

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'Hamed BOUGARA – Boumerdes



Institute of Electrical and Electronic Engineering
Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of
the Requirements for the Degree of

MASTER

In Computer Engineering
Option: Computer Engineering

Title:

**FPGA-Based Phasor Measurement Unit
Prototype**

Presented by:

- **HAMRIT Yazid Taha**

Supervisor:

Dr. MAACHE Ahmed

Registration Number:...../2020

Abstract

This report describes the design and implementation of an SoPC-based Phasor Measurement unit which is required in electronic applications where a synchronous relationship between the signals needs to be preserved, using the Field Programmable Gate Array (FPGA). However, due to the constraints imposed by the covid19 health crisis, the project only covered the evaluation usage of the FFT core using an analogue input from a potentiometer. This signal is sampled using the Analog to Digital Converters(ADC) on the FPGA board. The design then stores digital data into a local FIFO, which is passed to a 1024-Point FFT hardware core to get the spectrum of the signal and hence calculate the main frequency. The system uses the Intel DE10 FPGA board (donated by the Intel University Program) and the Quartus Prime suite to design and implement the system and the model was synthesized using Quartus II and targeted at Cyclone-V FPGA. The design was successfully implemented.

Dedication

This dissertation is dedicated to my family, which is without doubt, the greatest wealth that I will ever possess. To my parents, Dad Lakhdar HAMRIT and Mom Nabila DJEBBARI, for making me the person I am and supporting me all the way. Words cannot describe my love for you. To my sister, the Princess Djalila, and my two little cute brothers Zaki and Mahdi for shedding a soft glow on my life with their true love and innocent smiles. I love you and cherish you infinitely. To my brother Hakim, his wife Meriem and their baby girl, you are the perfect family, because whether it is the middle of a midst storm or calm as a sea, you will always be there for each other and for me. To the superheroes, my sisters Rania and Sabrin who stood by me from the very first beginnings. You fill my life with joy and your existence is priceless. To all my wonderful family, particularly, To my friends and everyone else, especially, my friend Anis Slimatni and my childhood friend and brother Mohamed Elhadi MANSOUR.

Acknowledgements

Firstly, I am grateful and thankful to Allah for the blessings and wellbeing that were necessary for me to complete this work.

I place on record my sincere thanks and appreciation to my supervisor Dr. A. Maache for sharing expertise and valuable guidelines which were of an extreme help to me. We take this opportunity to express my deep and special thanks to Slimatni and Khelifa for their continuing technical support in the implementation of my project and their valuable support throughout this work. I am also grateful to the teaching staff for their encouragement and support and to all members of the Department of Electronics and everyone that belongs to the Institute.

Finally, a thank you to everyone else who, directly or indirectly, has contributed and helped me accomplish my project.

Contents

Abstract	i
Acknowledgements	iii
List of Tables	vi
List of figures	vii
List of Abbreviations	viii
1 Introduction	1
1.1 Overview	1
1.2 Literature Review	2
1.3 Motivation	3
1.4 Project Objectives	4
1.5 Organization of the Report	4
2 Theoretical Background	5
2.1 Signal Model	5
2.2 Phasor	5
2.3 Fast Fourier Transform	7
2.4 Phasor Calculation for 3-phase System	9
2.5 Field Programmable Gate Array	10
2.5.1 Overview	10
2.5.2 Applications of FPGAs	11
2.5.3 DE10-Board	11
2.5.4 Nios II Processor	13
2.5.5 Quartus PRIME Software	14
2.6 Analog to Digital Converter	14

CONTENTS

2.6.1	Overview	14
2.6.2	The LTC2308	15
2.7	Qsys Tool	16
3	Hardware System Design	17
3.1	Overview	18
3.2	Signal Acquisition and Sampling	18
3.3	System Implementation	19
3.3.1	SoPC System	19
3.3.2	Sopc Main Entity	20
3.4	FFT BLOCK	21
3.4.1	Streaming FFT	22
3.4.2	Functional Description	22
3.4.3	Using the Streaming FFT	24
3.5	ADC Block	25
3.6	ALTERA NIOS II JTAG DEBUG MODULE	27
3.7	PLL System Block	27
4	Implementation and Results	29
4.1	Overview	29
4.2	The Firmware	29
4.3	Top Level File with RTL Viewer	30
4.4	Top Level File Compilation	30
4.5	Results of the Implemented System	31
4.5.1	Testing the ADC Block Reading	31
4.5.2	System Output	32
4.6	Hardware Limitations	33
	Conclusion	34
	References	36

List of Tables

2.1 Comparison of Nios II processor Versions	14
--	----

List of Figures

1.1	PMU and SCADA response to a disturbance.	2
2.1	Phasor representation of a sinusoid.	6
2.2	Conceptual Structure of an FPGA Device.	11
2.3	DE10-Standard development board (top view).	13
2.4	LTC2308 block diagram.	15
2.5	The system contents tab of the Qsys tool.	16
3.1	Block diagram of our prototype PMU.	17
3.2	Overall system's block diagram.	18
3.3	Connections between the FPGA, 2x5 header, and the A/D converter.	19
3.4	Overall System Components.	19
3.5	Overall Block Diagram of the System.	20
3.6	FFT core integrated inside Quartus's platform designer. (Qsys)	21
3.7	FIFO with Avalon-MM Input Interface and Avalon-ST Output Interface.	22
3.8	FFT BLOCK interfacing design.	23
3.9	FFT block interface.	24
3.10	FFT Streaming Data Flow Simulation Waveform.	24
3.11	ADC state machine.	25
3.12	Overall system interconnection.	26
3.13	PLL intel FPGA IP used.	28
4.1	Overall Block Diagram of the On-Chip System.	30
4.2	Compilation Summary.	31
4.3	The sampling result captured by the ADC.	31
4.4	FIFO output generated level.	32
4.5	Connectivity warning message.	33

List of Abbreviations

AC	Analog Current
ADC	Analog to Digital Convertor
ASIC	Application Specific Integrated Circuit
ARM	Acorn RISC Machine
CPU	Central Processing Unit
DC	Direct Current
DFT	Discrete Fourier Transform
DE10	board Development and Education board Cyclone V
CT	Current Transformer
EMS	Energy Management System
EPROM	Erasable Programmable Read Only Memory
FFT	Fast Fourier Transform
FIFO	First-In, First-Out
FPGA	Field programmable gate array
FSM	Finite State Machine
GPS	Global Positioning System
GUI	Graphical User Interface
HDL	Hardware Description Language

List of Abbreviations

HPS	Hard Processor System
IC	Integrated Circuit
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronic Engineers
PC	Personal Computer
PDC	Phasor Data Concentrator
PLD	Programmable Logic Device
PMU	Phasor Measurement Unit
PPS	Pulse Per Second
PROM	Programmable Read Only Memory
PT	Potential Transformer
RAM	Random Access Memory
ROM	Read Only Memory
RTL	Register Transfer Level
SCADA	Supervisory Control And Data Acquisition System
SDRAM	ynchronous Dynamic Random Access Memory
SoC	System on Chip
SoPC	System on Programmable Chip
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
UTC	Universal Coordinated Time
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

Introduction

1.1 Overview

Traditionally, analog and digital information (status of circuit breaker, power flow and frequency) is measured at the substation level and transmitted to the load dispatch center using supervisory control and data acquisition system (SCADA) or energy management system (EMS). The major limitation of SCADA or EMS is the inability to accurately calculate the phase angle between a pair of substations. In SCADA or EMS, phase angle is either estimated from available data or is calculated offline. Phasor Measurement Units (PMU) overcome the limitations of SCADA and EMS by accurately calculating the phase angle between a pair of grids. Synchronized phasor measurement units were introduced in the mid-1980s as a solution for the need of more efficient and safer monitoring devices for Electric Power Systems (EPS). SCADA had minor issues, it cannot deal with transient response, it deals only with the magnitude. Also, it is for local monitoring. The figure below shows a contrast between PMU and Old monitoring Systems.

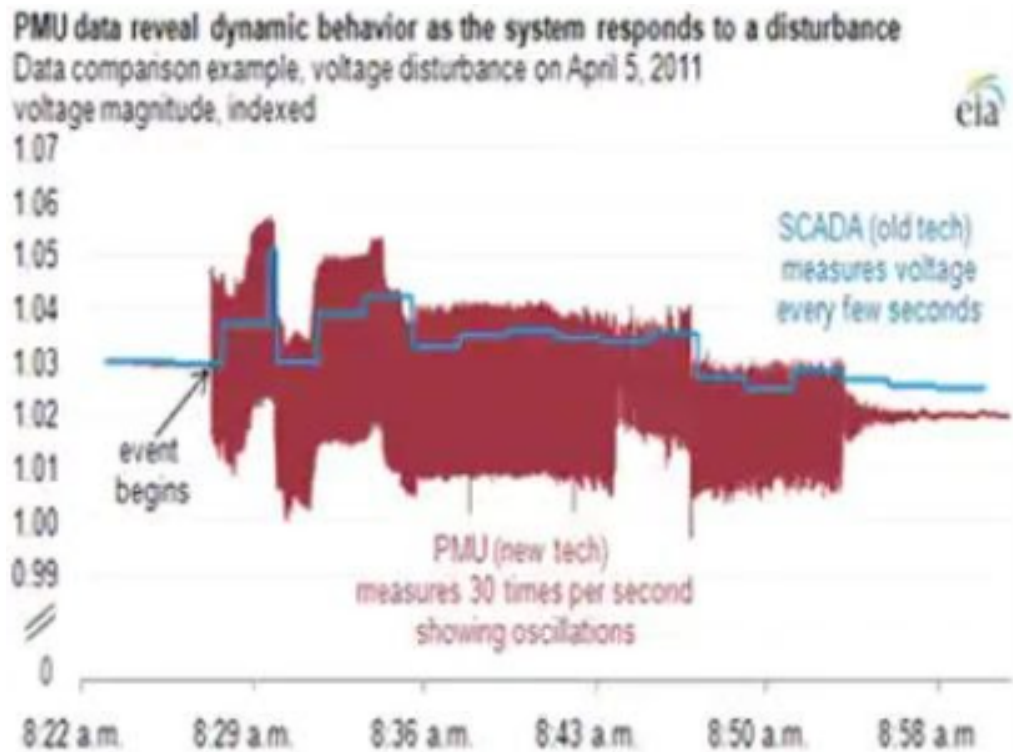


Figure 1.1: PMU and SCADA response to a disturbance.

Since then, measuring Electric Power System (EPS) parameters of voltage and current in relatively distant buses has received great attention from researchers. Such measurements are performed by phasor measurement units (PMUs), synchronized by Global Positioning System (GPS) satellites. A commercial PMU measures the voltage and angle of a particular grid at 25 samples per second. The phase information is synchronized with Global Positioning Systems (GPS) satellite and is transmitted to Phasor Data Concentrator (PDC) through a high-speed communication network. The time stamped phase information is called synchro phasor.

The measurement of voltage and current in the remote bus allows the operator to make a concrete decision about the maintenance and security of the system in the face of various uncertainties [1].

