Democratic and Popular Republic of Algeria

Ministry of Higher Education and Scientific Research
Abderrahmane Mira University
Faculty of Technology

Department of Automation, Telecommunication, and Electronics

Final Year Project

For the Master's Degree

Field: Automatic
Specialization: automatic and industrial computing

Theme

FPGA-based real time implementation of MPPT controller for photovoltaic energy conversion system

Prepared by :

- BOUDIAB Chahinaze
- GANI Adel

Supervised by :

Mr.F.YAHIAOUI

Mme.N.BERKANI

Examined by :

Mr H.BELKAID

Mme S.MEZZAH

Academic year: **2023/2024**

Acknowledgements

Firstly, we thank God for helping us and giving us the courage, will, and knowledge to complete this modest work.

Our deep and sincere thanks go first to our supervisors Mr. YAHIAOUI and Mrs. BERKANI for assisting and guiding us throughout this project. May they find here the expression of all our acknowledgments and deep respect for the trust they placed in us. Thank you for their interest in our work and for their valuable advice.

We also wish to express our gratitude to the jury members, Mrs. MEZZAH and Mr. BELKAID, for their scientific contributions during the evaluation of this work. Please accept our profound respect and sincere thanks.

Our heartfelt thanks also go to all the teachers in the Department of Automatics, Telecommunications, and Electronics, as well as to all our colleagues.

Finally, we extend our sincerest thanks to our parents; without them, we would not be where we are today, and to our brothers and sisters who accompanied, supported, and encouraged us throughout the completion of this final project

Dedication

I dedicate this work

To my father, the source of my strength

To my mother, who gives me hope

To my sisters, who taught me courage

To my brothers, source of my security

Chahinaze

I dedicate this work to

My cherished parents

My brothers Anis and Mehdi

My friends Mr.Djamyl, Lotfi, Dalim, Rayan, Walid, Said, my friends

from V1_gaming in Discord and some special person to me

ADEL

Table of contents

List of figures

List of tables

List of abbreviations

ABEL: Advanced Boolean Expression Language

AC: alternative current

AHDL: Altera HDL

AMD: Advanced Micro Devices

CUPL: Computer Aided Logic Design Programming

CLBs: Configurable logic blocks

DC: direct current

FCC :Current Coefficient Fraction

Fpga: Field Programmable Gate Arrays

GUI: graphical user interface

HCPLDs :Complex Programmable Logic Devices

HDL: Hardware Description Language

HLS :High-Level Synthesis

IDE: integrated development environment

IEEE : Institute of Electrical and Electronics Engineers

IOBs: Input/output blocks

INC: Incremental Conductance

IP: Intellectual Property

IPI:IP Integrator

Iph: current source

ISE: Integrated Software Environment

LUTs: Look-Up Tables

MPP: Maximum Power Point

MPPT: Maximum Power Point Tracking

PLDs: Programmable logic devices

PLL: Phase-Locked Loop

PVG:	Photovoltaic Generator
PWm:	pulse width modulation
P&O:	Perturb & Observe
Rs:	series resistance
Rsh:	internal shunt resistance
RTL:	Register Transfer Level code
SPLDs:	Simple Programmable Logic Devices
SDRAM:	Synchronous Dynamic Random Access Memory
VCO:	Voltage Coefficient Fraction
VHDL:	VHSIC Hardware Description Language
VHDL-AMS:	VHDL-Analog Mixed Signal
VHSIC:	Very High Speed Integrated Circuit
Vitis HLS:	Vitis High-Level Synthesis
Xadc:	Xilinx analog digital converter

General introduction

General introduction

The solar photovoltaic (PV) industry is currently undergoing rapid expansion amidst a global shift towards renewable energy. This growth is underscored by compelling statistics, solidifying solar PV's pivotal role in transitioning to cleaner and sustainable energy sources. According to recent data from Statistic, global PV capacity is projected to reach 1.3 terawatts (TW) by 2023, highlighting increasing global adoption driven by technological advancements, cost efficiencies, and supportive renewable energy policies.

The surge in PV capacity exemplifies solar energy's critical contribution to mitigating climate change through its provision of clean, renewable, and cost-effective energy. This trend signifies the escalating significance of solar power in the global energy landscape, promising a greener and more sustainable future. Algeria, with its abundant solar resources averaging between 2,000 to 3,900 sunlight hours annually and daily irradiation levels of 3,000 to 6,000 Wh/m^2, stands as a notable reservoir for solar energy.

In response to these favorable conditions, the Algerian government launched an ambitious long-term renewable energy program aiming to generate 22,000 MW of electricity by 2030, predominantly through solar thermal, PV, and wind power. This initiative not only aims to meet domestic electricity needs but also seeks to export an additional 10,000 MW to neighboring countries, thereby conserving natural gas reserves for domestic consumption and export.

Despite the promising outlook for solar PV, maximizing power extraction efficiency remains a critical concern due to inherent limitations in conversion efficiency. Consequently, the integration of Maximum Power Point Tracking (MPPT) controllers is essential for optimizing power output from PV systems.

This project focuses on implementing the Perturbation and Observation (P&O) MPPT method using Field-Programmable Gate Arrays (FPGAs), versatile programmable circuits known for their rapid development capabilities and modularity. The objective is to explore the implementation of this MPPT technique through VHDL programming, synthesis, testing, and simulation on an FPGA board. This study is structured into three main chapters:

- Chapter 1 provides a comprehensive overview of photovoltaic systems and DC-DC converters, covering the photovoltaic effect, solar cell characteristics, MPPT techniques, and various DC-DC converter topologies crucial for efficient power conversion.
- Chapter 2 focuses on FPGA technology and VHDL programming, detailing FPGA architecture, VHDL fundamentals, and their application in hardware design for MPPT control.
- Chapter 3 delves into the practical implementation of a PV panel system, discussing hardware components such as current and voltage sensors, as well as software aspects including the P&O algorithm implementation, Xadc functionality, and Pulse Width Modulation (PWM) techniques.

Each chapter concludes with a summary of findings and conclusions drawn from the respective implementation stages, contributing to the overall understanding and advancement of MPPT methodologies in solar PV applications.

Chapter 01
Photovoltaic and MPPT

I.1 Introduction

There are many technology that uses the solar energy like Solar Thermal Energy, Solar Architecture and Concerted Solar Power…

This chapter will focus only on one technology which is the Photovoltaic systems (PV). The block diagram of the standalone photovoltaic conversion energy system under consideration is shown in Figure 1.

The energy source of the PV system is constituted by a photovoltaic module, the Boost converter acts as interface between the solar panel and the load, and the MPPT controller allows reaching the available maximum power. **[1]**

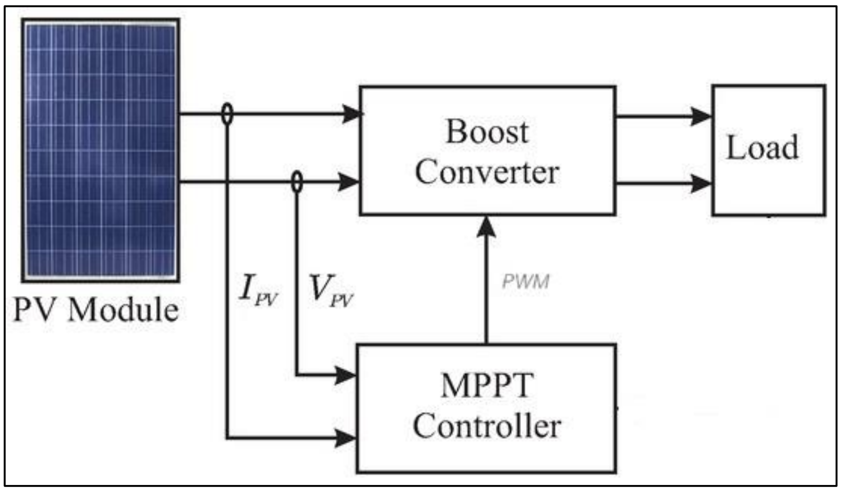


Figure I. 1: Block diagram of PV energy conversion system [1].

I.2 Photovoltaic systems :

I.2.1 Solar cells :

I.2.1.1 Solar cells structure :

The basic element of a solar PV system is solar cell. Solar cells consist of semiconductor materials (usually silicon) and are coated such that a negative electric field on the front (the side that faces the sun) and a positive electric field is created on the backside.

Electrons are loosened from the atoms in the semiconductor material when solar cells are exposed to photons (energy from the sun). This ‘loosening’ creates pairs of electron-holes and causes the electrical conductors to be attached to the positive and negative sides. In so doing, an electrical circuit emerges, within which electrons are captured as an electric current referred to as a photocurrent, Iph. The solar cell remains passive and functions like a diode during the night or other times when it is dark outside. In other words, it acts as a p–n junction, creating neither current nor voltage. Conversely, the cell creates a current known as a diode or dark current, if it is hooked to a sizeable external voltage supply. **[2]**

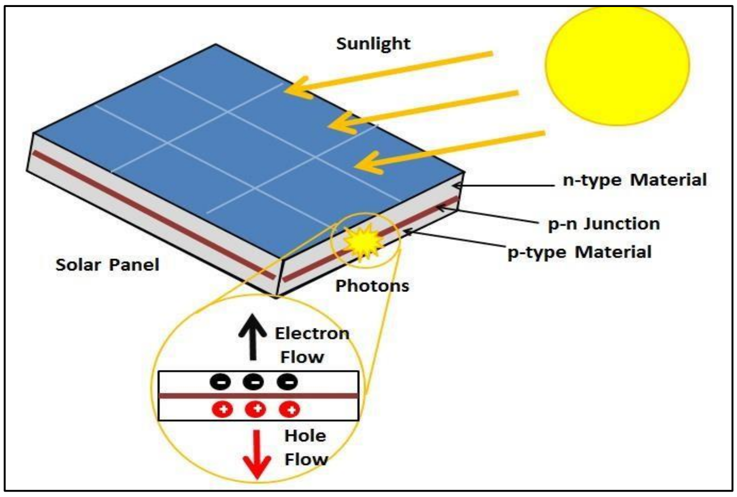


Figure I. 2: description of solar cell

I.2.1.2 Equivalent circuit model of solar cell :

Typically, a solar cell is represented by an electrical equivalent one-diode model as shown in Figure 3 (Lorenzo, 1994). This circuit model can be used for separate cells, a module including several cells, or an array consisting of numerous modules. **[2]**

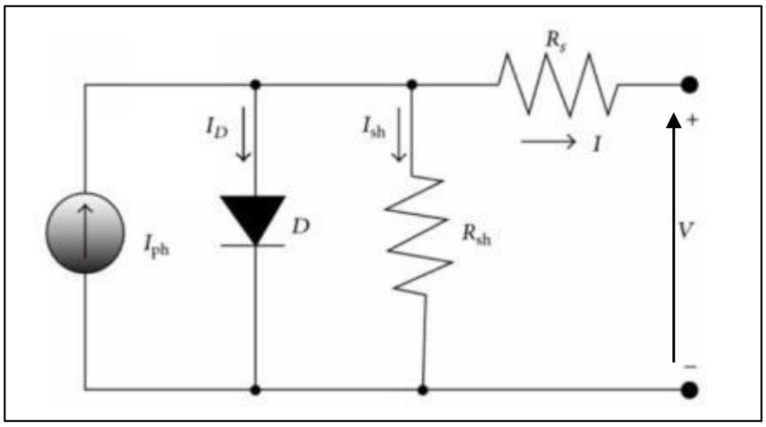


Figure I. 3: Photovoltaic cell equivalent circuit [2]

As shown in Figure 3, the model consists of a current source I_{ph}, a single diode, and a series resistance R_s that indicates the resistance level within the cells. The diode comprises an internal shunt resistance R_{sh} as well. The net current I which marks the difference between the photocurrent and the normal diode current, is calculated as :

$$I = I_{ph} - I_0 \left[exp\left(\frac{V + R_s\, I}{\alpha\, V_{th}}\right) - 1\right] - \left(\frac{V + R_s\, I}{R_{sh}}\right) \qquad (1)$$

where, I_0 is dark saturation current (A),

α is the diode's ideality factor, it is between 1 and 2,

V_{th} is the thermodynamic potential of the cell, expressed as follows :

$$V_{th} = \frac{KT}{q} \qquad (2)$$

where, K is Boltzmann's gas constant, $1.380658e^{-23}$J/K,

T est the cell's absolute temperature (Kelvin),

q is the charge of an electron, $1.60217733e^{-19}$C.

Load resistance is usually significantly smaller than shunt resistance. However, load resistance is usually much greater than series resistance, which means that a relatively small amount of power actually dissipates within the cell. If these two resistances are ignored, (1) is reduced to:

$$I = I_{ph} - I_D = I_{ph} - I_0 \left[exp\left(\frac{V+R_s\, I}{\alpha\, V_{th}}\right) - 1\right] \qquad (3)$$

I.2.1.3 Characteristics of solar cells

An illuminated solar cell can provide a certain photovoltage at a given photocurrent. A combination of values of photocurrent and photovoltage at which a solar cell can be operated is called a working point. A particular working point of a solar cell is fixed with a load resistance (R_L) due to the Ohm's law. **[3]**

$$R_L = \frac{U}{I} \qquad (4)$$

In terms of Ohm's law, the photovoltage is very low at very low R_L, and the photocurrent is very low at very high R_L. The short- and open-circuit operation conditions of a solar cell are defined as a R_L which is equal to zero or which is infinitely high, respectively. The values of the photocurrent and of the photovoltage at short- and open-circuit conditions are called short-circuit current (I_{SC}) and open-circuit voltage (V_{OC}), respectively. The electric power is equal to zero at short- and open-circuit operation of a solar cell.

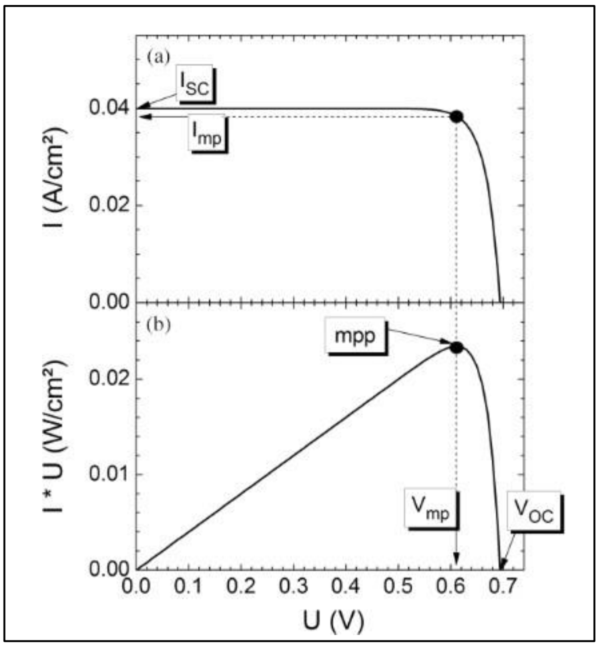


Figure I. 4 :Example of a current–voltage characteristic (a) and of the corresponding power–voltage characteristic (b) of a solar cell under illumination [3]

A current–voltage characteristic (I–V characteristic) of a solar cell is a plot of all possible working points in a considered range. Figure 4 shows schematically the I–V characteristic of a solar cell under illumination.

. Whith :

I_{SC} : The short-circuit current density
V_{OC} : The open-circuit voltage
V_{mpp} : the voltage at the maximum power point
I_{mpp} : The current at the maximum power point
MPP : maximum power point

There is one combination of current and voltage at which the power of the solar cell has its maximum (I_{mpp} and V_{mpp}, respectively). This point on the I–V characteristic of an illuminated solar cell is called the maximum power point. **[3]**

$$P_{mpp} = I_{mpp} \cdot V_{mpp} \qquad (5)$$

The values of I_{SC} and V_{OC} can be measured easily. Therefore, it is convenient to characterize the maximum power of a solar cell with I_{SC}, V_{OC} and an additional parameter instead of characterizing the solar cell with I_{mpp} and V_{mpp}. The additional parameter sets the product of I_{mpp} ,

V_{mpp} in relation to the product of I_{SC} and V_{OC}. This parameter describes the amount by which the I_{SC}–V_{OC} rectangle is filled by the I_{mpp}–V_{mpp} rectangle and is therefore called the fill factor (FF). **[3]**

$$FF = \frac{I_{mpp} \cdot V_{mpp}}{I_{sc} \cdot V_{oc}} \tag{6}$$

The η of a solar cell is defined as the ratio between the power extracted at the mpp of the solar cell and the power of the sunlight at which the solar cell is illuminated (Psun).

$$\eta = FF\ \frac{Isc \cdot Voc}{Psun} \tag{7}$$

The solar energy conversion efficiency is a decisive parameter for costs and sustainability of PV energy production. The higher the value of η, the lower the amount of material and area needed for a PV power plant. **[3]**

I.2.2 PV module Model

A single cell is not of much practical use, producing less than a volt, that's why several cells have to be connected to produce a useable voltage.

PV cells are interconnected to form modules, also known as solar panel, these modules are also connecter together to form arrays as illustrated in figure 5.

