Solar Energy Control System Design
Solar Energy Control System Design

SUN YANG

Master's Thesis in System-on-Chip Design
Supervisor: Björn Jernström
Examiner: Zhonghai Lu

TRITA-ICT-EX-2013: 273
Abstract

This thesis covers design, simulation and implementation of a solar energy control system for an on grid energy storage device. The design covers several control methods such as energy balance control, operating mode switching and data exchange. A genetic algorithm was designed to optimize the control system parameters design, and the algorithm’s simulation and real time operating system implementation showed comparable results. The control system was implemented to connect a power supply to the grid. The power supply simulated a solar panel and connected to an electrical grid via Energy Hub equipment, and the energy transfer characteristics of designed control system were tested. The results showed that the selected algorithm matches the target performance criteria.
Contents

1 Introduction
   1.1 Background .................................................. 1
   1.2 Problem ..................................................... 2
   1.3 Goals ........................................................ 4
   1.4 Method ...................................................... 4
   1.5 Report Outline ............................................. 4
   1.6 Summary .................................................... 5

2 Design Entity
   2.1 Function Introduction ....................................... 7
   2.2 Hardware Description ...................................... 8
       2.2.1 Device Architecture ..................................... 8
       2.2.2 FPGA soft processor .................................... 9
   2.3 Functional block ........................................... 10
       2.3.1 Controllers ............................................. 10
       2.3.2 Interfaces ............................................. 11
   2.4 Summary .................................................. 11

3 Algorithm Design
   3.1 MPPT ....................................................... 13
   3.2 Energy Balance ............................................. 20
   3.3 Summary .................................................. 23

4 Algorithm Comparison and Optimization
   4.1 Comparison Method .......................................... 25
   4.2 Model ....................................................... 25
   4.3 Comparative Optimization .................................. 30
       4.3.1 MPPT .................................................. 30
       4.3.2 Energy Balance ......................................... 32
   4.4 Simulation ................................................ 42
       4.4.1 Software Simulation .................................... 42
4.4.2 Hardware Simulation ........................................... 42
4.5 Summary ............................................................ 45

5 Implementation and Result ......................................... 47
  5.1 Necessity Of Using RTOS ....................................... 47
  5.2 RTOS Comparison ................................................. 48
  5.3 Tasks Assignment ............................................... 51
  5.4 Tracing Analyze .................................................. 55
  5.5 Result and Conclusion ......................................... 56
  5.6 Summary ........................................................... 60

6 Summary and Future Work .......................................... 61

Bibliography ........................................................... 63
Chapter 1

Introduction

1.1 Background

Today, coal, petroleum and natural gas are still the main characters in electricity generating. Since 1980, the alarming decrease in fossil fuel quantities and increase in atmospheric carbon dioxide composition has been reported many times. So far, utilization efficiency has significantly improved, but it is not a sustainable solution, and renewable energy sources are a more sustainable solution in the long run.

Energy sources such as nuclear, hydroelectric, solar, wind, and geothermal have been introduced. Among these solutions, solar energy using photovoltaics (PV) have an advantage due to the ability of installing small scale systems and have the potential for scaling these systems and connecting them to the existing grids.

According to the European Photovoltaic Industry Association (EPIA)[2], until 2012, the cumulative PV capacity of the world has reached to 102 GW, which is equal to the annual energy generation of 16 coal power stations. These PV installations save more than 53 million tons $CO_2$ each year and the numbers are rapidly growing. In 2012, there are 31.1 GW of PV systems installed around the world, 17.2 GW of which are installed in Europe. Germany was the top market in last year, followed by China, Italy, USA and Japan.
CHAPTER 1. INTRODUCTION

Figure 1.1. Evolution of global PV cumulative installed capacity 2000-2012 (GW)

PV market has growth potential. EPIA provides a PV cumulative capacity forecast against three scenarios developed by National Renewable Energy Action Plans of European Commission. Potential PV market is influenced considerably by political decisions. In the Policy-Driven scenario, the European market will grow slowly from 16-17 GW in 2013 to around 25-28 GW in five years. In this case, the global market would be more than 84 GW in 2017. Europe will represent less than half of the global PV market since the APAC region represents 20 to 30 GW each year. Chinese authorities claimed that China could add 10 GW of PV installations each year.

1.2 Problem

There are many PV application categories which have been developed.

- batteries
- electric vehicles
1.2. PROBLEM

- sending to grid

Extra generated electric power can be used to store in battery for emergency power cut, charging electric drive vehicle and sending to electrical grid. In a local residential areas, when a power input of solar grid decreases, an energy storage unit can supply its stored energy to decrease the energy variation in the power station.

The report of EPIA also provide another scenario called off-grid solar power system where the PV system can be installed in areas that are difficult to connect to the grid such as islands or mountainous areas.

Many devices have been developed for household solar power system. In this thesis topic, the design target, called Energy Hub, is a grid-tie battery storage photovoltaic system.

A grid-tie battery storage photovoltaic system consists of multiple modules. Photovoltaic modules, or solar panels, are used to convert sunlight to DC electricity. Every grid-tie photovoltaic system needs one or more inverters to transform DC power received from solar panels to AC for connecting with the electrical grid.

Photons in sunlight radiation that hits a solar panel are absorbed as energy in the semiconducting material. The semiconducting material consists of two layers, an N-type semiconductor layer, and a P-type semiconducting layer. When a photon is absorbed, the potential barrier of the P-N junction is lowered, which helps the excess electrons from the N-type material flow to the P-type material. This electron flow generates a current. The PV power output maximum point varies according to the radiant sunlight intensity and temperature, therefore the current flow is controlled by the Energy Hub to track the changing maximum power point.

For each energy controller module, there are plenty of optimized or ameliorated algorithms. It is necessary to use a method to give a suitable group of parameters for each algorithm. These parameters can represent the best performance of each algorithm in order to compare and decide which algorithm is best fit in the application scene.

To operate these power module controllers on a processor, the policy used to implement each of them is another problem need to be solved. In a time critical case, it is common to use a real time operating system (RTOS) to keep the device operating in its expected state so that every task can meet its deadline. The advantages of using RTOS will be discussed in a later chapter.
1.3 Goals

The goals of this thesis are listed below.

- Theoretical analysis of the Energy Controller functions
- Designing an Energy Controller
- Designing different operating modes
- DC-link balance
- Implementation of the energy controller in a Microblaze system on an FPGA
- Testing and verifying functions on actual hardware
- Feasibility study RTOS vs Interrupt based control loop

1.4 Method

A quantitative research methodology is used in the design of control algorithms. First, the mathematical model is generated using a python script. The suitable algorithm and parameters were simulated and implemented in hardware in the form of a task in RTOS. The simulated and implemented model results were compared.

1.5 Report Outline

The thesis report covers the design process from model construction to hardware simulation. The first chapter introduces the background, explains the issue of this thesis topic and specifies which method is adopted to solve the problem. In the second chapter, the hierarchy of the device is described. The third chapter introduces algorithms for each control module. The fourth chapter provides optimized methods to compare algorithms and test them in both software and hardware environments. Then the fifth chapter is about how the control algorithm is implemented on hardware and how the real time operating system increased the performance of the firmware. After that, the result of the control algorithm both from software and hardware simulation are provided and the results are discussed. The last chapter gives a summary and a discussion about future work on this topic.
1.6 Summary

With the continuous consumption of fossil fuel, sustainable resources become a serious problem for the humanity. Among all sorts of sustainable resources, photovoltaic energy is applicable and also has a huge market potential. A type of device, designed for a household photovoltaic installation, is used to convert energy from solar panels to electrical grid systems and household equipments. This device needs control algorithms for stable operation a high energy efficiency ratio. The target of this thesis work is to design, simulate and implement these algorithms.
Chapter 2

Design Entity

This chapter will provide a basic description about the designed energy control device. The introduction consists of two sections; function description and hardware hierarchy.

2.1 Function Introduction

The Energy Hub device is connected in parallel with the power circuit of household. The basic operating scene of this device is working as a photovoltaic inverter, that regulates the energy generated by the solar panel and provides AC power to the household circuit, and sends the excess power to the electricity grid.

Power outage is an annoying issue which happens occasionally in daily life. In a metropolitan area, a possible reason might be power station malfunction, repair or overloading. More accidents might happen in the countryside since the area is more likely to have accidents and natural disasters, and they have less priority of power reconnection and repairs than urban areas. For these scenarios the Energy Hub includes a lithium battery stack. This battery stack can be charged by energy from solar panels or electrical grid. When the power from the grid goes down, the house electric equipment can still operate with the power stored in the battery. A more common usage of the battery stack is during the night or on cloudy days, when the solar panel does not generate sufficient power.
2.2 Hardware Description

2.2.1 Device Architecture

Besides the household circuit, the Energy Hub measures the 3-phase grid current, then either sends or receives power from the grid using the patented current equalizing technology. In a practical installation, the Energy Hub connects to a solar cell, battery stack and the power grid.