1.2 Literature Review

The measurement of voltage phase angles using synchronized clocks for power system applications dates back to the early 1980s when measurements of voltage phase angles were carried out between Montreal and SEPT-ILES [2, 3], and par-

allel efforts by Bonanomi in 1981 [4]. However, the synchrophasor technology available today emerged from the early efforts by Phadke et al. at Virginia Tech as described in [5, 6]. Phadke demonstrated the first synchronized PMU in 1988, and in 1991 Macrodyne Inc. launched the first commercial PMU product [7]. Due to the cost of early PMU devices, PMU technology has historically been limited to transmission system applications where the business case justified expensive phasor analysis equipment.

One of the early applications that is important to mention is the implementation of the wide-area protection system Syclopes in France in the early 1990s, which was the first functional application of early forms of PMUs [8]. The cost of the components from which PMUs are assembled (such as GPS receivers, microprocessors, and storage devices) have dropped significantly due to recent developments across the electronics sector. As a consequence, PMUs have reached price points that have made them an attractive tool for the distribution systems and embedded generation. Many PMUs are sold as dedicated devices which offer event recorder type functionality. Costs for such units vary between US \$6000 and US \$15000 depending on the specification. Many equipment vendors have begun to offer PMU functionality a supplementary feature on other products in their range, such as protection relays [9].

The standard for PMU devices is maintained by the IEEE C37.118 Working Group. IEEE Std. C37.118 [1] was released in 2005 and subsequently updated in 2011. The latest release comes in two parts; IEEE C37.118.1 -2011 [1] describes how synchrophasors should be estimated and gives certification requirements while IEEE C37.118.2-2011 [10] describes data representation and data transfer. Concerns have been raised regarding the transient performance of PMUs under the 2005 standard [1, 11, 12]. These concerns are addressed in the 2011 release of the standard. IEEE C37.118.1 -2011 states that it defines synchrophasors, frequency, and rate-of-change-of frequency measurement under all operating conditions [1].

1.3 Motivation

Many researchers designed PMU based on microcontrollers, but microcontrollers are sequential in nature thereby degrading the efficiency of the system. By using FPGA, we are capable of measuring currents and voltages with parallel

measurement, in other words the data for current and voltage will be read at the same time and at the same clock. It's different from a microcontroller that uses a sequential programming language. In every task, it needs a couple of execution times, from first to the last task must be done sequentially. Since there exists some gap of measurement, it will give us an uncertainty of the accurate time between compared values. It also takes longer to delay than FPGA has. The advantages of using FPGA in phasor estimation is that the 1024-point FFT hardware core can be pipelined. In other words, it can accept input data every clock cycle and generate output data every clock cycle, after a certain time delay. These huge computations can be handled well with a parallel processor such as FPGA. [11]

1.4 Project Objectives

The main goal of this project is to design and implement a Phasor Measurement Unit (PMU) prototype using the DE10 Standard FPGA.

1.5 Organization of the Report

The report is divided into four chapters. The first chapter is the introduction to the project and the report. The second chapter deals with the theoretical background where it describes all the used materials, terms and components with their principle of operation. The third chapter titled the Hardware System Design where it shows the design and the implementation of the project's hardware in detail. The fourth chapter deals with the software part of the project including the firmware. At the end of the report, a general conclusion is added to summarize and provide suggestions for further work.

Theoretical Background

2.1 Signal Model

Electrical power is traditionally delivered from the generators to the end users through an infrastructure that is mainly composed by AC power systems. As a consequence, during normal operating conditions of the power system, voltage and current waveforms are usually modeled as signals characterized by a single sinusoidal component with constant parameters:

$$y(t) = Y_m \cos(\omega t + \phi) \quad (2.1)$$

2.2 Phasor

For a detailed analysis of an AC circuit, it is useful to measure the magnitude, frequency and phase angle of the time-varying quantities during a specific interval. Phasors allow us to easily accomplish it.

Let us consider equation 2.1, where Y_m represents the maximum value or peak amplitude; $\omega = 2\pi f_0$ is the angular frequency of the signal in radians per second (f_0 is the fundamental frequency); and ϕ is the phase angle in radians. Keeping in mind the Euler's identity ($e^{jx} = \cos x + j \sin x$), one can observe that equation 2.1 can also be rewritten as:

$$y(t) = \text{Re}[Y_m e^{j(\omega t + \phi)}] = \text{Re}[(e^{j2\pi f_0 t}) Y_m e^{j\phi}] \quad (2.2)$$

When the system frequency is known, the term $e^{j2\pi f_0 t}$ can be neglected. There-

fore equation 2.2 may be represented by a complex number V given by:

$$y(t) \iff V = Y_m e^{j\phi} = Y_m [\cos \phi + j \sin \phi] \quad (2.3)$$

Assuming that both voltage and current signals are given by equation 2.3, one can observe that this representation is at odds with the calculation of average power, therefore the RMS quantities must be taken into account for the correct phasor representation of sinusoidal signals, as illustrated by the complex number Y that follows.

The phase angle of a phasor brings the information about the fraction of the sinusoid's period in which the time, or the angular displacement ωt , is advanced or delayed to an arbitrary reference. It is very important to correlate different alternating-waves between them, thus, the phasors represent an equilibrium point or the steady-state condition of the AC circuit.

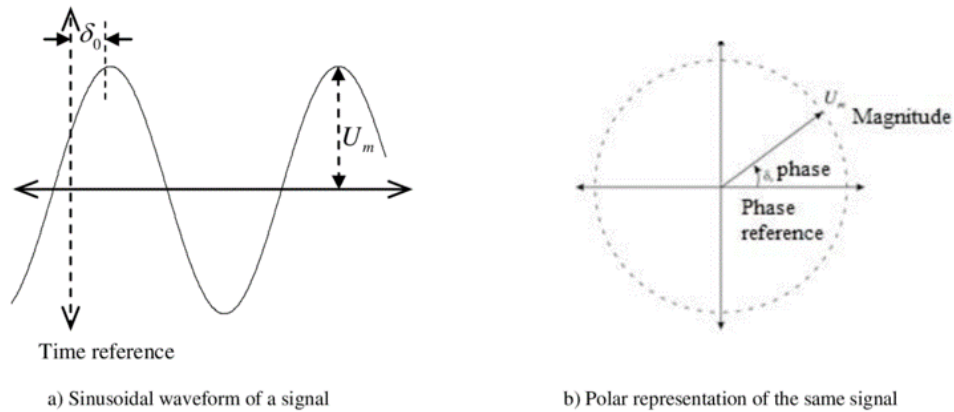


Figure 2.1: Phasor representation of a sinusoid. [13]

In practical situations, however, to perform the phasor calculation, a time interval must be considered. This time interval is often referred to as a "data window" or "observation interval," which is essential for practical waveform phasor estimation. In essence, the phasor representation is related to a pure sinusoidal signal, but harmonics can distort the existing signals in electric power systems. In this way, it is recommended that the envisaged frequency component(s) of the signal be extracted to be represented by phasor notation as well. These tasks have been properly performed by the classical Fourier's theory. Due to the fact, in the later sections, the main key points regarding phasor representation using the aforementioned theory are presented and discussed in greater detail.

2.3 Fast Fourier Transform

We should mention that the Fourier transform is a very important part of many engineering applications. FFTs are an important part of any digital spectrum analyzer. FFTs can also be used when implementing a spectrogram. Such spectrograms make it easier to understand artifacts of speech and other sounds, or even radio frequency waveforms, by visual inspection [14].

Convolutions and/or correlations can often be implemented much faster and cheaper using an FFT implementation of the Fourier transform. This means that digital filters can be implemented with Fourier transform enabled convolutions faster/better/cheaper. Fourier transforms are used to understand and analyze control systems. Fourier transforms are used not only in filter implementations, but they are also used in the filter design process. And finally, like we used it in our design Fourier transform can be used to evaluate phasor measurements. The Fast Fourier Transform (FFT) is a specific implementation of the Fourier transform, that drastically reduces the cost of implementing the Fourier transform Prior to the invention of the FFT, a Discrete Fourier transform could only be calculated the hard way with N^2 multiplication operations per transform of N points. Since Cooley and Tukey published their algorithmic implementation of the Discrete Fourier transform, they can now be calculated with $O(N \log N)$ multiplies. Needless to say, the invention of the FFT immediately started to transform signal processing. But before talking about the FFT we should understand a little more about what a Fourier transform is first [14]. A Fourier transform is a linear operator that decomposes a signal from a representation in time, to a time-less representation in frequency.

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft} dt \quad (2.4)$$

This is the definition we will first come across when studying Fourier transform. This form above is easy to work with mathematically with just pen and paper. There are two problems with this mathematical definition when it comes to working with an engineering reality. The first problem is that digital algorithms do not operate upon continuous signals very well. Computers and other digital signal processors do a much better job with sampled signals. Hence, we'll switch from discussing the pure Fourier transform shown above and examine the Discrete-time Fourier transform instead. For this, we will switch from a continuous incoming signal, $x(t)$ to its sampled representation, $x[n]$. The Discrete-time

Fourier transform can then be applied to our signal.

$$X(e^{j2\pi f}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi f n} \quad (2.5)$$

While this discrete-time transform works well for representing the response of certain digital filters, but it is not practical. This brings us to the second problem: Computers can't handle an infinite number of samples, nor can they handle an infinite number of potential frequencies. Both of these need to be sampled and finite. Fixing this second problem brings us to the Discrete Fourier transform.

$$X\left(\frac{k}{N}\right) = \sum_{n=0}^{N-1} x[n]e^{-j2\pi \frac{k}{N} n} \quad (2.6)$$

Now, not only is the $x[n]$ used in this transform discrete, but the frequency index, $\frac{k}{N}$, is as well. All three of these representations are very tightly related. Mathematically, there are major and significant differences between these transforms. Practically, however, only this last transform can ever be computed digitally. Therefore, the first two expressions of the Fourier transform and then the discrete time Fourier transform can only be digitally approximated by the Discrete Fourier transform [14].

It is this third representation of the Fourier transform, known as the Discrete Fourier transform, that we will be discussing the implementation. We are also going to argue that this is the only representation of the Fourier transform that can be numerically computed for any sampled finite sequence.

If we look at equation 2.6, we can see it takes as input N data samples, $x[n]$, and calculates one summation across those inputs for every value of k to produce N samples out, $X[\frac{k}{N}]$. Given that there's a complex multiplication required for every term in that of N numbers, and that there are N relevant outputs, this will cost N^2 painful multiplications to calculate. If we just left things there, this transform would be so hard to calculate that no one would ever use it. Cooley and Tukey, however, described a way that the Discrete Fourier transform can be decomposed into two transforms, each of them being half the size of the original, for the cost of only N multiplies. If you then repeat this $\log_2(N)$ times, you'll get to a one-point transform, for a total cost of $N \log_2(N)$ multiplies. At this cost point, the Discrete Fourier transform becomes relevant. Indeed, it becomes significant and a fundamental DSP operation. An FFT rapidly computes such transformations by

factorizing the DFT matrix into a product of sparse (mostly zero) factors [10]. As a result, it manages to reduce the complexity of computing the DFT from $O(N^2)$, which arises if one simply applies the definition of DFT, to $O(n \log n)$, where n is the data size. The difference in speed can be enormous, especially for long data sets where N may be in the thousands or millions. In the presence of round-off error, many FFT algorithms are much more accurate than evaluating the DFT definition directly. There are many different FFT algorithms based on a wide range of published theories, from simple complex-number arithmetic to group theory and number theory [14].

