complete this work.

I would like to express my sincere gratitude to my supervisor, Dr.belkairi Mohamed for their invaluable guidance and support throughout my master's project. Their expertise and encouragement helped me to complete this research and write this project.

I would also like to express my sincere appreciation to the members of the jury for their interest in my research and for agreeing to evaluate and enhance my work with their valuable suggestions. Their contribution is highly valued and appreciated.

Furthermore, I would also like to thank my friends and family for their love and support during this process. Without them, this journey would not have been possible.

Finally, I would like to thank all of the participants in my study for their time and Willingness to share their experiences. This work would not have been possible without their contribution.

CONTENTS

List of Figures

Creating and Implementing ADC

Creating and Implementing FFT & Cordic

List of symbols:

FPGA	Field programmable gate array
ADC	Analog-to-Digital Converter
SPI	Serial peripheral interface
FFT	Fast Fourier Transform
SOC	System on Chip
ICs	Integrated circuits
UART	Universal Asynchronous Receiver/Transmitter
I2C	Inter-Integrated Circuit
CS	Chip select
SCLK	Synchronous Clock
MOSI	Master Out Slave In
MISO	Master In Slave Out
DFT	Discrete Fourier Transform
CORDIC	Coordinate rotation digital computer
RISC	Reduced Instruction Set Computer
DSP	Digital Signal Processor

I.1 Introduction:

Power outages remain the most common cause of outages in data centers. The direct and indirect costs of such events are increasing every year. To avoid losing thousands of dollars and impacting continuity, end-user productivity, equipment lifecycles, and other key business metrics, companies leverage the system insights that power monitoring brings.

Power monitoring is one of the keys to avoiding unplanned downtime and the huge costs associated with it. In addition to detecting power issues that can cause outages, power monitoring solutions also play an important role in addressing other major challenges in data centers, namely improving energy efficiency and enabling better capacity planning. For data center managers who need to ensure 24/7 system availability, efficient operations, and future-proof facilities, power monitoring is an essential tool for protecting and optimizing data center environments.

Energy monitoring systems are widely used in industrial plants and buildings to monitor energy consumption. Compared to the commercial and industrial sectors, the residential sector consists of many small energy consumers, such as houses, mobile homes, and apartments. Studies have shown that these residential energy consumers waste nearly 41% of household electricity. Buildings need to measure voltage variations, energy consumption, power factor, and current parameters.

I.2 Project Objectives:

The project objective is developing an FPGA-based power monitoring system using the DE0-Nano-SoC board and an AD converter SPI. the goal is to analyze the voltage and current inputs to the device and send the harmonics to the graphical user interface

I.3 Project outline:

The starting point is the AD Converter, which will be two-channel: one for current and the other for voltage. We will use FFT to convert the time-domain signal into the frequency domain, extracting the magnitude of the power frequency (50 Hz) for both voltage and current, which is part of the required information. This data will then be transmitted through the embedded processor via UART. To obtain the phase, we will use the CORDIC algorithm, and all the collected information will be sent over UART from the embedded processor to the graphical user interface.

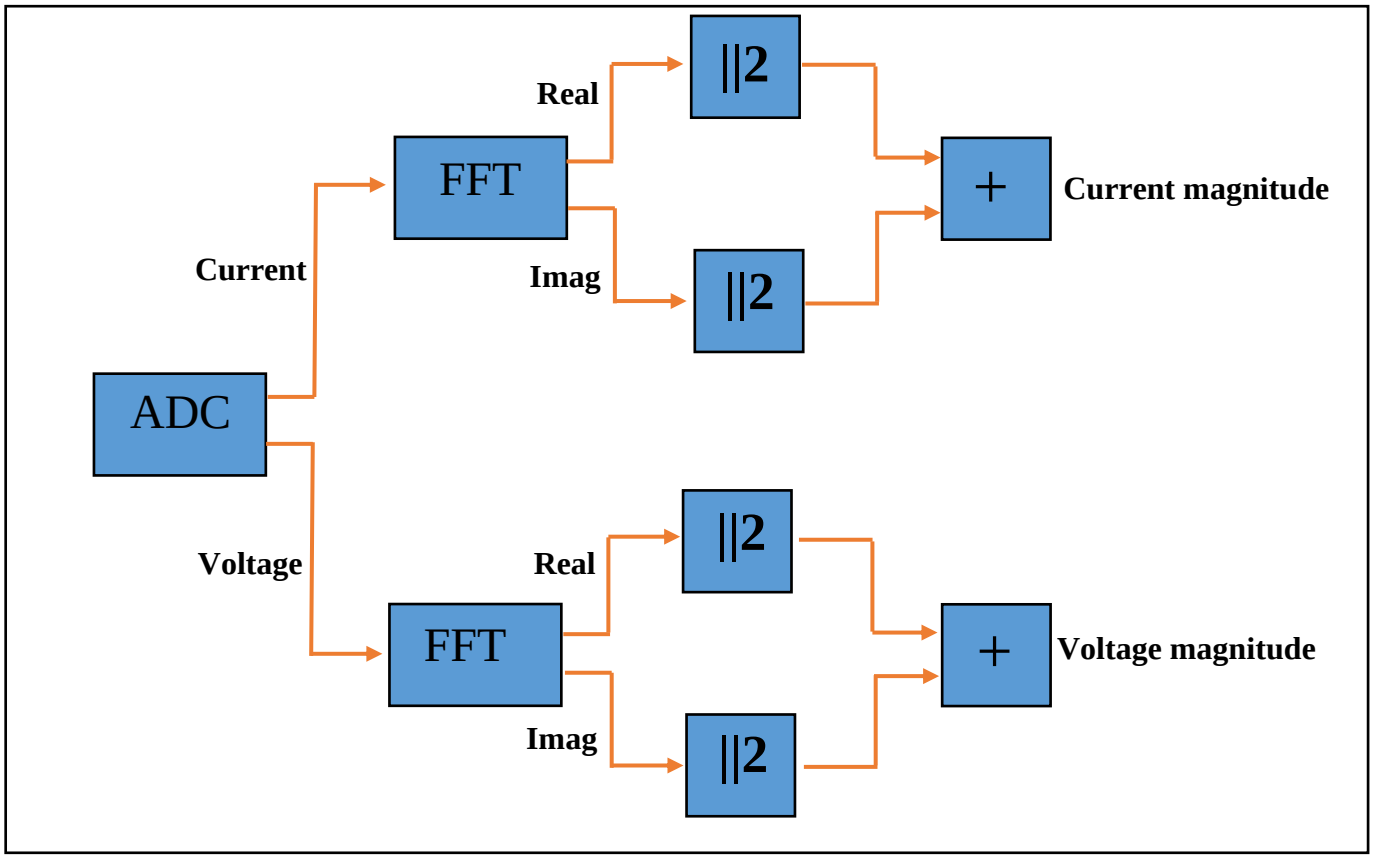


Figure I.1: FFT-Based Magnitude Extraction for Voltage and Current with UART Transmission

CHAPTER 1
Theoretical Background

I) Field Programmable Gate Array :

A field programmable gate array (FPGA) is a logic device that contains a two-dimensional array of general-purpose logic cells and programmable switches. The conceptual structure of an FPGA device is shown in Figure 1. The logic cells can be configured (i.e., programmed) to perform simple functions, and the programmable switches can be adjusted to establish connections between logic cells. Once the design is completed and synthesis is complete, a simple adapter cable must be used to download the desired logic cell and switch configuration to the FPGA device and obtain the custom circuit.[1]

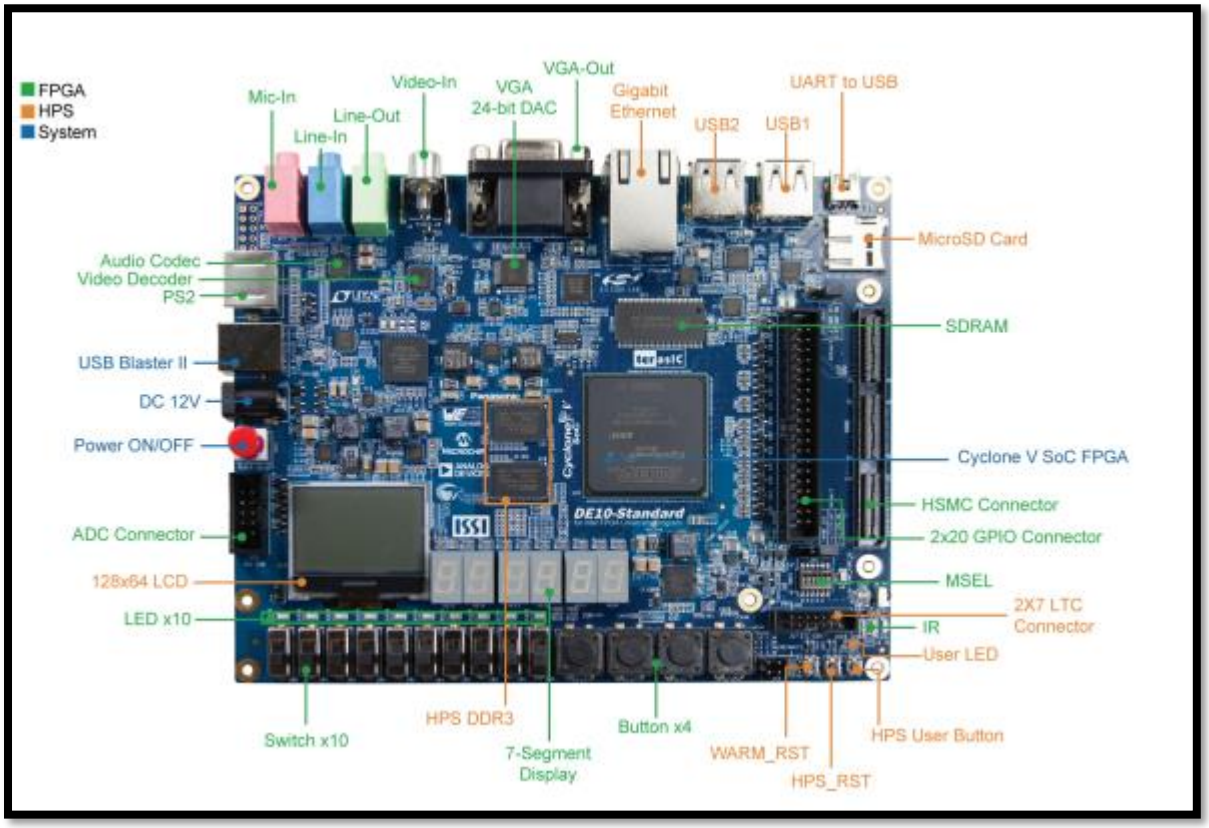


Figure I.2.1 field programmable gate array (FPGA)