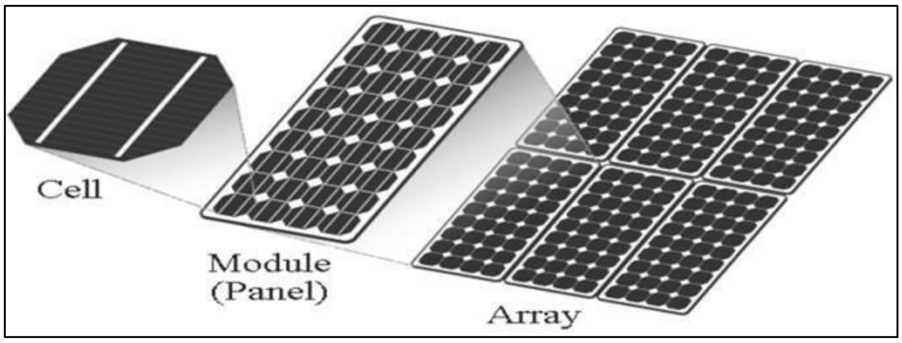


Figure I. 5: cell, module and array.

A PV module consists of a set of solar cells that are connected both in series and in parallel. If the solar cells are connected in series, the voltage of the PV module is increased while the current keeps constant. Conversely, if a parallel connection is considered, the PV module current is increased while the voltage remains constant.(figure 6)

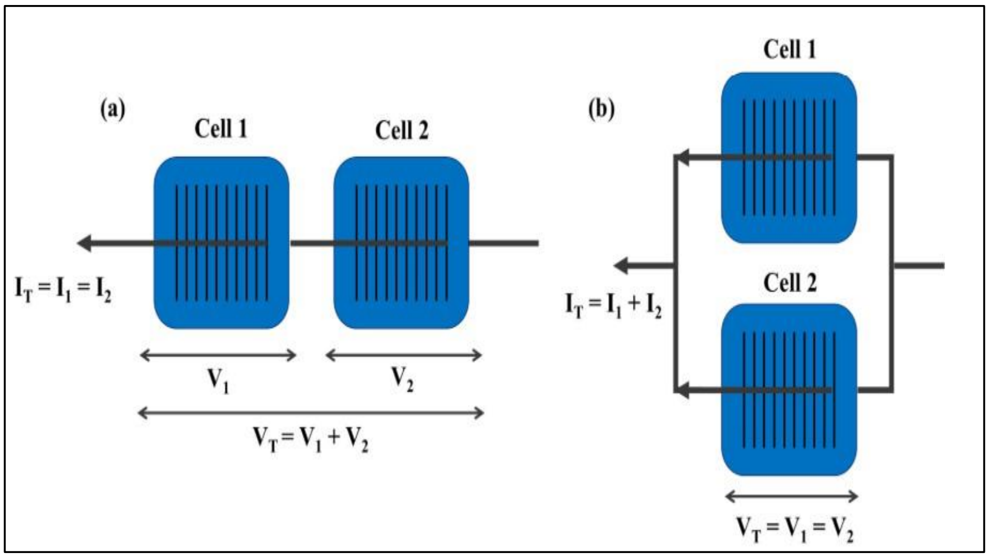


Figure I. 6: Schematics of (a) series- and (b) parallel-connected solar cells.

The characteristics of a PV module are plotted in Figures 7 and 8.

Figure 7a and 7b shows the I-V curves and P-V curves respectively for different values of solar irradiance considering a constant temperature of 25 °C, and Figure 8 shows the I-V curves considering different temperature values and a constant solar irradiance of 1000 W/m^2 . **[1]**

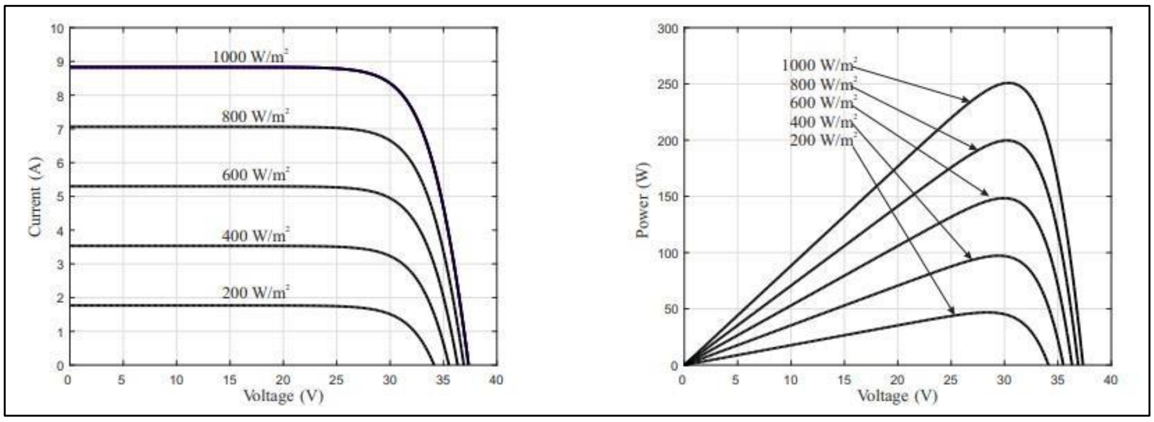


Figure I. 7: Curves of the PV panel under different irradiance levels and a constant temperature of 25 °C [1]

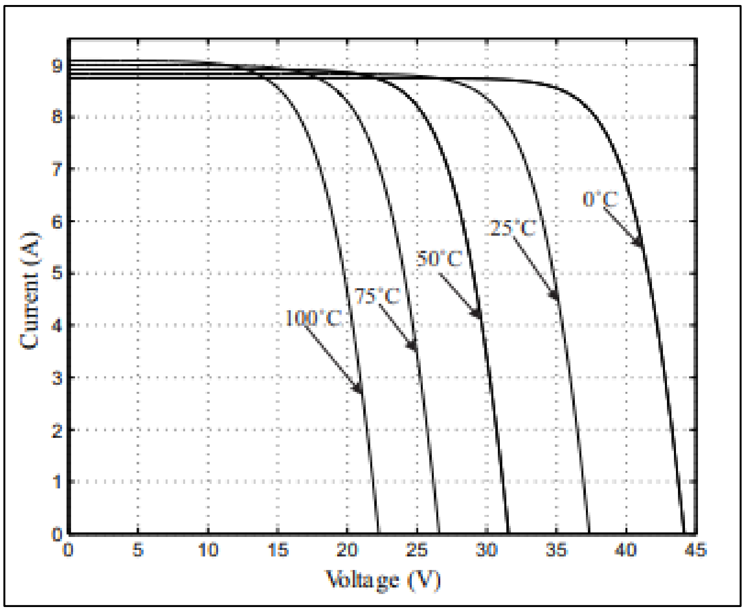


Figure I. 8 : I-V curves of the PV panel under different temperature levels and a constant irradiance of 1000 W/m² [1]

Notice that the voltage is most likely to be influenced by temperature variations, and the current by solar irradiance changes. As illustrated in Figures 7 and 8, high solar irradiance values increase the current, and high temperatures reduce the output voltage of a PV model. Then, the power conversion capability can be affected under conditions of high temperature and low solar irradiance.

I.3 Maximum Power Point Tracking :

I.3.1 Definition of MPPT :

The Maximum Power Point Tracking (MPPT) command is a method that allows a PV generator to operate at its maximum power regardless of meteorological conditions such as solar irradiation and temperature. Its principle is based on the automatic variation of the duty cycle α of a DC-DC converter to an appropriate value in order to continually maximize power at the PV panel output.

Depending on the type of MPPT controller used, several methods can be considered to extract the maximum power from a solar panel.

I.3.2 Criteria for evaluating an MPPT command:

The quality of the MPPT control can be defined as the position of the operating point of the system relative to the Maximum Power Point (MPP).

The actual power provided by the Photovoltaic Generator (PVG) depends on the control methods used at the converter level (MPPT, voltage control, direct connection, etc.). The efficiency of the resulting operating point (η_{MPPT}) is used to measure the efficiency of the power converter control.

η_{MPPT} determines the percentage (%) of power loss from the PV module compared to the maximum power that can be produced. **[4]**

$$\eta MPPT = \frac{Peff}{Pmax} \quad (8)$$

with :

η_{MPPT} : The operating point efficiency.

P_{eff} : The power delivered by the PV array.

P_{max} : The maximum power potentially available at the output of a PV panel

The evaluation of the performance of an MPPT command is not limited to the single parameter η_{MPPT}. Other criteria, equally important, are used for assessing the control quality : **[5]**

a) Simplicity and cost :

The MPPT control should be of simple design, energy-efficient, and have reasonable operating costs so that its use offsets the additional expenses invested.

b) Dynamic response :

The MPPT control must have good dynamic performance to control the adaptation stage and ensure that, following changes in illumination or temperature, a new MPP is found as quickly as possible.

c) Flexibility :

Regardless of the operating conditions, the MPPT control must be precise and stable. It should be designed to operate accurately and robustly universally with panels of different technologies without requiring significant modifications.

d) Competitive across a wide range of power :

The MPPT control used must track the MPP generated by the photovoltaic module, regardless of the level of sunlight.

The MPPT control is considered competitive if the MPP is reached with a static error, which corresponds to the operating point's position relative to the MPP, relatively low across a wide power range.

I.3.3 The various MPPT control algorithms :

Over the past few decades, numerous methods for finding the MPP have been developed. Among these methods, the most commonly encountered ones are commonly referred to as: Perturb & Observe (P&O), Incremental Conductance (INC), Hill Climbing, Voltage Coefficient Fraction (VCO), Current Coefficient Fraction (FCC), and MPPT controls based on different artificial intelligence methods.

I.3.3.1 Perturb and Observe (P&O) Algorithm :

P&O method is the most common MPPT method today. It is a renowned tracker based on creating a perturbation, observing the results, and comparing the outcome with the previous ones. In this case, the algorithm changes the duty cycle step size and remeasures the PV panel's current, voltage, and power, then compares them with the previous values. This algorithm measures the PV panel's voltage and power, and then it computes dV and dP to determine the perturbation sign, which can be positive or negative. The path the subsequent perturbation will take is described in table1. The objectif is to obtain the result expressed in (9) and (10) to reach the maximum output power that can be obtained from the PV module. **[6]**

$$\frac{dP}{dV} = 0 \qquad (9)$$

$$P(k) - P(k-1) = 0 \qquad (10)$$

Perturbation	Change in power	Next perturbation
Positive	Positive	Positive
Positive	Negative	Negative
Negative	Positive	Negative
Negative	Negative	Positive

Table 1: Table of operation for the P&O method

When analyzing the power curve illustrated in Figure 9, it becomes evident that it allows for voltage adjustments in both upward and downward directions until reaching the MPP. If there is a decrease in the PV array’s operating voltage in a specific direction, resulting in increased power extraction from the PV array, it indicates that the operating point is approaching the MPP. As a consequence, the operating voltage further decreases in the same direction. **[6]**

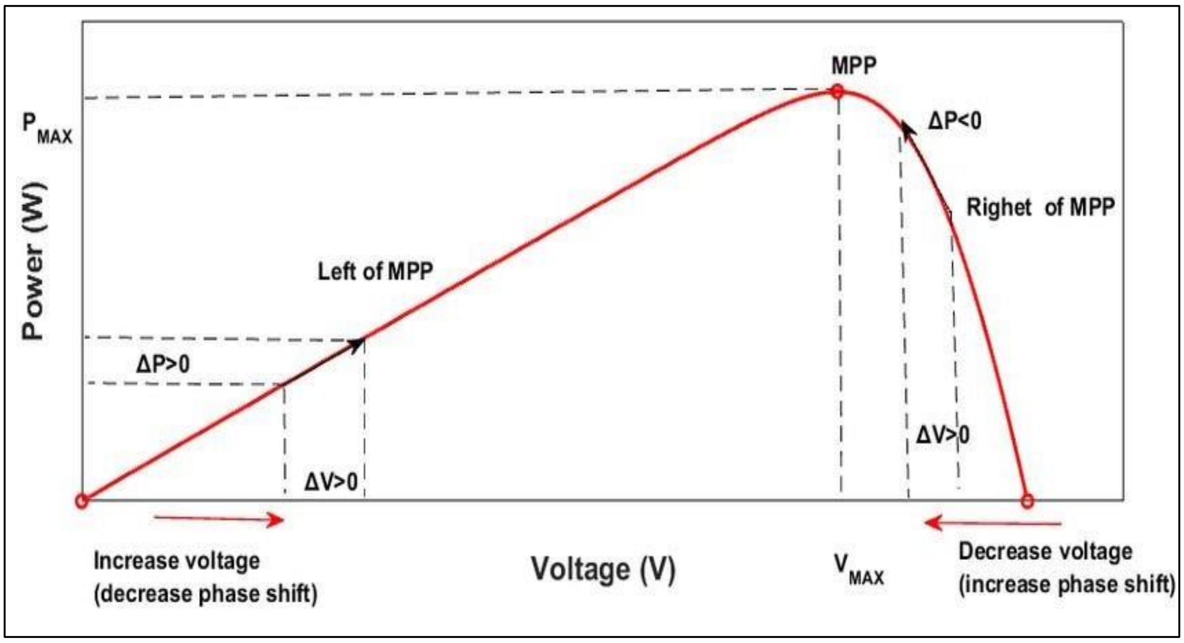


Figure I. 9: P&O power curve and MPP.

The P&O algorithm capitalizes on the power–voltage (P-V) curve, making it a widely adopted technique due to its straightforward implementation.

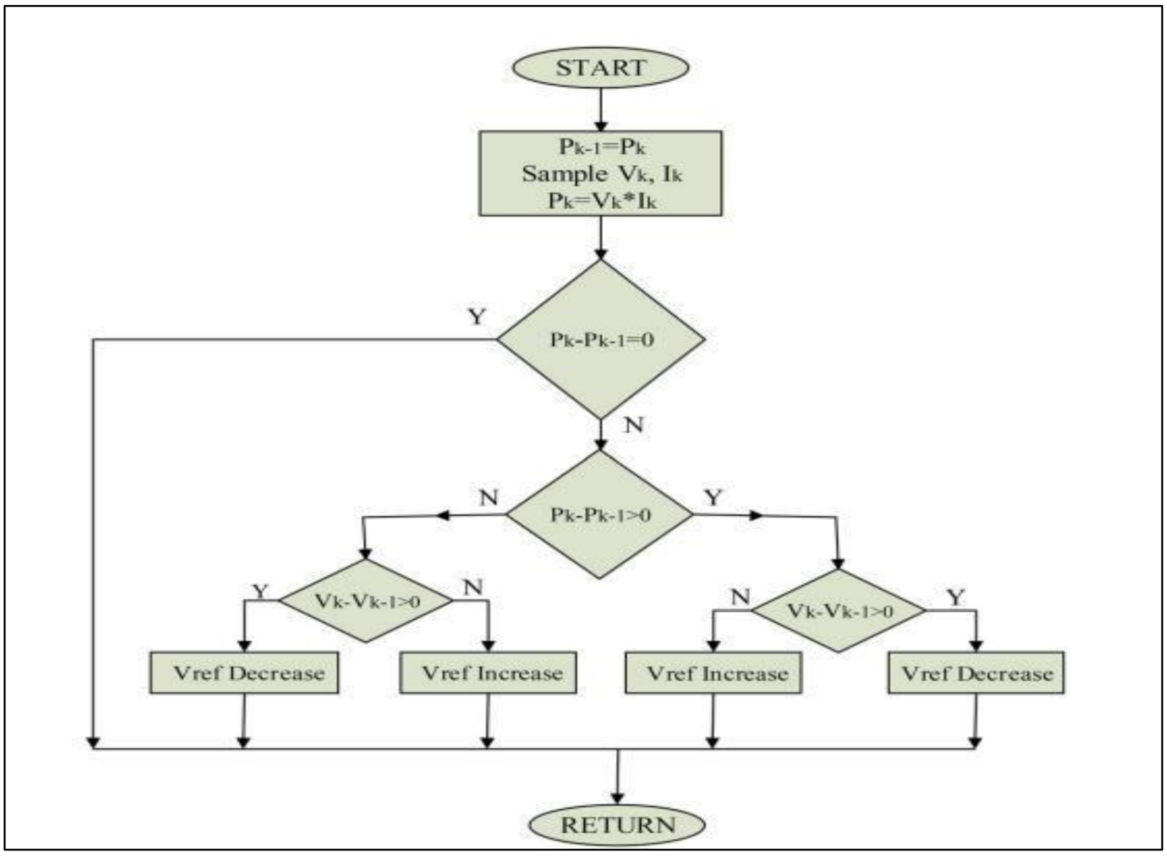


Figure I. 10 : P&O MPPT algorithm

As illustrated in Figure 10, the P&O algorithm start with the measurement of instantaneous voltage and current values obtained from the PV panel, followed by the calculation of instantaneous power values. This computational sequence repeats itself during each switching cycle until the MPP is reached.

The alteration in power value, referred to as ΔP, is computed by subtracting the current power value from the preceding one. Similarly, the procedure is replicated for panel voltage, yielding ΔV, which signifies the voltage change, achieved by deducting the current voltage value from the previous voltage value. This iterative process persists until $\Delta P/\Delta V = 0$.