2.4 Phasor Calculation for 3-phase System

Consider a balanced 3-phase power system operating at a nominal frequency of f_0 , then the voltage waveform can be represented as:

$$\begin{cases} x_1(t) = X_m \cos(2\pi f_0 t + \phi_1) \\ x_2(t) = X_m \cos(2\pi f_0 t + \phi_2) \\ x_3(t) = X_m \cos(2\pi f_0 t + \phi_3) \end{cases} \quad (2.7)$$

Here X_m represents the maximum amplitude of the signal and ϕ represents the phase angle. The phase angles are 120 degrees or $\frac{2\pi}{3}$ radians apart. The time domain sample of the power system can be represented as:

$$\begin{cases} x_{n1}(t) = X_m \cos(\frac{2\pi n}{N} + \phi_1) \\ x_{n2}(t) = X_m \cos(\frac{2\pi n}{N} + \phi_2) \\ x_{n3}(t) = X_m \cos(\frac{2\pi n}{N} + \phi_3) \end{cases} \quad (2.8)$$

Here N is the number of samples, which is an integer multiple of fundamental frequency f_0 and n represents the sample index in the array which ranges from 0 to $N - 1$.

The generalized expression for N-point can be represented as:

$$X = \frac{1}{N} + \sum_{n=0}^{N-1} x_n \left(\cos \frac{2\pi n}{N} - j \sin \frac{2\pi n}{N} \right) \quad (2.9)$$

N-point DFT of the signal can be found out using:

$$X_k = \frac{\sqrt{2}}{N} \sum_0^{N-1} x_n \left(\cos \frac{2\pi n}{N} - j \sin \frac{2\pi n}{N} \right) \quad (2.10)$$

$$X_{nominal} = \frac{\sqrt{2}}{N} \sum_0^{N-1} x_n \left(\cos \frac{2\pi n}{N} - j \sin \frac{2\pi n}{N} \right) \quad (2.11)$$

The real and imaginary part of the above expression can be rewritten as:

$$X_{real} = \frac{\sqrt{2}}{N} \sum_0^{N-1} x_n \left(\cos \frac{2\pi n}{N} \right) \quad (2.12)$$

$$X_{img} = -j \frac{\sqrt{2}}{N} \sum_0^{N-1} x_n \left(\sin \frac{2\pi n}{N} \right) \quad (2.13)$$

The phasor estimate at nominal frequency is represented by this complex quantity $X_{nominal}$, whose magnitude $\|X_{nominal}\| = \sqrt{X_{real}^2 + X_{img}^2}$ gives the RMS magnitude of the signal. The phase angle can be computed using the trigonometric property, $\phi_{nominal} = \tan^{-1} \frac{X_{real}}{X_{img}}$.

2.5 Field Programmable Gate Array

2.5.1 Overview

A field-programmable gate array (FPGA) is a logic device that contains a two-dimensional array of generic logic cells and programmable switches. The conceptual structure of an FPGA device is shown in figure 2.2. A logic cell can be configured (i.e., programmed) to perform a simple function, and a programmable switch can be customized to provide interconnections among the logic cells. Once the design and synthesis are completed, a simple adaptor cable has to be used to download the desired logic cell and switch configuration to the FPGA device and obtain the custom circuit [15].

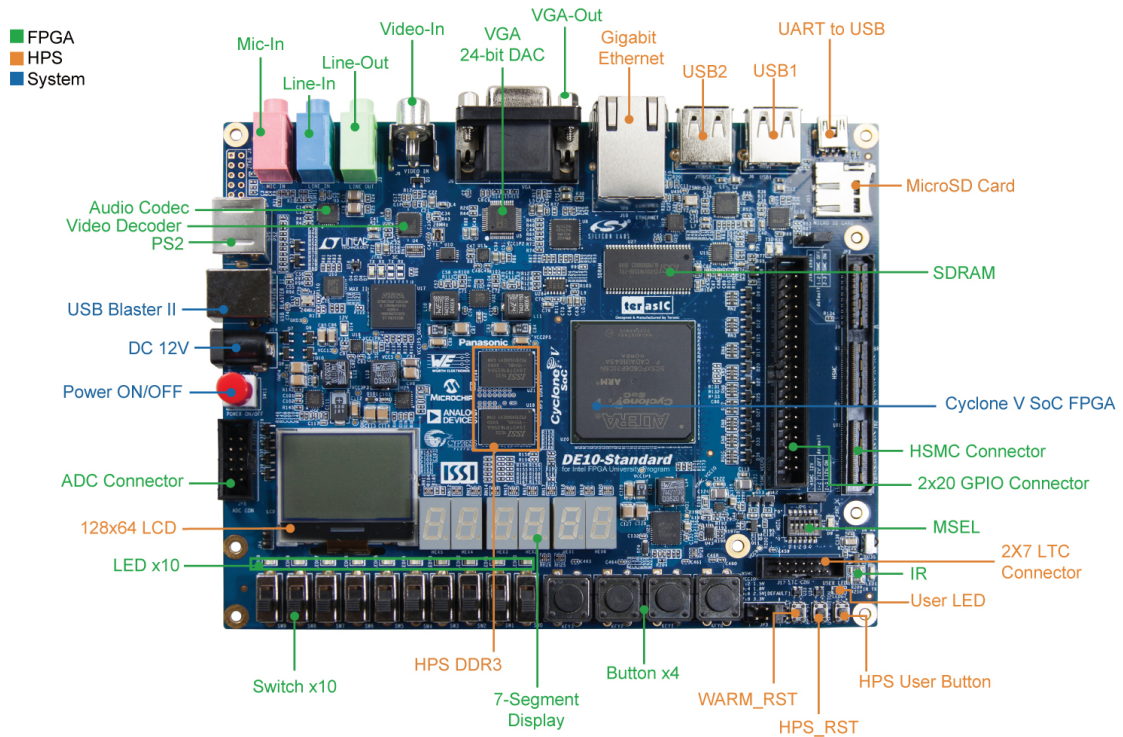


Figure 2.2: Conceptual Structure of an FPGA Device. [16]

An FPGA can be used to solve any problem which is computable. This is trivially proven by the fact FPGA can be used to implement a soft microprocessor, such as the Xilinx MicroBlaze or Altera Nios II. Their advantage lies in that they are sometimes significantly faster for some applications because of their parallel nature and optimality in terms of the number of gates used for a certain process [17].

2.5.2 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 [17].

2.5.3 DE10-Board

The DE10-Standard Development Kit presents a robust hardware design platform built around the Intel System-on-Chip (SoC) FPGA, which combines the latest dual-core Cortex-A9 embedded cores with industry-leading programmable logic for ultimate design flexibility. Altera's SoC integrates an ARM-based hard

processor system (HPS) consisting of processor, peripherals and memory interfaces tied with the FPGA fabric using a high-bandwidth interconnect backbone. The DE10-Standard development as shown in figure 2.3 board includes hardware such as high-speed DDR3 memory, video and audio capabilities, Ethernet networking, and much more. [18] The following hardware is provided on the board:

- Intel Cyclone V SE 5CSXFC6D6F31C6N device
- Serial configuration device - EPCS128
- USB-Blaster II onboard for programming; JTAG Mode
- 64 MBSDRAM (16-bit data bus)
- 4 push-buttons
- 10 slide switches
- 10 red user LEDs
- Six 7-segment displays
- Four 50MHz clock sources from the clock generator
- VGA DAC (8-bit high-speed triple DACs) with VGA-out connector
- PS/2 mouse/keyboard connector
- IR receiver and IR emitter
- One HSMC with Configurable I/O standard 1.5/1.8/2.5/3.3
- A/D converter, 4-pin SPI interface with FPGA

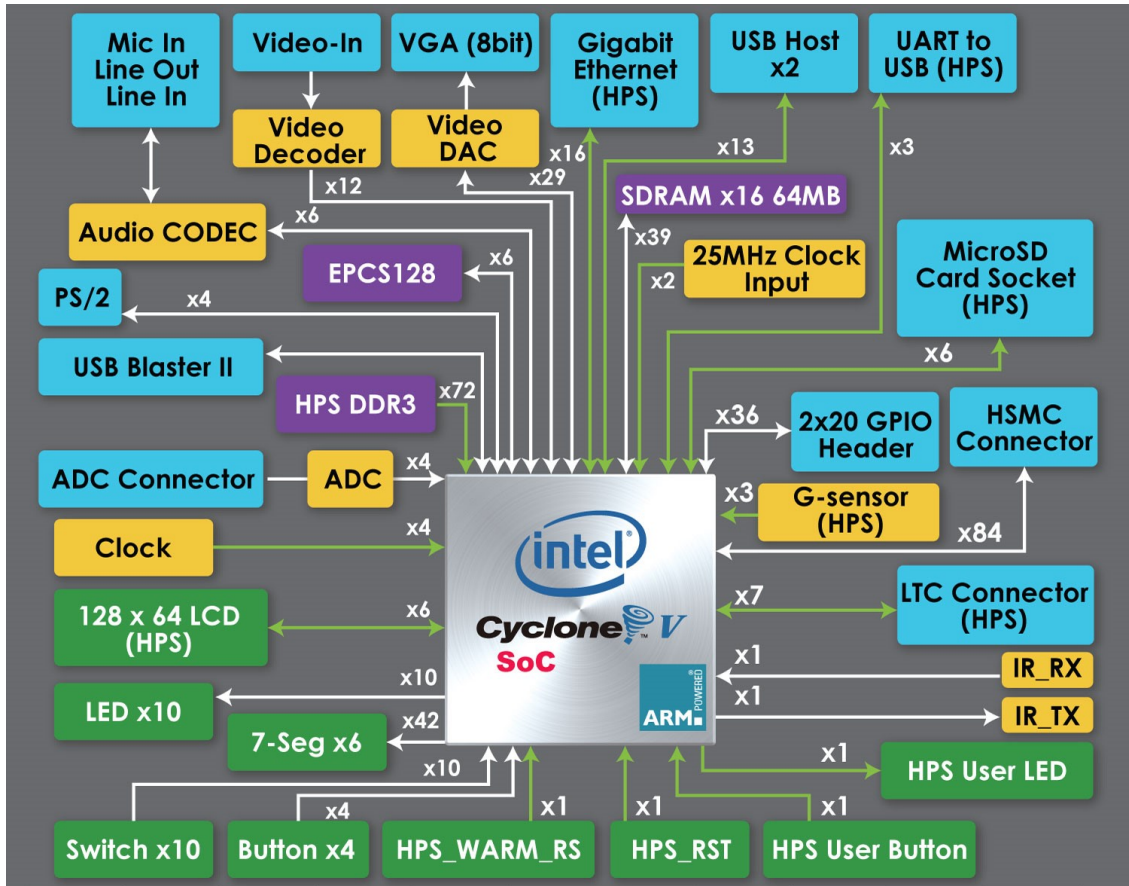


Figure 2.3: DE10-Standard development board (top view). [18]

2.5.4 Nios II Processor

Nios II processor is the most widely used soft processor in the FPGA industry. As opposed to a fixed prefabricated processor, a soft-core processor is described by HDL codes and then mapped onto FPGA's generic logic cells. This approach offers more flexibility. A soft-core processor can be configured and tuned by adding or removing features on a system-by-system basis to meet performance or cost goals. The Nios II processor follows the basic design principles of a RISC (Reduced Instruction Set Computer) architecture and uses a small, optimized set of instructions [20].