I.1 DE-0 Soc Nano Board:

The DE0-Nano-SoC Development Kit presents a robust hardware design platform built around the Altera System-on-Chip (SoC) FPGA, which combines the latest dual-core Cortex-A9 embedded cores with industry-leading programmable logic for ultimate design flexibility. Users can now leverage the power of tremendous re-configurability paired with a high-performance, low-power processor system. Altera's SoC integrates an ARM-based hard

processor system (HPS) consisting of processor, peripherals and memory interfaces tied seamlessly with the FPGA fabric using a high-bandwidth interconnect backbone.[2]

I.1.1 Layout:

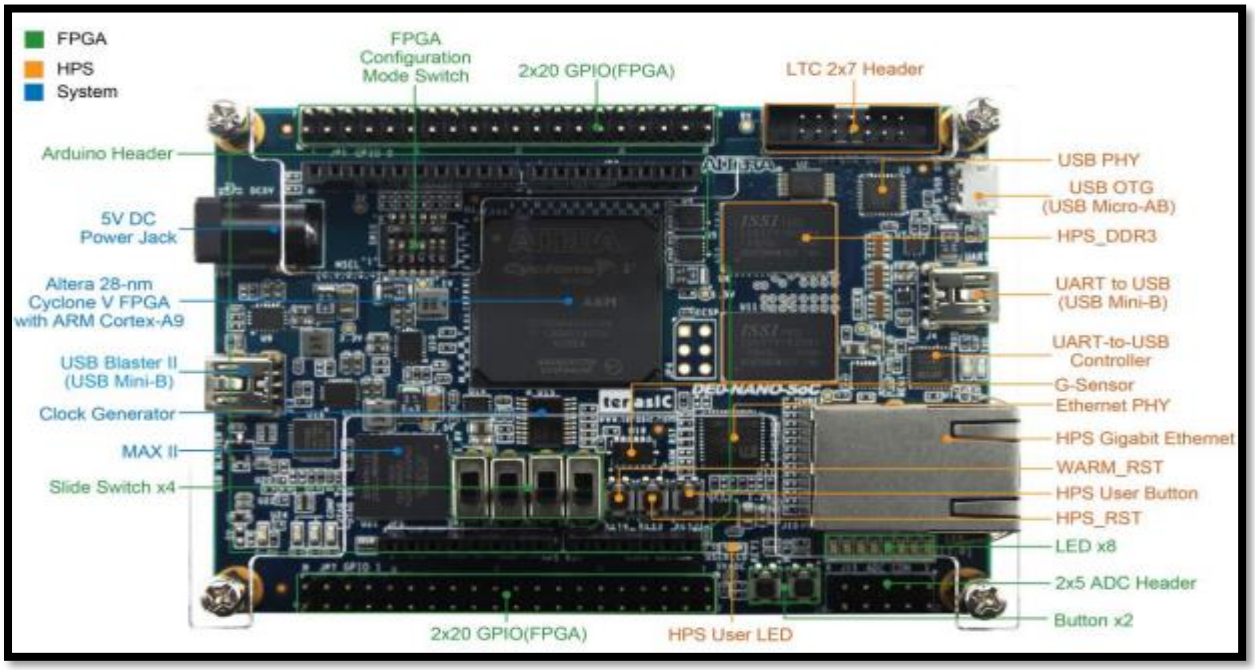


Figure I.2.2 DE0-Nano-SoC development board (top view).[2]

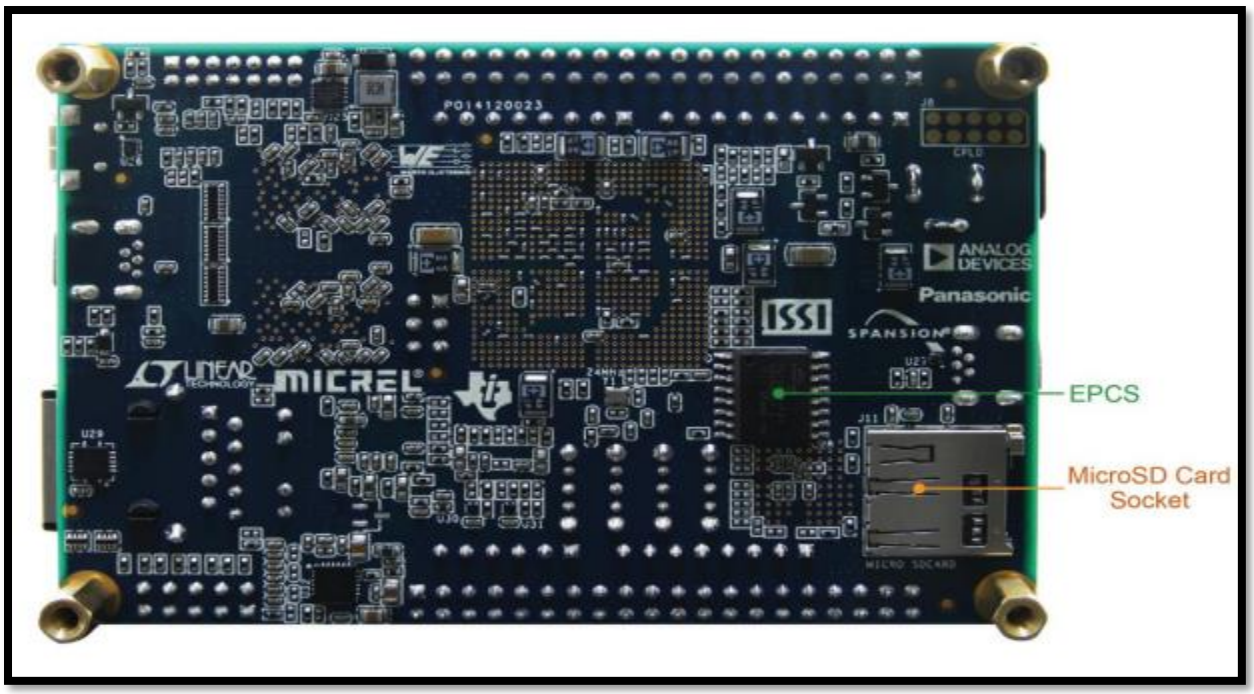


Figure I.2.3 DE0-Nano-SoC development board (bottom view).[2]

I.1.2 Applications of FPGAs:

Specific applications of FPGAs include digital signal processing, software defined radio, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation, radio astronomy, metal detection and a growing range of other areas.[1]

I.1.3 Block Diagram of the DE0-Soc Nano Board:

Figure 2-3 shows the block diagram of the DE0-Soc Nano board. To provide maximum flexibility for the user, all connections are made through the Cyclone V FPGA device. the user can configure the FPGA to implement any system design.[2]

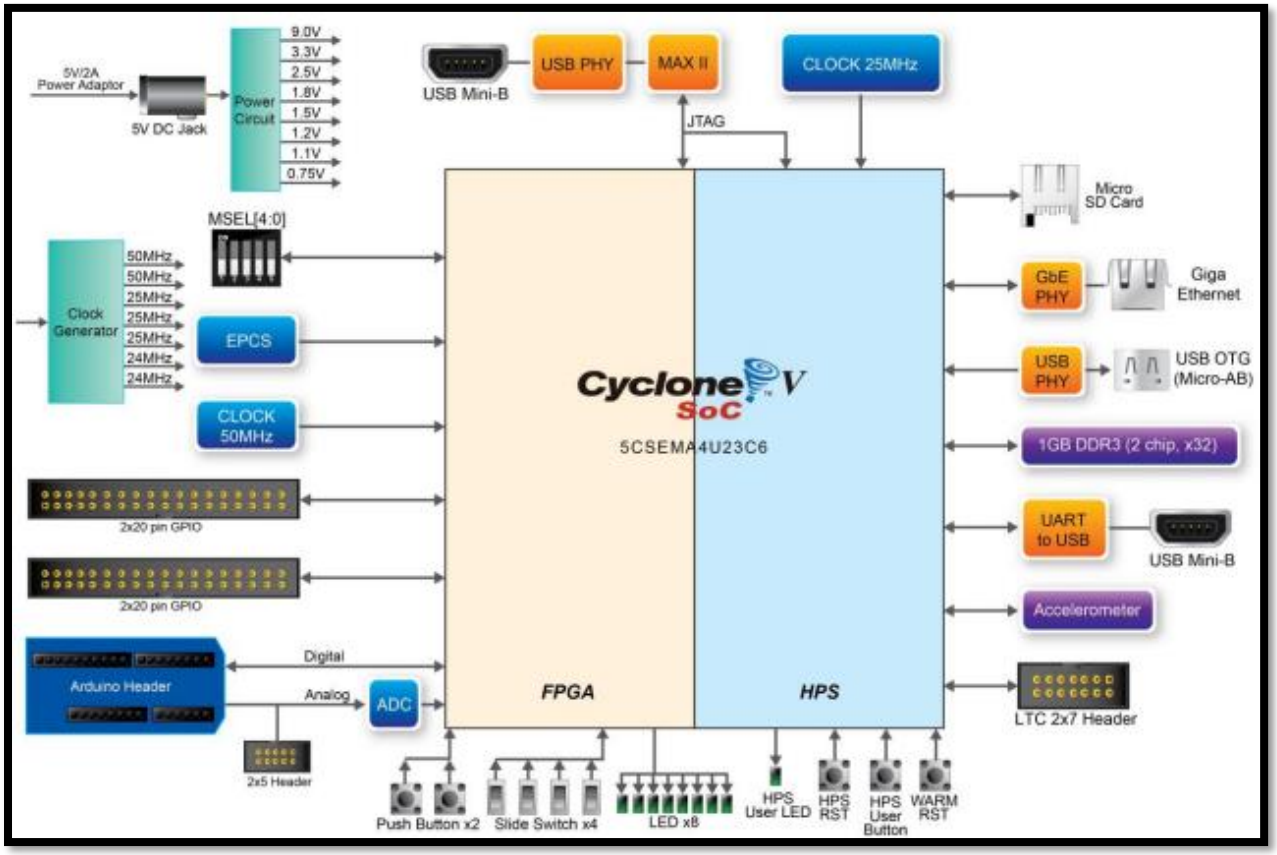


Figure I.2.4 Block diagram of DE0-Nano-SoC

I.1.4 Specifications:

- Cyclone V SoC 5CSEMA4U23C6N Device
- Dual-core ARM Cortex-A9 (HPS)
- 40K programmable logic elements
- 2,460 Kbits embedded memory
- 5 fractional PLLs
- 2 hard memory controllers

I.1.4.1 Configuration status and settings:

- Serial configuration device – EPCS on FPGA
- Onboard USB-Blaster II (Mini-B USB connector

I.1.4.2 Storage Devices:

- 1GB (2x256Mx16) DDR3 SDRAM on HPS
- Micro SD card socket on HPS

I.1.4.3 Communication:

- One USB 2.0 OTG (ULPI interface with USB Micro-AB connector)
- UART to USB (USB Mini-B connector)
- 10/100/1000 Ethernet

I.1.4.4 Connectors:

- Two 40-pin expansion headers
- Arduino expansion header
- One 10-pin ADC input header
- One LTC connector (one Serial Peripheral Interface (SPI) Master, one I2C and one GPIO interface)

I.1.4.5 ADC:

- 12-Bit Resolution, 500Ksps Sampling Rate. SPI Interface.
- 8-Channel Analog Input. Input Range: 0V ~ 4.096V.

I.1.4.6 Switches, Buttons, and Indicators:

- 3 user Keys (FPGA x2, HPS x1)
- 4 user switches (FPGA x4)
- 9 user LEDs (FPGA x8, HPS x 1)
- 2 HPS reset buttons (HPS_RESET_n and HPS_WARM_RST_n)

I.1.4.7 Power

- 5V DC input

I.2 AD Converter:

An ADC (Analog-to-Digital Converter) is a device or circuit that converts an analog signal, such as voltage or current, into a digital representation. In simpler terms, Analog-to-Digital Converters are used to connect analog devices (such as a microphones) to a digital system. The ADC performs the function of converting a continuous-valued analog signal into a discrete-valued digital one. Conversion involves a series of steps, including sampling, quantization, and coding.[3]

I.2.1 Applications for the ADC:

- Industrial Process Control
- Motor Control
- High Speed Data Acquisition
- Battery-Operated Instruments
- Isolated and/or Remote Data Acquisition
- Power Supply Monitoring
- Digital audio workstations
- TV tuner cards
- Microcontrollers

I.2.2 SPI (Serial Peripheral Interface):

A serial peripheral interface (SPI) is an interface commonly used in computers and embedded systems to facilitate short-distance communication between a microcontroller and one or more peripheral integrated circuits (ICs).[6]

- The SPI provides speed improvement over other serial data protocols such as UART or I2C and can support full duplex communication.
- The SPI most often used for transferring data between a smart controller and less smart peripheral device.

I.2.3 Applications for the SPI:

- The common applications include sensors, displays, ADC, real time clocks and wired gaming controllers and Battery Management Systems …etc

I.2.4 How Serial Peripheral Interface (SPI) work:

SPI components are commonly implemented in a 4-wire configuration. Each wire carries a specific type of signal between the controller and the peripherals and it uses the Master & Slave protocol to work.

I.2.5 SPI Master & Slave Protocol:

It's a protocol that uses the master as the controller and the slave as peripherals. Normally the master & slave are connected by four wires.

I.2.5.1 CS: chip select

The chip select wire is used to select which slave the master wants to communicate with.

I.2.5.2 SCLK: Synchronous Clock

The Synchronous Clock signal is used to provide timing and synchronization.

I.2.5.3 MOSI: Master Out Slave In

The data from the master to the slave is sent on MOSI.

I.2.5.4 MISO: Master in Slave Out

The data received by the master.

I.2.6 Simple SPI Protocol:

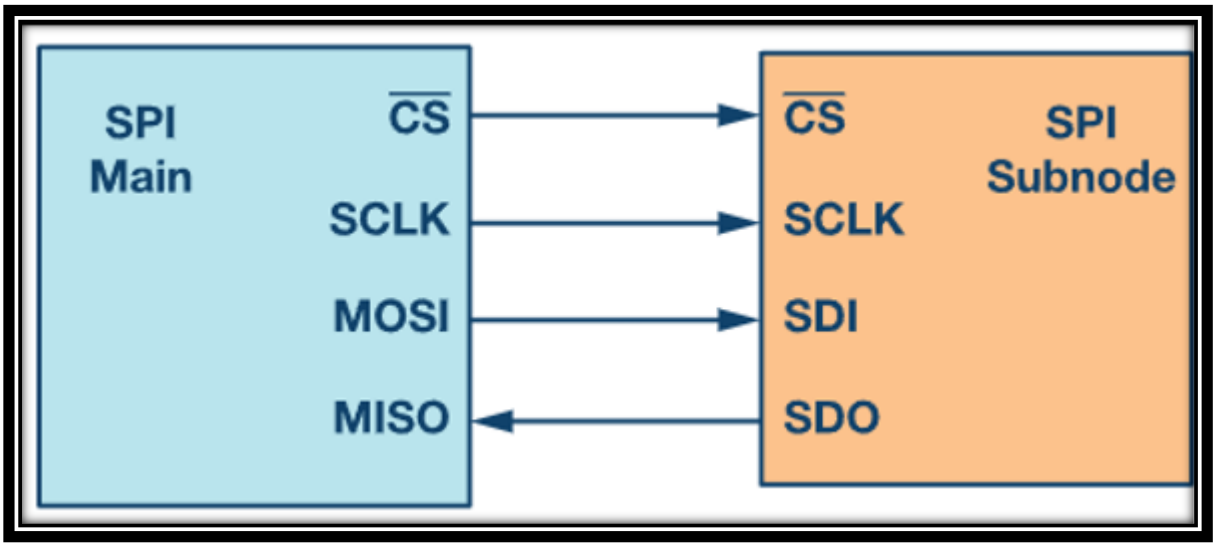


Figure I.3 SPI configuration with main (master) and a sub node (slave)

I.2.6.1 Multi Configuration SPI:

I.9 Cordic Algorithm:

I.9.1 Definition

CORDIC (coordinate rotation digital computer) , is a simple and efficient algorithm to calculate trigonometricfunctions, hyperbolicfuction, squareroots, multiplications, divisions, and exponentials and logarithms with arbitrary base, typically converging with one digit (or bit) per iteration. CORDIC is therefore also an example of digit-by-digit algorithms. CORDIC and closely related methods known as pseudo-multiplication and pseudo-division or factor combining are commonly used when no hardware multiplier is available (e.g. in simple microcontrollers and field-programmable gate arrays or FPGAs), as the only operations they require are additions, subtractions, bit shift and lookup tables. As such, they all belong to the class of shift-and-add algorithms. In computer science, CORDIC is often used to implement floating-point arithmetic when the target platform lacks hardware multiply for cost or space reasons.[12]

I.9.2 Cordic Overview:

- CORDIC is a collection of iterative algorithms using shifts and adds in lieu of multiplications in order to compute a number of vector rotations
- At each step it rotates vector by predetermined angle so as to eventually achieve a target angle
- Larger of steps yields a better precision, but at the cost of higher area or latency in HW

I.9.3 Angle Calculation:

Many applications require angle detection using contact free sensor solution products

I.9.4 Cordic Cosine and Sine Calculation:

- Applying the rotation transformation matrix on the unit i-direction vector results in the cosine and sine of the rotation angle

$$X' = X\cos\theta - Y\sin\theta$$

$$Y' = Y\cos\theta + X\sin\theta$$

The algebra behind this procedure is fairly straightforward. To rotate a vector by a given angle, we can use the following vector transformation, where alpha represents our target angle step. The result of each transformation takes on this new form. We are now left with two trigonometric equations derived from our known rotation angle that can be applied to any vector to perform this transformation.

To expedite the binary search for our target angle, we must determine a way to assess the value of both cosine and tangent of alpha

I.9.5 Vector Rotation:

$$x' = \cos\alpha(x + ytan(\alpha))$$

$$y' = \cos\alpha(x - ytan(\alpha))$$

$$m = \frac{1}{\prod_{n=0}^{i}\cos(\alpha i)}$$

Each step of the algorithm will transform the vector by a known value. This means that over the course of the entire process, we can obtain a fixed cumulative product. This scalar, represented as M, can be predetermined based on any known pattern of step sizes and may be

applied at the end of our algorithm once the required number of clockwise or counterclockwise rotations is complete. For now, let's temporarily ignore the cosine factor and shift focus to the remaining tangent portion of the equation. With our new simplified equations:

$$x' = x - y\tan(\alpha)$$

$$y' = y - x\tan(\alpha)$$

α	$\tan(\alpha)$	2^(-n)
45	1	1
22.5	0.414	0.5
11.25	0.199	0.25
5.625	0.098	0.125
2.8125	0.049	0.0625

Let's examine the impact of a true binary search. For each step, we can calculate the new vector component based on the tangent of our specific rotation angle. Here we can see that as alpha is halved with each step, the resulting tangent value progressively approaches zero. This pattern is the most efficient process to determine any angle, but it would require accessing a lookup table of known tangent values for every transformation step. While this is the ideal step pattern, let's explore what happens if we use a different pattern that similarly converges to zero with step sizes slightly larger than what we see here.

Consider instead if we were to replace the tangent of alpha by a factor of 2 raised to the negative nth power. We see that, similar to the tangent, this approaches zero as the number of iterations increases. Additionally, the value of this factor provides us with large enough step sizes to accomplish the full rotation to the x-axis while simultaneously allowing the algorithm to converge. The new step size is no longer a perfect binary search and may require a step or two more to converge, but it is simpler to implement using digital logic. We can perform this adjustment using just a rightward bit shift.