![Figure 2.1. Current Equalizing Technology](image)

The whole device can be divided into three parts.

- Interface board: it contains all connectors, and most of the energy components such as capacitors and DC relays.

- Power modules: the power module is designed to connect with different types of power devices such as solar panel, battery, electrical grid and so on. Every power module board is in a same size and using same communication method.

- Control board: in the current prototype, a Xilinx FPGA development board is using as the control board to process all data and control all analog devices.

The control logic and power transfer is well isolated as this design. DC relays and fuses is used to protect the control circuit from high voltage circuit. The interface board and power modules have already been designed before the thesis started and those are evaluated during this thesis work. The main topic and actual work of this report is to design a program for controlling on FPGA. The other parts are used in evaluation and verification.
2.2. HARDWARE DESCRIPTION

![E HUB Diagram]

**Figure 2.2.** Hardware Hierarchy

### 2.2.2 FPGA soft processor

The FPGA includes three main function blocks. It has one external data processor, one main control processor and a hardware layer block.

The hardware layer block has been finished before this thesis work started. It handles all data transmission with the interface board, such as I2C interface with power modules control, current sensor access and some other settings. It uses a shared block memory to exchange all data with the control processor. The shared block memory is memory-mapped to the internal AXI bus directly, which connects the processor and all the other IP cores such as timer or GPIO.

Programming the control processor is the main scope of this thesis work. It controls the whole system via hardware block RAM. In the operating state, it controls each power module and peripheral by implementing them as tasks. All tasks are managed by a real time operating system. The processor is implemented as a Microblaze soft core in the FPGA.

The external data exchange processor is used to dealing with all the data output, which includes LCD display and Ethernet. This processor can be implemented on another internal Microblaze core or on an external chip.

Since control tasks is the most critical part, the data transfer task is isolated from the control processor in case of disturbing the control tasks. Power must be controlled to run in a regular range to avoid serious errors. Therefore, it is
reasonable to isolate the external data exchange and energy control by using dual processors to take response each of them.

2.3 Functional block

![Functional Blocks Diagram]

**Figure 2.3.** Functional Blocks

### 2.3.1 Controllers

Here is the list of necessary control functions. Some of them might be trimmed or merged with others in the implementation.

- MPPT controller is one of the main components. As the maximum power operating point is drifting depends on climate, which is unpredictable in short time scale, this controller keeps the solar panels working in the highest energy efficiency ratio.
2.4. SUMMARY

- The energy balance controller keeps the energy stored in the DC link at a constant level by controlling the output current. It is because in most of the case the energy comes from solar panel which current is controlled by the MPPT controller.

- The DC energy input from the solar panel is bipolar - it has three pins that are positive, negative and neutral. Due to sensor measurement error, the positive and negative power sent to the DC link might be unsymmetrical. Because the energy sent to the electric grid is AC with positive and negative cycles, a symmetric controller is necessary for balancing the energy storing in DC link.

- The device has a number of operating modes that depend on the type of power sources and loads. These modes are defined for easier control.

- Before the device starts, the DC link is designed to charge to the operating level smoothly to protect the device from damaging power surges.

2.3.2 Interfaces

For decoupling the controller and external hardware from shared hardware resources, interface accessing functions have to be isolated from control tasks.

- UART interface is connected with the external data processor. It takes commands such as mode switching, or setting operating parameters commands and sends to each control tasks besides communicating with UART.

- A CAN bus interface is connected with the battery controller.

2.4 Summary

This chapter is mainly about introducing the Energy Hub. This device is connected in parallel with the household power circuit. It gets energy from PV panels and injects to the household grid as three phase AC power.

The whole device is separated into three parts - interface board, power module and control board. The designed control algorithm will be implemented on the control processor which is on the control board. Functions of the control processor include MPPT, energy balance, symmetric, mode switch, device booting and two interfaces - UART and CAN bus.
Chapter 3

Algorithm Design

This chapter will introduce the basic algorithm of MPPT and Energy Balance, then several improved methods for each of them will be presented.

3.1 MPPT

The output power of photovoltaic solar cell has non-linear characteristics. Solar cells have a complex relationship between solar radiation and temperature. To increase output efficiency and decrease generated power cost, the Maximum power point tracking (MPPT) technique is widely used. With MPPT controller can track the maximum power point without measuring weather conditions.

A MPPT system has a sequential workflow. First, it reads the output from the PV panels, analyzes and calculates the output power, then changes the value to obtain the maximum power of any given environmental conditions. Traditional MPPT controllers usually track by changing the duty rate of a transistor that controls the current level. With a software implementation, a selection of algorithms can be chosen to achieve a higher efficiency.

The graph below is an ordinary photovoltaic panel IV curve. From this graph, it can be seen that the IV curve effected by solar radiance and temperature. To obtain the maximum power, the product of current and voltage has to reach the maximum level. When the weather is changing, for instance, the sun blocked by a piece of cloud will decrease solar radiance sharply, then the maximum power point will have corresponded change. The purpose of MPPT is to locate and keep tracking the maximum power point even if the solar radiance or temperature change rapidly.
There are many algorithms which have been proposed in previous literature study.

The Perturb and Observe (P&O) method is being used in most of MPPT products because its simple principle and easy implementation. When this algorithm is operating it will perturb the controlled value periodically. By the perturb operation, it will read the operating parameters and calculate the power. With the perturbed power value it will compare to the previous power and decide whether if the controlled value should increase or decrease. For example, if the power increases after perturbing, the controlled value will keep its changing direction in the perturbation period or vise versa. The flow chart

Figure 3.1. Solar Cell IV Curve[3]
3.1. MPPT

below describes the classic P&O algorithm.

\[ P_{P1}(n) = V_{P1}(n) \cdot I_{P1}(n) \]

\[
\frac{P_{P1}(n) - P_{P1}(n-1)}{\delta(n) - \delta(n-1)} > 0
\]

\[ k = \begin{cases} 
-1 & \text{NO} \\
1 & \text{YES}
\end{cases} \]

\[ \delta(n+1) = \delta(n) + \Delta\delta \cdot k \]

**Figure 3.2. P&O Algorithm**

The problem in the perturb and observe algorithm is that, it has to perturb all the time, which means when the maximum power point is under steady state, this algorithm will generate oscillations. Another problem is that it is unable to track the maximum point in a rapidly changing weather because of its step change value is determined.

To solve the first problem, if the measured power value is steady, or if measurement noise is considered, changed in a tiny range, the controlled value does not change. Then it will stop the oscillation near the maximum power point. To solve the second problem, the step variable can be set according to the power change level, for example the step controlled variable can be calculated by using a constant as a slope times power change value. Then
when the power change rapidly, controlled value changes in large step and vice versa. The flow chart of Improved P&O algorithm becomes like the chart below.[6]

\[
\frac{dP}{dI} = 0
\]  
\[dP = P_n(n) \cdot P_{n-1}(n-1)\]
\[dV = P_n(n) \cdot V_{n-1}(n-1)\]
\[d(n) = M \cdot \left| \frac{dP}{dV} \right|\]
\[k\rightarrow 0\]  
\[d(n+1) = d(n) + k \cdot a(n) \cdot \Delta \delta\]

**Figure 3.3. Improved P&O Algorithm**

The Incremental Conductance (INC) algorithm is based on observation of IV curve. The maximum power point must happen on the top of IV curve so here is the following equation

\[
\frac{dP}{dI} = 0
\]  
(3.1)

Since power can be represented as current multiply by voltage, so

\[
\frac{dP}{dI} = \frac{d(IV)}{dI} = V \cdot \frac{dV}{dI} = 0
\]  
(3.2)

When the operating point is at right of maximum power point, this equation will less than zero. So for getting closer to the maximum power point the current must increase. Next figure shows the INC algorithm flow chart.
3.1. MPPT

The incremental conductance method has advantage over P&O method that it does not have the oscillating around maximum power point and it can track under rapidly varying irradiation conditions. However, this method has oscillations and abnormal performance under rapidly changing conditions. For instance, the output value is implemented through the control of the duty cycle. If the duty cycle is less than 0.6, the change of controlled value is not significant. [11]

To improve dynamic performance, there are several algorithms proposed. For example, fractional-order incremental method try to describe dynamical behaviors as fractional-order diffusion to improve dynamic performance.[9] It uses the fractional-order method to compare the I-V changes towards high order change.