Nios II embedded processors provide an ideal embedded solution offering flexibility, high performance, low cost, and a long-life cycle [?]. There are three basic versions of Nios II:

- **Nios II/f:** The fast core is designed for optimal performance. It has a 6-stage pipeline, instruction cache, data cache, and dynamic branch prediction.

- **Nios II/s:** The standard core is designed for small size while maintaining good performance. It has a 5-stage pipeline, instruction cache, and static branch prediction.
- **Nios II/e:** The economy core is designed for optimal size. It is not pipelined and contains no cache. These processors' key characteristics are summarized on the top of table 2.1 and their sizes and performances (which are based on the Cyclone II family) are listed on the bottom [20].

Table 2.1: Comparison of Nios II processor Versions

	Nios II/f	Nios II/s	Nios II/e
Processor Pipeline	6 stages	5 stages	1 stage
Branch prediction	dynamic	static	-
Multiplication	1-cycle multiplier	3-cycle multiplier	software
Shift	1-cycle barrel shifter	3-cycle barrel shifter	software
Instruction cache	0.5 KB to 64 KB	0.5 KB to 64 KB	-
Data cache	0.5 KB to 64 KB	-	-
MMU/MPU	optional	-	-
Circuit size	1600 LEs	1030 LEs	540 LEs
Max clock rate	140 MHz	110 MHz	195 MHz
Performance	145 MIPS	55 MIPS	18 MIPS

2.5.5 Quartus PRIME Software

Quartus prime is a software development suit tool developed by **Intel FPGA** (former Altera) Company. It provides a graphic interface for users to access tools and display relevant files. Some differences may exist between different versions.

2.6 Analog to Digital Converter

2.6.1 Overview

Analog to digital converter converts continuous analog signal to discrete digital numbers. ADC's differ from each other through two main parameters, the resolution which indicates the number of discrete values it can produce over the range of analog values and the step size (quantization value) which is based on the reference voltages of the ADC and it can be found as:

$$\Delta V = \frac{V_{ref+} - V_{ref-}}{2^{n_{bits}} - 1} \quad (2.14)$$

2.6.2 The LTC2308

The DE-10 Board FPGA has a 12 Bit ADC, The LTC2308 is a low noise, 500ksps, 8-channel, 12-bit successive approximation register (SAR) A/D converter (figure 2.4). The LTC2308 includes a precision internal reference, a configurable 8-channel analog input multiplexer (MUX) and an SPI-compatible serial port for easy data transfers. The ADC may be configured to accept single-ended or differential signals and can operate in either unipolar or bipolar mode. A sleep mode option is also provided to save power during inactive periods. Conversions are initiated by a rising edge on the CONVST input. Once a conversion cycle has begun, it cannot be restarted until the current conversion is complete. The time taken by each conversion for each channel is $13\mu s$ [19].

It is configured as 8 signal-end channels in the Verilog code. Users can change SW [2:0] to measure the corresponding channel. The default reference voltage is 4.096V. The formula of the sample voltage is:

Sample Voltage = ADC Data / full scale Data * Reference Voltage.

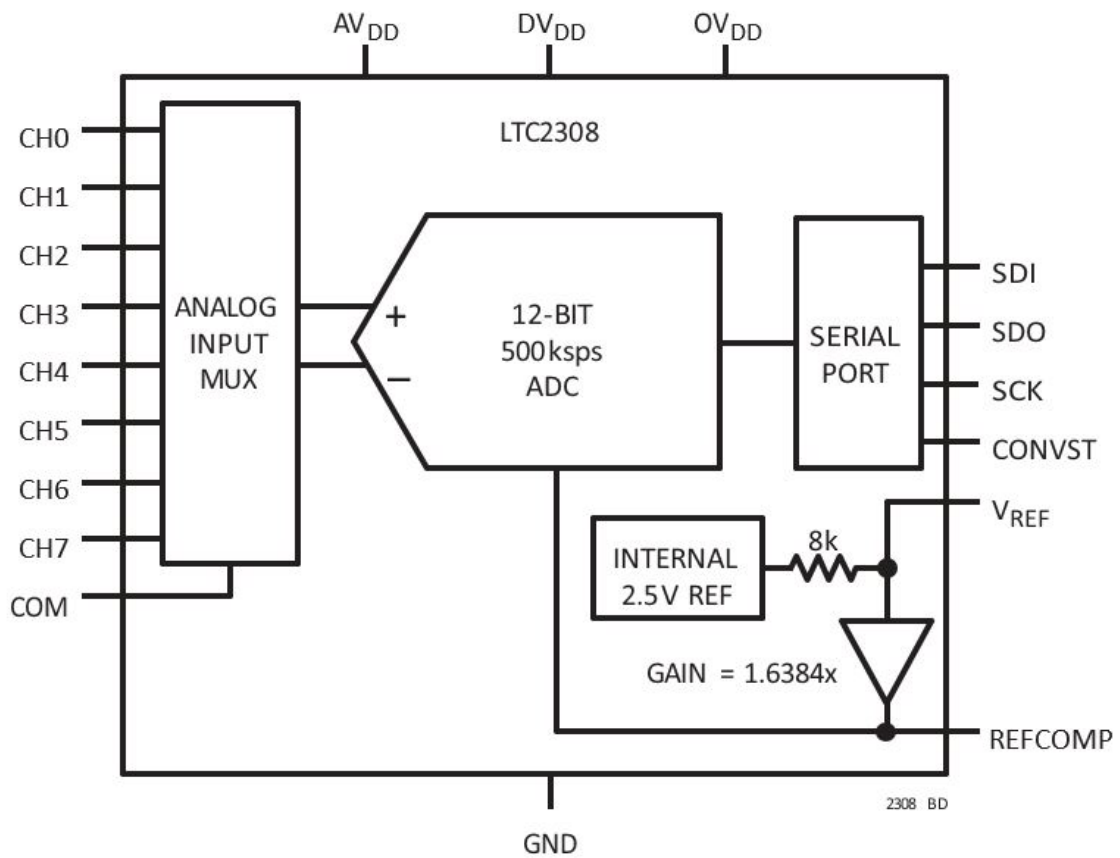


Figure 2.4: LTC2308 block diagram. [19]

2.7 Qsys Tool

The Qsys tool is used in conjunction with the Quartus II software. It allows the user to easily create an SoPC system based on the Nios II processor which enables you to define and generate a complete system-on-a-programmable-chip (SOPC) in much less time than using traditional, manual integration methods.

By Simply selecting the desired functional units and specifying their parameters. The Qsys tool allows a designer to choose the components that are desired in the system as processors, memories, input/output interfaces, timers..., by selecting these components in a graphical user interface. It then automatically generates the hardware system that connects all of the components [21].

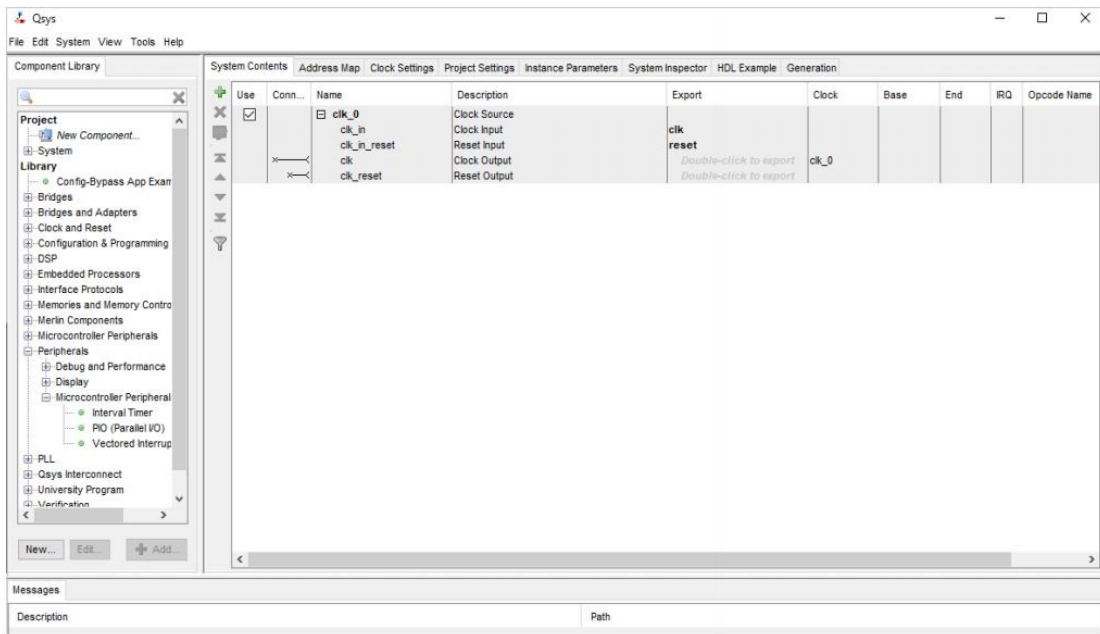


Figure 2.5: The system contents tab of the Qsys tool.

The System Contents tab of the Qsys tool appears as shown in figure 2.5, which is used to add components to the system and configure the selected components to meet the design requirements. The available components are listed on the left side of the window [21].

Hardware System Design

This chapter describes the design and implementation of a part of a Phasor Measurement Unit using FPGA. The block diagram of such a unit is shown in figure 3.1.

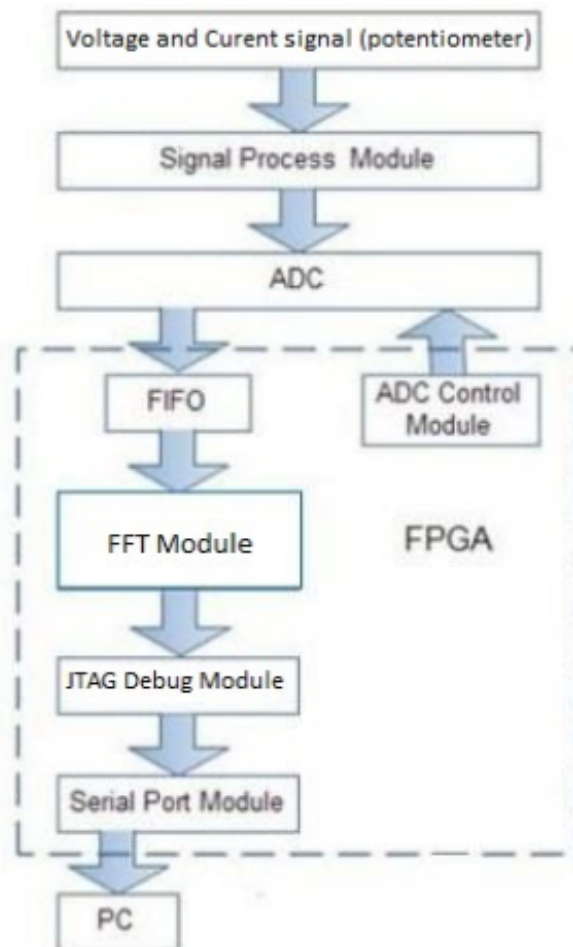


Figure 3.1: Block diagram of our prototype PMU.