Bringing everything back together, we can now define the logic for this calculation of both the angle and magnitude of the starting vector. We must first define a variable that will indicate the direction of rotation. This must be determined at each step by checking which quadrant the vector is in before executing either a clockwise or counterclockwise rotation. We will remember that we defined a scalar M to help account for the cosine factor, which will be needed to ensure the final vector magnitude is correct. We can define the next iterative step for both vector components to follow the simplified form with direction and a rightward bit shift applied.

Thus, the angle is equal to the sum of all the predetermined angle shifts. The magnitude is the final one-dimensional vector divided by M. The final algorithm follows these steps for our original vector:

Let d_n= {1 for cw rotation ^ (_1) for cw rotation

$$x_{n+1} = x_n - y_n \times d_n \times 2^{-n}$$

$$y_{n+1} = x_n + y_n \times d_n \times 2^{-n}$$

$$\alpha_n = \alpha \tan(2^{-n})$$

$$m = \frac{1}{\prod_{n=0}^{i} \cos(\alpha i)}$$

$$magnitude = \frac{x_i}{m}$$

$$\theta = \sum_{n=0}^{i} d_n \times \alpha_n$$

I.9.6 Benefit of Cordic:

it is simpler to implement in an end system. It reduces the burden on the microcontroller and decreases calculation time by providing a pre-processed angular output. This relieves the microcontroller from the task of performing these calculations, allowing the entire process to be executed more quickly.

Let's first consider the total additional time for the microcontroller to gather the data after the magnetic field has been sampled and process the result. In a system reading from the TMAG 5170 using a 5-megahertz SPI clock, the MCU would need to perform two 32-bit reads, each requiring a minimum of 6.4 microseconds. There would be some additional delay between each read. Then, the MCU would need to perform the necessary arctangent calculation. In the end, the angle information would be available to the MCU after a time period greater than 12.8 microseconds. Depending on operating efficiency and speed, this delay might increase significantly.

If instead, we allow the device to integrate the CORDIC algorithm:

I.9.7 Microcontroller Calculations:

$$2*6.4 \text{ us} + T_{read\_delay} + T_{atan2} > 12.8 \text{ µs}$$

I.9.8 Cordic Calculations:

$$3 \text{ µs} + 6.4 \text{ µs} = 9.4 \text{ µs}$$

The device can accomplish the complete calculation in about 3 microseconds. Then, only a single SPI read is required to gather the angle information. Here, the time required to determine the angle information is predetermined, and the MCU is not burdened by this task. As the

rotation speed of the magnet increases, this delay will represent a fixed error that becomes increasingly significant. All efforts to reduce this delay benefit overall system accuracy.

For instance, consider a magnet spinning at a low speed of 100 rpm. This is equivalent to 600 degrees per second, and a three-microsecond delay represents only 1.8 millidegrees of error. If that rotation speed increases to 5000 rpm, the delay now results in about 0.1 degrees of change. Again, 3 microseconds here assume no additional delay between read events or in the arctangent calculation.

I.10) The Embedded Processor Nios II:

An embedded processor is a microprocessor that is used in an embedded system. An embedded system is a computer system that is designed to perform a specific task within a larger system. It is a self-contained system that is embedded within a larger device or system and is used to control the operation of the device.

The **Nios** II processor is the most widely used FPGA processor in the industry. Unlike a fully structured program, the logic is simply defined in HDL code and mapped to a common FPGA cell. This fer style is more than flexible. A simple core program can be configured and integrated by adding or removing system-wide features to match the performance or cost of.

The **Nios** II processor follows the basic principles of Reduced Instruction Set Computer (RISC) design and uses a small, well-designed instruction set.

Nios II is a 32-bit embedded memory controller designed for Altera's family of field-programmable gate arrays (FPGAs). **Nios II** includes many improvements over the original Nios architecture, making it more versatile for a wide range of embedded computers, from digital signal processing (DSP) to system control.[13]

I.10.1 Nios II Family:

Nios II is generally offered in 3 different models:

- Nios II/f (**fast**)
- Nios II/s (**standard**)
- Nios II/e (**economical**)

I.10.2 Nios II Features:

- **I.10.2.1** Nios II/f (**fast**):

- Discrete instruction and data cache (512 B - 64 KB).
- High-speed MMU or MPU.
- Up to 2 GB external address space.
- Onboard memory and instruction priority Six-phase network for DMIPS/MHz.

- access - hardware multiplexing and floating barrels.

- **I.10.2.2** Nios II/s (**standard**):

- Instruction cache.
- Up to 2 GB external address space.
- Instruction priority unified memory.
- Five-level network.
- Hard branch prediction.

- **I.10.2.3** Nios II/e (**economical**):

- Up to 2 GB of external address space.
- JTAG debug module.
- Complete systems in fewer than 700 LEs.
- Optional debug enhancements.

CHAPTER 2

II) Creating and Implementing ADC :

II.1 Creating ADC (spi) Interface:

1. Creating state machine for the ADC SPI
2. Write VHDL module for the interface code
3. Create BDF, import our working interface module and apply a clock
4. Assign the correct pins
5. Program the DEO nano board
6. Check whether its working by using Signal tap

II.1.1 ADC SPI State Machine:

- A state machine is a behavior model. It consists of a finite number of states and is therefore also called finite-state machine (FSM). Based on the current state and a given input the machine performs state transitions and produces outputs. [8]

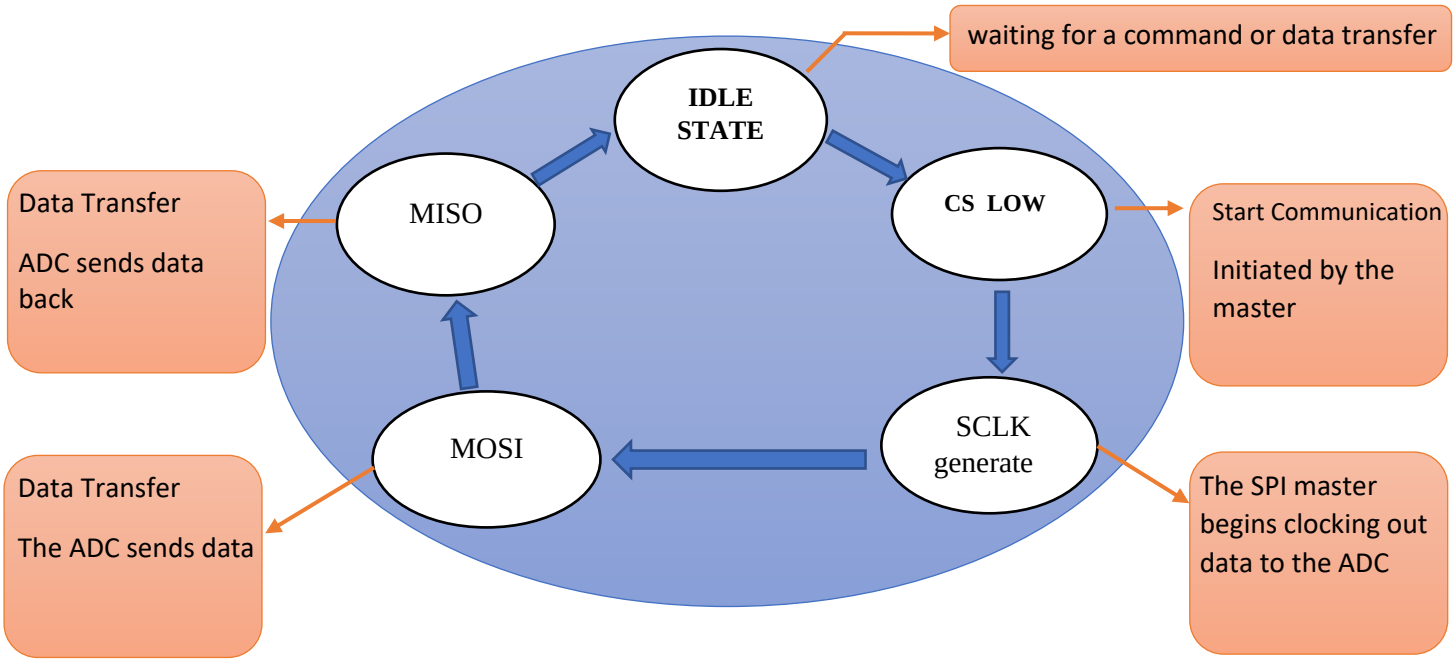


Figure II.1 ADC SPI State Machine

II.1.2 The Vhdl Code:

II.1.2.1 Module Ports:

clk: Input clock signal.

sclk: Output wire representing the SPI clock.

din: Output wire representing serial data input.

dout: Input wire representing serial data output.

cs: Output wire representing the chip select signal.

count: Output wire representing the count.

dataout: Output wire representing the output data.

clock_out: Output wire representing an additional clock signal.

II.1.2.2 Internal Signals and Registers:

data_temp: 12-bit register used for storing temporary data.

ADD2, ADD1, ADD0: Control signals.

count: 4-bit register for counting clock cycles.

II.1.2.3 Initial Block:

Initializes various registers and signals.

II.1.2.4 Processes:

- **Process 1**: This process is sensitive to the clk signal's falling edge. It checks the value of count_reg and sets the chip select (cs) signal low when count_reg equals "0001".

- **Process 2**: This process is sensitive to the rising edge of clk. It increments the count_reg by 1.

- **Process 3**: This process is sensitive to the clk signal and uses a case statement to assign values to the din signal based on the value of count_reg.

- **Process 4**: This process is also sensitive to the clk signal and utilizes a case statement to handle the data output (dataout) and an additional clock output (clock_out) based on the value of count_reg

II.1.2.5 Behavioral Logic:

- The sclk signal is assigned based on the logical OR operation between cs and clk.
- The din signal is assigned different values (ADD2, ADD1, or ADD0) based on the value of count_reg.
- The dataout signal is assigned the value of data_temp when count_reg is "0011", and data_temp is loaded with dout values when count_reg is between "0100" and "1111". Additionally, clock_out is controlled based on specific values of count_reg.