Assume that \( f(x) \) is a monomial of the form.

\[
 f(x) = x^k. \tag{3.3}
\]

The first derivative is as usual

\[
 f'(x) = \frac{d}{dx} f(x) = k x^{k-1}. \tag{3.4}
\]
Repeating this gives the more general result that

\[
\frac{d^a}{dx^a} x^k = \frac{k!}{(k-a)!} x^{k-a},
\] (3.5)

Which, after replacing the factorials with the Gamma Function, leads us to

\[
\frac{d^a}{dx^a} x^k = \frac{\Gamma(k+1)}{\Gamma(k-a+1)} x^{k-a}. \tag{3.6}
\]

According to incremental conductance equations, the I-V can be described as fractional-order differentiation.

\[
\frac{d^a}{dV^a} I(V) = \lim_{\Delta V \to 0} \frac{I(V) - aI(V - \Delta V)}{\Delta V^2}
\]

\[
\frac{d^a}{dV^a} I \approx \frac{I - aI_O}{(V - V_O)^a} \tag{3.7}
\]

\[
\frac{d^a}{dV^a} \left( -\frac{I_O}{V_O} \right) = \left( -\frac{1}{V_O} \right) \frac{d^aI_O}{dV^a} + \left( -\frac{1}{V_O} \right) \frac{d^aV^{-1}}{dV^a} \tag{3.9}
\]

\[
= \left( -\frac{1}{V_O} \right) \frac{\Gamma(2)}{\Gamma(2-a)} (I_O)^{1-a} + \left( -\frac{1}{V_O} \right) \frac{\Gamma(0)}{\Gamma(-a)} V_O^{-1-a} \tag{3.10}
\]

Where \( \Gamma(n) = (n-1)! \) and \( \text{Res}(\Gamma, -n) = \frac{(-1)^n}{n!} \), \( n = 0, -1, \ldots \)

The flow chart below shows that the fractional-order incremental conductance algorithm. It adjust controlled value using the instantaneous values \( I \) and \( V \) at present cycle and their corresponding values \( I_O \) and \( V_O \) stored at the end of previous iteration.

\[18\]
3.1. MPPT

![Fractional Order Incremental Conductance Algorithm](image)

Figure 3.5. Fractional Order Incremental Conductance Algorithm
The incremental changes in current and voltage become 
\[ d^a I \approx (I - aI_O) = \delta I \]
and 
\[ dI^a \approx (V - V_O)^a = \delta V^a, \]
respectively.

### 3.2 Energy Balance

The power from the power modules is transferred to a circuit through DC Link. The DC Link acts as a DC to AC convertor then sends the power to the electrical grid. It can be seen as a node that has large capacitance. To keep the DC link operating safely, the voltage stored in its capacitor has to be controlled at a stable value. To avoid the voltage going too high or too low, the input and output power levels must be balanced.

Beginning with the energy balance controller design, the simplest case to start with is to receive energy from a solar panel and send it to the grid. In this case, the energy level can be easily calculated from the output of the MPPT controller. We assume that the DC link is an ideal node with no power lost during the voltage conversion.

The grid voltage level is around 220 to 240 volts. To keep an energy balance, the output current is equal to solar panel power divided by 230. This feed forward solution works perfectly in the ideal case, although in the actual case there are uncalculated energy losses during transformation. Another problem faced is that unpredictable noise might lead to voltage drifting away to levels high enough to damage the hardware. To overcome these negative effects, a PID controller is added so that the feed forward equation can adjust the DC Link voltage to stay in nominal operating levels and make the switching gradient smooth. [12]

The PID controller has a control feedback loop mechanism that is widely used in industrial control systems. It takes an input which is the difference between expect 'setpoint' value and the measured value, which is referred to as 'error'. The output of a PID controller is the sum of three terms which are the proportional, the integral and derivative values, denoted P, I and D. Proportional value P is multiplied with the error with a proportion parameter. I is calculated by using the integral gain with the integration of error, or the sum of all previous error values discretely. D is the product of the gain and the rate of change of error. The output of the PID controller when fed into a system can control the output levels to be within the nominal values and prevent the output from overshooting.

\[ u(t) = P + I + D = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \]  
(3.11)
3.2. ENERGY BALANCE

The discrete form of the PID equation is

\[ u(n) = K_p e(n) + K_i \sum_{t=0}^{t=n} e(t) + K_d [e(n) - e(n-1)] \]  

(3.12)

There are several ways to improve the PID performance. For instance, a fast swinging current might also damage the hardware. To smooth the incremental step value of a PID controller, the derivative of previous equation can be used to adding to the step changing value. The step change value is

\[ u(n) = K_p \{e(n) - e(n-1)\} + K_i e(n) + K_d [e(n) - 2e(n-1) + e(n)] + u(n-1) \]  

(3.13)

The equation above does not have integral term. The standard PID algorithm use the sum of errors to be the integral term. If the PID control takes a long period to stabilize itself the Integral term might be too large then overflow.

A phenomenon happens when the set point is switched in a large step. Since \( Error = Setpoint - Input \), if the controlled value is already stable around the previous set point, the change of set point would cause an instantaneous error. Then the degree of how the derivative term is close to infinity depends on the time step coefficient precision. This would result in an undesirable spike in the output.

![Figure 3.6. Spike Caused By Setpoint Change](image-url)
For solving this issue, there is an easy solution.

\[
\frac{d\text{Error}}{dt} = \frac{d\text{Setpoint}}{dt} - \frac{d\text{Input}}{dt} \quad (3.14)
\]

When Setpoint is a constant:

\[
\frac{d\text{Error}}{dt} = -\frac{d\text{Input}}{dt} \quad (3.15)
\]

Since the setpoint would not drift as a curve, it can be considered as a constant and the phenomenon does not happen. This solution is also available in this design. The setpoint is set to 80 percent of the DC link capacity and the hardware parameters will not drift in a normal case. Therefore, the derivative term will be modified from

\[
D\text{term} = K_d \cdot (\text{Error} - \text{Error}_{prev}) \quad (3.16)
\]

to

\[
D\text{term} = -K_d \cdot (\text{Input} - \text{Input}_{prev}) \quad (3.17)
\]

If the PID parameters is tuned while the system is operating, it might generate a bump as the graph illustrated below:

![Figure 3.7. Bump Caused By Parameters Change](image-url)
3.3. SUMMARY

This is because the new I parameter times the history error while it does not affect the history since it is used to be another value. To solve this issue only need to modify the integral term from use KI times the history error to current error. For instance, it can be done by changing Integral term equation from

\[ I_{\text{term}} = K_i \cdot (Error + ErrorSum_{\text{prev}}) \]  \hspace{1cm} (3.18)

To

\[ I_{\text{term}} = K_i \cdot Error + I_{\text{term prev}} \]  \hspace{1cm} (3.19)

The integral term is used for accelerating the movement towards set point and eliminating steady-state error. However, whenever there is a large error, for instance, power from solar panel dropped sharply, the integral term might generate over-shoot and long time vibrating. It can be solved by adding a gate function on integral term, so the integral term is reset when it exceeds the gate limit.

\[ I_{\text{term}} = K_i K_g \sum_{t=0}^{t=n} e(t) \]  \hspace{1cm} (3.20)

\[ K_g = \begin{cases} 1, & |e(t)| \leq A \\ 0, & |e(t)| > A \end{cases} \]  \hspace{1cm} (3.21)

Merge the last three improve, it can result to the third PID algorithm into algorithm test.

\[ u(n) = K_p Error(n) + K_g \sum_{t=0}^{t=n} K_i \cdot Error(t) - K_d [Input(n) - Input(n-1)] \]  \hspace{1cm} (3.22)

\[ K_g = \begin{cases} 1, & |e(t)| \leq A \\ 0, & |e(t)| > A \end{cases} \]  \hspace{1cm} (3.23)

3.3 Summary

This chapter explains the control algorithm adopted in the design. The perturb and observe algorithm is a basic MPPT. The principle is to perturb the working condition then observe it and decide whether the controlled variable increases or decreases. It is easy to implement but generates oscillations. An improved algorithm is built by adding a condition to keep it stable at the maximum power level. The incremental conductance measures the step change of the voltage and current separately instead of the power in perturb and
observe algorithm. By doing so, it eliminates the oscillation and increases the
energy transfer efficiency. To increase its sensitivity, the improved incremental
conductance method adds fractional-order differentiation to it.