I.3.3.2 The Incremental Conductance (INC) Algorithm :

The Incremental Conductance (INC) MPPT technique is one of the most widely used MPPT strategies, which offers the advantage of fast tracking of the MPP. Compared to the P&O MPPT strategy, INC combines and utilizes the unique characteristics of both the P-V output curve and the I-V curve of the photovoltaic generator, enabling it to track the MPP more rapidly and accurately. **[7]**

This technique relies on the slope of the P-V characteristic curve (Figure 9), where the MPP is tracked when dP/dV = 0 as follows:

$$\frac{dP}{dV} = \frac{d(V.I)}{dV} = I\frac{dV}{dV} + V\frac{dI}{dV} = I + V\frac{dI}{dI}dV$$

$$\frac{dP}{dV} \simeq I + \frac{\Delta I}{\Delta V} \tag{11}$$

As shown in Figure 9, the slope of the PV panel's power curve is zero at the MPP, positive to the left, and negative to the right. The previous Equation indicates that the MPP is reached when $\Delta I/\Delta V = -I/V$.

If $\Delta I/\Delta V > -I/V$, the operating point is to the left of the MPP on the P-V curve.
If $\Delta I/\Delta V < -I/V$, the operating point is to the right.

The MPP can thus be tracked by comparing the instantaneous conductance (I/V) to the incremental conductance (ΔI/ΔV) as indicated in the flowchart in Figure 11.

The size of the increment determines the tracking speed of the MPP. Faster tracking can be achieved with larger increments, but the system may not precisely operate at the MPP and may oscillate around it. **[7]**

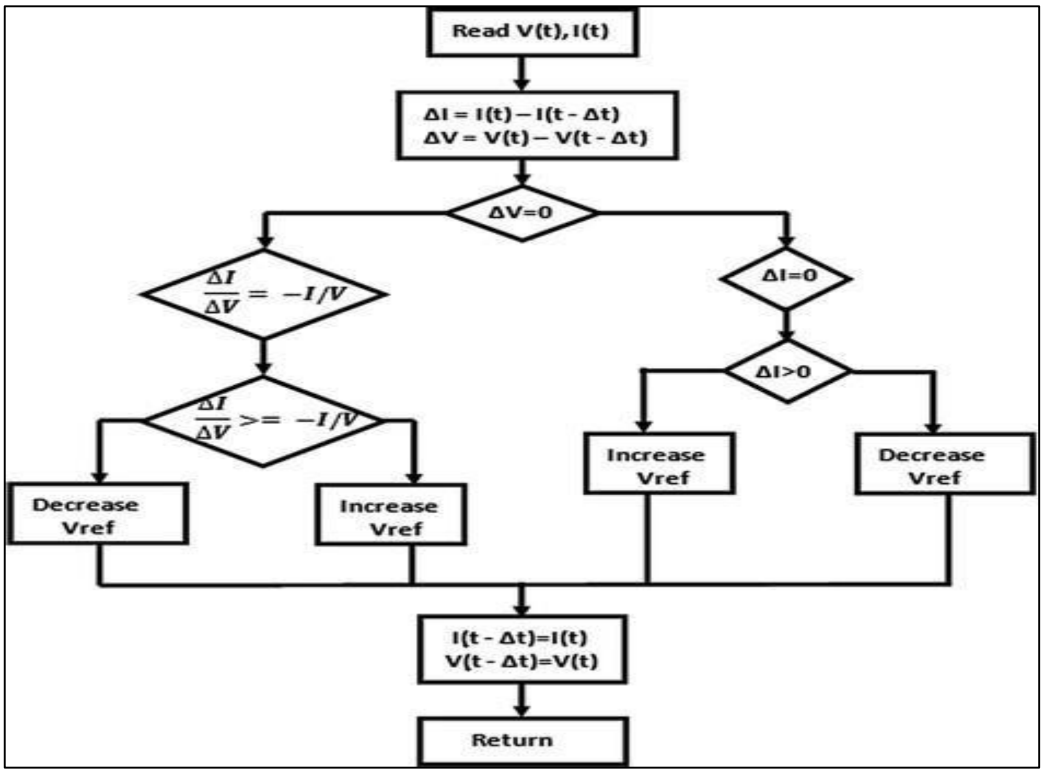


Figure I. 11: The Incremental Conductance (INC) Algorithm.

I.3.3.3 Particle swarm optimization (PSO) algorithm :

To effectively monitor the MPP of a PV system, it is advised to integrate a PSO method, which is also referred to as cooperative particles. This PSO technique aims to solve the nonlinear system optimization problem by utilizing a swarm of Np particles. The cooperative particles of the PSO algorithm work together to find and follow the MPP, guaranteeing the PV system's optimal performance. **[8]**

The diagram of the PSO controller is given in figure 12.

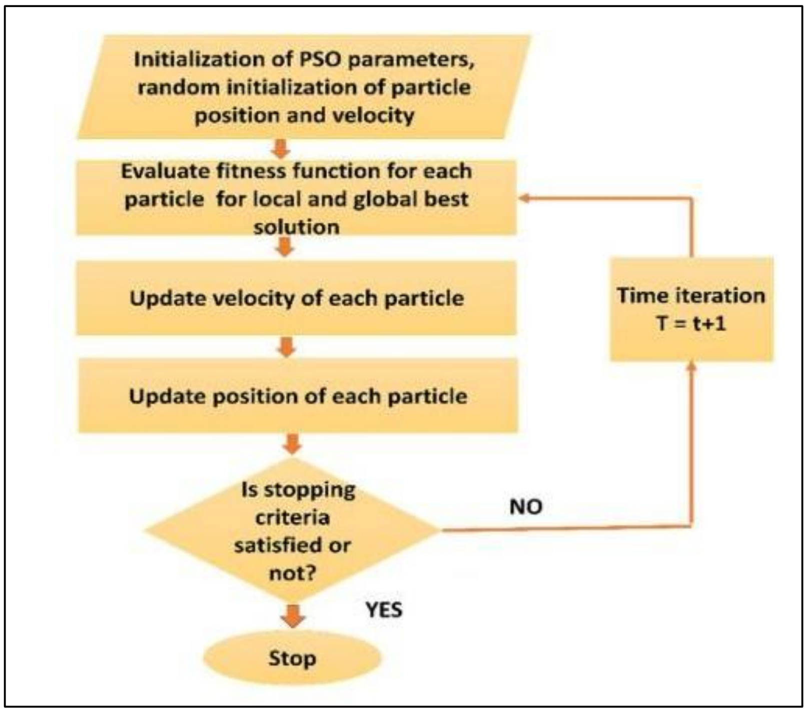

Initialization of PSO parameters, random initialization of particle position and velocity
Evaluate fitness function for each particle for local and global best solution
Update velocity of each particle
Update position of each particle
Is stopping criteria satisfied or not?
NO
YES
Stop
Time iteration T = t+1

Figure I. 12 : The diagram of PSO [8]

For discrete or mixed-variable optimization issues, PSO may need to be modified or adjusted from its initial design for continuous optimization problems. PSO's handling of discrete variables can be challenging and can produce unsatisfactory results.

I.3.3.4 Fuzzy logic controller algorithm :

Fuzzy logic controller (FLC) is an artificial intelligence system that uses fuzzy logic principles to make decisions depending on input and output parameters. Due to its ability to accommodate many input variables and efficiently consider the dynamic nature of the system, FLC outperforms other MPPT approaches in terms of capabilities. Due to this quality, FLC is more adaptable and durable when it comes to maximizing power extraction from solar panels under various environmental circumstances. **[8]**

Working principle of FLC :

FLC typically comprises three fundamental stages: fuzzifcation, rule base, and defuzzifcation. The controller takes variations of errors as inputs and produces the duty ratio variation of the DC/DC Boost converter as the output. The inputs of the fuzzy controller are defined by (12) and (13):

$$\frac{dP}{dV} = 0 \text{ at the MPP}$$

$$\frac{dP}{dV} > 0 \text{ to the left of MPP}$$

$$\frac{dP}{dV} < 0 \text{ to the right of MPP}$$

$$e(t) = \frac{\Delta P(t)}{\Delta V(t)} = \frac{P(t) - P(t-1)}{(V(t) - V(t-1)} \tag{12}$$

$$\Delta e(t) = e(t) - e(t-1) \tag{13}$$

The error in this case is denoted by "e(t)", which is the ratio of the power change (ΔP(t)) to the voltage change (ΔV(t)) between successive time steps (t and t−1). With the help of this formulation, the fuzzy logic controller can efficiently ascertain the proper duty ratio modification for the DC/DC Boost converter, guaranteeing optimal power point tracking, and analyze and interpret fluctuations in the system's performance. **[8]**

I.3.4 Advantage and disadvantage of MPPT methods :

There is so many different MPPT techniques that has been researched and developed over the years, and depending on these many criteria (complexity, cost, speed of convergence, sensors required...), each of those techniques has his advantage and disadvantage. Some of them are shown in the table 2 :

MPPT techniques	Advantages	Disadvantages
P&O	- Reliable result. - It is not dependent on the panel properties and on characteristics.	- not suited in fast changing environmental conditions. - At the steady state condition the output voltage and current signals of PV panel oscillates which causes losses.
INC	- Higher accuracy than P&O	- Tuning trade-off related to system dynamics and accuracy of the steady-state
FLC	- There is no need for a mathematical model - Tuned based on human knowledge of the system	- High computational resources when the number of membership function increase
PSO	- Simple for solving complex problems of optimization - High speed convergence	- Complex implementation in hardware - Risk of falling in a local MPP

Table 2. Summary of advantages and disadvantages of some MPPT techniques. **[9]**

I.4 DC-DC converter :

A DC-DC converter, also called a DC-DC power converter or voltage regulator, is an electrical system (device) which converts direct current (DC) sources from one voltage level to another. As the name implies, a DC-DC converter only works with direct current (DC) sources and not with alternative current (AC) sources.

There are several types of DC-DC converters, each with unique circuit topologies and characteristics, and the output DC voltage can be higher or lower than the DC input voltage depending on the type used. The types generally used for MPPT at the outputs of PV systems are “Buck”, “Boost” and “Buck- Boost” configurations.

I.4.1 Step-up (Boost) converter :

Figure 13 depicts a step-up or a boost converter. It is comprised of d_c input voltage source V_S ,boost inductor L, controlled switch S, diode D, filter capacitor C, and load resistance R. **[10]**

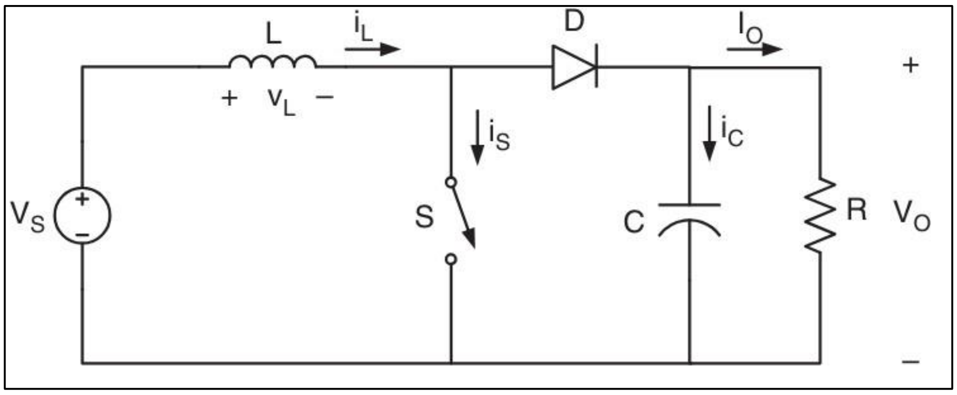


Figure I. 13: circuit diagram of Boost converter [10]

The converter waveforms in the CCM are presented in figure14. When the switch S is in the “on” state, the current in the boost inductor increases linearly. The diode D is “off” at the time. When the switch S is turned “off”, the energy stored in the inductor is released through the diode to the input RC circuit.

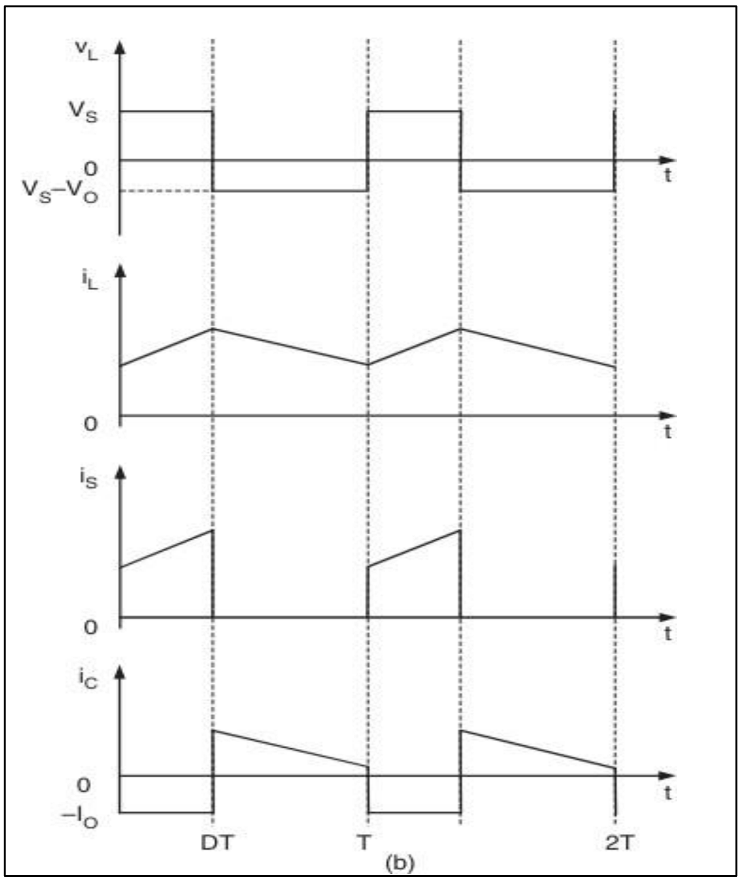


Figure I. 14 : waveforms of Boost converter. [10]

Using the Faraday's law for the boost inductor :

$$V_S DT = (VO - VS)(1 - D)T \tag{14}$$

from which the DC voltage transfer function turns out to be :

$$M_V \equiv \frac{V_s}{V_o} = \frac{1}{1-D} \tag{15}$$

As the name of the converter suggests, the output voltage is always greater than the input voltage. The boost converter operates in the CCM for L > L_b where :

$$L_b = \frac{(1-D)^2 DR}{2f} \tag{16}$$

As shown in figure 14, the current supplied to the output RC circuit is discontinuous. Thus, a larger filter capacitor is required in comparison to that in the buck-derived converters to limit the output voltage ripple. The filter capacitor must provide the output DC current to the load when the diode D is off. The minimum value of the filter capacitance that results in the voltage ripple Vr is given by. **[10]**

$$C_{min} = \frac{DV_O}{V_r R f} \tag{17}$$

I.4.2 DC-DC converter topology and MPPT working zone :

The MPPT operating zone for solar PV is dependent on DC– DC converter topology and restricts the value of resistive load for which MPPT become effective. The MPPT scheme is implemented in the control of DC–DC converter (it varies the duty cycle). The basic principle of adjusting the duty cycle is to match load impedance with input impedance seen by the DC– DC converter, as shown in Figure 15. **[11]**

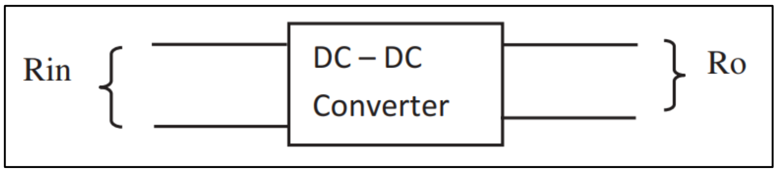


Figure I. 15 : Block diagram of DC–DC converter

Rin (the input impedance seen by the converter) and Ro (the output impedance connected with converter) are related with characteristic equation. This mathematical equation varies with DC–DC converter topologies.

I.5 Conclusion

This chapter shows the different part of which the PV system is consisted. First the source of the energy of the system that comes from the panels , then the software part where there is many methods and algorithms that can maintain the maximum power that can be provided by the panel, and finely the DC/DC converter who will adapt the its input from the panel to the its output (load) with the help of mppt controller.

Chapter 02
FPGA and VHDL

II.1. Introduction :

FPGA (Field Programmable Gate Arrays) is one of the most powerful PLDs (Programmable logic devices) available today, with technology that enables users to build systems to meet design requirements.

FPGAs play a crucial role in automation systems, revolutionizing the way tasks are performed and controlled across various industries. With their unique ability to be reconfigured and programmed to perform specific functions, FPGAs offer unparalleled flexibility and adaptability in automation applications. They are also widely employed in the creation of electronic prototypes and systems. Engineers can quickly test and iterate ideas thanks to their re-programmability without needing to create new chips for each iteration.