II.2 Create BDF and Import Our Working Interface:

✦ creating the block diagram file can be done by going to:

FILES > NEW > DESIGN FILES > BLOCK DIAGRAM

- By doing this we create an empty new block diagram page.

✦ inserting the symbol for our ADC by:

- Setting the ADC code as top entity

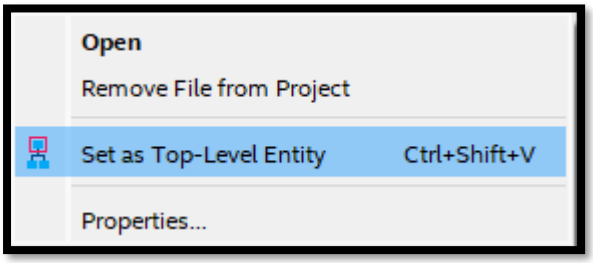


Figure II.1.1 top level entity

- Clicking on analysis & elaboration

Figure II.1.2 analysis & elaboration

- Right click on the ADC CODE file and Create symbol file for current file

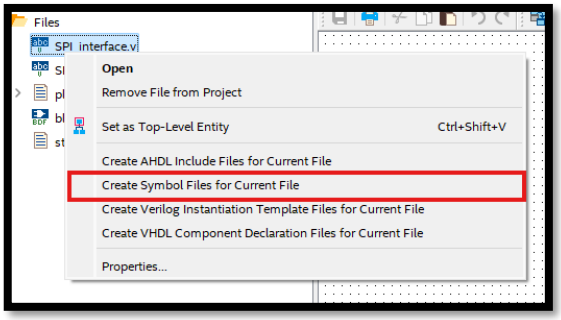


Figure II.1.3 Creating symbol file

- After creating the symbol file for the ADC interface, we go back to the block diagram we created in the previous steps and insert the symbol file inside the blank space on the block by double clicking the space and choose the interface block (SPI_Interface)

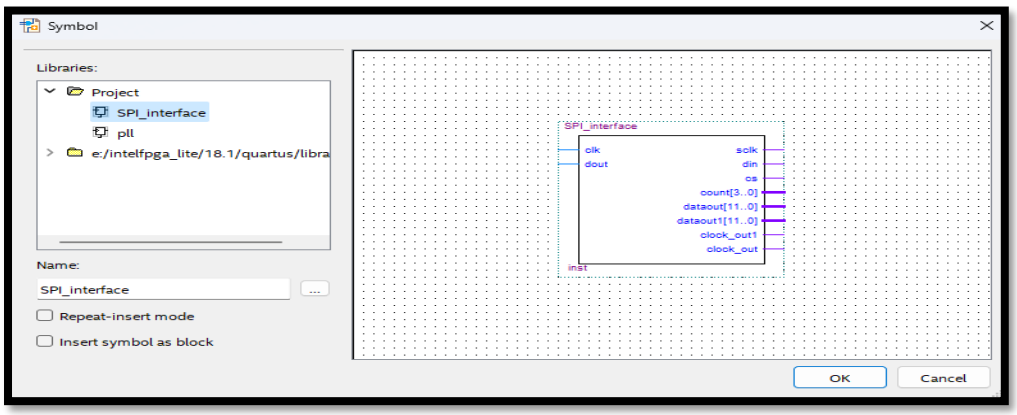


Figure II.1.4 ADC SPI_interface block symbol.[16]

- The PLL will be the next thing to add in the block diagram from the IP Catalog in quartus II which is located on the right side.

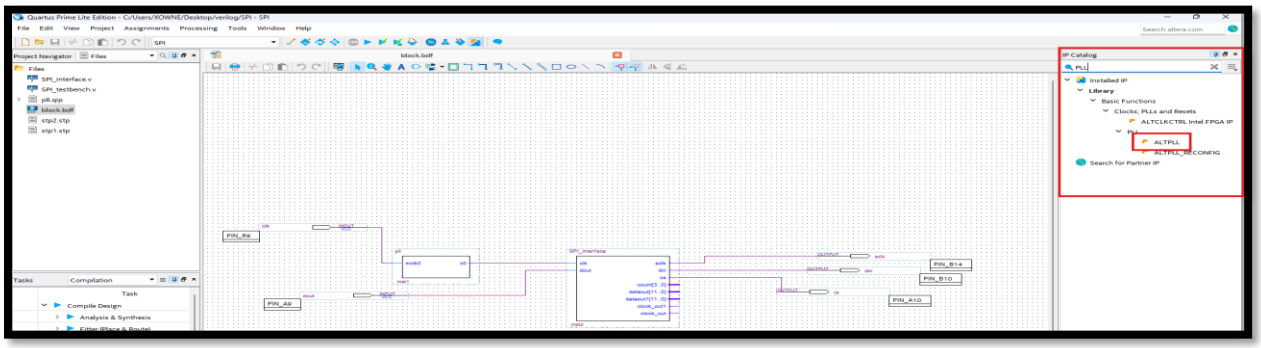


Figure II.1.5 phase locked loop location

II.2.1 Assign the Correct Pins:

Assigning the correct pins in Quartus II is crucial because it ensures that the FPGA design interfaces properly with the external hardware.

Assigning the pins can be done by clicking on:

ASSIGNMENT > PIN PLANER

That is lead us to this window:

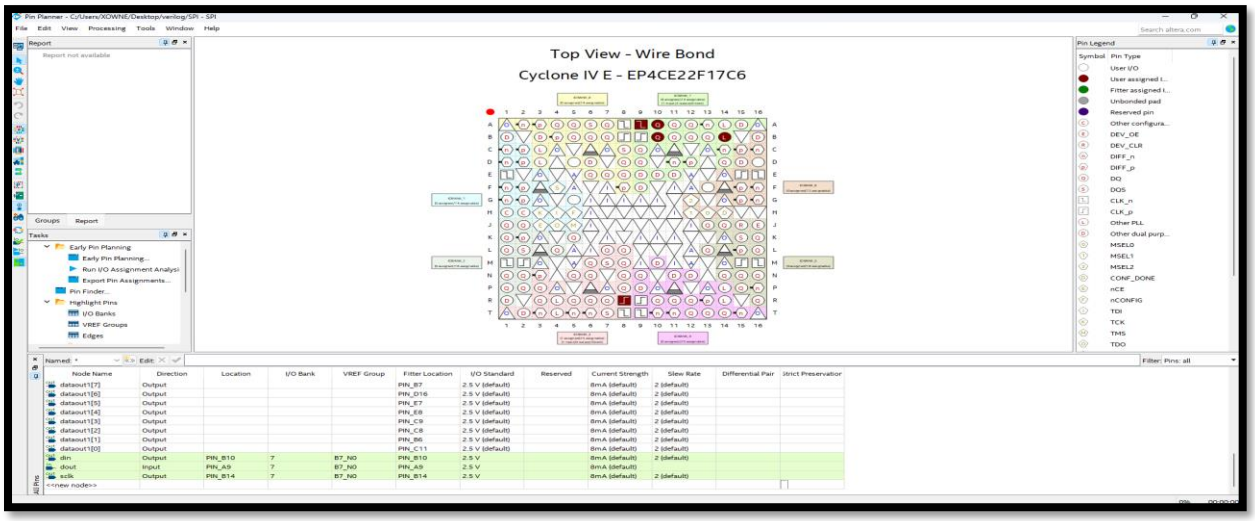


Figure II.1.6 Cyclone IV E Pin_Planner

II.2.1.1 the Pin Assignment for the Inputs and Outputs

CLK: PIN_R8

DOUT: PIN_A9

CS: PIN_A10

SCLK: PIN_B14

DIN: PIN_B10

II.2.2 Adc Block Diagram File:

To create ADC block diagram file, we need 4 things:

- Input signal.
- PLL (phased locked loop).
- ADC interface block.

- PIN ASSIGNMENT.

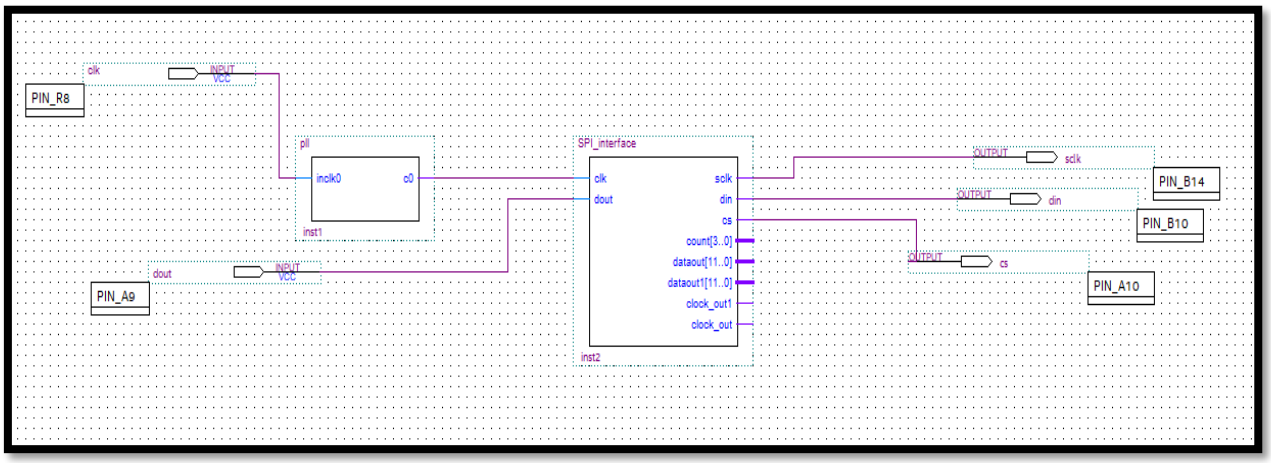


Figure II.1.7 ADC interface quartus ii block diagram.[16]

II.2.3 Program the DEO Nano Board & Signal Tap:

Programming SignalTap on Quartus II involves setting up the SignalTap Logic Analyzer to capture and analyze signals in your FPGA design. Here are the steps to set it up:

- **II.2.3.1 Step 1: Create Quartus Project**
  - Open Quartus II and create a new project or open an existing project.
  - Make sure your files (VHDL, Verilog, etc.) are added to the project.

