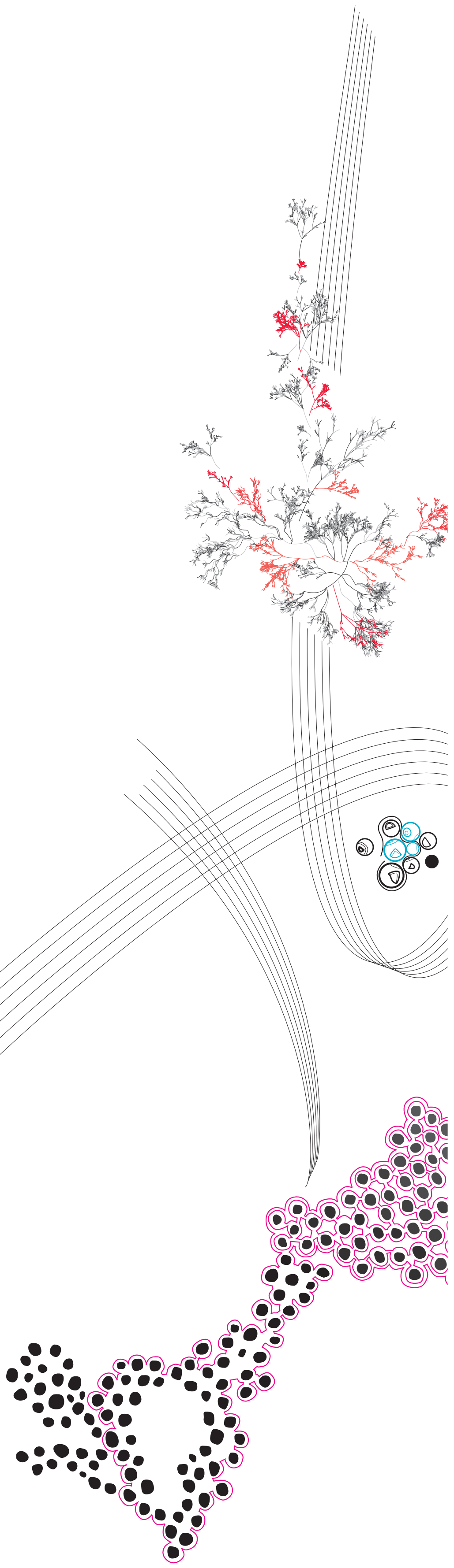
BSc Thesis Electrical Engineering



Design of a Fast Control System Reacting to Non-sinusoidal Voltage Fluctuations

Bram Bremer

Supervisors: A. Matthee, R.A. Vogt-Ardatjew

July, 2022

Power Electronics & EMC
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Summary

Inrush currents can cause significant issues in weak grids. Therefore a concept device has been created that can actively compensate for these peak currents. In this report a parameterisable control system for this concept device is developed. A simulink model has been created and using the Simulink extension HDL Coder, converted to synthesisable HDL Code. With the use of Xilinx Vivado this model is programmed onto a FPGA. The model was verified to work correctly in simulation as well as on the FPGA. Additional work is needed to meet the timing constraints in Vivado as well as to integrate the model with the other subsystems.

Contents

Chapter 1

Introduction

1.1 Problem statement

Microgrids that function in island-mode most often have limited power capabilities. Due to this, these weak grids are susceptible to short transient load peaks. In commonly used DC-AC converters, the inverters rely on their filter and control system to respond quickly to a sudden peak load. For both inverters and conventional rotating generators, these peaks can exceed the units capability. This can activate the protection system of the generator, shutting it down and causing instability within the microgrid. Even if the generator is not shut down, severe voltage sags and swells can occur, possibly damaging internal components and connected equipment.

Light emitting diode (LED) drivers are for example notorious for creating large inrush currents, as can be seen in figure 1.1. Large sets of LEDs are often controlled using a single switch and will thus add up their inrush current when started. For microgrids this inrush current can cause significant issues, emphasising the importance of mitigating these currents.