FPGA programming or configuration is typically done using a Hardware Description Language (HDL) such as Verilog or VHDL. The HDL code describes the desired behavior and functionality of the circuit, and it is synthesized and mapped onto the FPGA using specialized software tools provided by the FPGA manufacturer.

II.2. Field Programmable Gate Arrays

II.2.1. Programmable Logic Device

Programmable logic devices (PLDs) encompass all circuit components that enable engineers to simulate, design, and process real-world information in digital format [12]

These integrated circuits are used to create different types of digital circuits. In general, these devices allow for the programming of a variety of logic components, ranging from the simplest elements like logic gates to more complex structures such as combinational and sequential circuits. In other words, these circuits provide the necessary flexibility to implement a wide range of logical functionalities in electronic systems [13]

PLDs are commonly classified into two types: Simple Programmable Logic Devices (SPLDs) and Complex Programmable Logic Devices (HCPLDs). The figure 1 shows the PLDs classification [12]

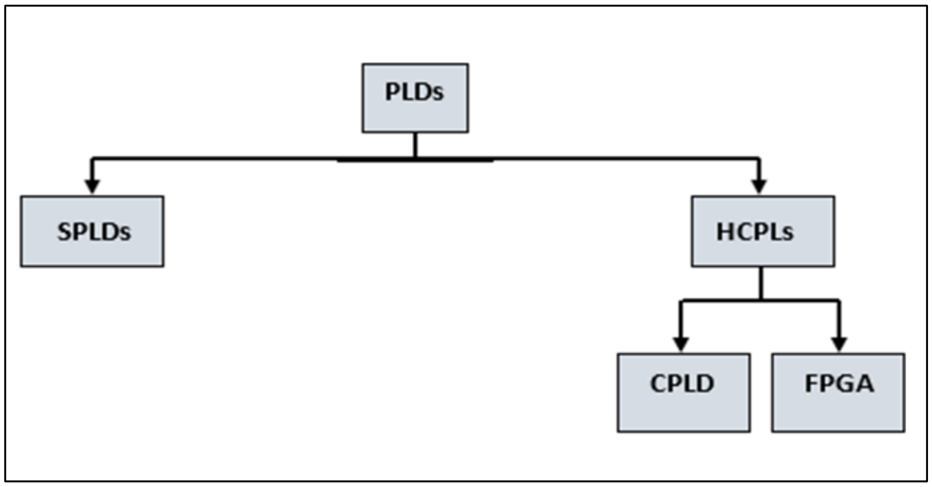


Figure II. 1: PLDs classification.

II.2.2. FPGA card

A Field Programmable Gate Array (FPGA) is a type of semiconductor device that can be programmed and reprogrammed after manufacturing. It is a configurable integrated circuit that can be used to implement a wide range of digital functions, from simple logic gates to complex digital signal processing algorithms[14] [15]

The first major FPGA manufacturer are Ross Freeman and Bernard Vonderschmitt, founders of Xilinx, in the early 1980s. They proposed a new type of programmable logic device (PLD) that users could program to execute diverse digital logic functions. In 1984, Xilinx released the XC2064, the world’s first field-programmable gate array. It was based on the concept of a sea of programmable gates and programmable interconnects, providing greater flexibility compared to traditional PLDs [16]

Figure II. 2: The first FPGA card (XC2064).

II.2.3. FPGA internal architecture:

The architecture of an FPGA consists essentially of three elements: configurable logic blocks, input/output blocks and interconnection resources. The figure 3 shows the FPGA internal architecture [17]

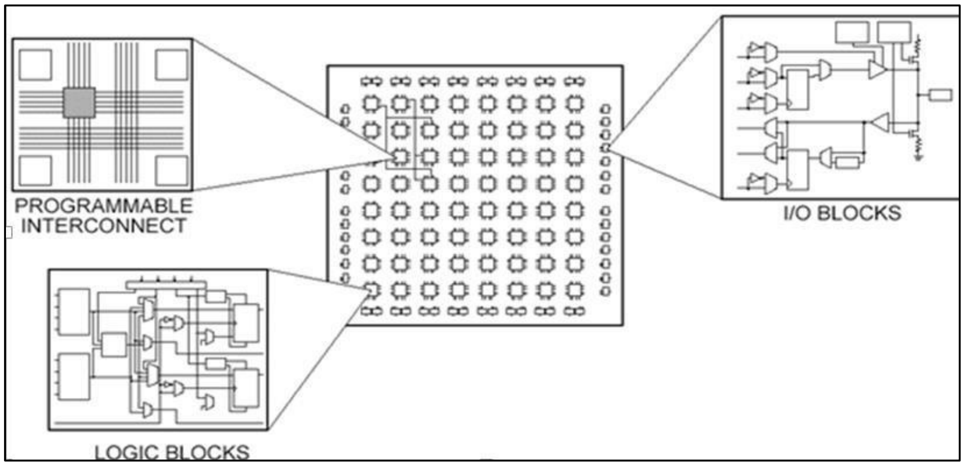


Figure II. 3 : FPGA internal architecture

II.2.3.1. Configurable logic blocks

Every FPGA is equipped with configurable logic blocks, serving as the fundamental building blocks for implementing both combinational and sequential circuits. They consist of a configurable array of logic elements, such as Look-Up Tables (LUTs), multiplexers, flip-flops, and other components.Various types of CLBs exist, with their specifications depending on the type of FPGA used by the manufactures [14]. Figure 4 chows the CLBS architecture [18]

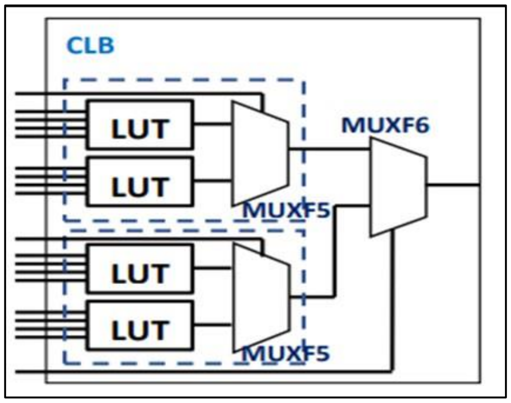


Figure II. 4 : CLBs architecture

II.2.3.2. Input/output blocks (IOBs)

These adaptable blocks facilitate the connection between the user-developed internal logic and external modules .I/O blocks can be configured as inputs, outputs or bi-directional signals and have their own configuration memory. [14]

The input/output blocks integrate a combination of components including a 3-state gate and multiple D flip-flops, serving purposes of output and input synchronization. These blocks offer additional capabilities such as the utilization of reset resistors on input/output pins, enabling voltage supply (via pull-up resistors) or grounding (via pull-down resistors). Furthermore, they provide the flexibility to program the rise time of output signals, enhancing the adaptability and performance of the digital circuit. [18]

II.2.3.3. Interconnections

The role of interconnects is to link logic blocks with input/output blocks in order to implement any user-defined circuit[19]**.**The interconnection resources encompass a programmable network that occupies roughly 90% of an FPGA component's surface area; consequently, the flexibility of the FPGA primarily hinges on this programmable network . The interconnection network consists of horizontal and vertical routing tracks interconnected through switching blocks. [20]

II.2.4. The Leaders in the FPGA Market

The leaders in the FPGA market, such as Xilinx (now part of AMD), Intel (including Altera), and Microchip Technology, Lattice Semiconductor Corporation are key players in the industry according to 2023 statistic, and have continued to innovate and maintain their positions as industry leaders. [21]

II.2.4.1. XILINX

Historically one of the biggest players in the FPGA market, Xilinx has been known for its high-performance FPGAs targeting various applications including data centers, networking, telecommunications and automotive. In 2020, AMD (Advanced Micro Devices) announced its acquisition of Xilinx, further strengthening its position in the semiconductor industry. [21]

II.2.4.2. Intel

Intel's FPGAs find applications in data centers, networking infrastructure, automotive, and industrial sectors. Intel entered the FPGA market in the late 1980s, offering its first programmable logic devices. This marked the beginning of competition in the FPGA industry, further driving advancements and innovation. Intel acquired Altera in 2015, integrating its FPGA capabilities with Intel's technologies .Altera was historically known for its FPGA families “Stratix” and “Cyclone” [21]

II.2.4.3. Microchip Technology

Microchip Technology, formerly known as Microsemi Corporation, is a leading provider of semiconductor solutions. Historically, Microchip has been known for its FPGA families the Smart Fusion and IGLOO series. [21]

II.2.4.4. Lattice Semiconductor Corporation

Lattice Semiconductor offers several FPGA families, each designed to cater to specific applications and markets. The iCE40 HX Series is a family of low-cost FPGAs that feature low-power consumption, high-speed interfaces, and integrated Phase-Locked Loop (PLL). These FPGAs are ideal for applications such as wearable technology, industrial control, and consumer electronics. [21]

II.2.5. Nexys 4dd card

The Nexys 4 DDR board is a complete, ready-to-use digital circuit development platform based on the latest Artix-7™ Field Programmable Gate Array (FPGA) from Xilinx®. Generous external memories, and collection of USB, Ethernet, and other ports, Several built-in peripherals, including an accelerometer, temperature sensor, MEMs digital microphone, a speaker amplifier, and several I/O devices allow the Nexys4 DDR to be used for a wide range of designs without needing any other components. [22]

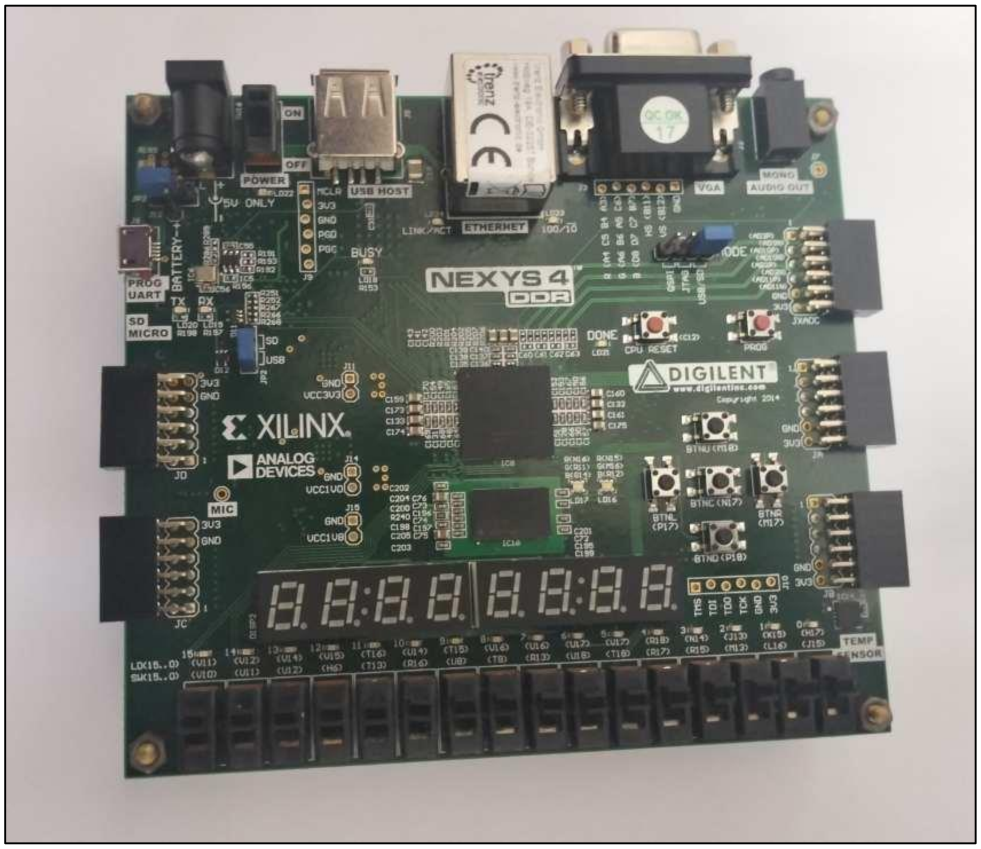


Figure II. 5 : Nexys 4 ddr card [22]

II.3. FPGA Hardware Description Languages and Design Tools

Hardware Description Language. It's a specialized programming language used to describe the structure of digital electronic circuits and systems. FPGA manufacturers have launched several tools thus reducing FPGA design time and making implementations easier.

II.3.1. FPGA Design Tools

The programming tools for FPGAs vary depending on the manufacturer and the type of FPGA. here is a general list of some of these commonly used tools

II.3.1.1. Xilinx ISE

Xilinx ISE (Integrated Software Environment) is a software suite used for designing, testing, and implementing FPGA (Field-Programmable Gate Array) designs. It provides tools for HDL (Hardware Description Language) synthesis, simulation, and implementation for Xilinx FPGAs. ISE includes tools for timing analysis, debugging, and programming FPGAs [23].

II.3.1.2. Vivado software

Vivado, the robust design software engineered by Xilinx for FPGAs, combines design entry, synthesis, place and route, and verification/simulation tools. Its advanced capabilities assist hardware designers in optimizing compiler times. [24]

The Vivado software platform includes the following tools [24].

- **Vivado Design Suite**: The main graphical user interface (GUI) for designing FPGAs, providing access to all other tools and features.
- **Vivado HLS (High-Level Synthesis)**: Allows designers to describe hardware functionality using C, C++, or SystemC languages, converting high-level code into RTL (Register Transfer Level) code.
- **IP Integrator (IPI)**: A tool for creating complex FPGA designs by assembling pre-designed IP (Intellectual Property) cores and custom logic blocks using a graphical interface.
- **Synthesis**: Transforms RTL code written in languages like Verilog or VHDL into a gate-level netlist, optimizing the design for area, timing, and power.
- **Implementation**: Places and routes the synthesized design onto the target FPGA device, considering factors like timing constraints, resource utilization, and routing congestion.
- **Timing Analysis**: Analyzes the timing performance of the design to ensure that all timing requirements are fulfilled, identifying critical paths and potential timing violations.
- **Simulation**: Allows designers to simulate the behavior of the FPGA design before synthesis and implementation, using tools like Vivado Simulator or third party simulators.
- **Debugging and Verification**: Provides features for debugging and verifying the FPGA design, including waveform viewers, RTL-level debugging, and integrated logic analyzers.

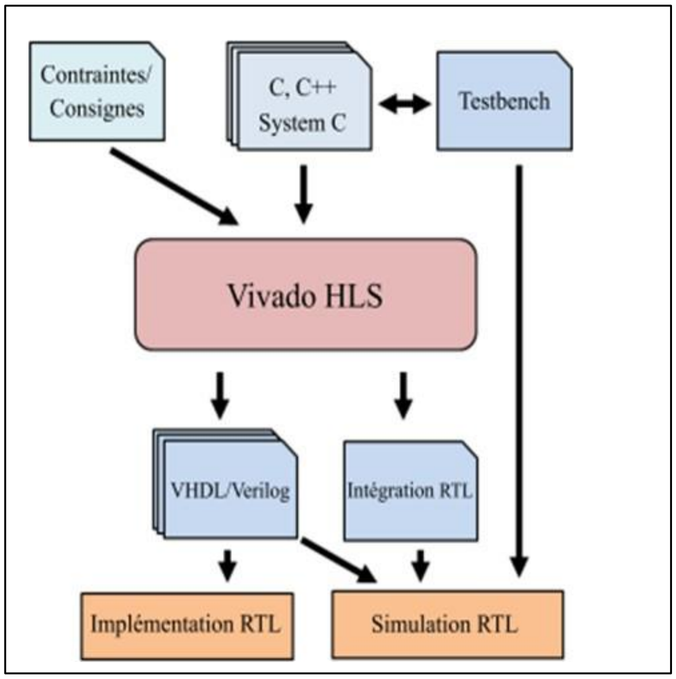


Figure II. 6 : Designing with Vivado HLS

II.3.1.3. Vitis software

The AMD Vitis software platform is a development environment for developing designs that includes FPGA fabric, Arm processor subsystems, and AI Engines. The Vitis tools work in conjunction with AMD Vivado design Suite to provide a higher level of abstraction for design development. [25]

The Vitis software platform includes the following tools [25]:

- **Vitis Embedded** :For developing C/C++ application code running on embedded Arm processors
- **Compiler and simulators** :For implementing designs using the AI Engine array
- **Vitis HLS** (High-Level Synthesis)**:**: For developing C/C++ based IP blocks that target FPGA fabric
- **Vitis Model Composer**: A model-based design tool that enables rapid design exploration

II.3.1.4. Quartus Prime design

Quartus Prime Design is an integrated development environment (IDE) by Intel for FPGA design. It offers tools for designing, compiling, and programming FPGA devices. It supports various FPGA families, uses HDLs like Verilog or VHDL, and provides features for synthesis, simulation, place-and-route, timing analysis, and programming. It is user-friendly with a graphical interface and comprehensive documentation.

II.3.1.5. HDL Coder in the MATLAB environment

HDL Coder is a tool developed by MathWorks that allows the generation of HDL (VHDL/Verilog) code from Simulink models and MATLAB functions [26]. The HDL Coder tool also allows for the verification of the generated HDL code and testing it against the original MATLAB/Simulink model.