The energy balance controller is the sum of the feedforward result and a PID
controller output. Since the amount of injecting power is measurable, the
output current can be determined with a simple calculation. Considering the
energy transfer loss, a PID controller is used for fine tuning. Several improved
methods are used to trim undesirable pulses.
Chapter 4

Algorithm Comparison and Optimization

4.1 Comparison Method

From the previous chapter, there are several control algorithms for each control module with different methods to compare algorithms.

MPPT algorithms are easy to analyze and optimize since they have two parameters. These parameters are the step value and algorithm update frequency. The difference between each MPPT algorithm is the condition of step value applied, while the effects of the step value itself is the same.

The relationship of updating frequency and step value are linear. In another word, the actual parameters affecting the performance of algorithm is the product of update frequency and step value.

Tuning PID parameters is complex and can be a subject of an independent study project. Some PID algorithms might have more parameters than the regular three parameters. Therefore, applying the same set of parameters on different PID algorithms might lead to a large variance of results. One set of parameters having good performance on one algorithm might not be good for the other. To solve this problem and make each algorithm present a rather good performance and then compare them, a general optimization method is necessary.

4.2 Model

For simulation, a model that represents the hardware model is used. For instance, the input values of a MPPT algorithms are the voltage and current of photovoltaic panel. As environmental variables such as temperature and solar
irradiation power affect the photovoltaic panel power curve, a mathematical model of photovoltaic panels is built to observe the effects on MPPT tracking. Semiconductor materials such as monocrystalline silicon or polycrystalline silicon is used to making solar cells. This semiconductor converts radiation to electric currents. An electron-hole pair is created by capture a photon. Then there is an energy greater than the energy of band gap. But electrons and holes will only exist since their metastability. Afterwards, P-N junction generated from these carriers will spatially separate the electron and the hole the collection and prevents this recombination. Since the light-generated carriers do not have the ability of leaving the solar cell, the collection of carriers result to an electrons increment on the N-type side and a similar increment of holes in the P-type side. In another word, it will increase the potential energy. Then the a current that moves through semiconductors is produced by conduction band electrons.

In an actual product, the light absorbed by solar cells as a combination of direct solar radiation. The temperature increase the energy of electrons and reflect on velocity. And these two parameters becomes the most important parameters that could affect PV panels operating state.

PV cells can be configured in series to form modules. Then modules can be connected in parallel or series to form arrays. When modules are connected in series, they must have the same current to produce an additive voltage output. Similarly, to produce larger currents, modules must have the same voltage when connected in parallel.

PV cells can be modeled as a current source in parallel with a diode. When there is no light present to generate any current, the PV cell behaves only like a diode. As the intensity of incident light increasing, current is generated by the PV cell.

![Figure 4.1. Electrical Diagram of PV Cell](image)
4.2. MODEL

In an ideal cell, the total current \( I \) is equal to the current \( I_l \) generated by the photoelectric effect minus the diode current \( I_D \), according to the equation:

\[
I = I_l - I_D = I_l - I_O \left( e^{\frac{qV}{kT}} - 1 \right)
\]  

(4.1)

Where \( I_O \) is the saturation current of the diode, \( q \) is the elementary charge \( 1.6 \times 10^{-19} \) coulombs, \( k \) is a constant of value \( 1.38 \times 10^{-23} \) J/K, \( T \) is the cell temperature in kelvin, and \( V \) is the measured cell voltage that is either produced (power quadrant) or applied (voltage bias). A more accurate model will include two diode terms. However, we will concentrate on a single diode model at here.

Expanding the equation gives the simplified circuit model shown below and the following associated equation, where \( n \) is the diode ideally factor (typically between 1 and 2), and \( R_S \) and \( R_{SH} \) represents the series and shunt resistances:

\[
I = I_l - I_O \left( \exp \left( \frac{q(V+I \cdot R_S)}{nkT} \right) - 1 \right) - \frac{V + I \cdot R_S}{R_{SH}}
\]  

(4.2)

Figure 4.2. Simplified Equivalent Circuit Model for a PV Cell

The short circuit current \( I_{SC} \) corresponds to the short circuit condition when the impedance is low, and it is calculated when the voltage equals 0. \( I_{SC} \) occurs at the beginning of the forward-bias sweep, and it is the maximum current value in the power quadrant. For an ideal cell, this maximum current value is the total current produced in the solar cell by photon excitation.

\[
I_{SC} = I_{MAX} = I_l
\]  

(4.3)

27
CHAPTER 4. ALGORITHM COMPARISON AND OPTIMIZATION

The open circuit voltage $V_{OC}$ occurs when there is no current passing through the cell. $V_{OC}$ is also the maximum voltage difference across the cell for a forward-bias sweep in the power quadrant.

Hence, the mathematical model can be built by the equations above and the datasheet from solar panel manufacture. In this case, the data from SweModule AB is taken since it is used in Glava energy centers where is the device will be actually tested.

<table>
<thead>
<tr>
<th>Number of cells</th>
<th>60 pcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{OC}$</td>
<td>38.0 V</td>
</tr>
<tr>
<td>$I_{sc}$</td>
<td>8.8 A</td>
</tr>
<tr>
<td>Temperature coefficient of $V_{OC}$</td>
<td>-0.382 %/K</td>
</tr>
<tr>
<td>Temperature coefficient of $I_{sc}$</td>
<td>-0.320 %/K</td>
</tr>
<tr>
<td>Nominal operation cell temperature</td>
<td>47.5 °C</td>
</tr>
</tbody>
</table>

Table 4.1. SweModule PV Panel Parameters[14]

Environmental variables such as temperature and solar irradiation need to be considered in the model. The temperature coefficient has been given in the datasheet. Irradiation effect can be considered as linear. Then the previous equations become:

\[
I_t = I_{sc} \cdot (1 + (47.5 - T) \cdot (-0.32)) \cdot (s/1000) \quad (4.4)
\]

\[
I_O = \frac{I_t}{\exp\left(\frac{qV \cdot (1 - 0.382 \ast (47.5 - T) \ast (s/1000))}{k(T + 273.5)}\right) - 1} \quad (4.5)
\]

Where T represents temperature and s represents solar irradiation level. The unit of s is watt per square meter.

With an instant current value, the voltage can be calculated with known $I_t$ and $I_O$.

\[
U = (T + 273.5) \cdot k \cdot q \cdot \lg\left(\frac{I_t - I}{I_O}ight) + 1 \quad (4.6)
\]

For comparing result easily, a theoretical MPP current can be calculated by derivating the previous formulas.

\[
I_{mpp} = I_t + I_O - I_O \cdot e^{\omega\left(\frac{(I_t + I_O)}{I_O}\right) - 1} \quad (4.7)
\]

Where $\omega$ is the omega function.
4.2. MODEL

Figure 4.3. Ordinary PV Panel I-P Curve

This figure shows the simulation model of photovoltaic panel. It represents the current-power relationship in different irradiation level.

For the energy balance simulation, the dc link model is necessary as well as solar panel model. As the part of controlled unit in closed loop system, this model provides the input of energy balance controller and acts as the controlled target.

Figure 4.4. Simplified Equivalent Circuit Model for a PV Cell

The DC Link is modelled as a group of capacitors that takes energy from a power supply, such as solar panels, and sends out energy out. The power stored in the capacitors must not exceed its nominal power capacitance or else the battery will explode.
CHAPTER 4. ALGORITHM COMPARISON AND OPTIMIZATION

In this design, the DC link voltage value is measured as the input of the energy balance controller. It is need to be noticed that the voltage of solar panel is bipolar, and the energy is stored in two groups of capacitor. While the output is a 3-phase AC voltage, each phase voltage will take the power from both capacitor groups. The formula of the DC link model is:

\[
W_{DC} = \frac{U_{DC_{pre}}^2 * C_{dc}}{2} \quad (4.8)
\]

\[
dP_{DC} = P_{PV} - (I_{L1}U_{L1} + I_{L2}U_{L2} + I_{L3}U_{L3}) \quad (4.9)
\]

\[
U_{DC} = \sqrt{\frac{2(W_{DC} + dP_{DC} \cdot dt)}{C_{dc}}} \quad (4.10)
\]

The model first calculates the power of last condition stored in the capacitor. The power increment is then calculated by subtracting the sum of the power from each phases power comes in from PV panels. Since the voltage is separated into three phases, in every time segment there is always one phase taking energy from positive or negative capacitor groups, which helps making the energy balance more stable. Finally, the DC link voltage is updated by the sum of previous energy and the new energy segment.