- **II.2.3.2 Step 2: Install the Signal Tap Logic Analyzer**
  - **Open Signal Tap**: Go to Tools > SignalTap II Logic Analyzer.
  - **Create a new file**: Create a new SignalTap file (.stp) if desired.
  - **Set Time**: Assign a time signal to the time line in the SignalTap window. This clock should be a free clock in your drawing.
  - **Add a node to transform**: Click in the node finder (or right-click in the data window and select Add Node or Bus). Select the signal you want to monitor in the Node Finder and add it to the Data window.

- **II.2.3.3 Step 3: Compiling the Design**
  - **Compiling the Project**: Make sure your design has been created using the included SignalTap Logic Analyzer.

    Go to Edit > Start Collection.
- **II.2.3.4 Step 4: Modify the FPGA program FPGA:** Use the Programmer tool to configure the FPGA with your design.

  Go to Tools > Applications.

  Confirm the selected file and prepare the device.

- **II.2.3.5 Step 5: Start the SignalTap Logic Analyzer.**
  - **Start the analyzer:** Return to the SignalTap window. Make sure the settings and triggers are correct. Click the Run Analysis button (the green one).
  - **Data Capture:** The analyzer begins capturing data according to the defined parameters and displays the results in the stamp window.
  - **Analyze results:** Use the row view to analyze recorded data. You can zoom in and out, pan, and use shape controls to adjust your design.

II.2.4 The Nodes for The ADC Test:

- Conv
- Conv_Clock
- SCLK
- SCLK2
- Serial_Data
- Serial_Prog
- STMP
- STMP2

II.2.5 Adc Signal Tap Result:

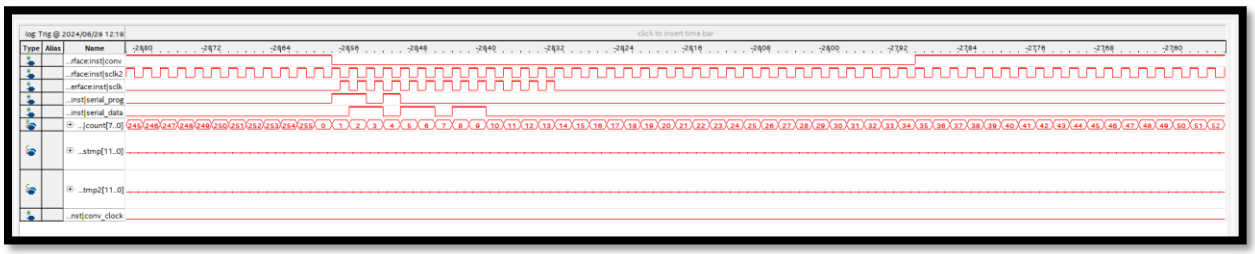


Figure II.1.8 ADC SignalTap Logic Analyzer timing diagram channel - 1

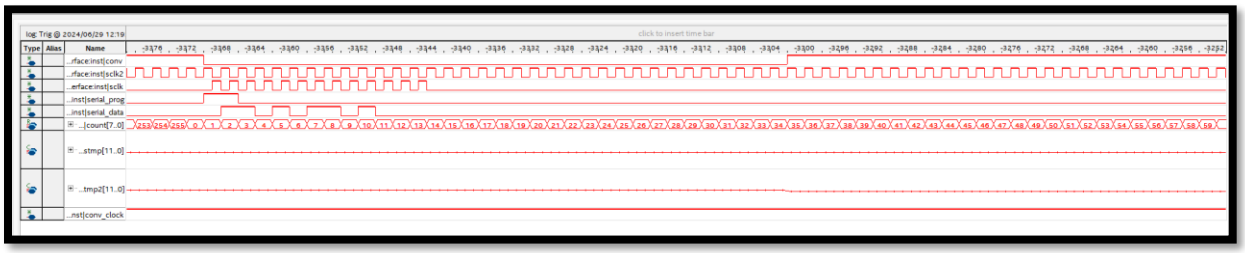


Figure II.1.9 ADC SignalTap Logic Analyzer timing diagram channel – 2

We are looking at the pins associated with the ADC including the SCLK, the conv (convert pin and also called chip select), serial prog (Dout) and serial data (Din).

- The conv will initiate the start of the conversion
- Serial data carries the 12-Bits of data.
- Serial prog carries the programming with it to the ADC
- SCLK is the 12 rising edges

II.3 Creating and Implementing 2 FFTs with Interface Code:

- Creating FFT interface code
- Create BDF, import our working interface module
- Connecting the FFT block with the ADC block
- Testing the FFT

II.3.1 Creating FFT Interface Code:

The interface code must work as channel to send the voltage into FFT block and the current into another FFT block.

The fft_wrapper module is designed to perform a Fast Fourier Transform (FFT) on an input signal. It takes in a signal, processes it using an FFT, and then outputs the real and imaginary components of the frequency spectrum.[9]

II.3.2 Inputs and Outputs:

II.3.2.1 Inputs:

- clk: A clock signal that times the operations.
- in_signal: A 12-bit input signal that will be transformed by the FFT.

II.3.2.2 Outputs:

- real_power: The real part of the FFT output (the frequency spectrum).
- imag_power: The imaginary part of the FFT output.
- fft_source_sop: Signal that indicates the start of the FFT output packet.
- sink_sop: Signal that indicates the start of the input packet to the FFT.
- sink_eop: Signal that indicates the end of the input packet to the FFT.
- sink_valid: Signal that indicates the input data is valid and ready to be processed.

II.3.2.3 Internal Operations:

- **II.3.2.3.1 Reset and Counter :**

- At the start (initial block), a reset signal (reset_n) is set to 0 (active reset), and a counter (count) is initialized to 0.
- The counter increments with each clock cycle. After 10 cycles, the reset signal is deactivated (reset_n = 1), allowing the FFT to begin processing.

- **II.3.2.3.2 Control Signals for FFT:**

- The control_for_fft module generates control signals and prepares the input data for the FFT. It sets up when the input data starts (sink_sop), when it ends (sink_eop), and when it is valid (sink_valid).
- It also takes the input signal (in_signal) and separates it into real and imaginary components, which are then sent to the FFT.

- **II.3.2.3.3 FFT Processing:**

- The fft_inst is the main FFT block that performs the actual transformation. It takes the prepared input data and processes it according to the control signals.
- The FFT outputs the real and imaginary parts of the transformed data (real_power and imag_power), along with signals indicating the start and end of the output packet.

II.3.3 Overall Functionality:

The fft_wrapper module wraps around an FFT processor, managing the input signals and control signals required for FFT operation. It ensures the FFT starts processing at the correct time, and it outputs the frequency spectrum of the input signal in the form of real and imaginary parts.

II.3.4 Create BDF and Import the Interface Module:

creating block diagram file for the FFT interface block can be done by :

- Setting the FFT interface code as top entity.
- analysis & elaborate the code
- RHIGH CLICK ON THE FFT interface CODE FILE
- Selecting Create symbol file for current file
- insert the symbol file inside the block diagram by double clicking on the blank space and choose the FFT_interface file

insert (ADD, MULT, SQRT) blocks to the diagram and connecting them with the FFT_wrapper to get the fft part ready.

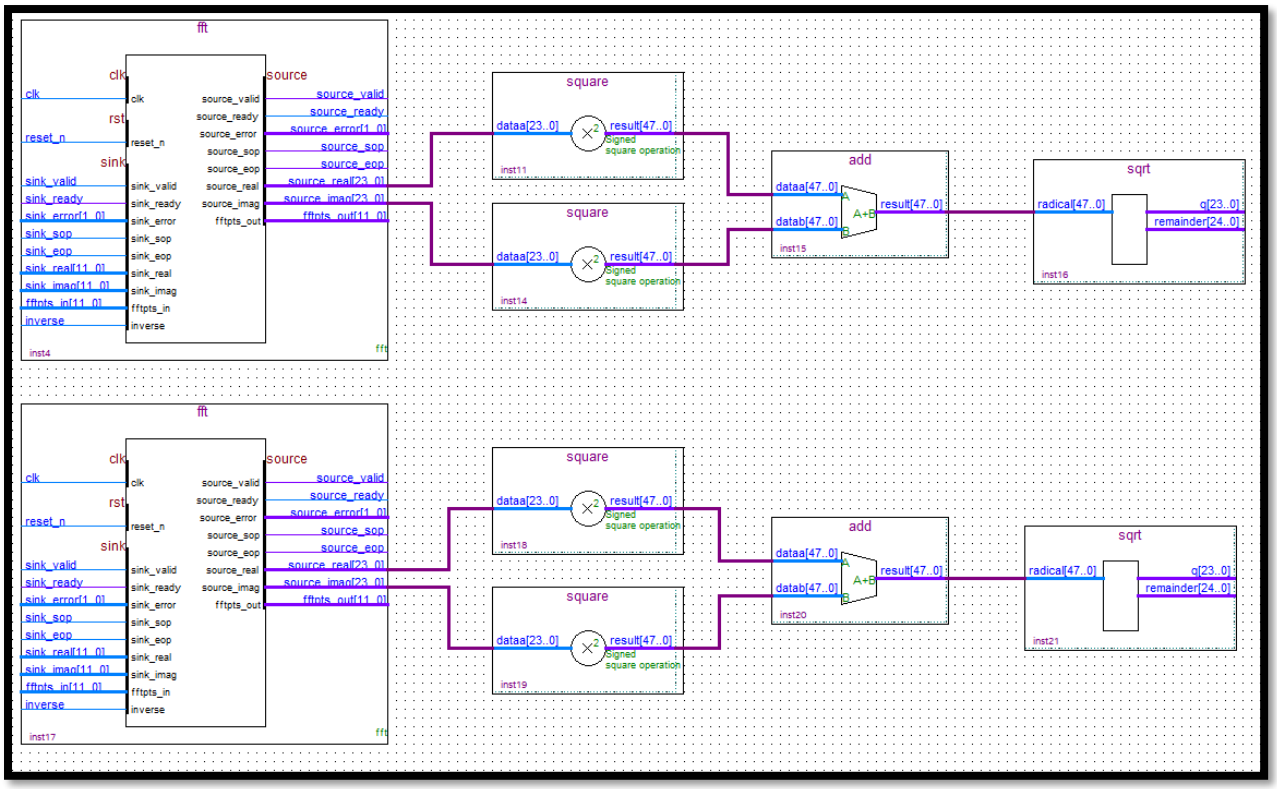


Figure II.2 FFT interface module.**[15]**

II.3.5 Connecting the FFT with the ADC:

The integration of **Fast Fourier Transform (FFT)** and **Analog-to-Digital Converters (ADC)** plays a critical role for the harmonics, as shown in the (Figure V-3)

1) the ADC convert the conditioned analog signal into a digital signal that can be processed by the system.