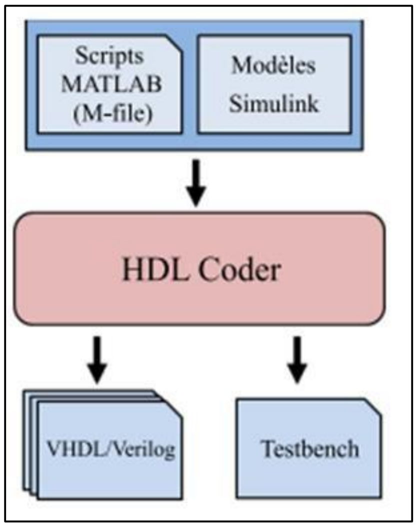


Figure II. 7 : Designing with HDL Coder [26].

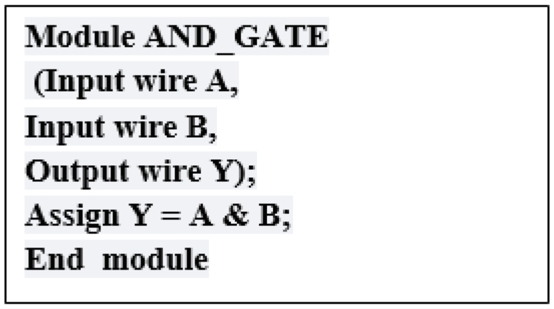
II.3.2. Hardware Description Language

A digital system design can be captured either as a schematic or by employing specialized Hardware Description Languages (HDL), such as VHDL (VHSIC Hardware Description Language),Verilog, ABEL (Advanced Boolean Expression Language), AHDL (Altera HDL), SystemC, Confluence, CUPL(Computer Aided Logic Design Programming),VHDL-AMS(VHDL-Analog Mixed Signal).These languages allow engineers to describe the behavior and structure of their digital circuits and systems, enabling synthesis into FPGA configurations using tools like Xilinx Vivado or ISE Design Suite. With these HDLs, developers can efficiently design and implement complex digital systems on Xilinx FPGAs .The most used are VHDL and Verilog.

II.3.2.1. Verilog

Verilog HDL is one of the two most common Hardware Description Languages (HDL) used to design electronic systems the other one is VHDL. The Verilog code is compiled and synthesized using these tools Xilinx Vivado, Xilinx ISE.

The Verilog code example:

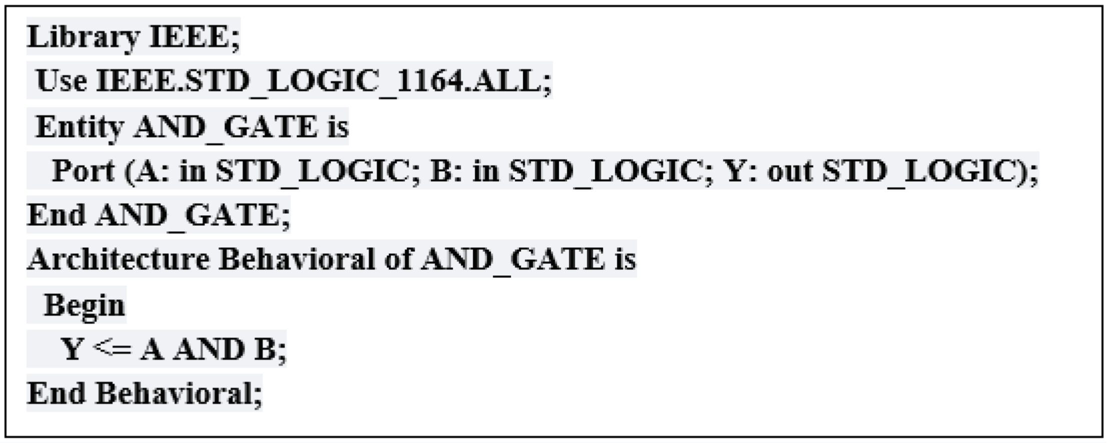


```
Module AND_GATE
 (Input wire A,
Input wire B,
Output wire Y);
Assign Y = A & B;
End  module
```

II.3.2.2. VHDL

VHDL (Very High Speed Integrated Circuit) Hardware Description Language is one of the two most common Hardware Description Languages (HDL) used to design electronic systems the other one is Verilog. The VHDL code is compiled and synthesized using these tools Xilinx Vivado, Xilinx ISE.

The VHDL code example:



```
Library IEEE;
 Use IEEE.STD_LOGIC_1164.ALL;
 Entity AND_GATE is
   Port (A: in STD_LOGIC; B: in STD_LOGIC; Y: out STD_LOGIC);
End AND_GATE;
Architecture Behavioral of AND_GATE is
  Begin
    Y <= A AND B;
End Behavioral;
```

II.4. About VHSIC Hardware description language

VHDL stands for VHSIC (Very High Speed Integrated Circuit) Hardware Description Language. It emerged from a project sponsored by the US Department of Defense in the 1980s VHDL was the first hardware description language to be standardized by the IEEE, achieved through standards 1076 and 1164. [27]

II.4.1. Structure of a VHDL program

To describe a circuit in VHDL, it is necessary to specify three fundamental parts: the libraries, the entity and the architecture.

The figure 8 shows the fundamental section of a basic VHDL code

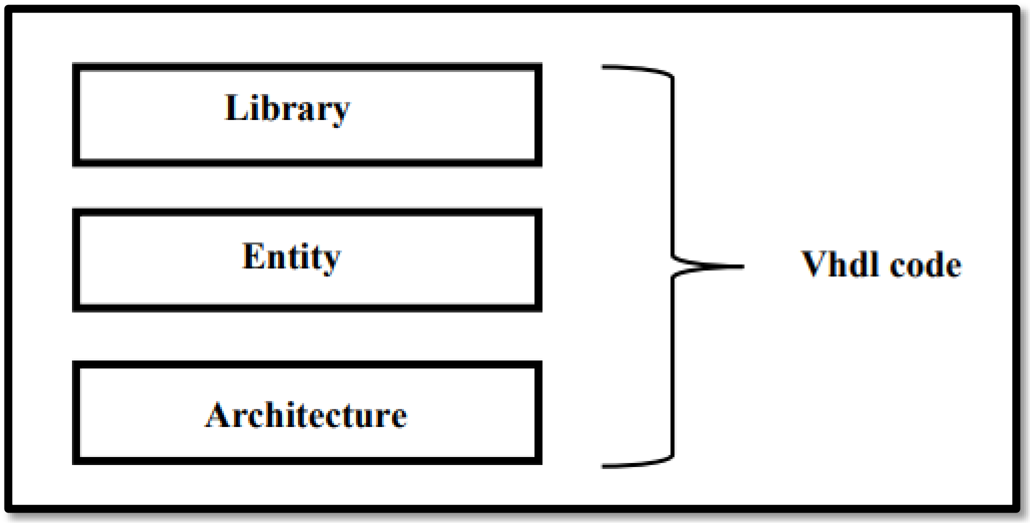


Figure II. 8 : Fundamental section of a basic VHDL code [27].

Library declaration: contains a list of all libraries to be used in the design.

For example: IEEE, STD, work.

ENTITY: specifies the I/O pins of the circuit.

Architecture: contains the VHDL code proper, which describes how the circuit should behave (function).

II.4.1.1. Library declaration:

The IEEE (Institute of Electrical and Electronics Engineers) has standardized libraries, in particular the IEEE1164 library. These libraries contain definitions of electronic signal types, functions and subroutines for arithmetic and logic operations, and the ability to store compilation results. The **use** directive is used to select the libraries to use. [27]

Example:

```
Library IEEE;
 Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
```

The IEEE library contains several packages, including the following:

Std_logic_1164: specifies the STD_LOGIC and STD_ULOGIC multi-valued logic systems. **Std_logic_arith :** specifies the SIGNED and UNSIGNED data types and related arithmetic and comparison operations it also contains several data conversion functions which allow one type to be converted into another conv_integer(p) ,conv_unsigned(p, b), conv_signed(p, b), conv _std _logic _vector(p, b).

STD_ logic_ signed: contains functions that allow operations with Std_ logic _vector data to be performed as if the data were of type signed.

STD _logic _unsigned: contains functions that allow operations with STD _logic_ vector data to be performed as if the data were of type unsigned.

II.4.1.2. Entity declaration:

An entity declaration describes a module (a digital circuit) from an external point of view, Giving its name, inputs/outputs and any parameters.

Inputs and outputs are called ports, and parameters are called generics. An entity declaration always has the following structure:

```
Entity entity_name is
  Generic (...);
  Port (...);
End entity_name;
```

The ports mode can be ‘IN’, ‘OUT’, ‘INOUT’ or ‘BUFFER’; we choose the mode of the port according to our needs:

- IN: the entity reads an external signal (unidirectional).
- OUT: the port is an output unidirectional.
- INOUT: the port is an input and output (bidirectional).
- BUFFER: we use it when the output signal should be read in the internal architecture.

The port type is scalar or vector:

- Scalar: BIT, STD_LOGIC, BOOLEAN and NTEGER.
- Vector: BIT_ VECTOR, STD_ LOGIC _VECTOR, SIGNED, UNSIGNED.

II.4.1.3. Architecture declaration:

The architecture syntax :

```
Architecture archihtecture_name of entity is
    (Declaration)
BEGIN
    (Code)
End archihtecture_name;
```

The architecture is divided into two parts

a) Declarative part

The declarative part reserve to declare the objects (signal, function, constant, and component….) between a header and the word BEGING.

b) Code part

In this part, we declare all the descriptive instructions of the circuit.

II.4.2. Concurrent and sequential operations

II.4.2.1. Concurrent operation

In a logic circuit, the gates operate simultaneously, so that all operations are performed simultaneously and the order in which they are written is irrelevant.

The VHDL concurrent instructions:

- **Simple affection**: In a VHDL description, this is the most commonly used operator.
  Example: A <= "0000";

- **Conditional assignment when ... else:**
  We note the absence of punctuation at the end of intermediate lines.

```
Signal_S <= Signal_A when Condition_A
Else Signal_ B when Condition_B
Else Signal_C when Condition_C
Else Signal_F;
```

- **Selective assignment with ... select:**

Note the presence of a comma at the end of intermediate lines.

```
With Vecteur_De_Commande select
Signal_Sortie <=Valeur_1 when Etat_Logique_1,
                Valeur_2  when  Etat_Logique_2,
                 Valeur_N when others;
```

- **The structural description is based on:**

Declaring components in the declarative part of the architecture

```
----- Component declaration:
Component_Name is
 Port (A: in std_logic; B: in std_logic;
       X: out std_logic_vector (1 downto 0);
End component;
```

The declaration of component interconnection in the code of the architecture.

```
Etiquette : Component_Name
PORT MAP (association_lists) ;
```

For example in Figure 9: Instead of writing a program for each D flip-flip , a single implementation can be created and used in various component.

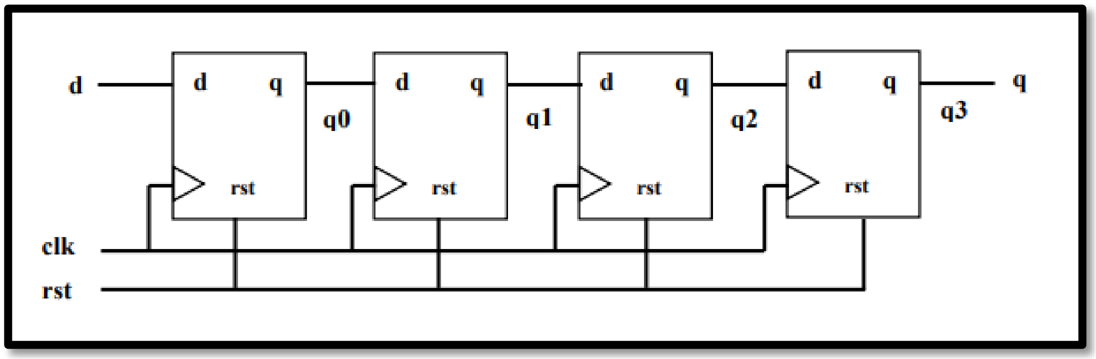


Figure II. 9 : shift register

- **Conditional generation :**

```
------ Structure conditionnelle:
 If condition generate
    concurrent_instructions;
End generate;
```

II.4.2.2. Sequential operation:

Sequential instructions appear only within processes or sub-programs (function, procedure). Within these descriptions, instructions are executed sequentially, one after the other. The order of declarations is important.

- **Process:**

A process is a part of a circuit description in which instructions are executed sequentially are executed sequentially, i.e. one after the other.

```
process (Sensitivity_List)
--zone de Déclarations
Begin
--Zone pour instructions Séquentielles
End process ;
```

- **Instruction case...is...when:**

```
Case number is
  When Valeur_1 => --Zone pour instructions séquentielles
  When Valeur_2 => --Zone pour instructions séquentielles
  When others => --Zone pour instructions séquentielles
End case ;
```

- **Loop instructions :**

The generic syntaxes for loop instructions are as follows:

```
-- For loop
 For variable in min_value to max_value loop
----instructions;
End loop;
```

```
--- while loop
 While conditions loop
----instructions;
End loop [label];
End case;
```

```
---general loop
Loop -- infinite loop
 ----instructions;
End loop;
End case;
```

II.5. Conclusion

FPGA circuits are programmed and configured using tools based on hardware description languages such as VHDL. In this chapter, we presented the internal architectures of FPGAs with market leaders such as AMD Xilinx, and then describe the Nexys 4 DDR Xilinx board used in this work. We also discuss FPGA tools and explain in detail the structure of VHDL language code instructions used in concurrent and sequential operating.

Chapter 03
Implementation of PV system with VHDL

III.1 Introduction

In the previous chapters, we learned about the photovoltaic system, including the different methods of MPPT, and the boost converter. Also about programming the FPGA board with VHDL.

In this chapter, we will present our work, which is divided into two part. The first one is implementing the Photovoltaic system with the help of the FPGA board, and the method we will be using in the program is the Perturb and Observe (P&O) Algorithm, the second one is to build a boost converter that will be commanded by the VHDL program.

III.2 Hardware

We will describe in this part all the materials used in the hardware side with their results which consists of the solar panel, the sensors used to read the current and the voltage generated from the panel and finely the boost converter.

III.2.1 Solar panel

In our work, we used the solar panel STP085-12/Bb with a maximum power rating of 80 W, which has the following parameters:

- Current at maximum power : 4.65A
- Voltage at maximum power : 17.2V
- Short-Circuit Current : 5.1A
- Open-Circuit Voltage : 21.6V
- Nominal Operating Cell Temp: 50±2°C
- Dimensions: 1195×541×30(mm)
- Maximum System Voltage: 715V
- Maximum Series Fuse Rating: 8A

We conducted an experiment to see the I-V curve and P-V curve using a variable resistor, voltmeter and an ammeter.

In this experiment we tried to simulate the effect of shading by covering different parts of the panel, and also simulate other naturel factors like rain, this time by covering it with different things (water, tree leaves, sand..).