4.3 Comparative Optimization

Either MPPT or Energy Balance has several ameliorate methods. Although there is a relative rational inference to support these algorithms, a method is still needed to compare their performance in different situations than the solar panel case. Moreover, each equation has several parameters that are needed to be determined for the comparison. It is necessary to use a generic method for tuning algorithms to optimize the performance of each algorithm. Since the model is built for the actual solar panel product, tuned gain can be found out using simulation and hardware calibration.

4.3.1 MPPT

Since no MPPT algorithm has more than two parameters, it is more convenient to analyze and reckon valid parameters.

For irradiation level, the active zone can be divided into two areas along the MPP(max power point position). Assuming the step changing current \( dI = I_{AA} \) where stands for the current sector in active zone after maximum power point \( I_{AA} = I_{SC} - I_{mpp} \), and \( I_{SC} \) stands for short circuit current. When
the current is less and close to MPP, the MPPT will request \( I_{n+1} = I_n + I_s \). Then in the next time step the current would come to very close to the cutoff area and operating power is close to zero. So to say, for keeping the power stable in MPP tracking process, \( dI \) has to be much smaller than the value of \( I_{AA} \). It must be noted that, \( I_{AA} \) becomes less because of the less irradiation.

For instance, when irradiation is 200 W/m\(^2\), \( I_{mpp} = 1.667A, I_{SC} = 1.760A, I_{AA} = 0.093 \). Taking step change current equal to the value of \( I_{AA} \), then \( dI = I_{AA}/10 = 0.01A \).

The MPPT controller executing frequency can be analyzed and determined as well. Assuming in an extreme whether condition, the irradiation would go from 1000 to 0 W/m\(^2\) in 10/sec. When irradiation is 1000 W/m\(^2\), \( I_{mpp} = 8.374A \). \( I_{sc} = 8.374 \) when irradiation is 951.6 W/m\(^2\) and \( I_{mpp} = 7.967A \). The irradiation decrease \( 1000 - 951.6 = 48.4 \) W/m\(^2\) takes \( 48.4/1000 \times 10 = 0.484/sec \). To adapt this changing speed, MPPT algorithm will execute \((8.374 - 7.967)/0.01 = 41 \) times. Then it result that the MPPT algorithm need to be executed in every \( 0.012/sec = 85Hz \).

The graph below shows the compare result of each MPPT algorithms.

![Comparison of MPPTs](image)

**Figure 4.5.** The comparison of each MPPT algorithms

Clearly, the first incremental conductance algorithm has the best performance. It is desirable to track the MPP during the rapidly changing weather conditions. The perturb and observe algorithm can track the irradiation level when
CHAPTER 4. ALGORITHM COMPARISON AND OPTIMIZATION

the weather condition is flat. However, due to its constant step change value, it could not adjust fast enough when there is a sharp irradiation decrease causing a drop in power output. Improved P&O algorithm solves the zero power issue but adds another issue that is slowing down the raise up speed. This algorithm might be selected if a static output power is required. Therefore, in the hardware implementation, the incremental conductance algorithm should receive more consideration than the others.

4.3.2 Energy Balance

To find PID gains, a genetic algorithm is adopted as the parameter algorithm. Since it has been proved that using genetic algorithm to tune the PID controller has good performance[8] and easy to implement.

Genetic algorithm is a heuristic search method that mimics the selection theory. It is a global optimization algorithm used to generate useful solutions for optimization and search problems. It uses techniques inspired by Darwin natural evolution, such as inheritance, mutation, selection and crossover.

![Figure 4.6. An Example of genetic algorithm, use polygons approach Mona Lisa [1]](image-url)
4.3. COMPARATIVE OPTIMIZATION

Genetic algorithm is a heuristic search that mimics the selection theory. It is a global optimization algorithm used to generate useful solutions for optimization and search problems. It uses techniques inspired by Darwin natural evolution, such as inheritance, mutation, selection and crossover.

In genetic algorithms, the solution of optimization problem is referred to as a population, and a group of genes form a chromosome, in our case, a chromosome is a group of PID parameters. In the beginning, the chromosome is generated randomly. The initial point is generated within a range to improve the quality of initial population. In each generation, every individual is evaluated to calculate a fitness value. The individuals in population are sorted by fitness.

The next step is to generate a next generation and form a new population. It is done by selection and reproduction which includes crossover and mutation. Selection is done by evaluating the fitness of each individuals. However, it does not mean that all low fitness individuals will be filtered. If only high fitness individuals are selected, that might lead to the algorithm converging towards a local optima rather than global optimum of the problem. This case is known as "over optimized". The genetic algorithm uses the principle that higher fitness individuals have a higher chance to be selected. On the other hand, a low fitness individual is more likely to be eliminated. After that, the selected population starts to crossover. Most of genetic algorithm has a crossover possibility, which means the possibility of two selected individuals crossing over and generating a next generation individual. For instance, if the crossover possibility is 0.8, then 80% population reproduce. Every two individuals generate two new individuals to replace the previous pair. Individuals that did not crossover keep their chromosomes. The crossovered individuals exchange their chromosomes then generate two new chromosomes. Part of chromosomes of each individual come from the father and the other part come from the mother. Since the reproduction probability is random, the percentage of genes that each parent contributes can vary from a 50% share. The next step is mutation, which would generate new individuals with random sets of chromosomes. Every genetic algorithm has a constant mutation coefficient. In this project, only one chromosome is randomly generated.

After the entire series of processes, every generation is different from the initial one, and approach the optimized target by increasing the fitness with each newer generation. This process repeats continually until satisfied the end condition.

The genetic algorithm has several limitations. [5] [7]
• Compared to the real application, the fitness function might be complex with long calculation time.

• The more parameters, or chromosomes that the individuals have, the longer and more memory intensive will be required by each iteration.

• The result is not guaranteed to be the best solution. The algorithm has a tendency to converge towards local optima.

• It is difficult to operating on dynamic data which means it is not suitable to use in real time auto tuning.

• The fitness measurement error would affect the optimization.

The above algorithm states are:

**Start** Generate population with random chromosome.

**Fitness** Evaluate the fitness of each chromosome in the population. The fitness value uses reciprocal variance.

**Selection** Select two parent chromosomes from the population according to their fitness.

**Crossover** With a crossover probability cross over the parents to form a new offspring. If no crossover was performed, offspring is a copy of parents.

**Mutation** With a mutation probability mutate new offspring at each locus of chromosomes.

**Accepting** Place new offspring in a new population.

**Replace** Use newly generated population to replace the previous one for the next iteration.

**Evaluate** If the end condition is satisfied, stop and return the best solution in current population.

The main principle of genetic algorithm is that a chromosome that has better fitness has more chances to crossover. The crossover gives a chance for the more fit gene to stay, which stands for PID gains and generates better species. Since random numbers are used in the initial population, mutation is used to import random new chromosome in iterations. This avoids missing finding higher fitness individuals. Compared to traversal for all possible parameters,
4.3. COMPARATIVE OPTIMIZATION

genetic algorithm decreases the executing time and provides an approximately optimized PID gains for each algorithms. However, there is possibility of missing better result even with mutation. It can be avoided by running the program several times or giving different inputs such as random or sine series.

The program is written in Python, using matplotlib to store results as plots. The input of PID controller is the variance input power and DC link operating voltage as setpoint.

Each iteration of generating new population is going through the selection, crossover and mutation for whole population. And the iteration of each generation generates new population and run the fitness method for every chromosome. The end condition is defined by the number of generations. Every generation the program records the chromosome in its generation who has the highest fitness and draws a plot for it. The program also records the fitness and chromosome of the champion and draws a plot in the end.

The pseudo code is given at below

\[
\begin{align*}
\text{InputData} &= \text{WaveGen(WaveType, WaveAmp, WaveFreq)} \\
\text{Population} &= \text{PopulationInitial()} \\
\text{while Counter equal 100 do} \\
\text{for Chromosome in Population do} \\
\text{\quad Chromosome.fitness} &\leftarrow \text{FitnessEval(PIDFunc, KP, KI, KD, InputData)} \\
\text{end for} \\
\text{for } i/2 \text{ in Population do} \\
\text{\quad Selection()} \\
\text{\quad Crossover()} \\
\text{\quad if random()} < \text{MutationRate then} \\
\text{\quad \quad Mutation()} \\
\text{\quad end if} \\
\text{end for} \\
\text{Plot()foreachPIDFunc} \\
\text{end while} \\
\text{PlotFitness()foreachPIDFunc}
\end{align*}
\]

To use genetic algorithms to optimize PID parameters specifically, there are customizations that were done. The input data is initialized in the beginning, by choosing the type, amplitude and frequency of the input waveform.