3.1 Overview

The hardware system of this project is divided into two parts, the off-chip and the on-chip systems. As shown in the block diagram of the system in figure 3.2 The ON-chip hardware system is composed of two parts; The SoPC system implemented using Qsys tool, which includes the Nios II processor, the on-chip memory, the ADC IP block, the FFT block and I/O peripherals. And the non SoPC system, which is a set of custom VHDL blocks.

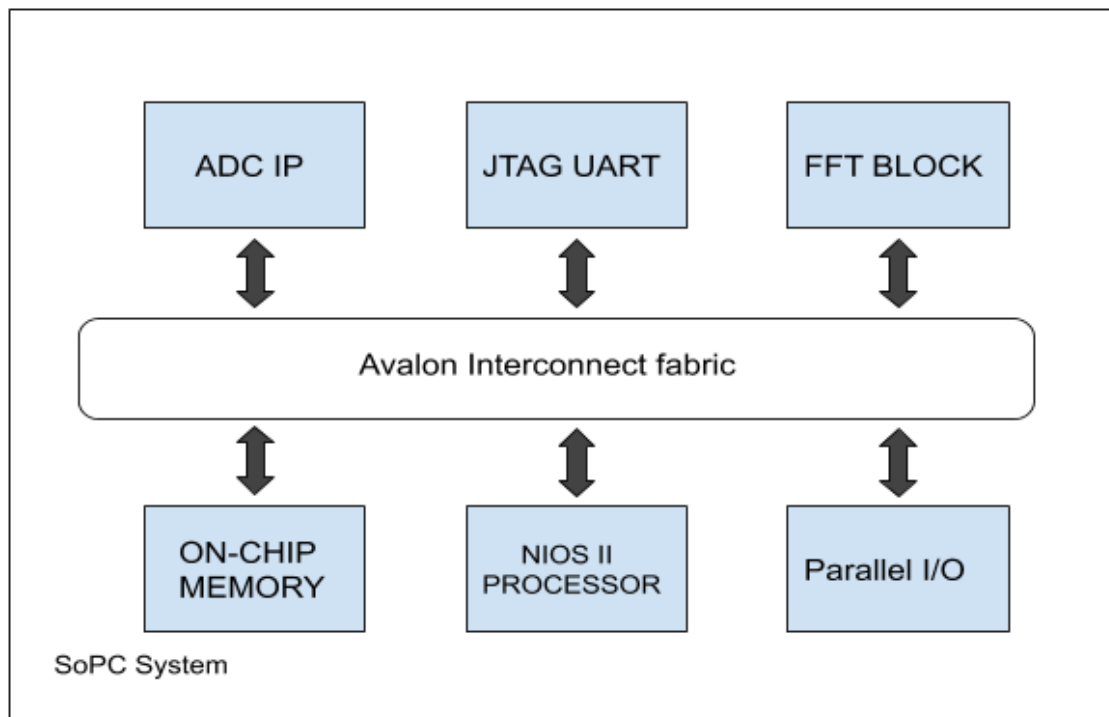


Figure 3.2: Overall system's block diagram.

3.2 Signal Acquisition and Sampling

For the calculation of a phasor, the data (i.e. the sampled voltage signal) must be acquired. When the PMU is tested in real-world scenarios a means of getting the signals from the transmission lines is necessary, which is accomplished using a Potential Transformer (PT) and a Current Transformer (CT) in the substations. This signal is further stepped down using the Hall Effect voltage sensors. In our laboratory setup, we were planning to use a function generator to generate sinusoidal input signals to mimic the signals read from voltage and current sensors. However, due to the access constraints imposed by the COVID19 health crisis, we only used a potentiometer for this purpose.

The data acquisition unit, built around the 12-bit analog to digital converter LTC2308CUF which is connected to the FPGA through an SPI-interface is seen in figure 3.3.

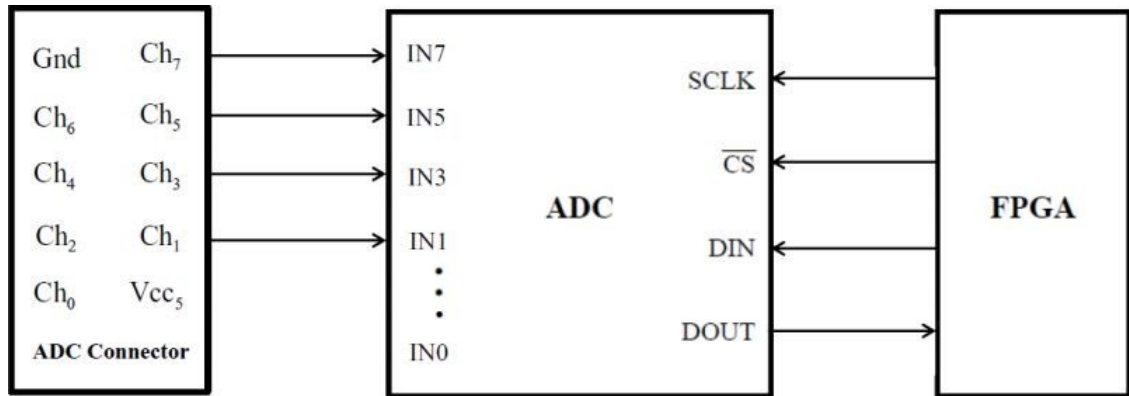


Figure 3.3: Connections between the FPGA, 2x5 header, and the A/D converter. [19]

3.3 System Implementation

The on-chip hardware system is implemented using the Cyclone V FPGA and it is composed around the SoPC system.

3.3.1 SoPC System

The SoPC system is built using the Qsys tool provided in Quartus II software, where the components of the system were selected as shown in figure 3.4.

Name	Description	E...	Clock	Base	End	IRQ
clk_50	Clock Source		exported			
pll_sys	PLL Intel FPGA IP		clk_50			
nios2_gen2_0	Nios II Processor		pll_sys	0x0000_0800	0x0000_0fff	
onchip_memory2	On-Chip Memory (RAM or ROM) Intel FPGA IP		pll_sys	0x0004_0000	0x0006_70ff	
sysid_qsys	System ID Peripheral Intel FPGA IP		pll_sys	0x0000_1028	0x0000_102f	
jtag_uart	JTAG UART Intel FPGA IP		pll_sys	0x0000_1030	0x0000_1037	
adc_ltc2308	adc_ltc2308		multiple	0x0000_1020	0x0000_1027	
sw	PIO (Parallel I/O) Intel FPGA IP		pll_sys	0x0000_1000	0x0000_100f	
fft_ii_0	FFT		clk_50			
SinkINT	Avalon FIFO Memory Intel FPGA IP		clk_50	0x0000_1018	0x0000_101f	
SourceINT	Avalon FIFO Memory Intel FPGA IP		clk_50	0x0000_1010	0x0000_1017	

Figure 3.4: Overall System Components.

The system consists of the clock source clk_50, Nios II processor, on-chip memory, GPIO port connected to Switches, an IP block to control the on-board ADC,

the FFT IP core, and two interfacing IP core to interface between the NIOS II and the FFT core which we will talk about more later. The system needs *jtag_UART* to exchange data serially with the PC. The system ID peripheral (*sysid_qsys*) also must be added to maintain the consistency between the hardware configuration file and software image.

The Overall Block Diagram of the System is shown in figure 3.5.

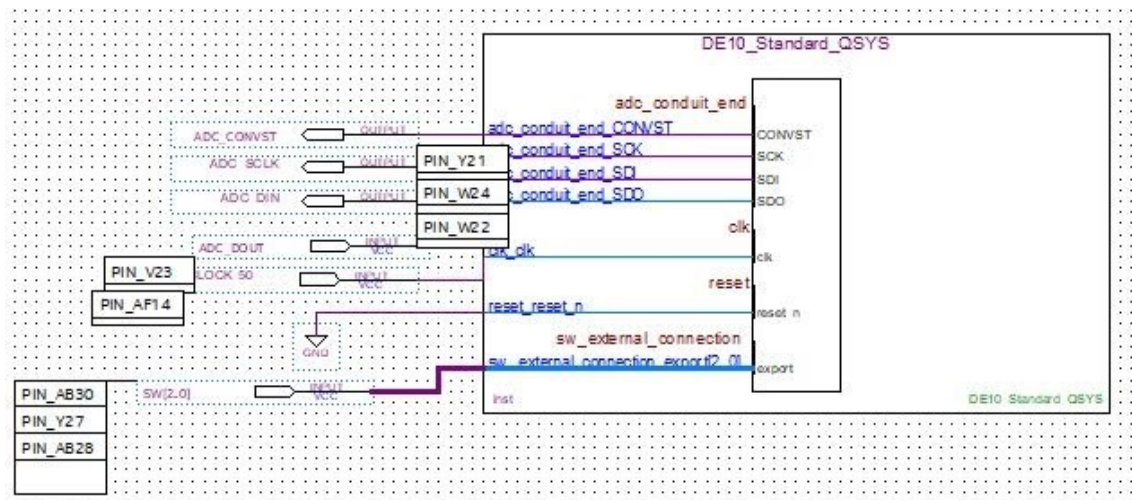


Figure 3.5: Overall Block Diagram of the System.

3.3.2 Soc Main Entity

The SoC system main entity has the following inputs and outputs:

- Data-in: 1-bit input port connected to the ADC block, which represents the serial digital data coming-in from the LTC2308 ADC to the FPGA.
- Data-out: 12-bit output port coming out of the ADC block, which represents the sampled parallel data.
- Data ready: 1-bit output ports from the ADC to tell if the Data in the output register is valid or not. It is also used for handshaking with the FIFO block.
- Convst: 1-bit output used to send the start of conversion signal to the ADC.
- Sclk: 1-bit output used to provide slower serial data clock for the ADC.
- Reset : 1-bit input asynchronous reset connected to the ADC and FFT block., to reset the hardware.

- clk: 1-bit input port which represents the 50MHz clock input used as the overall design clock source.
- SW: 3-bit input port used to select the ADC channel for conversion.

3.4 FFT BLOCK

The FFT unit is used to calculate the spectrum of the input signal. *Altera_fft_ii* core shown in figure 3.6 represents the best solution in terms of efficiency and reliability. Data could be sent from the processor to the FFT block for processing via the bus.