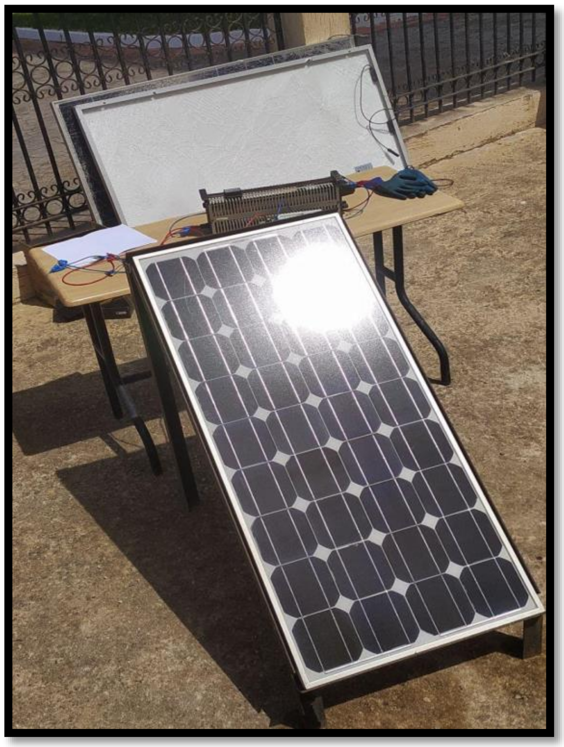


Figure III.1(a): The solar panel is shaded 100%

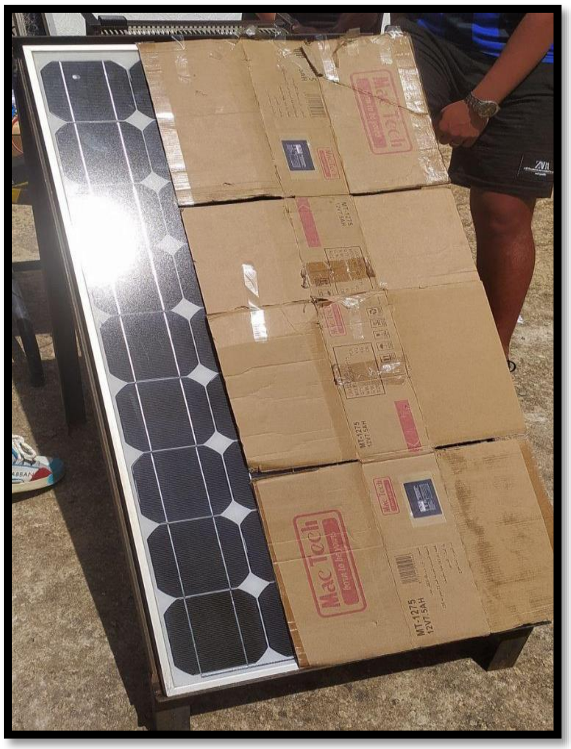


Figure III. 1(b): The solar panel is shaded 68%

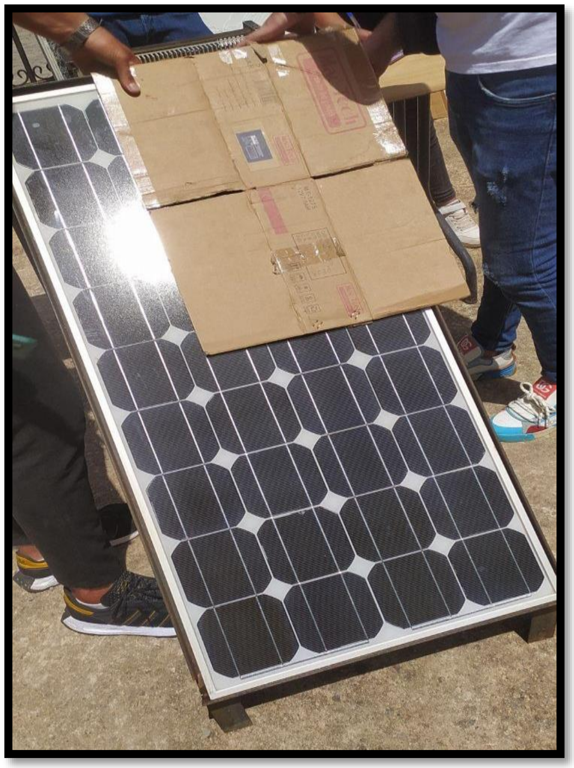


Figure III. 1(c): The solar panel is shaded 34%

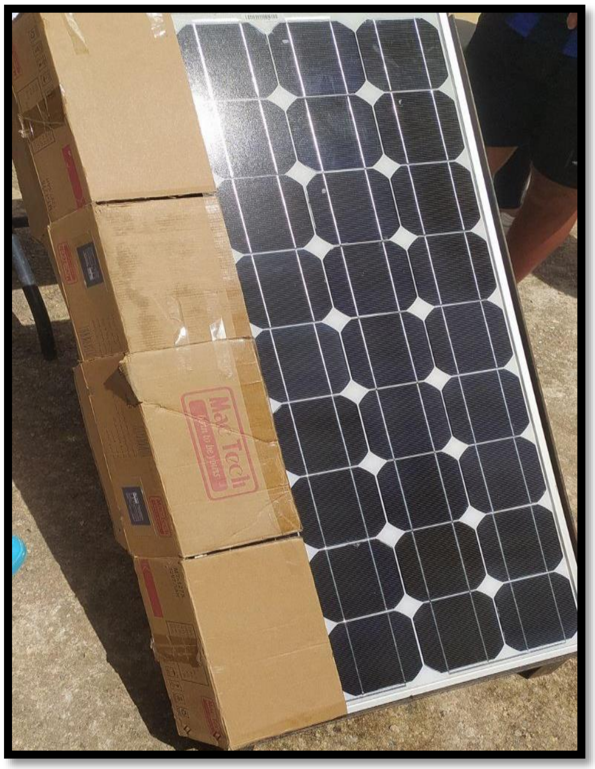


Figure III.1(d): The solar panel is shaded 25%

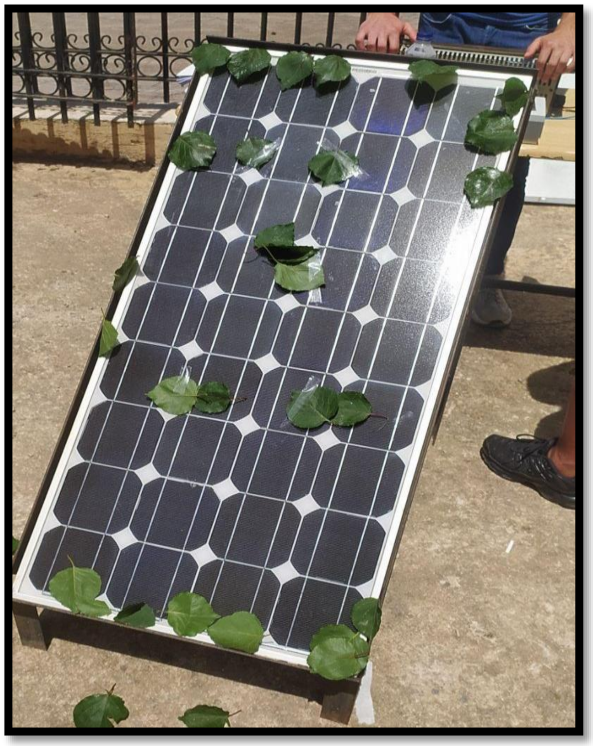


Figure III.1(e): The solar panel is covered with tree leaves

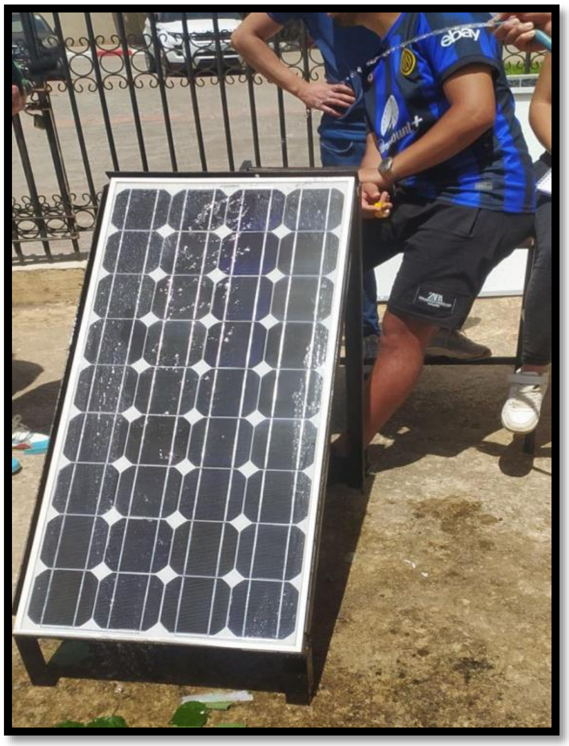


Figure III.1(f): The solar panel is covered with water

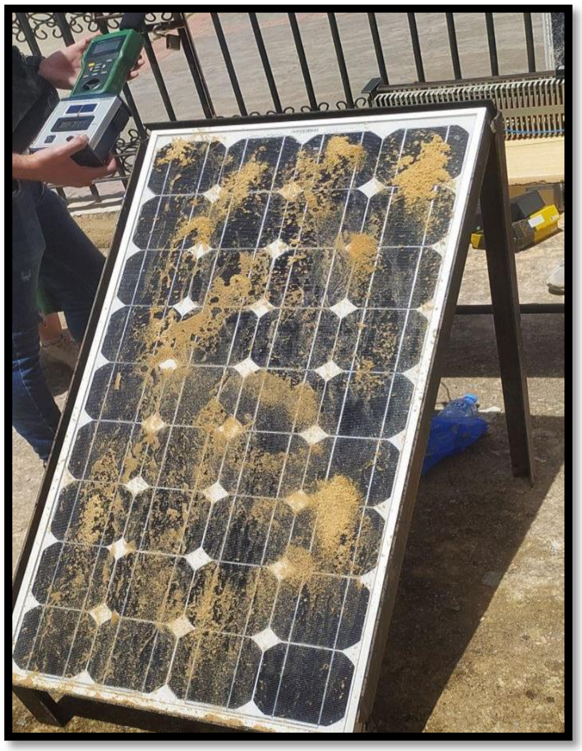


Figure III.1(g): The solar panel is covered with more sand

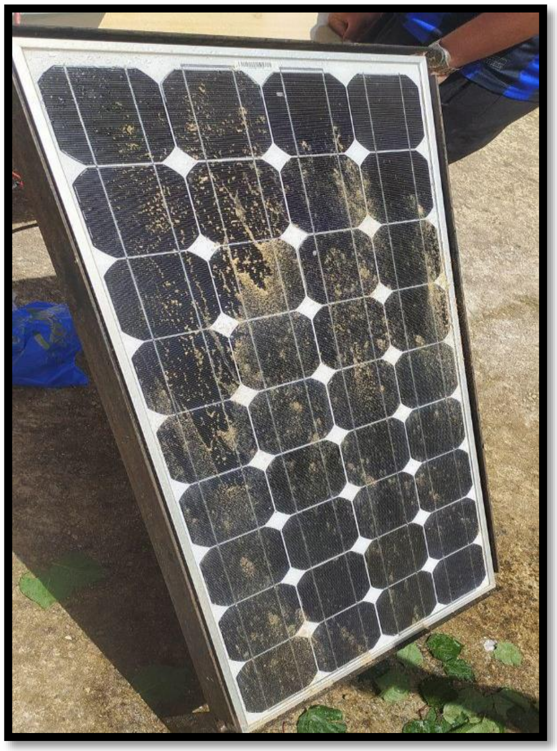


Figure III.1(h): The solar panel is covered with sand

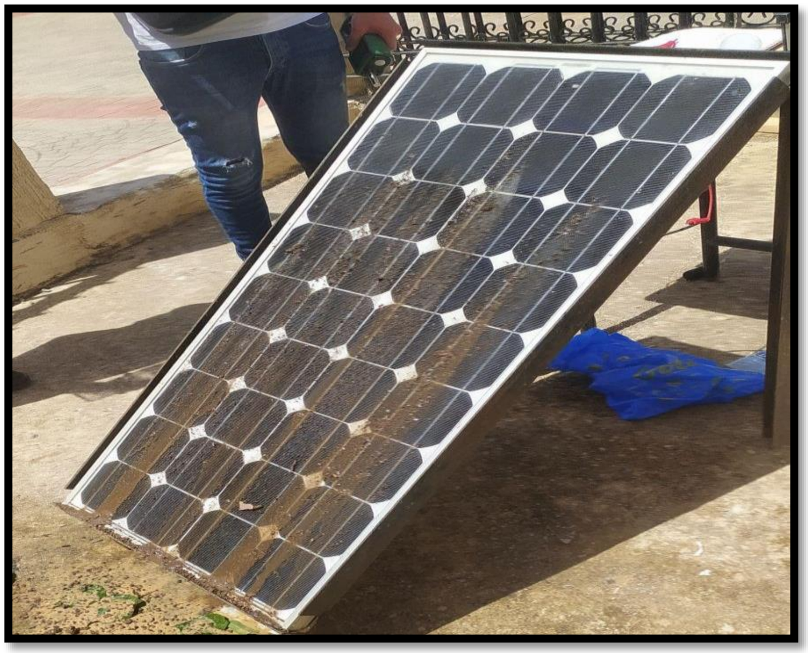


Figure III.1(i): The solar panel is covered with earth

Figure III.1: images of the panel in different situation

We can see from the results of this experiment in figure III.2 that the graphics are logical and similar to the graphic from the theory.

The result shows that when we cover different part of the panel completely, the power is much lower. That is because all the cells are connected in series, and the cells that are covered will not generate any current.

The curves that represent the experiment of the panel with water shows that the efficiently is better than the panel without water. And we conclude from this that the water keep the surface of the panel cooler (low temperature).

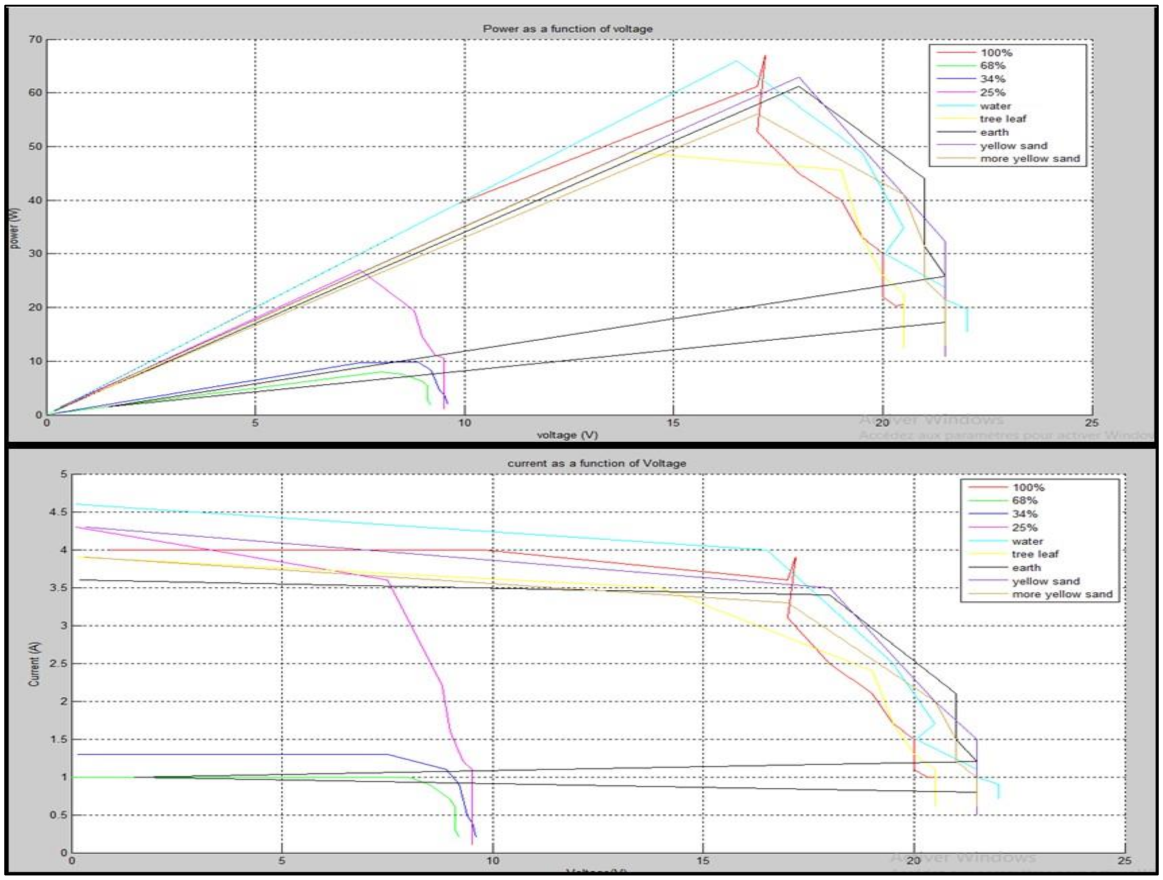


Figure III. 2: The I_V curve and P_V curve of the panel under varying conditions

III.2.2 Current sensor

The current sensor used is **ACS712**. It is an Arduino-compatible sensor designed to measure electrical current in a circuit.

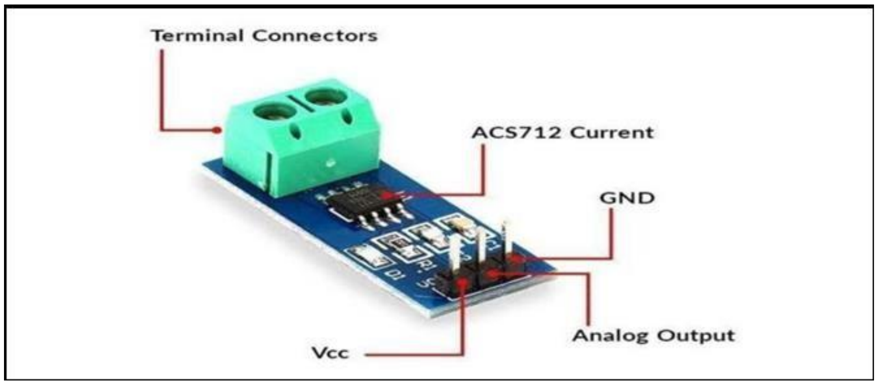


Figure III. 3: Current sensor

This current sensor has a supply voltage of 5volts (Vcc), and its analog output can reach 5 volts as well.

To use it with an FPGA board with Xilinx analog digital converter (Xadc) which can only receive voltage between 0 to 1 volt , a voltage divider must be employed to reduce the voltage range .