The first step of genetic algorithm is the population initialization. In the initial method, a chromosome is a class that includes PID coefficient and up/down bound of integral term for improved algorithm. Each parameters in a chromosome is initialized in a range in case of any abnormal value.
CHAPTER 4. ALGORITHM COMPARISON AND OPTIMIZATION

Require: $0 < KP < MaxKP, 0 < KI < MaxKI, 0 < KD < MaxKD$

Ensure: Optimized $KP, KI, KD$

for Chromosome in Population do
    Chromosome.$KP \leftarrow$ random$(0, MaxKP)$
    Chromosome.$KI \leftarrow$ random$(0, MaxKI)$
    Chromosome.$KD \leftarrow$ random$(0, MaxKD)$
end for

return Population

Then the program enters the nature selection iteration. The fitness method
that optimizes the PID controller for input data returns the reciprocal of the
standard deviation. Every PID algorithm function is called by the fitness
evaluation function as a parameter.

for i in InputData do
    $Sum \leftarrow Sum + PIDFunc(KP, KI, KD, i)$
end for

Return $1/SquareRoot(Sum)$

Selection, crossover and mutation methods are the same as the definition
of standard genetic algorithm. Selection method selects a pair of parents that
are selected according to their fitness value.

SortAnd Normalize(Population)
Parent1Probability $\leftarrow$ random()
Parent2Probability $\leftarrow$ random()
if ChromosomePair.1.fitness > Parent1Probability and ChromosomePair.2.fitness > Parent2Probability then
    Parents.append(ChromosomePair)
end if

Return Parents

The crossover method exchanges the parent chromosome to make a new off-
spring.

CrossoverProbability $\leftarrow$ random()
if CrossoverProbability > CrossoverRate then
    Return Parent1
end if

ExchangeProbability $\leftarrow$ random()
if ExchangeProbability < 1/7 then
    Return [Parent1.$KP$, Parent1.$KI$, Parent1.$KD$]
end if
if ExchangeProbability < 2/7 then
4.3. COMPARATIVE OPTIMIZATION

Return[Parent2.KP, Parent1.KI, Parent1.KD]
end if
if ExchangeProbability < 3/7 then
  Return[Parent1.KP, Parent2.KI, Parent1.KD]
end if
if ExchangeProbability < 4/7 then
  Return[Parent1.KP, Parent1.KI, Parent2.KD]
end if
if ExchangeProbability < 5/7 then
  Return[Parent2.KP, Parent1.KI, Parent1.KD]
end if
if ExchangeProbability < 6/7 then
  Return[Parent2.KP, Parent2.KI, Parent2.KD]
end if
if ExchangeProbability > 6/7 then
  Return[Parent2.KP, Parent2.KI, Parent2.KD]
end if

Mutation replaces one coefficient of a chromosome with a random generated value.

MutationProbability = random()
if MutationProbability < 1/3 then
  Return[KP + Random(), KI, KD]
end if
if MutationProbability < 2/3 then
  Return[KP, KI + Random(), KD]
end if
if MutationProbability > 2/3 then
  Return[KP, KI, KD + Random()]
end if

From the plot of each iteration it can be seen that parameters are optimized to let the output approach the inject power.
Figure 4.7. The 10, 50 and 200th iterative optimized results

The optimization process can be seen from the fitness graph as well. The fitness graph also shows the performance of each algorithm for comparison.
4.3. COMPARATIVE OPTIMIZATION

Figure 4.8. The average fitness values over time with sine wave input

In case of algorithm might have different performances with specific input data, four types of basic input are taken. These are sinusoidal wave, square wave, triangle wave and random input. The random input is generated via atmosphere noise by importing a library so called true random service. These four types of input are considered covering all possible conditions.
Figure 4.9. The iterative optimized result with sine and square input
4.3. COMPARATIVE OPTIMIZATION

These comparative results show that the ameliorated PID algorithm has the best performance.
4.4 Simulation

4.4.1 Software Simulation

The software simulation is done by using the models and algorithms that were implemented previously using python, combining every module, taking the parameter optimized by genetic algorithm and plotting. The target of the software simulation is to verify the simulated control algorithm functionality. Since we are running at high voltage any unnoticed bugs or improper parameters might result in device damage and there is a risk of fire and explosions. Building a mathematical model of the hardware and algorithm simulation based on this model can help avoiding accidents.

The parameters of the MPPT controller are determined by calculation, while the PID controller gains are obtained using genetic algorithms.

The simulation model is constituted with four parts:

**PV panel model** This model takes operating sample data as input, PV current from MPPT controller as feedback input. It provides voltage to MPPT controller and power to DC Link Model.

**MPPT Controller** Controller The MPPT controller uses the PV Panel operating status to track the MPP position and control the PV Panel with maximum power output.

**DC Link model** The DC Link Model calculates DC Link voltage with input and output power. The output power is the sum of three phases products of grid current and voltage instant value.

**Energy Balance Controller** This Controller uses the sum of feedforward and PID controller to keep DC Link voltage stable.

4.4.2 Hardware Simulation

The hardware simulation is used to test the algorithms translated from Python to C code and implemented on ‘Microblaze’. In this step, the simulation environment is set up. The hardware simulation result is discussed in the results chapter.

The hardware simulation environment is built by using a programmable power supply to be the input of Energy Hub. The test system is connected as the diagram shows below.
4.4. SIMULATION

Figure 4.11. The Hardware Simulation Block Diagram

The programmable power supply can be controlled via serial port, ethernet or custom assembly code storing in internal RAM to generate a specific output. In this case, the ethernet method is used since it is easy to setup and program. The internal program is not used in here because the instructions are limited and unsuitable for performing complex calculations such as simulating a PV panel model. There is also a limitation of the ethernet method, that is the socket communication delay. For the same reason, the output waveform from the power supply can not be sufficiently smooth without adding external capacitors.

The power supply control script is written in a command line style. It is self-documented with comments in the script and a simple manual is given when type in the name of the control script in command line window. It is designed to have two working modes which are a waveform generator and a PV panel simulator. The waveform generator generate sinusoidal wave, square wave, saw wave and constant value. The PV simulator reads current and voltage
values from internal sensors, and send these values into the PV model. Then it calculates the new voltage values and writes back to the power supply. The flow chart of test script is given at below.

**Figure 4.12.** The Hardware Simulation Script Flow Chart
4.5 SUMMARY

The function of test script has to be evaluated before applying on the simulation. It is finished by connecting to a set of power resistors. Switching the connection from serial to parallel makes the resistance change from high to low, and results to the current changing from low to high. Therefore the reading voltage behaviors as expected.

The socket communication takes about 40 ms per time which includes one socket read and one socket write. And also the power supply takes a short period to reach the set voltage level. This delay can be ignored by decreasing the MPPT update frequency to synchronize. It is because the purpose of the hardware simulation is evaluating the implementation and fine tuning. The update frequency can be raised up easily because the MPPT calculation takes only a few milliseconds.

4.5 Summary

This chapter describes the method used in algorithm comparison and optimization. The mathematical model used to provide input data is designed to be close to the real environment. The PV panel model is built to take solar radiation and temperature as inputs, and send the current-power relationship to the MPPT controller. DC link model takes the power increment and calculates the voltage for the energy balance model.

To compare each algorithm, a set of parameters should be chosen to make each algorithm has an average good performance. The MPPT parameters are inferred by analyzing since they are intuitive. The optimization of energy balance parameters is done by using genetic algorithm. Genetic algorithm is a global optimization to generate optimization solutions. The result shows that the incremental conductance for MPPT controller and second improved PID algorithm for energy balance controller have the best performance.

The performance of each algorithm is evaluated both in software and hardware. The software simulation takes the model built previously and modified slightly by giving bipolar voltage input and three phases output. For the hardware simulation, a set of power supplies is used as a solar panel simulator to provide the input data to the device. A program is written for controlling power supplies via ethernet.
Chapter 5

Implementation and Result

5.1 Necessity Of Using RTOS

In this design, each control task has to meet its time requirement. Whether the voltage level is too high or too low it would damage the hardware. Therefore its value must be checked. Each control task has its own time limit or it might lose tracking. In other words, all the control tasks have their own time constraints. The data output task might take a long time if using a slow speed port like UART. When one task is executed, other tasks still need to meet their deadlines.