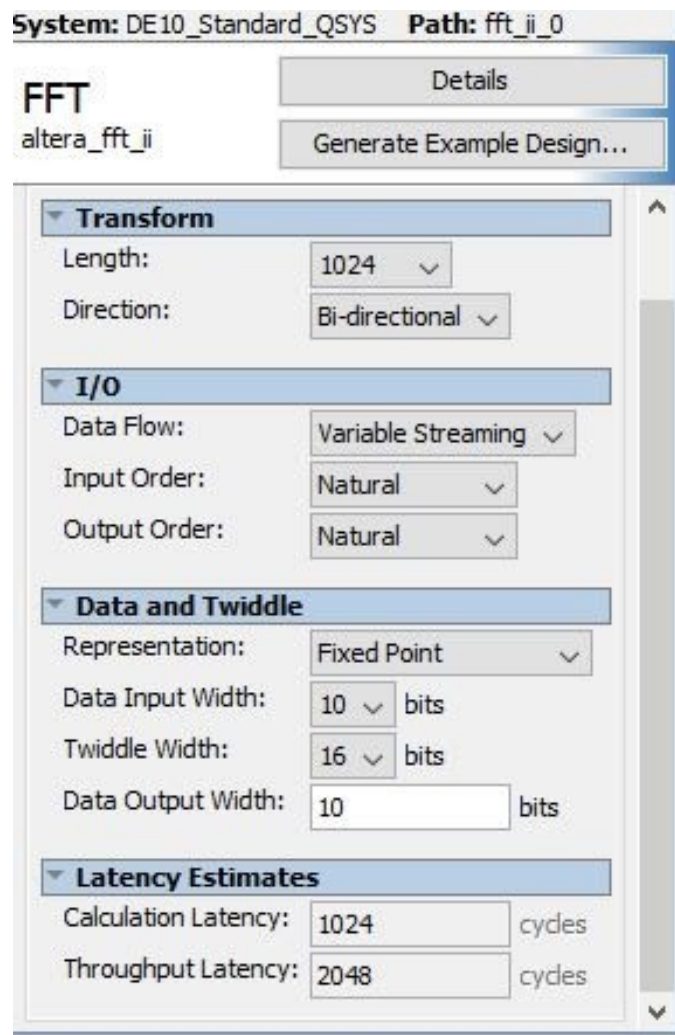


Figure 3.6: FFT core integrated inside Quartus's platform designer. (Qsys)

3.4.1 Streaming FFT

The streaming FFT allows continuous processing of input data, and outputs a continuous complex data stream without the need to halt the data flow in or out of the FFT IP core.

The streaming FFT generates a design with a quad output FFT engine and the minimum number of parallel FFT engines for the required throughput. A single FFT engine provides enough performance for up to a 1,024-point streaming I/O data flow FFT.

To implement the phasor calculation unit, the DE10-board FPGA has been used as the computational unit. For a 3-phase system, the voltage samples must be stored in FIFOs which are updated every time a new sample comes in. Here , we used The Intel FPGA Avalon FIFO memory core (figure 3.7) that buffers data and provides flow control in the Platform Designer system. The core can operate with a single clock or with separate clocks for the input and output ports.

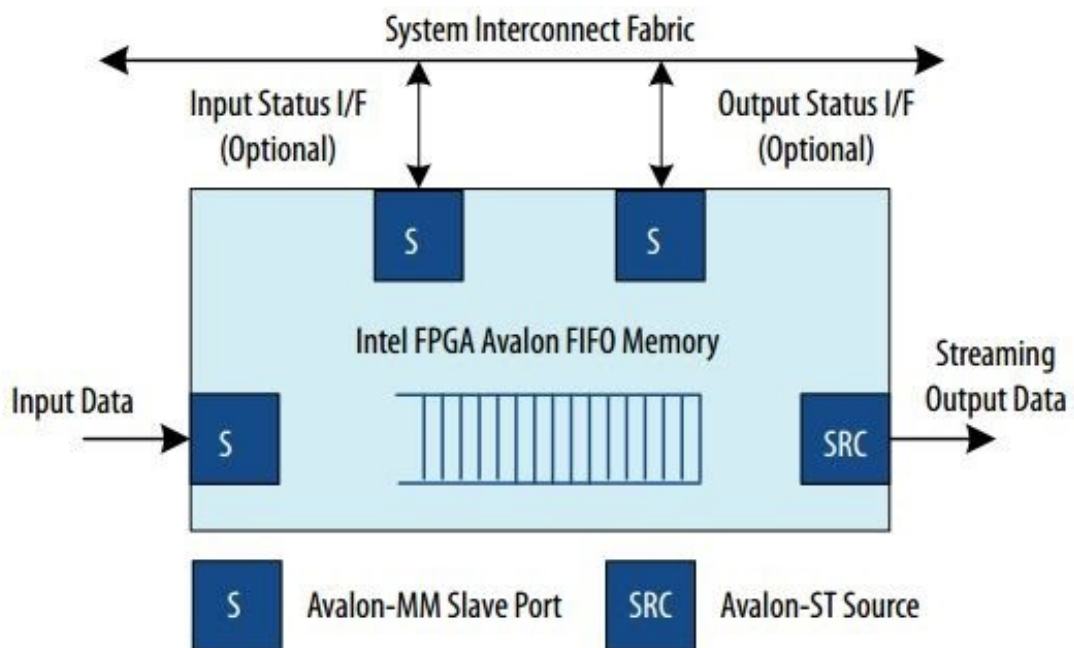


Figure 3.7: FIFO with Avalon-MM Input Interface and Avalon-ST Output Interface. [22]

3.4.2 Functional Description

The input interface to the Intel FPGA Avalon FIFO memory core may be an Avalon Memory Mapped (Avalon-MM) write slave or an Avalon Streaming (Avalon-

ST) sink. The output interface can be an Avalon-ST source or an Avalon-MM read slave. The data is delivered to the output interface in the same order that it was received at the input interface. See figure 3.8 which shows such design.

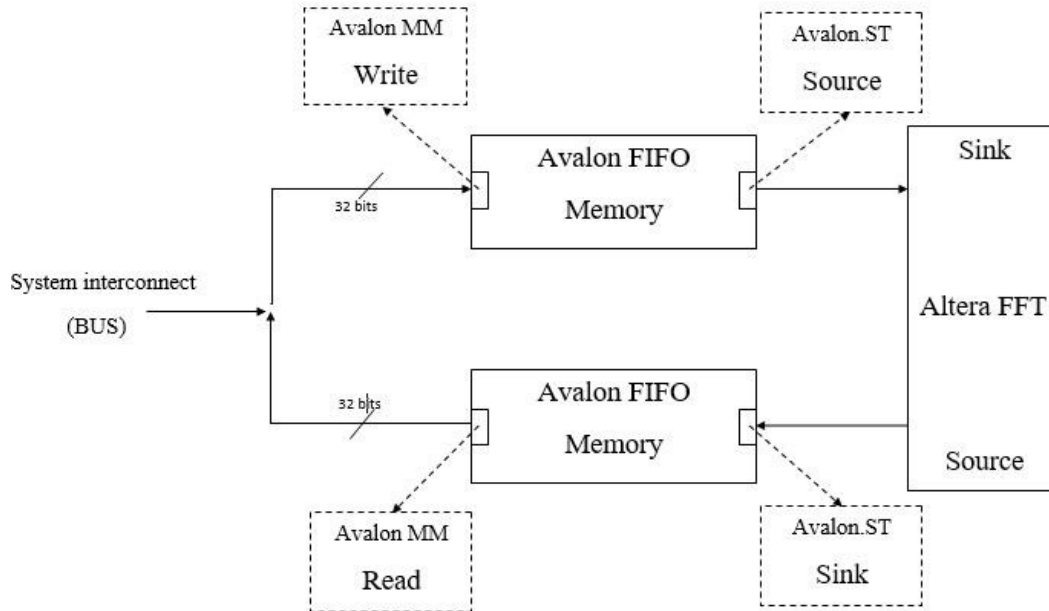


Figure 3.8: FFT BLOCK interfacing design.

We used two FIFO memory cores to interface with the FFT, ie. the sink and the source data streaming:

- **Avalon-MM Write Slave to Avalon-ST Source:**
In this mode, the FIFO's input is an Avalon-MM write slave with a width of 32 bits. The Avalon-ST output (source) data width must also be 32 bits.
- **Avalon-ST Sink to Avalon-MM Read Slave:**
The FIFO's input is an Avalon-ST sink and the output is an Avalon-MM read slave with a width of 32 bits. The Avalon-ST input (sink) data width must also be 32 bits.

Figure 3.9 shows the successful parameters matching.

We can configure output interface parameters, including: bits per symbol, symbols per beat, and the width of the channel and error signals. The FIFO performs the endian conversion to conform to the output interface protocol.

Type	Path	Message
	1b into Messages	
	DE10_Standard_QSYS.SinkINT	For Avalon-ST and Avalon-MM combinations, data width is 32 bits.
	DE10_Standard_QSYS.SourceINT	For Avalon-ST and Avalon-MM combinations, data width is 32 bits.
	DE10_Standard_QSYS.fft_ii_0	Radix-2 digit reverse implied

Figure 3.9: FFT block interface.

3.4.3 Using the Streaming FFT

The data source asserts `sink_valid` to indicate to the FFT function that valid data is available for input. Assert both the `sink_valid` and the `sink_ready` for a successful data transfer.

When the data transfer is complete, the FFT asserts `sink_sop` and loads the data samples in natural order.

The simulation waveform below in figure 3.10 shows the process.

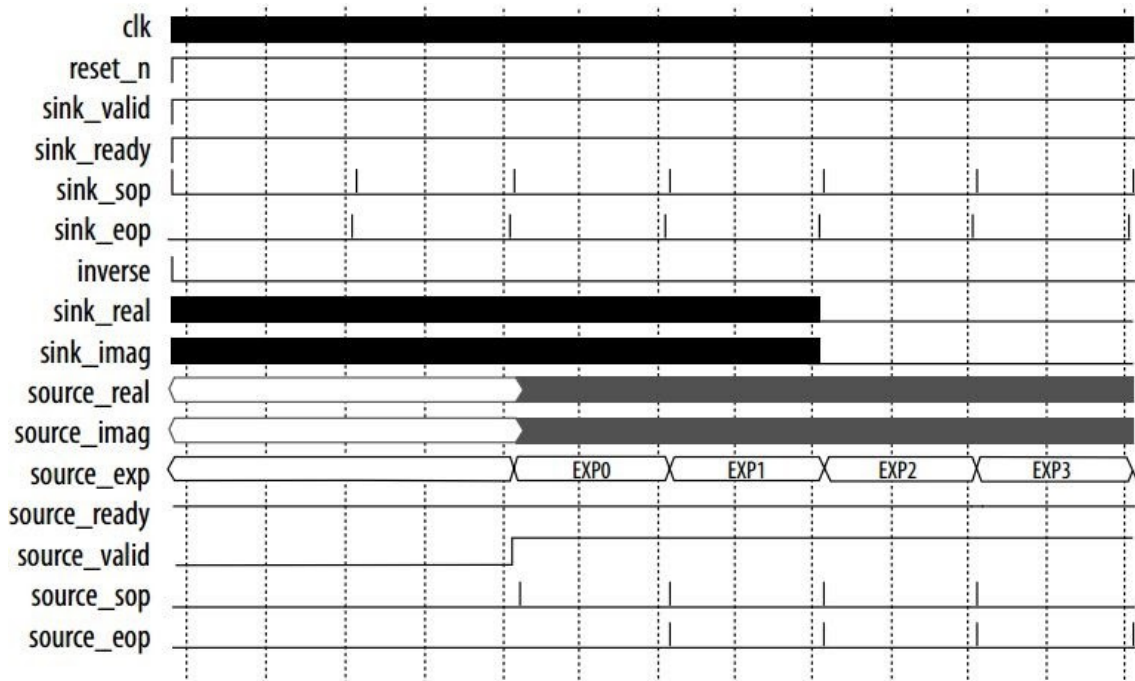


Figure 3.10: FFT Streaming Data Flow Simulation Waveform. [23]

When the final sample loads, the source asserts `sink_eop` and `sink_valid` for the last data transfer.