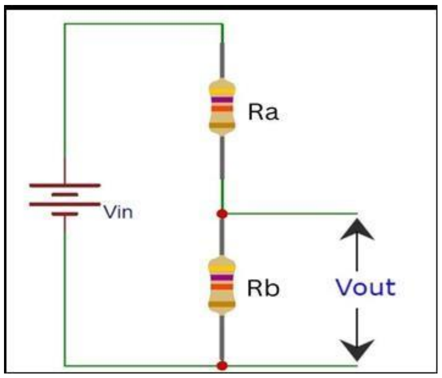


Figure III. 4: voltage divider

The equation of voltage divider is :

$$\mathbf{Vout} = \frac{Rb}{Rb+Ra}\ \mathbf{Vin}$$

With : Vin : input voltage ; Vout : output voltage ; Rb,Ra : resistors

To go from a rang of (5v – 0v) to (1v – 0v) we need to choose Rb and Ra to have

$$\frac{Rb}{Rb+Ra} = \frac{1}{5}$$

III.2.3 Voltage sensor

To capture the voltage, we used a voltage divider setup to adapt the solar panel's voltage range from 0 to 17 volts to a range of 0 to 1 volt, making it compatible with an Fpga board equipped with XADC.

The voltage divider we will use needs to have a ratio of $\frac{1}{17}$ .That means we choose the resistors Ra and Rb so that $\frac{Rb}{Rb+Ra} = \frac{1}{17}$.

III.2.4 Boost converter

The materials we used to build the boost in figure III.5 :

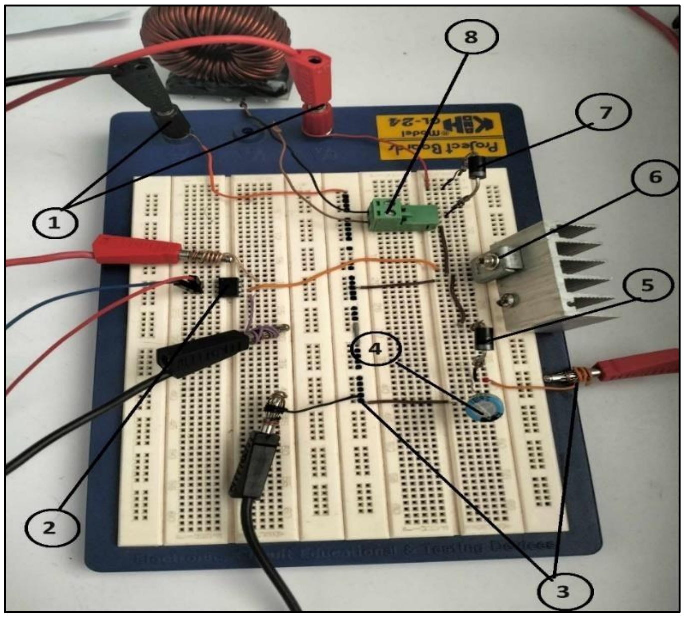


Figure III. 5 : Boost converter

1. **Input of the boost**
2. **Optocopler (HCPL_3120)**
3. **Load**
4. **Capacitor**
5. **Diode (1N5408)**
6. **Transistor (mosfet) IRFZ44N**
7. **Diode (1N5408)**
8. **Inductor**

III.2.4.1 Boost implementation

In this implementation we took the duty cycle of the mosfet D = 0,5 with a frequency of 10Khz from the Low-frequency generator. As for the voltage, we took from the generator (Figure.III.6) V0 = 17v for the input of the boost and Vcc = 15v for optocolper.

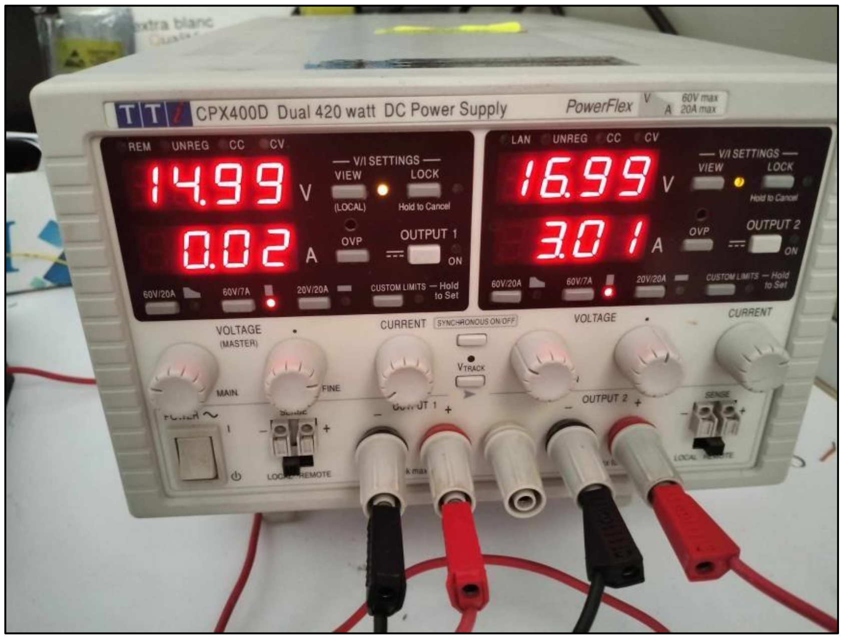


Figure III. 6 : the Generator used in the experiment

The boost simulation in proteus :

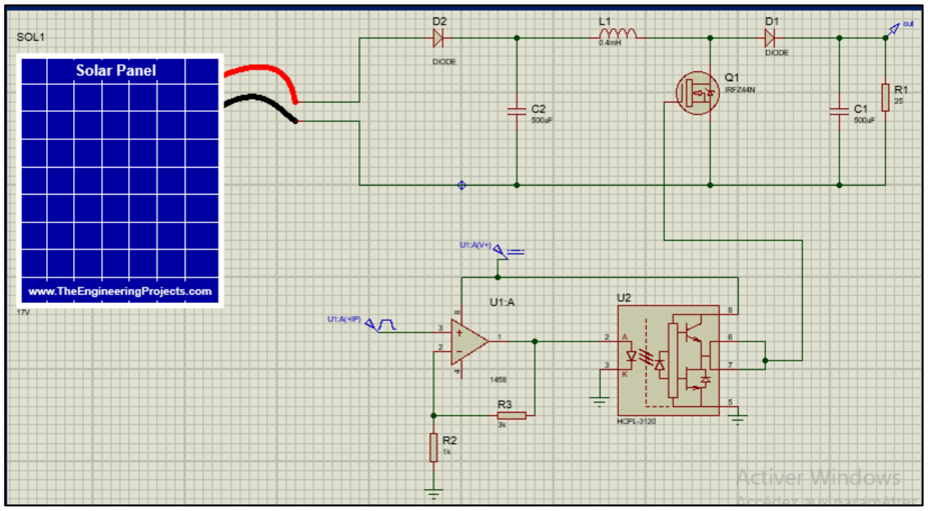


Figure III. 7: Boost diagram in proteus

III.2.4.2 Results

From the function (15) of DC voltage transfer function in chapter 01, we have:

$$V_S = \frac{1}{1-D} V_O$$

$$V_S = 2V_O = 2 * 17 = 34 \text{ V}$$

Figure III.8 shows the graphic of the result from the proteus simulation of the our boost, and we can see that the load voltage stabilizes at Vs ≃ 34V :

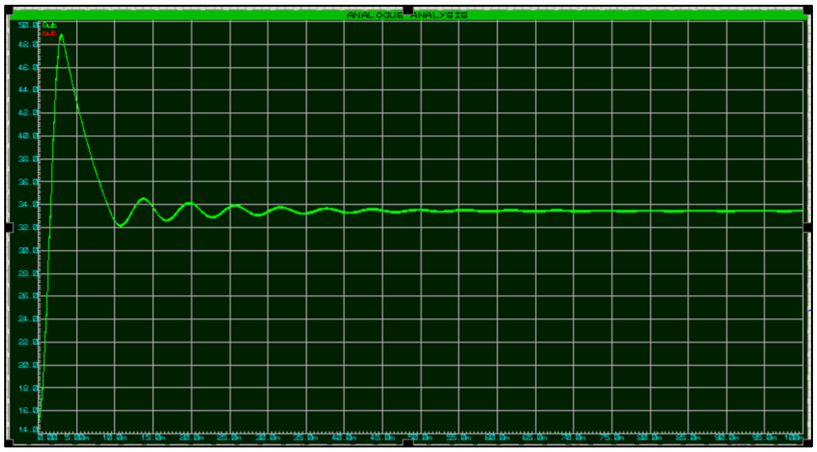


Figure III. 8 : Result of proteus simulation

The result we obtained from the implementation shows that the voltage measured between the terminals of the load (Vs) are: Vs =31 V (figure III. 9).

This result was what we were expecting to get because they are close to the theoretical results, and also the result of the simulation with proteus.

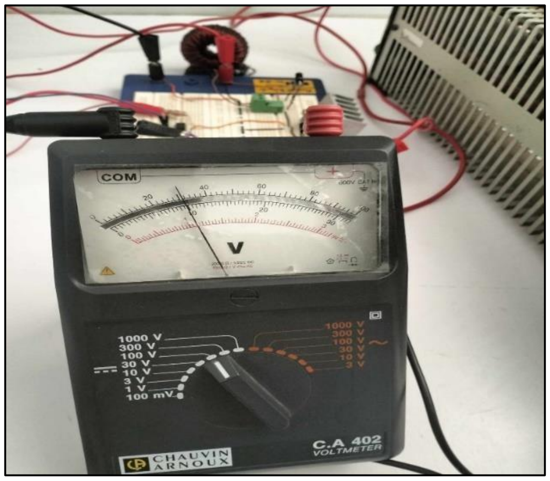


Figure III. 9: Result of the boost implementation

III.3 Software

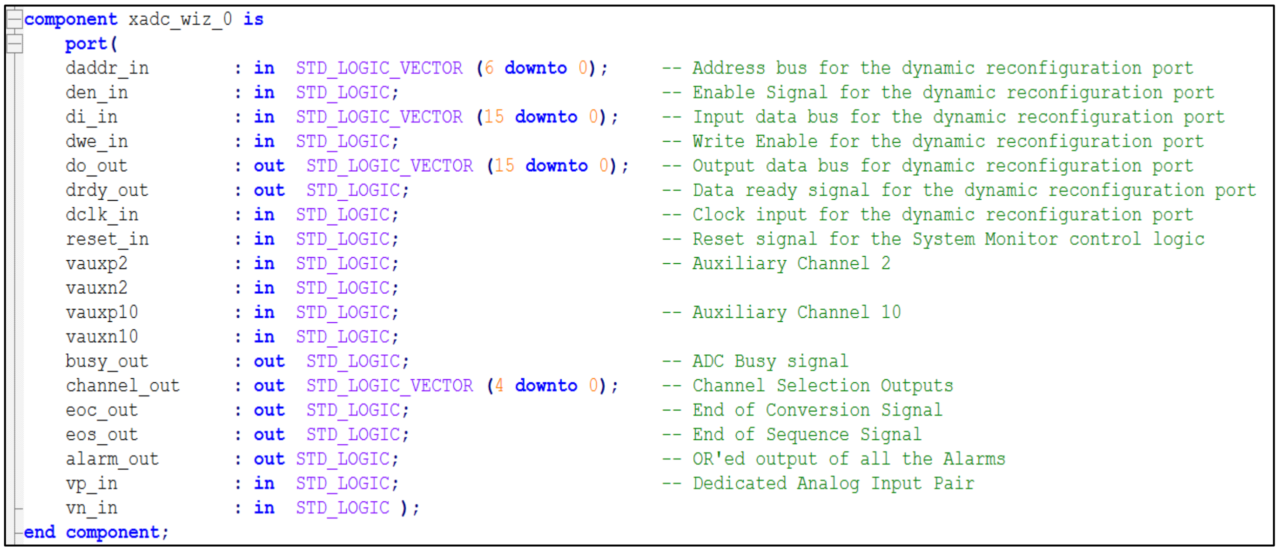
III.3.1 VHDL program

III.3.1.1 XADC code

XADC (Xilinx analog digital converter) is a specialized component who will be responsible of receiving the analog signals from current and voltage sensors and convert them into digital values that can be processed by FPGA card.

The code of this block is integrated in VIVADO. But the component is provided so that we can change its parameters (inputs and outputs).

Code 01:



```vhdl
component xadc_wiz_0 is
    port(
    daddr_in        : in  STD_LOGIC_VECTOR (6 downto 0);     -- Address bus for the dynamic reconfiguration port
    den_in          : in  STD_LOGIC;                         -- Enable Signal for the dynamic reconfiguration port
    di_in           : in  STD_LOGIC_VECTOR (15 downto 0);    -- Input data bus for the dynamic reconfiguration port
    dwe_in          : in  STD_LOGIC;                         -- Write Enable for the dynamic reconfiguration port
    do_out          : out  STD_LOGIC_VECTOR (15 downto 0);   -- Output data bus for dynamic reconfiguration port
    drdy_out        : out  STD_LOGIC;                        -- Data ready signal for the dynamic reconfiguration port
    dclk_in         : in  STD_LOGIC;                         -- Clock input for the dynamic reconfiguration port
    reset_in        : in  STD_LOGIC;                         -- Reset signal for the System Monitor control logic
    vauxp2          : in  STD_LOGIC;                         -- Auxiliary Channel 2
    vauxn2          : in  STD_LOGIC;
    vauxp10         : in  STD_LOGIC;                         -- Auxiliary Channel 10
    vauxn10         : in  STD_LOGIC;
    busy_out        : out  STD_LOGIC;                        -- ADC Busy signal
    channel_out     : out  STD_LOGIC_VECTOR (4 downto 0);    -- Channel Selection Outputs
    eoc_out         : out  STD_LOGIC;                        -- End of Conversion Signal
    eos_out         : out  STD_LOGIC;                        -- End of Sequence Signal
    alarm_out       : out STD_LOGIC;                         -- OR'ed output of all the Alarms
    vp_in           : in  STD_LOGIC;                         -- Dedicated Analog Input Pair
    vn_in           : in  STD_LOGIC );
end component;
```

In our case, we have two analog signals to convert, but the Nexys 4DDR have only one ADC block that can do the conversion. Therefore, the solution was that we programed a small state machine. This state machine will read **port "vaux2"** for current and "**vaux10"** for voltage simultaneously, depending on the "**daddr", "eoc", "drdy"**, and "**den"** signals of the XADC port.

Code 02:

```
if clk'event and clk='1' then
    case state is
    when idle =>  -- waiting for eoc
        den <= '0';
        daddr <= "0010010"; -- Vaux2
        if eoc = '1' then
            state <= read1; -- read the first data
            den <= '1';
        end if;
    when read1 =>
        den <= '0';
        if drdy = '1' then
            data1 <= data;
            daddr <= "0100000"; -- vaux10
            state <= tempo;
        end if;
        if eoc = '1' then  -- normally it never happens
            state <= idle; -- but better to put it
        end if;
    when tempo =>
        den <= '1';
        state <= read2;
    when read2 =>
        if drdy = '1' then
            data2 <= data;
            state <= idle;
        end if;
    end case;
end if;
```

In this code, “data” represent the output of the ADC block, “data1” and “data2” represent the signal of current and signal of voltage respectively.

III.3.1.2 Current sensor’s code

“data1” is a signal type std_logic_vector and his size is 12 bits. So when we convert it to type integer, its range become between 0 to 4094.

The first rule of thee will give us the voltage that entered the XADC from the current sensor. In addition, the second rule of three will give us real value of the current generated by the panel.

Code 03:

```
ibrut <= conv_integer(unsigned(data1))*1000;
voltageadc<=(ibrut/4094)*5000;
current_out <= ((voltageadc-2400000)*5000)/2500000;
```

III.3.1.3 Voltage sensor's code

"data2" is also a std_logic_vector with 12 bits in size, and as mentioned earlier, its represent the signal that XADC receive from the voltage sensor, and by converting this signal into an integer and the do the rule of three, we will have the real value of the voltage generated by the panel.

Code 04:

```
vbrut <=conv_integer(UNSIGNED(data2))*1000;
voltage_out <= (vbrut * 17000) / 4094000;
```

III.3.1.4 P&O code

First, we calculate the power P, ΔP and ΔV

Code 05:

```
power <= (current_out * voltage_out)/1000;
 -- Calculate power and voltage variations
delta_power <= power - previous_power;
delta_voltage <= CONV_INTEGER(voltage_out) - previous_voltage;
```

Then the program of P&O inside a process

Code 06:

```
process (delta_power, delta_voltage, voltagepo)
    begin
        -- If power increased, reduce output voltage
        if delta_power > 0 then
            -- Compare voltage delta
            if delta_voltage > 0 then
                voltagepo <= voltage_out - step_size;
            else
                voltagepo <= voltage_out + step_size;
            end if;
        -- If power decreased, increase output voltage
        else
            if delta_voltage > 0 then
                voltagepo <= voltage_out + step_size;
            else
                voltagepo <= voltage_out - step_size;
            end if;
        end if;
end process;
```

After the process is over, “previou_power” and “previous_voltage” will be updated. And the value of the voltage obtained at the end of this process will be converted to std_logic_vector.

Code 07:

```vhdl
-- Update previous values
  previous_power <= power;
  previous_voltage <= voltage_out;
  mppt_voltage <= std_logic_vector(conv_UNSIGNED(voltagepo,16));
```

III.3.1.5 PWM code

This part of the program is the last one. Here, the program receive “mppt_voltage” which is a vector of 16 bits and convert it to integer. Then, with the rule of thee, we obtain the value of the duty cycle with a range between 0 and 65535.