2) The **control_for_ff**t is used to control the input and output flow for an FFT (Fast Fourier Transform) operation. It generates control signals like start of packet (SOP), end of packet (EOP), and validity signals that are used by the FFT processing block.

3) The **FFT IP block** (Intellectual Property block) is a pre-designed and optimized software component used to perform Fast Fourier Transform (FFT) operations in digital systems.

4) the Square, Addition and Square Root blocks are meant for the real and imag outputs to represent the magnitude equation:

$$MAGNTUD = \sqrt{REAL^2 + IMAG^2}$$

5) the Serie to parallel module is used to process voltage and current signals in series, and extract their harmonic components, and convert this information to a Parallel output format.

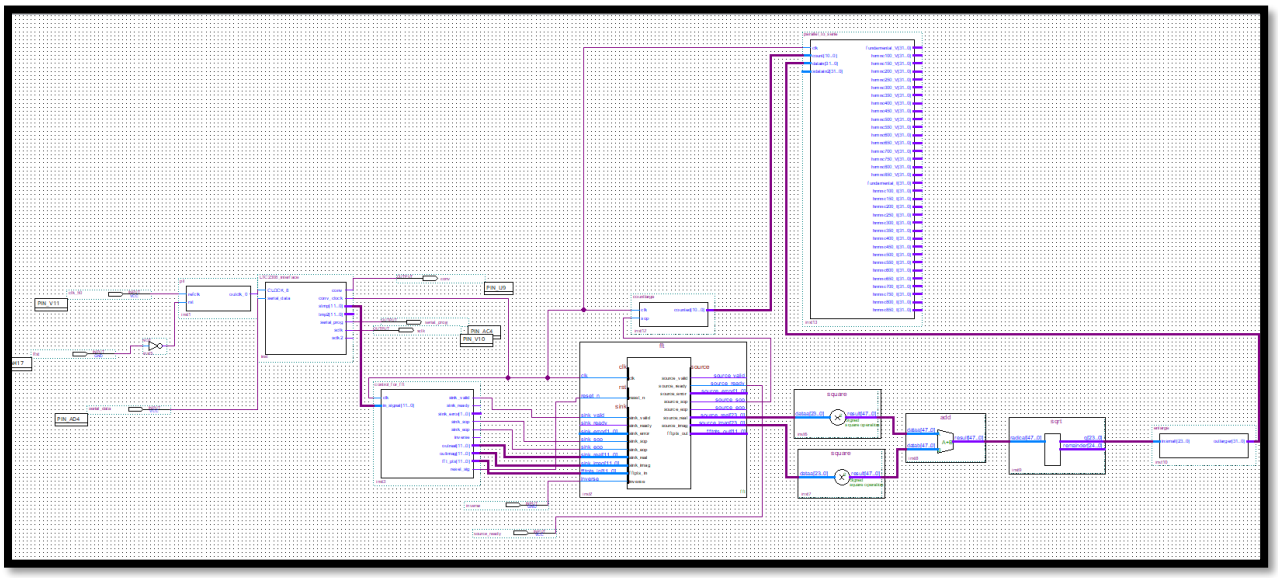


Figure II.2.1 ADC & FFT.[15]

II.3.6 Testing Results:

FIGURE II.2.2 FFT Signal Tap result Overview -1

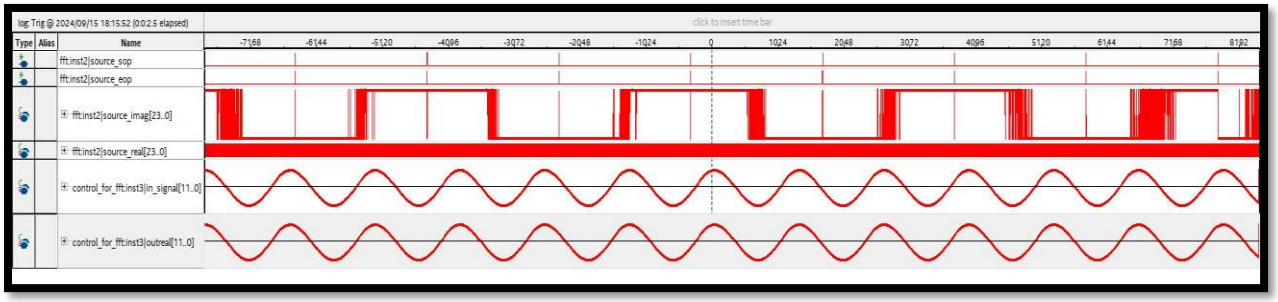


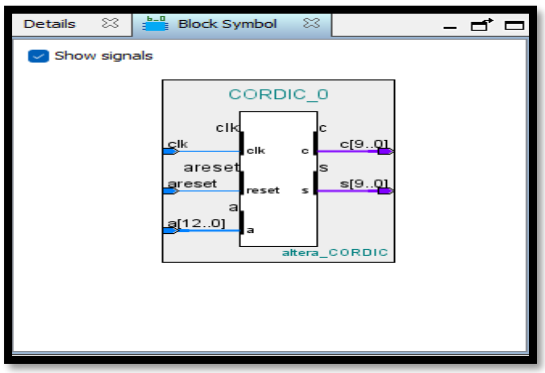
FIGURE II.2.3 FFT Signal Tap result Overview -2

II.4 Creating and Implementing Cordic :

The cordic will hold the role of converting from rectangular (Real and imaginary) to polar (Magnitude and phase) . so this is why we are using two cordic blocks , one of them for the voltage channel and one for the current .

II.4.1 Implementig the Cordic :

The cordic block can be found in the IP catalog and its easy to insert it to the block diagram



II.4.2 Connecting the Cordic :

This can be achived by connecting the output of the FFT, the real and imaginary of the two FFTs (voltage and current) to the input of the cordic .

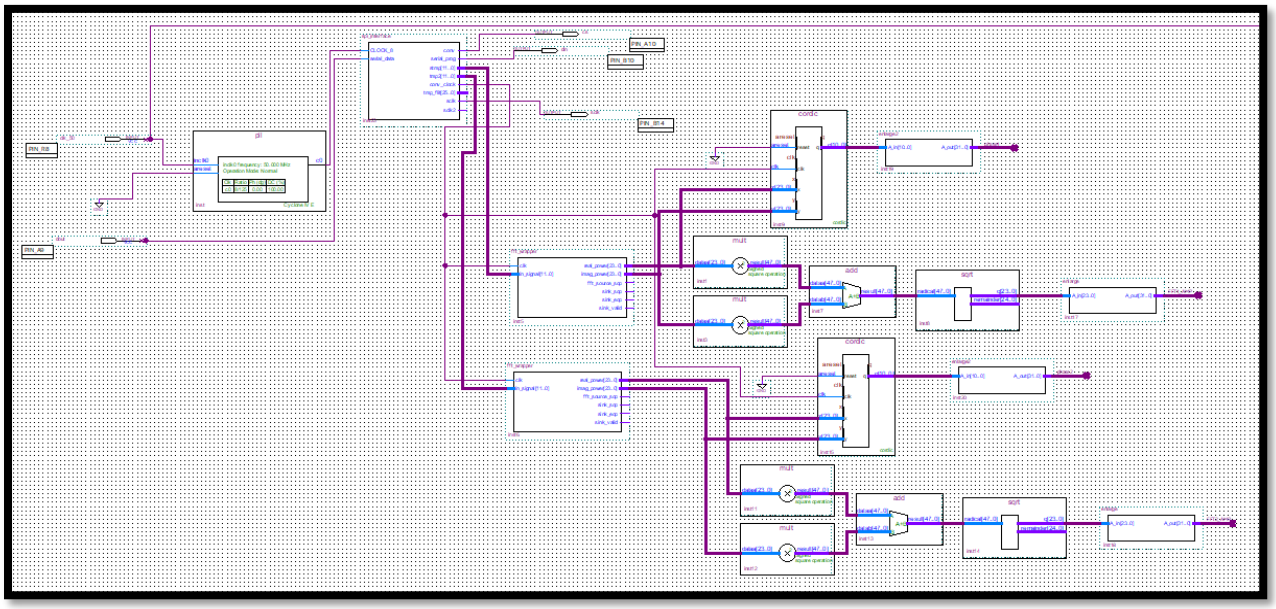


Figure II.3 CORDIC algorithm implementation.[15]

CHAPTER 3

III) Creating an Embedded Processor on an FPGA

Simply the embedded processor is using the logic elements to preforme the function of a processor and the number of logic elements necessary is around 3000 logic elements for a basic embedded processor . but the soon you have this it will allows you to run C and C++ code which is advantage because it gives you something that the hardware in the FPGA cant give you .[13]

III.1 Communicate with the Embedded Processor :

This can be done through " WRITE Master " .

The WRITE MASTER it's a piece of IP which is produced by intel so we have to embed it into the embedded processor and then we can write the information into the memory scope of the embedded processor from outside . so with our code we can target a particular base address within the embedded processor.