3.5 ADC Block

Since the ADC in the DE-10 board FPGA is a 12-bit ADC, it gives a data reading of 0 to 2^{12-1} for an input voltage range of 0 volt to 4.096 volt. Hence, it is necessary to map the digital data readings with the actual measured values using the linear relationship between them.

Once the data is sampled, it needs to be stored temporarily in order to be processed by the FFT. The storage process begins once the ADC sends to the FIFO a "dataready" signal. The FIFO block used was initially generated from the IP core library in Quartus Prime. It was configured to accept a 12-bit width and 1024 words depth. This block is also provided with full and empty signals which are used to enable the FFT block. Once the data is available, This IP block is clocked at the same clock as the ADC. The sampling process then begins as the state machine in figure 3.11 depicts.

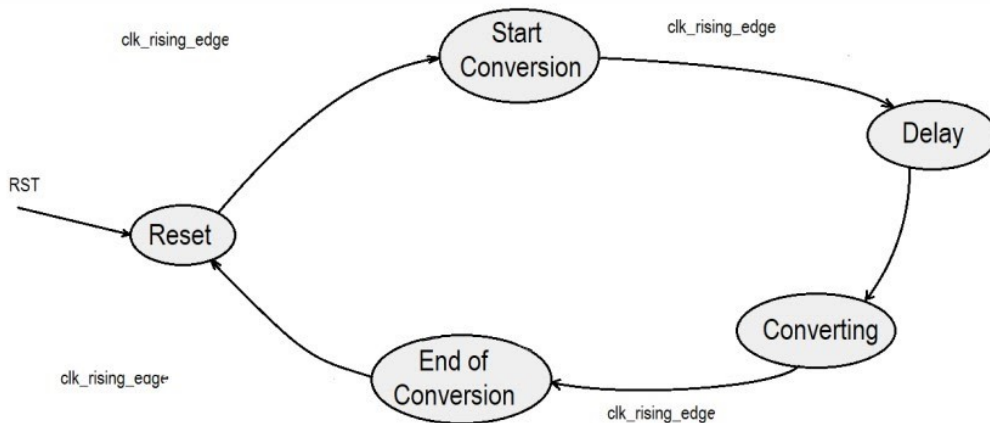


Figure 3.11: ADC state machine.

The previous blocks interconnection design is further shown in figure 3.12. It also includes JTAG IP for communication which we will discuss later.

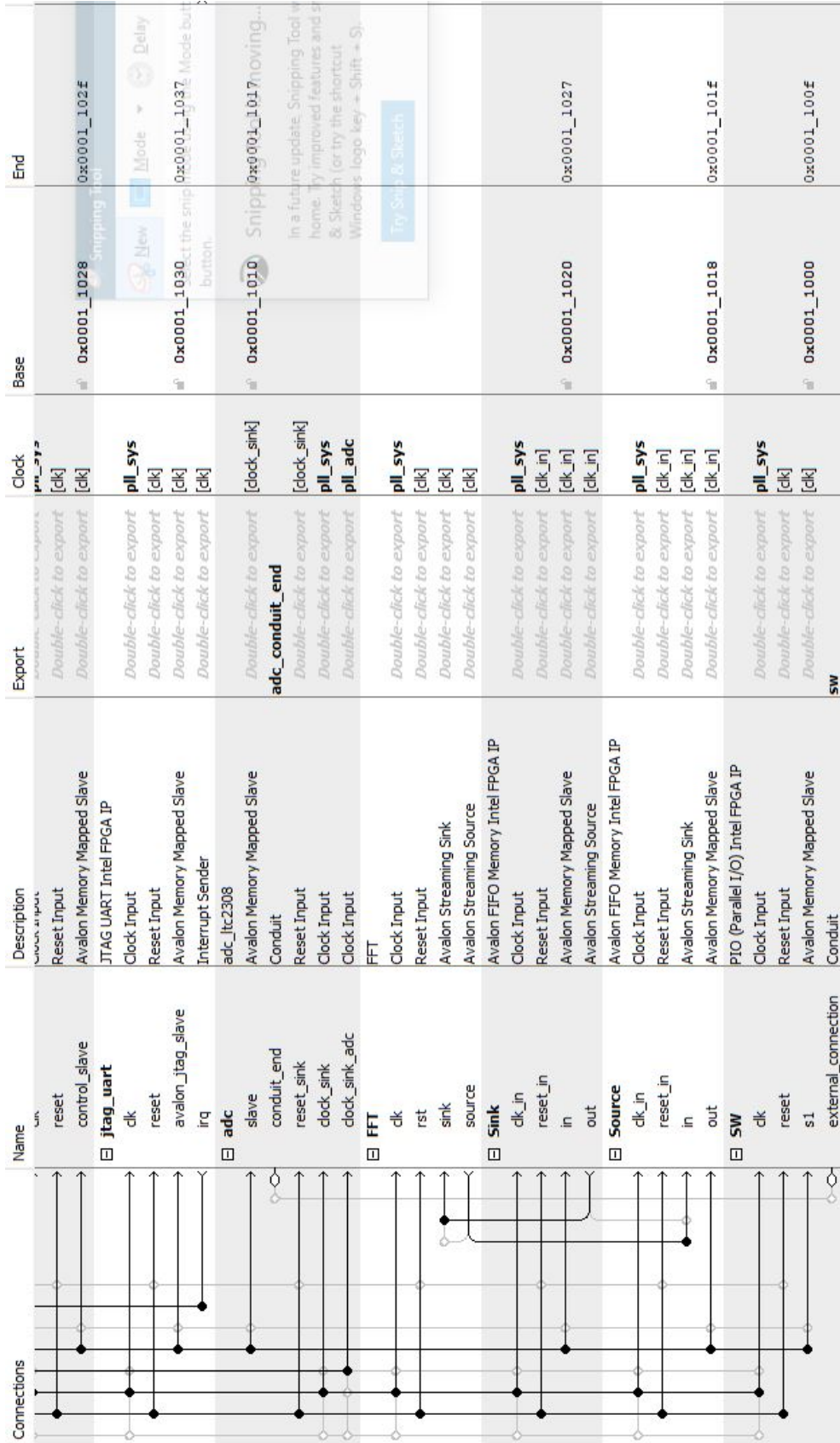


Figure 3.12: Overall system interconnection.

3.6 ALTERA NIOS II JTAG DEBUG MODULE

The Nios II a supports a JTAG debug module which gives on-chip emulation features to control the processor from a host PC. PC-based software debugging tools communicate with the JTAG debug module and provide facilities, such as the following features:

- Downloading programs to memory
- Starting and stopping execution
- Setting breakpoints and watchpoints
- Analyzing registers and memory

3.7 PLL System Block

The core takes an Platform Designer system clock as its input and generates PLL output clocks locked to that reference clock .The main frequencies generated by this block were: 100Mhz which was supplied to the processor and the FIFO block, the 40 Mhz sampling frequency which was supplied to the slow clock of the ADC as shown in figure 3.13.

Device Speed Grade:	6
PLL Mode:	Integer-N PLL
Reference Clock Frequency:	50.0 MHz
Operation Mode:	normal
<input type="checkbox"/> Enable locked output port	
<input type="checkbox"/> Enable physical output clock parameters	
Output Clocks	
Number Of Clocks:	2
outclk0	
Desired Frequency:	100.0 MHz
Actual Frequency:	100.000000 MHz
Phase Shift units:	ps
Phase Shift:	0 ps
Actual Phase Shift:	0 ps
Duty Cycle:	50 %
outclk1	
Desired Frequency:	40.0 MHz
Actual Frequency:	40.000000 MHz
Phase Shift units:	ps
Phase Shift:	0 ps
Actual Phase Shift:	0 ps
Duty Cycle:	50 %
<input type="button" value="Copy"/>	

Figure 3.13: PLL intel FPGA IP used.

Implementation and Results

4.1 Overview

Software design is the process of envisioning and defining software solutions to one or more sets of problems. In embedded systems, after building the hardware system, the software takes its roles to manipulate the data and take decisions accordingly. The software part of this project The firmware. which is a hard-coded program, or set of instructions, in the on-chip memory of the FPGA system, provides the necessary instructions for how the device communicates with the other computer hardware (It can be thought of as the software that allows hardware to run). It is programmed using the C language.

4.2 The Firmware

In any microprocessor-based system, the hardware interconnections on their own cannot provide the functionality of the system. In order for the system to be fully operational, a firmware must be included. The Firmware is the software that provides control and monitoring of the system and manages all the communications and data manipulation between the processor and the peripherals. It can be programmed using Nios II Assembly language or high-level language, containing multiple routines shared between different tasks.

The firmware function of this project depends on the received converted data From FFT.

4.3 Top Level File with RTL Viewer

Using the RTL Viewer is a good way to view your initial synthesis results to determine whether you have created the necessary logic, and that the logic and connections have been interpreted correctly by the software. It allows you to view a register transfer level (RTL) graphical representation of your Quartus II integrated synthesis results or your third-party netlist file in the Quartus II software. The top-level file containing the SoC overall project is illustrated in figure 4.1.

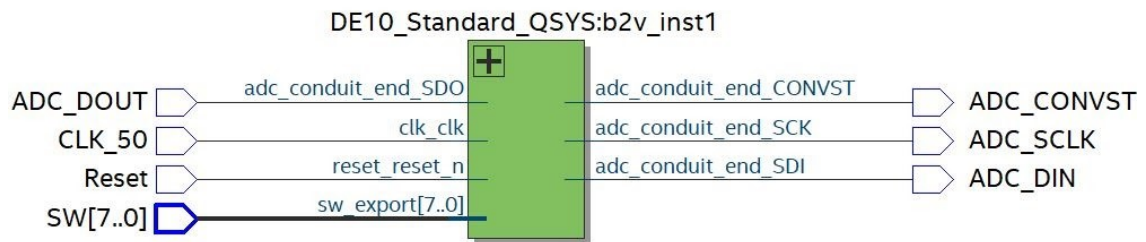


Figure 4.1: Overall Block Diagram of the On-Chip System.

4.4 Top Level File Compilation

The compilation of the whole system was successful and the summary of the compilation is shown in figure 4.2. The entire system uses 8% of the total logic elements, 6580 registers, 3% of the total pins and 8% of the total memory bits with one PLL because of the use of PLL system ip.