To meet these time deadlines, RTOS is the solution. The main function of a RTOS is the scheduler, which assigns a priority level and execution time for each task. There are many disciplines for scheduler such as round-robin, rate-monotonic scheduling, preemptive and so on. It uses a hardware timer to generate ‘tick’ interrupts. Every time when the tick timer is triggered, the scheduler of RTOS would check if there are any pending higher priority tasks. When there is another task waiting for being executed, the scheduler will do the context switch which means switch the executing task currently to the pending task.

RTOS also provides other benefits. Many RTOS provide variety optional packages, such as network protocol stacks, file systems, device drivers for building application faster and easier. Some RTOS provide testing tools to provide detail information of how tasks are executing.

On the other hand, using a ‘super loop’ might be small code size, no reliance on third party source code, no RTOS RAM, ROM or processing overhead. But it will be difficult to cater for complex time requirements. It does not scale well without an increase in complexity. And the most important thing
is that timing is hard to evaluate or maintain due to the interdependencies between the different functions.

5.2 RTOS Comparison

There are many choices of RTOS. Deciding which RTOS is suitable for development, they need to be compared in different ways.

- License Property. Although there are many RTOSs are open source and have free licenses, they will charge when they are used in a profitable product or provide more technical support. In this product, the company prefers a free RTOS.

- Context Switching Time. As a RTOS, the context switch happens very frequently. The operating efficiency is highly depended on time cost of context switch.

- Footprint Size. For most of embedded products, the storing space is always restricted. Many RTOSs claim that they are scalable which means developer is able to choose the only part they need for scaling down the footprint. In this report the minimal footprint is chosen to compare.

- Schedule Discipline. Every RTOS has its own policy of scheduling. Some RTOSs have its own specific schedule discipline.

- Additional Function. Multiple interface library and testing tools provide a great help in developing. At least the RTOS must have Xilinx Microblaze port or the developer has to migrate by themselves and it take extra time to finish.

- User community. Owning a large user group means the RTOS is easy to use, smooth learning curve, more documents, variety third party libraries and easy to acquire help. On the contrary, a RTOS might cease maintaining if the user group is small.

The following table shows a list of RTOSs which has Xilinx Microblaze port and their licenses.
### 5.2. RTOS COMPARISON

<table>
<thead>
<tr>
<th>Name</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embox</td>
<td>BSD</td>
</tr>
<tr>
<td>FreeRTOS</td>
<td>GPL</td>
</tr>
<tr>
<td>µCOS</td>
<td>Prorietary</td>
</tr>
<tr>
<td>Linux</td>
<td>GPL2</td>
</tr>
<tr>
<td>ThreadX</td>
<td>Prorietary</td>
</tr>
<tr>
<td>Xilkernel</td>
<td>Open Source</td>
</tr>
</tbody>
</table>

**Table 5.1.** RTOSs and Licenses Compatible with Microblaze

Considering there is no budget for using RTOS, the µCOS ($5k for kernel License fee, and additional for other modules) and ThreadX are excluded. The development team of Embox consists primarily of students of Saint-Petersburg State University. From gathered information, it is mainly used in education rather than product development. Linux can support some real time functions by add real time patches and options during the kernel build. There are several Linux embedded distributions such as µClinux but not RTOS. FreeRTOS has a very large user group since its open source, small kernel, and support variety cores. It still keeps receiving updates with new features such as third party library support of multicore. Xilkernel is an open source kernel shipped with EDK, which supports the core feature required in a lightweight embedded kernel with a POSIX API. But it is not well documented and all technical support is handled by Xilinx.

Many benchmarks compare the performance between each RTOS has been done. A Mälardalens University student did a benchmark with µClinux and Xilkernel in his thesis. [13] The result shows that µClinux minimal kernel size is more than 1MB while Xilkernel is only 20kB large. For interrupt latency uClinux takes 10 times ticks than Xilkernel. Another student did his thesis in TU Delft compare Xilkernel and FreeRTOS on Multicore processors. [4] In his conclusion FreeRTOS is slightly faster and its code size is larger than Xilkernel. Xilkernel and µCOS is compared in a paper post in IEEE conference.[15] It presents that Xilkernel has 10 times faster context switch time and slightly larger code size than µCOS. A master thesis gives many detail benchmark of FreeRTOS using specific hardware task such as Uart, I2C etc.[16]

The comparison mainly happens between FreeRTOS and Xilkernel. Because both of them are free, small footprint and short context switching time. The result of memory footprint is shown in the diagram below. In this diagram, TEXT segment is used for storing program code. DATA and BSS(Block Started by Symbol) are used for storing uninitialized global and static variables.
CHAPTER 5. IMPLEMENTATION AND RESULT

The project name ending with small_core generates the board specific platform project in XPS uses a minimized core which only contains a 64kB BRAM. The custom FreeRTOS project removes semaphore and mutex support in the FreeRTOS kernel. And the constrained FreeRTOS project reduces the size of stack and heap.

<table>
<thead>
<tr>
<th>text</th>
<th>data</th>
<th>bss</th>
<th>dec</th>
<th>hex</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>20950</td>
<td>432</td>
<td>19216</td>
<td>40598</td>
<td>9e96</td>
<td>xilkernel_demo_0.elf</td>
</tr>
<tr>
<td>20698</td>
<td>416</td>
<td>19214</td>
<td>40328</td>
<td>9d88</td>
<td>xilkernel_demo_small_core.elf</td>
</tr>
<tr>
<td>72476</td>
<td>1604</td>
<td>82716</td>
<td>156796</td>
<td>2647c</td>
<td>freertos_hello_world.elf</td>
</tr>
<tr>
<td>29800</td>
<td>1480</td>
<td>82656</td>
<td>113936</td>
<td>1bd10</td>
<td>freertos_hello_world_custom.elf</td>
</tr>
<tr>
<td>31080</td>
<td>1484</td>
<td>4876</td>
<td>37440</td>
<td>9240</td>
<td>freertos_sem_test_constrained.elf</td>
</tr>
</tbody>
</table>

Table 5.2. Footprint Size Of XilKernel And FreeRTOS In Different Settings

From the diagram, it is obvious that the size of target file is slightly affected by removing IP devices from the core. Costumed RTOS kernel may result a significant effect to the size of code segment. The actual memory footprint of FreeRTOS can be done by checking map file. In this case, the FreeRTOS kernel memory footprint is 13716B.

The BSS segment is actually combined with three segments, BSS, stack and heap. For instance, the BSS segment in FreeRTOS custom kernel is 82,656 bytes. From the ELF file it is easy to see that the sum of BSS, stack and heap is \(0 \times \text{2c0} + 0 \times \text{a000} + 0 \times \text{a000} = 0 \times \text{142c0} = 82656\). Since every task must have its own stack, the best solution to constrain the stack size is reducing the number of tasks.

Through the comparison above, the sizes of these two RTOS are very close. Both of these two RTOS are allowed to use in commercial usage without any extra licensing fees and both of them provide technical support. While Xilkernel is integrated with Xilinx SDK, FreeRTOS can be also easily imported into SDK software. FreeRTOS is much easier to customize or debug because it is open source. FreeRTOS is more activated than Xilkernel. The latest edition v7.5.2 of FreeRTOS is released in Jul 24 2013. Xilkernel does not have an official site. The latest release of Xilkernel that can be found is 5.01.a updated in Oct 19 2011. FreeRTOS has more documents and discussion than Xilkernel. It is possible to acquire support from mailing list, sourceforge and IRC. Since FreeRTOS supports variety hardware, if there would be any product upgrade, the code with FreeRTOS can be migrated to another hardware platform easily.

From the comparison above, FreeRTOS is chosen for this project.
5.3. TASKS ASSIGNMENT

5.3 Tasks Assignment

The main function is used to initialize the whole system. It will setup all connected hardware and the operating system. The system itself also takes the responsibility for setting several hooks to exceptions such as malloc fail, stack overflow and watchdog timer.

![Main function Flow Chart](image)

Figure 5.1. Main function Flow Chart

Tasks are created with previous specification. There are several tasks sharing the same priority since the requirement of decoupling. It could be merged if the ram space is in tense. In the design of FreeRTOS, each task must take its own stack which has a minimum stack size is 120 bytes. Therefore, task merge can save footprint size and decrease context switch as well.
CHAPTER 5. IMPLEMENTATION AND RESULT

<table>
<thead>
<tr>
<th>Priority</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>UpdateAndProtectTask</td>
</tr>
<tr>
<td>3</td>
<td>EnergyBalanceTask, MpptTask</td>
</tr>
<tr>
<td>2</td>
<td>UartTask, CanTask</td>
</tr>
<tr>
<td>1</td>
<td>DisplayTask</td>
</tr>
<tr>
<td>0</td>
<td>IdleTask</td>
</tr>
</tbody>
</table>

Table 5.3. Tasks And Priorities

UpdateAndProtectTask takes the highest priority and the highest executing frequency. It has two main functions, interface board accessing and fault protection. When it is executing it reads all values from external components such as voltage and current sensors of each modules. Then it detects if there is any fault happened. For instance, if the voltage exceed the safety voltage level it would suspend the whole device and send a message to the data output processor warning about the type of happened exception.