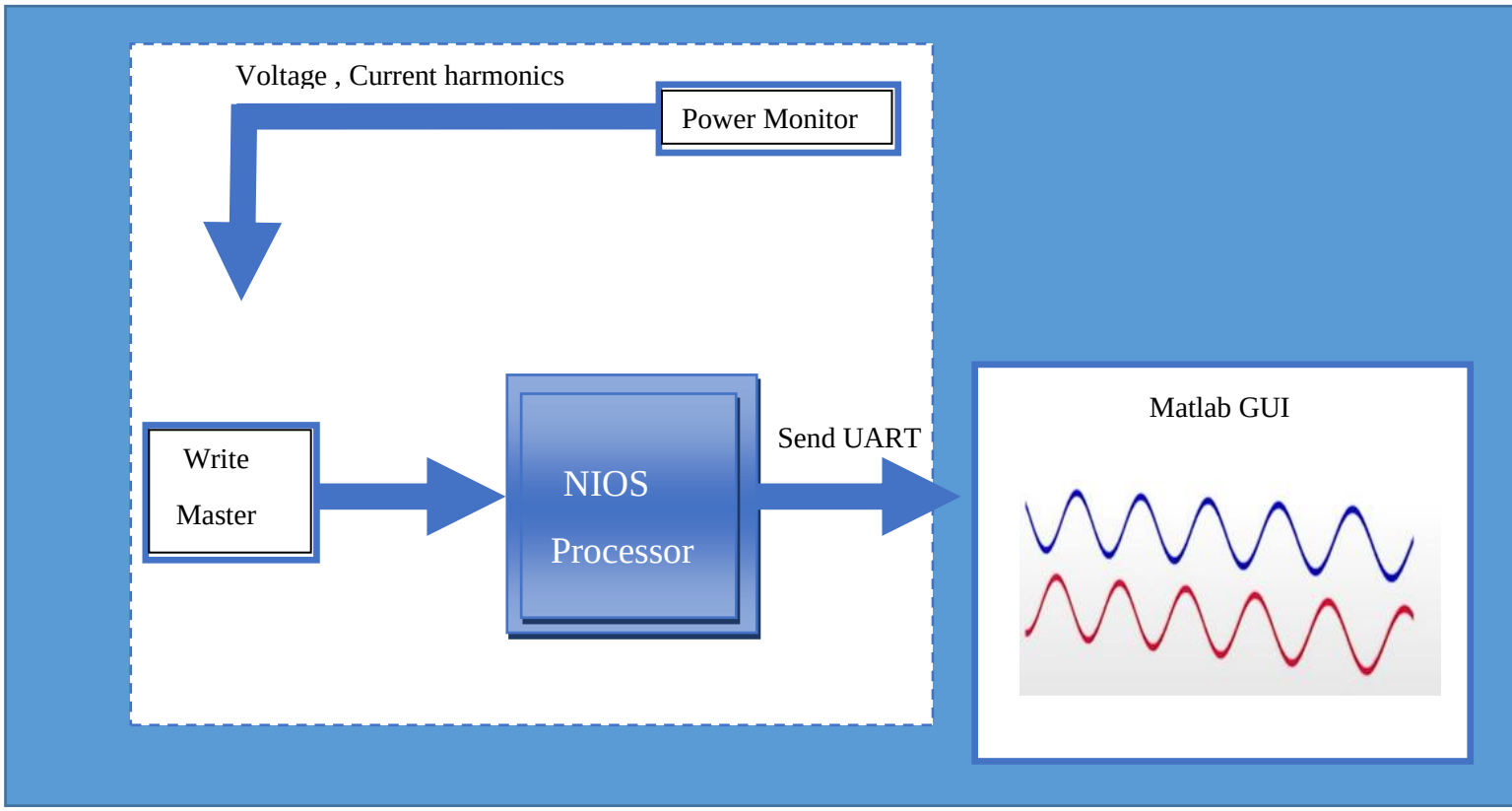
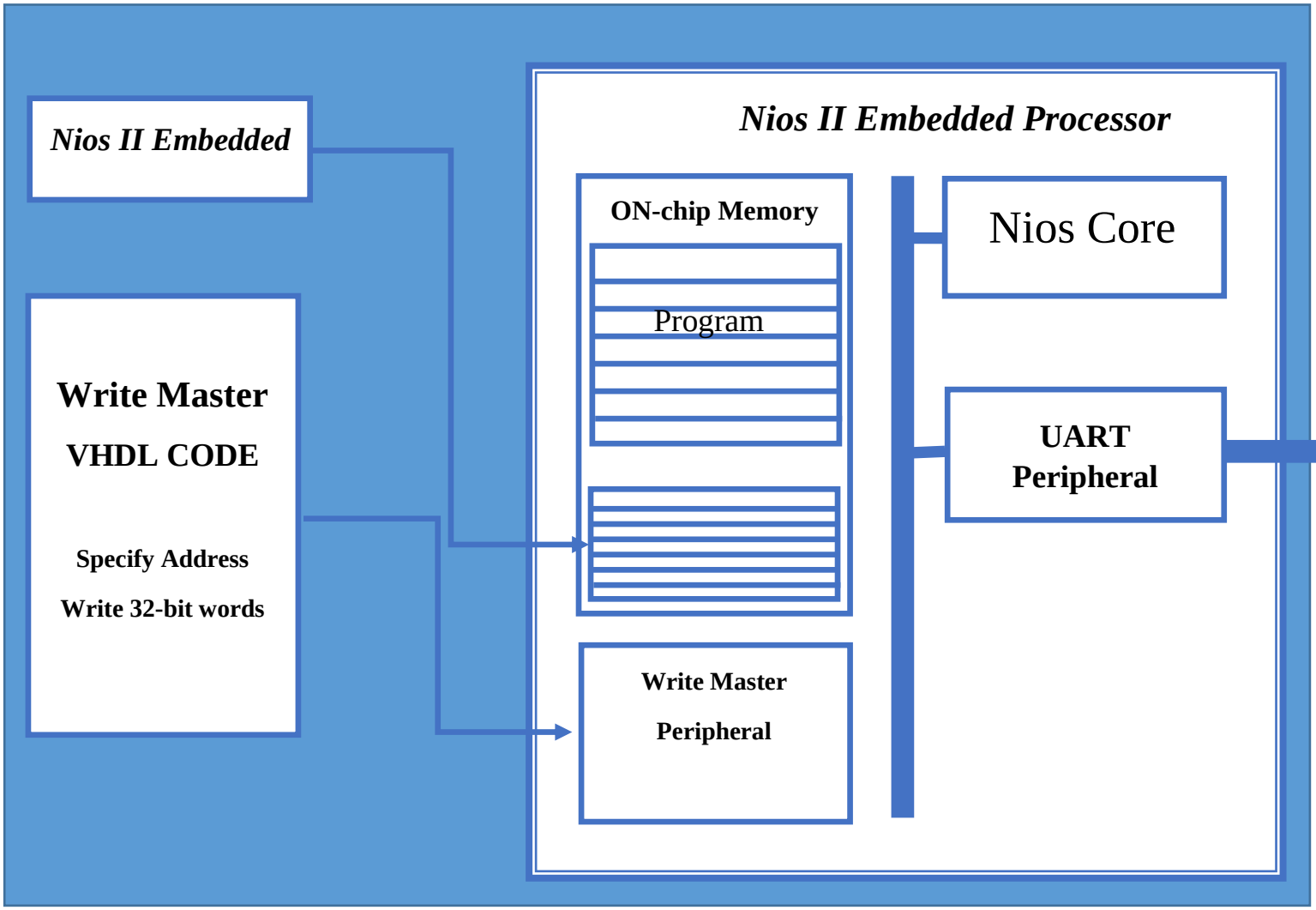


Figure III.1 Write master overview

Our processor is made up of peripherals so we've got the nios Core it self and we've got the UART for sending the information out and we also have ON-chip memory and we know a portion of this ON-chip memory will be used for the program it self.

Now we need to be carful because our desing is all on ship memory because the memory address we write to is away from where the program is stored .

When we use the Write master we can tell it to write to a specific address on the ON-chip memory so both the write master and our VHDL code and the Nios processor know the base address of the ON-chip memory so we can actually write to a specific address and then as we've seen it's quite trivial then to use the PRINT F command which is the easiest and the most basic command we in software to send the contants of those memory addresses over the UART so now its about targeting the right adresses .[14]



III.2 Conclusion:

the system begins by utilizing a two-channel ADC to measure both current and voltage signals. By applying FFT, the time-domain signals are transformed into the frequency domain, enabling the extraction of the power frequency magnitude (50 Hz) for both parameters. This vital information is then communicated to the embedded processor via UART for further processing, ensuring accurate and efficient data transmission for the intended application.

III.3 Source :

[1] 2022_06_15_FPGA_Lecture_HS (cern.ch)

[2] terasic.com.tw/attachment/archive/941/DE0-Nano-SoC_User_manual_rev.D0.pdf

[3] *"Jitter effects on Analog to Digital and Digital to Analog Converters"* ***(PDF)****.* ***Retrieved August 19, 2012.***

[4] **Lathi,** ***B.P. (1998). Modern Digital and Analog Communication Systems (3rd ed.). Oxford University Press.***

[5] **Paul denisowski, Product Management Engineer, understanding spi, Rohde & Schwarz**

[6] Lecture 12 - Serial Peripheral Interfacing (ic.ac.uk)

[7] LTC2308 - Low Noise, 500ksps, 8-Channel, 12-Bit ADC (analog.com)

[8] What is a state machine? (itemis.com)

[9] FFT Implementation on the TMS320VC5505, TMS320C5505, and TMS320C5515 DSPs (Rev. B)

[10] TI Precision Labs – ADCs: Fast Fourier Transforms (FFTs) and Windowing (youtube.com)

[11]https://www.ti.com/lit/ab/slya069a/slya069a.pdf?ts=1723886972842&ref_url=https%253A%252F%252Fwww.google.com%252F

[12] CORDIC algorithm for angle calculations (youtube.com)

[13] *Altera. "Nios II Embedded Processor Backgrounder" (PDF).*

[14] *"5 Reasons to Switch from SOPC Builder to Qsys"*. ***Altera. Retrieved 16 March 2012.***

[15] FFT based Frequency Detector using an FPGA -Intel Quartus (IT WORKS!!) (youtube.com)

[16] How to write SPI Interface code in Verilog HDL for a 12-bit ADC (using the DE0-Nano) (youtube.com)

[17] AN-878 High Speed ADC SPI Control Software Application Note (Rev. A) (analog.com)

[18] **Absolute Angle Measurement for Rotational Motion using Hall-effect sensors**

[19] **Linear Hall Effect Sensor Angle Measurement Theory implementation, and Calibration**

[20] **Example for FFT in MATLAB** (YouTube.com)