The top-level file containing the SoPC system is illustrated in figure 3.5.

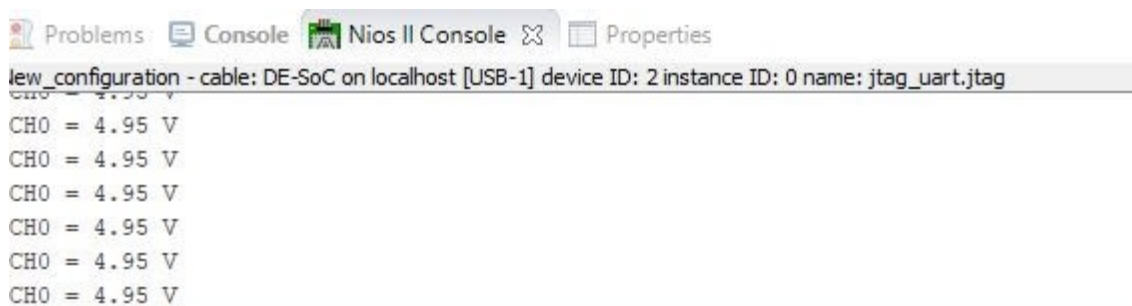
Flow Summary	
Flow Status	Successful - Sun Nov 15 15:30:29 2020
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Standard Edition
Revision Name	DE10_Standard_ADC
Top-level Entity Name	DE10_Standard_ADC
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	3,461 / 41,910 (8 %)
Total registers	6580
Total pins	14 / 499 (3 %)
Total virtual pins	0
Total block memory bits	406,940 / 5,662,720 (7 %)
Total DSP Blocks	11 / 112 (10 %)

Figure 4.2: Compilation Summary.

4.5 Results of the Implemented System

4.5.1 Testing the ADC Block Reading

After the design has been compiled, To prove if the implementation is working on the right track, an experiment to sampling voltage is applied to the circuit with peak 5.0V and altered with a potentiometer. On power-up, channel 0 is selected with SW switches. The sampling result is shown in figure 4.3.



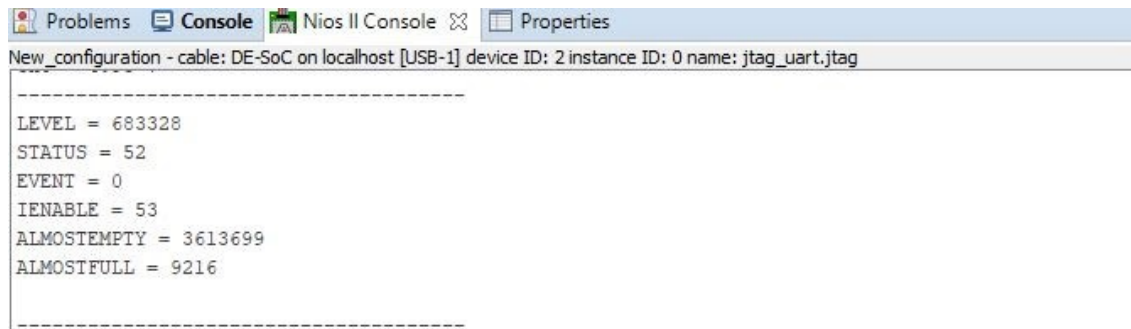
```

Problems Console Nios II Console Properties
new_configuration - cable: DE-SoC on localhost [USB-1] device ID: 2 instance ID: 0 name: jtag_uart.jtag
CH0 = 4.95 V
CH0 = 4.95 V
CH0 = 4.95 V
CH0 = 4.95 V
CH0 = 4.95 V
CH0 = 4.95 V

```

Figure 4.3: The sampling result captured by the ADC.

4.5.2 System Output



```
Problems Console Nios II Console Properties
New_configuration - cable: DE-SoC on localhost [USB-1] device ID: 2 instance ID: 0 name: jtag_uart.jtag
-----
LEVEL = 683328
STATUS = 52
EVENT = 0
IENABLE = 53
ALMOSTEMPTY = 3613699
ALMOSTFULL = 9216
-----
```

Figure 4.4: FIFO output generated level.

As seen in the output generated in figure 4.4 I am getting some data inputs from the interfaced FFT block which is connected to the data acquisition unit (ADC), hence the data from the ADC is well received. Also, these fields which are described below produced the processed data from the FFT..

LEVEL - The instantaneous fill level of the FIFO

STATUS - A 6-bit register that shows the FIFO's instantaneous status

ienabl - the value to write to the interrupt enable register

ALMOSTEMPTY - the value for the almost empty threshold level

ALMOSTFULL - the value for the almost full threshold level

4.6 Hardware Limitations

After we generate an IP for FFT from mega wizard.sof is not generated. The Compiler only generates a time-limited device programming file (`<projectname>_time_limited.sof`) that expires at the time limit.

We are assuming that this problem is due to the unavailability of the licensed version. When the evaluation time expires for any licensed Intel FPGA IP in the design, the design stops functioning. All IP cores that use the Intel FPGA IP Evaluation Mode time out simultaneously when any IP core in the design times out. Therefore we could not verify our proposed architecture. But individually we could verify many components of our proposed architecture as for example, if the data could be saved in the SD-RAM and FFT core is working. See figure 4.5 below.

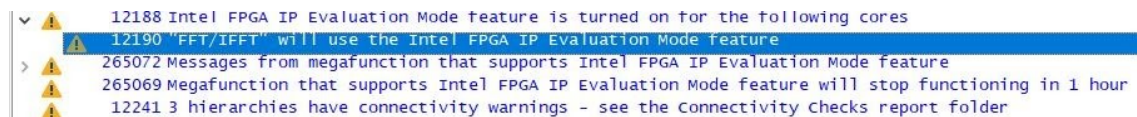


Figure 4.5: Connectivity warning message.

Conclusion

The design and implementation of a Nios II-based system for the operation of the phase measurement unit system has been realized in this project. The FPGA-based PMU was based on the Cyclone V Assembly, VHDL, Verilog HDL. Hence, the overall implementation is low cost and can be easily configured.

Although the problems and obstacles that I faced, especially with the health conditions that the world is going through and the effect of it on me when it reached my family , It can be concluded that SoC-based PMU was designed and implemented and the objectives of this work have been successfully met and they are as follows:

- Implementation of a part of a Phasor Measurement Unit System based on FPGA by exchanging and processing the data received from the system.
- Apply every single knowledge of Hardware design and computer engineering that was acquired during the past five years.

Effort and time were spent to debug software bugs and hardware problems to improve the system and make it operational .However, the project still lacks many features that can be added to improve its performance and make it suitable to be realized on the ground.

Future Work

The project was designed using an analogue input from a potentiometer for study purpose, for wide area monitoring purpose, the prototype can be implemented using function generator with some modifications, for example, by testing the system with 3 phases, increasing the sampling rate for better accuracy, attaching a GPS module in the system for time stamping etc.

In the future scope of research, the prototype is to be used in studying the effects of faults on the Phasor estimates, power monitoring system, and other benefits that could be gained from the phasor measurement unit.

References

- [1] IEEE Standard for Synchrophasor Measurements for Power Systems, IEEE Std. C37.118.1 -2011. [Online]. Available: <http://standards.ieee.org/findstds/standard/C37.118.1-2011.html>.
- [2] IEEE Standard for Synchrophasor Data Transfer for Power Systems, IEEE Std. C37.118.2- 2011. [Online]. Available: <http://standards.ieee.org/findstds/standard/C37.118.2-2011.html>
- [3] G. Missout and P. Girard, Measurement of bus voltage angle between montreal and SEPT-ILES, IEEE Trans. Power App. Syst., vol. PAS-99, no. 2, pp. 536-539, Mar. 1980
- [4] G. Missout, J. Beland, G. Bedard, and Y. Lafleur, Dynamic measurement of the absolute voltage angle on long transmission lines, IEEE Trans. Power App. Syst., vol. PAS-100, no. 11, pp. 4428-4434, Nov. 1981.
- [5] P. Bonanomi, Phase angle measurements with synchronized clocks principle and applications, IEEE Trans. Power App. Syst., vol. PAS-100, no. 12, pp. 5036-5043, Dec. 1981.
- [6] A. G. Phadke and J. S. Thorp, History and applications of phasor measurements, in Proc. IEEE PES PSCE, 2006, pp. 331-335.
- [7] A. G. Phadke, Synchronized phasor measurements A historical overview, in Proc. IEEE/PES Transmiss. Distrib. Conf. Exhib. Asia Pacific, Oct. 6-10, 2002, vol. 1, pp. 476-479.
- [8] A. G. Phadke and J. S. Thorp, Synchronized Phasor Measurements and Their Applications. New York, NY, USA: Springer-Verlag, 2008.

- [9] P. Denys, C. Counan, L. Hossenlopp, and C. Holweck, Measurement of voltage phase for the French future defence plan against losses of synchronism, *IEEE Trans. Power Del.*, vol. 7, no. 1, pp. 62-69, Jan. 1992.
- [10] Paolo Castello, Algorithms for the synchrophasor measurement in steady-state and dynamic conditions, University of Cagliari, March 2014.
- [11] A. Agarwal, N. Verma, H. Tiwari, J. Singh, Varun Maheshwari Design and Development of Phasor Measurement Unit on FPGA, December 2016.
- [12] B. Kasztenny and M. Adamiak, Implementation and performance of synchrophasor function within microprocessor based relays, in *Proc. 61st Annu. Georgia Tech. Protect. Relaying Conf.*, Atlanta, GA, USA, May 2-4, 2007, pp. 1-43.
- [13] Bogdan Vicol, modern technologies for power systems monitoring, *ELS International Symposium*, September 2013.
- [14] A. J. Roscoe, I. F. Abdulhadi, and G. M. Burt, P-class phasor measurement unit algorithms using adaptive filtering to enhance accuracy at offnominal frequencies, in *Proc. IEEE Int. Conf. SMFG*, Nov. 14-16, 2011, pp. 51 -58.
- [15] Open source FFT core, [Online]: <https://zipcpu.com/dsp/2018/10/02/fft.html>. Accessed on 30-06-2019
- [16] Own Work, Xilinx Inc., 29 October 2012, 15:30:46.
- [17] R. Wisniewski, in *Synthesis of compositional microprogram control units for programmable devices*, Zielona Góra: University of Zielona Góra, 2009, p. 153.
- [18] Intel FPGA, DE10 Standard User Manual, 2017.
- [19] Linear Technology, LTC2308 ADC datasheet, 2007.
- [20] P. P. Chu, “Embedded SoPC design with Nios II processor and VHDL examples”, A JOHN WILEY & SONS, INC., PUBLICATION, 2011.
- [21] Altera, “Introduction to the Altera Qsys System Integration Tool”.
- [22] Intel Corporation, Embedded Peripherals IP User Guide, Updated for Intel®Quartus®Prime Design Suite: 20.2, UG-01085 | 2020.09.21
- [23] Intel Corporation, FFT IP core user guide, UG-FFT | 2017.11.06