![Update and Protect Task Flow Chart](image)

Figure 5.2. Update and Protect Task Flow Chart

EnergyBalanceTask maintains the energy transfer operation. It combines the
functions of energy balance, symmetric and precharge controller. Actually, the precharge controller is trimmed since the PID controller already cover its function. Symmetric controller is merged since its control the same resource. This task works as the flow chart showing below.

At first EnergyBalanceTask initializes all parameters and variables and enable power modules used for energy transfer. Then it suspends itself to let other tasks execute until it is called to start. If it is called then it will start its main while loop. In the beginning of this loop, it will detect is there any messages such as emergency issue or stop manually. If there was an exception, it would quit the main loop and go back to suspend. If there is no stop message it will come to the second condition of whether there is a new power setpoint. This condition is related the energy flows inside. Since whenever the energy flows in is set to a new power level, it should raise up or set down with a gradient curve to trim any glitches and also help to improve the performance of the PID controller. The symmetric controller is combined into the fix power control part since they are actually controlling a same group of current.

After the function received power control is called, the send power control function is executed. This is the energy balance controller that equalizes the input and output power flow levels and stabilizes the DC link voltage. The sum of a feedforward calculation is used for rough tuning and PID controller for fine tuning, and then it goes back to the start of while loop which is the restart detection function. The precharge controller is removed since it can be done by optimizing the energy balance algorithm.
The MPPT task controls the power output from solar panel. It is used to keep the solar panel’s power stay at the highest value regardless of weather conditions. The control flow is similar to energy transfer task.

UART function is separated into a single task because the decoupling requirement. It uses the interrupt for receive command to decrease execution time.

The message received from interrupt subroutine transfer the message to UART function with message queue method provided by the operating system. After the command is received, it is parsed and then sent to the destination module to complete the target of command. The CAN bus function is similar with UART task. The flowchart is shown at below.
5.4. TRACING ANALYZE

The display task is used to print messages to the user interface. This task has two options. In development mode procedure it prints out operating condition via serial port and in production mode it shows all values on an external display screen. Since print executing time can be a very long time, this task updates every 5 seconds.

The Idle task is set up by the operating system. It is completely empty by default. It is because its function is to denote that the scheduler is not full in a specific period for debugging. If it is executing that means the scheduler is not full and can be occupied by any other tasks or the task delay can be set to less to increase its frequency.

5.4 Tracing Analyze

To ease development and testing, FreeRTOS has a trace library that can record the conditions of each tasks during debug mode. To get more detailed information such as Gnatt chart of task operating in a period, a company in Västrås called Percepio developed a tracing tool for RTOS.

At first, the tracing library needs to be added into the software program.
CHAPTER 5. IMPLEMENTATION AND RESULT

project and then it is set to record tasks, semaphores and similar objects. For some processors, their debuggers have specific functions that can be called during debugging. Their software supports directly imported record data from the debugger. However, in this case, the debugger and the development environment do not have this function. The record data is stored in a data structure. To acquire the data, GDB can dump a piece of content out of the memory and store it locally. Then the tracing software will analyze this memory data and generate a chart automatically.

![Gantt Chart of Program Operate in First 100ms](image)

**Figure 5.5.** Gantt Chart of Program Operate in First 100ms

From this chart it can be seen that every task meets its deadline and display task is always interrupted by the other three tasks.

5.5 Result and Conclusion

After the thesis work the operating system is deployed and control algorithms are implemented. This chapter is about the result with software and hardware simulation, then analysis.

The result of software simulation shows that chosen algorithm and parameters of controller gives an acceptable output. The MPPT controller can track MPP drifting. The energy balance controller keeps DC Link voltage in a stabilized level.
5.5. RESULT AND CONCLUSION

Figure 5.6. The Simulation result of 1 phase grid voltage

Figure 5.7. The Simulation result of 3 phases grid voltage

The graphs below is the result hardware simulation of Energy Control with 200V DC input voltage and transferring 700W and 1000W power separately. The red curve is the current of fix DC power supply. The black one is the current of controlling ac current. And the purple and yellow curves are positive and negative DC link voltage separately.
The DC link is charged smoothly as shown in figure 5.6. This is the result of optimizing the PID algorithm and adding a limit to the integrator factor.
5.5. RESULT AND CONCLUSION

Since in the precharge period the actual voltage is far from the set point, it generates a large error. If the integrator factor does not have any limitation it will saturate rather fast and generate a large peak when the DC link voltage is close to the setpoint. This implementation result is similar to the simulated results.

The power supply is constant in this case, therefore the result of feed forward is proportional of input current. The PID controller trimmed the peak after DC link precharge successfully. After adding the PID controller, the output has added noise. One possible reason is that the input current values of the energy controller are calculated value rather than measured. The current sensor is not in the thesis scope, and needs further calibration. A finer tuning of the current sensor will significantly reduce the noise in the output of the system.

The table below gives parameters and standard deviations of the control values between the simulation model and the hardware test.

<table>
<thead>
<tr>
<th></th>
<th>KP</th>
<th>KI</th>
<th>KD</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>simulation</td>
<td>0.27</td>
<td>0.12</td>
<td>0.0</td>
<td>1.039</td>
</tr>
<tr>
<td>hardware</td>
<td>0.04</td>
<td>0.003</td>
<td>0.0</td>
<td>1.06</td>
</tr>
</tbody>
</table>

Table 5.4. PID Parameters Using In Simulation And Hardware Test

Although noise is considered in the simulation model, the parameters still have large variance. It has been detected that there is an error with the current sensors and the error is non linear. The parameter optimization results of the mathematical model can not be used on the hardware test directly and need further adjustments.

According to the results above it can be concluded that, the simulation model can be used to compare between each algorithm. It is helpful for theoretical analysis and identification of the best performing algorithm. This method is general and effective in the case of small amount parameters such as PID controller. However, using genetic algorithm to optimize parameters can be only used in algorithm selection. Since the simulation is built based on an ideal mathematical model, unpredictable differences between the ideal model and the real hardware are existed. Moreover, in a real hardware these parameters need to be fine tuned due to the differences in each set of hardware.
5.6 Summary

Controllers are implemented on the hardware using a real time operating system. The reason of using a real time operating system is the processor is required to handle simultaneous time constrained tasks. The operating system is decided according to cost, performance and available support. Tasks are assigned to four priority levels according to their usage. Tasks that have similar usage are merged to save memory resources, while Interface communication tasks are separated for decoupling. The data exchange between each task uses semaphores and message queues to avoid deadlock. Task assignment and context switch are verified using tracing software. The result of software simulation shows a good performance. The hardware result is not working as expected at high power conditions since the measurement equipment has not been carefully calibrated and the parameters need to be fine tuned.
Chapter 6
Summary and Future Work

The planned tasks for this thesis work were completed. First, the Energy Hub equipment are organized as control modules and hardware interfaces. After that, many control algorithms for the MPPT and energy control modules have been analyzed theoretically and also for the simulation point of view. A software mathematical model and hardware simulation script were designed for system simulation. With the software simulation, algorithms are compared and chosen to use the maximum fitness method and genetic algorithm. For the function requirements, a real time operating system is selected for product development. A suitable operating system is selected considering several reasons. The control algorithm is implemented as multitasks on the operating system. Part of them is evaluated and simulated with the programmable power supply and remote test script.

In the study, several points have been proved as listed below.

- A general method of using genetic algorithm to compare and optimize control algorithms is of great help in the prototype development stage.

- Software simulation has detected unexpected bugs in design. The hardware test has verified the algorithm comparison.

- Real time operating system is suitable for energy control system application.

When input power is more than 1kW the operating result could exceed the safety limit. To solve this the PID gains need to be fine tuned more precisely and decrease the measuring errors. Another solution could be implemented using more advanced control algorithms such as fuzzy logic or particle swarm
optimization. Since the simulation model has already been built, performance evaluation of any new algorithms becomes much easier.

The development of this prototype has not finished. There are several operating modes left. The previous test used a power supply as a PV simulator. It will test on real solar panels after the performance is improved. To finish this prototype a user interface development needs to be designed and implemented.
Bibliography