Meanwhile, a counter is running. This counter will count until it reach a constant named “counter_max” and then initialize the counter.

Code 08:

```vhdl
process (sclk)
    begin
        if rising_edge(sclk) then
            -- Update the counter
            counter <= counter + 1;
            if counter >= counter_max then
                counter <= 0;
            end if;

            -- Calculate duty cycle based on mppt_voltage
            -- Assuming mppt_voltage varies from 0 to 65535 (16 bits)
            duty_cycle <= conv_integer(unsigned(mppt_voltage)) * counter_max / 65535;
```

“sclk” is a clock signal. This signal is the result of Code 10 that divide the clock of our FPGA board “clk” (100Mhz) to “sclk” (2550Khz). This code is programmed separately from the main program, and then we called it with Component instructor in the Architecture before “begin”

We want to have the result (PWM) with a frequency of 10 KHz, so the value of “counter_max” mast be 250.

Code 09:

```vhdl
component clk_div2 is
    Port (
        clk : in std_logic;
        sclk : out std_logic );
end component;
```

Code 10:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity clk_div2 is
    Port (  clk : in std_logic;
           sclk : out std_logic);
end clk_div2;
architecture my_clk_div of clk_div2 is
  constant max_count : integer := (20);
  signal tmp_clk : std_logic := '0';
begin
  process (clk,tmp_clk)
   variable div_cnt : integer := 0;
   begin
      if (rising_edge(clk)) then
         if (div_cnt = MAX_COUNT) then
            tmp_clk <= not tmp_clk;
            div_cnt := 0;
         else
            div_cnt := div_cnt + 1;
         end if;
      end if;
      sclk <= tmp_clk;
   end process ;
end my_clk_div;
```

After that, a comparison will be made to compare duty_cycle with the counter:

code 11:

```
         -- Generate the PWM signal
         if counter < duty_cycle then
             pwm_out <= '1'; -- High state
         else
             pwm_out <= '0'; -- Low state
         end if;
     end if;
```

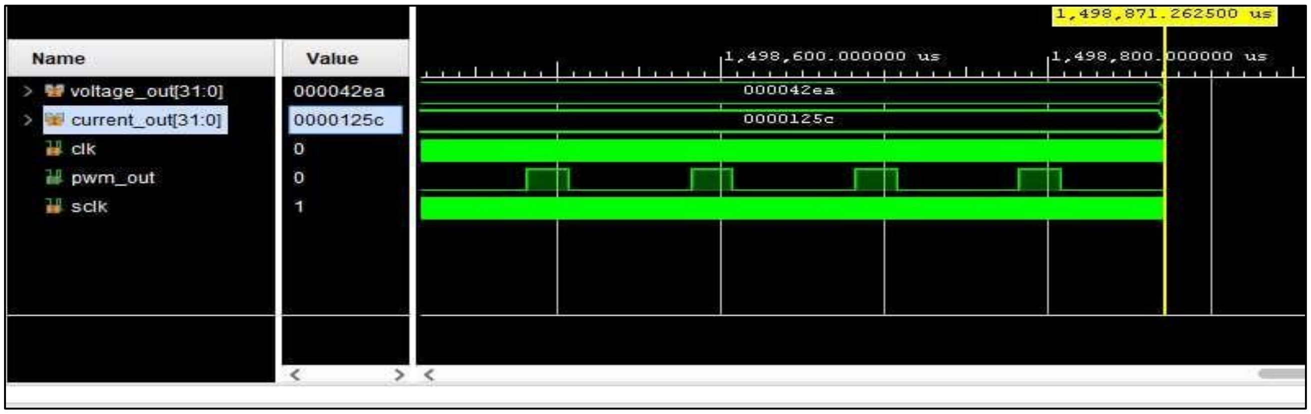
In addition, that is how the pwm signal is made which is a square signal that will vary depending on the variation in the value of “mppt_voltage” from the output of the P&O part. This signal will exit from the FPGA cart with a frequency of 10Khz and then go to the Gate of our Mosfet (transistor) in the boost.

In some part of our program, we multiplied the integer values by 1000. And reason is that we didn’t want to deal with float type.

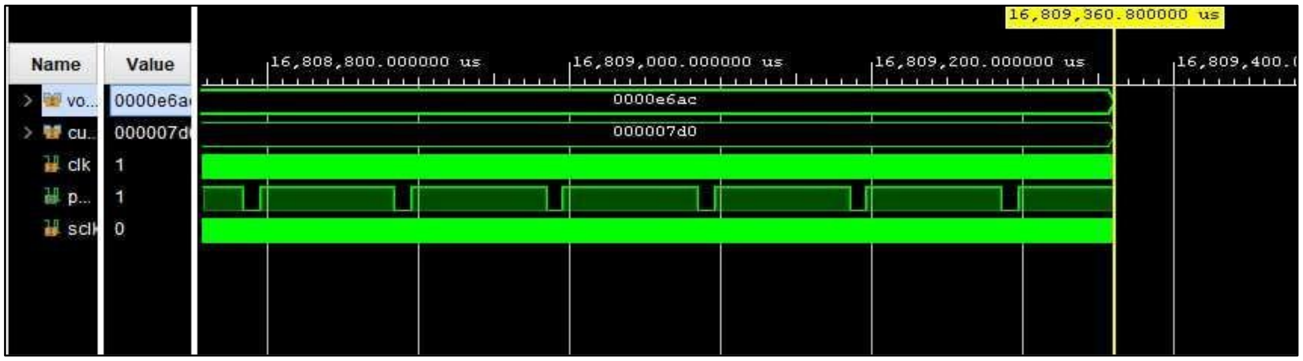
III.3.2 Results of the program

With a simulation tool in VIVADO, we were able to see the result of the program, which is the PWM signal.

Without using the FPGA card and the two sensors, we put manually different value in the inputs that represent the current and voltage (Figures III.10)



a) PWM signal with the increase of “mppt_voltage”



b) PWM signal with the decrease of “mppt_voltage”

Figure III. 10: the results of PWM signal with diffrent inputs

From the Figure III.10, we can see that our program will successfully change and adapt the PWM signal in every change in the value of our inputs.

The variation in inputs (voltage and current) can either increase or decrease the “mppt_voltage”. In Figure III.10 (a), the “HIGH” state of PWM signal decrese because “mppt_voltage” increased. And it is the opposite in Figure III.10 (b).

III.3.3 Diagram of the program

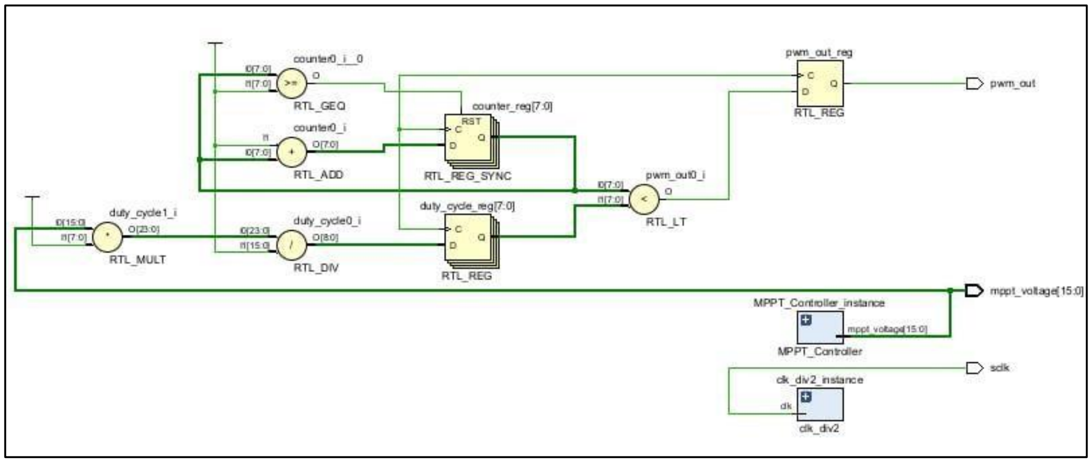


Figure III. 11 : Vivado diagram

III.4 Conclusion

During our discussion, we explored practical aspects related to photovoltaic, including PV panel characterization, hardware components (such as current and voltage sensors), and software considerations (such as algorithms like P&O and PWM). These insights are valuable for designing effective photovoltaic systems.

General conclusion

General Conclusion

FPGAs have become essential components in numerous digital systems due to their flexibility and high performance compared to traditional microcontrollers. For applications such as Maximum Power Point Tracking (MPPT) in photovoltaic (PV) systems, implementing on FPGA offers significant advantages.

Implementing MPPT, using FPGA and VHDL, with P&O method, represents a significant advancement in optimizing photovoltaic systems. This approach combines the high processing power of FPGAs with the flexibility of VHDL to create robust and adaptive systems capable of maximizing solar energy production even under variable environmental conditions. These technologies contribute to more efficient use of renewable resources and pave the way for innovations in solar energy.

Bibliography

[1] *Gil, Leopoldo & Saldivar, Belem & Portillo-Rodríguez, Otniel & Avila Vilchis, Juan & Martinez Rodriguez, Panfilo & Martinez Mendez, Rigoberto. (2019). Flatness-Based Control for the Maximum Power Point Tracking in a Photovoltaic System. Energies. 12. 1843. 10.3390/en12101843.*

[2] *mer M. Elbadri, Khalifa M. Mohammed, and Zuhir I. Saad. 2021. Comparison of P&O and PSO Algorithms using Matlab/Simulink. In The 7th International Conference on Engineering amp MIS 2021 (ICEMIS'21), October 11–13, 2021, Almaty, Kazakhstan. ACM, New York, NY, USA, 7 pages.*

[3] *Materials Concepts for Solar Cells Downloaded from www.worldscientific.com by 105.108.183.141 on 05/27/24. Re-use and distribution is strictly not permitted, except for Open Access articles*

[4] *T.-Y. Kim, H.-G. Ahn, S. K. Park, and Y.-K. Lee, "A novel maximum power point tracking control for photovoltaic power system under rapidly changing solar radiation," in ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570), vol. 2, pp. 1011–1014, IEEE, 2001.*

[5] *W. Xiao and W. G. Dunford, "Evaluating maximum power point tracking performance by using artificial lights," in 30th Annual Conference of IEEE Industrial Electronics Society, 2004. IECON 2004, vol. 3, pp. 2883–2887, IEEE, 2004*

[6] *Fevzi Çakmak, Zafer Aydoğmuş & Mehmet Rıda Tür (25 Dec 2023): Analysis of Open Circuit Voltage MPPT Method with Analytical Analysis with Perturb and Observe (P&O) MPPT Method in PV Systems, Electric Power Components and Systems, DOI: 10.1080/15325008.2023.2296958*

[7] *M. Al-Dhaifallah, A. M. Nassef, H. Rezk, and K. S. Nisar, "Optimal parameter design of fractional order control based inc-mppt for pv system," Solar Energy, vol. 159, pp. 650–664, 2018*

[8] *Sarang, S.A., Raza, M.A., Panhwar, M. et al. Maximizing solar power generation through conventional and digital MPPT techniques: a comparative analysis. Sci Rep* ***14****, 8944 (2024). https://doi.org/10.1038/s41598-024-59776-z*

[9] *Derbeli, M.; Napole, C.; Barambones, O.; Sanchez, J.; Calvo, I.; Fernández-Bustamante, P. Maximum Power Point Tracking Techniques for Photovoltaic Panel: A Review and Experimental Applications. Energies 2021, 14, 7806*

[10] *H. RASHID, MUHAMMAD (2001). POWER ELECTRONICS HANDBOOK, DC-DC Converters (p. 211-216). PRESSE ACADÉMIQUE 2001, SAN DIEGO, SAN FRANCISCO, NEW YORK, BOSTON, LONDRES, SYDNEY, TOKYO*

[11] *Ahteshamul Haque (2014) Maximum Power Point Tracking (MPPT) Scheme for Solar Photovoltaic System, Energy Technology & Policy, 1:1, 115-122, DOI: 10.1080/23317000.2014.979379*

[12] *Kevin Skahill, VHDL for programmable logic, Massachusett: Addison-Esley, 1996.*

[13] *Enrique Mandado Pérez, ValdésD. and Jacobo, L., Dispositivos lógicos programables, Madrid: Thomson, D.L., 2003.*

[14] *N. Marques, Méthodologie et Architecture Adaptative pour le Placement Efficace de Tâches Matérielles de Tailles Variables sur des Partitions Reconfigurables, Thèse de Doctorat, l'Université de Lorraine, 2012.*

[15] *Denis Rabasté, programmation des CPLD et FPGA en VHDL avec Quartus II, IUFM d'Aix-Marseille « .*

[16] *Mithila kumanjana, «History of Field-Programmable Gate Arrays[1960-2022],» 09 07 2023. [En ligne]. Available: https://circuitprofessor.com/history-field-programmable-gate-arrays-fpga/#pal. [Accès le 06 06 2024].*

[17] *Lucas S. Parobek, RESEARCH, DEVELOPMENT AND TESTING OF A FAULT-TOLERANT FPGA-BASED SEQUENCER FOR CUBESAT LAUNCHING APPLICATIONS, MONTEREY.CALIFORNIA, march 2013.*

[18] *E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A.Tisan, M. W. Naouar, FPGAs in Industrial Control Applications, IEEE Transactions on Industrial Informatics, 2011.*

[19] *U. Farooq, Z. Marrakchi, H. Mehrez, Tree-Based Heterogeneous FPGA Architectures Application Specific Exploration and Optimization, Springer Science and Business Media, 2012.*

[20] *H. Parvez, H. Mehrez, Springer Science and Business Media, 2011.*

[21] *«embedded computing desing,» [En ligne]. Available: https://embeddedcomputing.com/technology/processing/semiconductor-ip/fpga-insights-and-trends-2023-unleashing-the-power-of-fpga. [Accès le 2024].*

[22] *«www.digillentinc.com,» 11 September 2014. [En ligne]. Available: https://www.eng.auburn.edu/~nelson/courses/elec4200/FPGA/nexys4ddr_rm.pdf. [Accès le 23 05 2024].*

[23] *E. O. Hwang, , Digital Logic and Microprocessor Design with Interfacing, Cengage Learning, 2016.*

[24] *Xilinx, «Vivado Overview,» 2022. [En ligne]. Available: https://www.xilinx.com/products/design-tools/vivado.html. [Accès le 19 05 2024].*

[25] *« Vitis unified softwar platform,» AMD, 2024. [En ligne]. Available: https://www.xilinx.com/products/design-tools/vitis.html. [Accès le 07 06 2024].*

[26] *L. H. Crockett, R. A. Elliot, M. A. Enderwitz, R. W. Stewar, The Zynq Book,Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000, All Programmable SoC, Scotland UK: Department of Electronic and Electrical Engineering University of Strathclyde Glasgow, 2014.*

[27] *VOLNEIN A.PEDRONI, Circuit desing with VHDL, London,England: Massachusetts Institute of Technology, 2004.*

Abstract

This work presents a method to design a Maximum Power Point Tracking (MPPT) controller based on the Perturb and Observe (P&O) method for optimizing the maximum power point of a solar panel, using an FPGA for implementation. The input variables typically include the voltage and current of the solar panel, while the output is the optimal duty cycle of a DC-DC converter to maximize energy conversion efficiency. The different parts of the controller, such as voltage and current measurement, disturbance analysis to adjust the operating point, and observation of variations to determine the maximum power point, are implemented using the VHDL hardware description language. This approach aims to enhance and improve the overall system performance. The VHDL design is then simulated using Vivado software, confirming that the results obtained are accurate for various simulation values, ensuring the robustness and effectiveness of the MPPT P&O controller under different operating conditions.

Keywords: Photovoltaic, Maximum Power Point Tracking, Perturbation & Observation, Boost, Field Programmable Gate Arrays, VHSIC Hardware Description Language

Résumé

Ce travail présente une méthode pour concevoir un contrôleur de Point de Puissance Maximale (MPPT) basé sur la méthode Perturbation et Observation (P&O) afin d'optimiser le point de puissance maximale d'un panneau solaire, en utilisant un FPGA pour l'implémentation. Les variables d'entrée comprennent généralement la tension et le courant du panneau solaire, tandis que la sortie est le cycle de fonctionnement optimal d'un convertisseur DC-DC pour maximiser l'efficacité de conversion d'énergie. Les différentes parties du contrôleur, telles que la mesure de la tension et du courant, l'analyse des perturbations pour ajuster le point de fonctionnement, et l'observation des variations pour déterminer le point de puissance maximale, sont implémentées en utilisant le langage de description matériel VHDL. Cette approche vise à améliorer les performances globales du système. La conception VHDL est ensuite simulée à l'aide du logiciel Vivado, confirmant que les résultats obtenus sont précis pour différentes valeurs de simulation, assurant la robustesse et l'efficacité du contrôleur MPPT P&O dans différentes conditions de fonctionnement.

Mots clés : Photovoltaic, Maximum Power Point Tracking, Perturbation & Observation, Boost, Field Programmable Gate Arrays, VHSIC Hardware Description Language