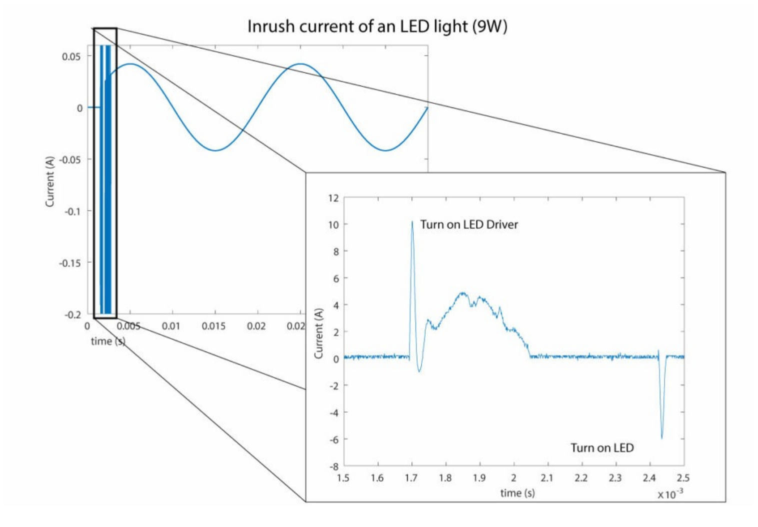


FIGURE 1.1: An example of inrush currents in LED lighting [1] [2]

1.2 Current state of research

From earlier research, it was found that inrush currents can have significant effect on the stability of microgrids [3]. A traditional approach is oversizing the generator/inverter to cope with the peak currents. This solution is not cost-effective and also increases the weight, size and maintenance costs [3].

A possible solution to this problem is explored by H. Huang et al. [4]. They designed an active current eliminating device built using modular multilevel cascaded converters (MMCC). From their simulations they showed that using MMCC's in combination with a prediction based control system can significantly reduce harmonic currents. However, in contrast to harmonic currents, inrush currents are not predictable. Therefore the implemented prediction controller is not suitable for inrush current mitigation.

Another approach has been taken in the paper "Active Transient EMI Stabilization" by B. Ihsan et al. [5]. In this paper a concept device was designed specifically aimed for transient current peaks. Using simulations they showed that the device can significantly reduce the stress on the main inverter and improve the stability of the microgrid in situations with large inrush currents or other transient peaks.

1.3 Design approach

In the current project the device that was explored by B. Ihsan et al. is further developed. This paper is specifically focused on a single subsystem of this device; the control system. In general, the stabilization device (SD) that is to be designed in this project should allow for quick stabilization of the grid voltage. The SD focuses on the mitigation of high frequency and short voltage sags by injecting short and timed current pulses. The system is divided into several subsystems as shown in figure 1.2. The Grid Voltage Tracker (GVT) subsystem measures the grid voltage and using a phase-locked loop (PLL) provides a clean in-phase reference sinusoid that is used in the controller. The controller generates a signal that drives the H-Bridge driver. The signal is used in addition to a sinusoidal pulse width modulated (SPWM) engine, generating an in-phase sine on the output of the H-Bridge. A low-pass filter is used after the H-Bridge to attenuate the switching frequency.

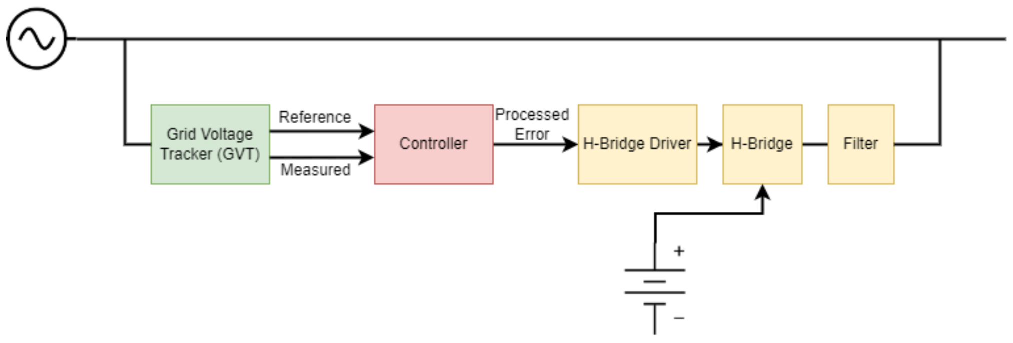


FIGURE 1.2: An overview of the complete system

Chapter 2

Design Aspects & Tools

In this chapter the system-level requirements will be discussed. With these requirements in mind, the general design method and associated tools are determined. Background is given on the involved tools and how these are used during the design of the controller.

2.1 Requirements

The controller has several requirements that it should fulfill. These requirements are derived from the main objective of the device and from the connected subsystems; the PWM generator and the GVT.

- **Low latency**

  With a focus on mitigating short-duration current spikes and high-frequency components, it is important to have minimal latency in the control loop. For this reason, the system was chosen to be implemented on a field-programmable gate array (FPGA). In comparison to microprocessors, FPGA's have the advantage that they can be directly programmed on a hardware level. By programming the internal connections between different hardware blocks, e.g. lookup tables, lower latencies can be achieved than would be possible using a general purpose embedded microcontroller.

- **Low frequency rejection**

  The device is not intended to deliver power at the fundamental frequency of the grid. For this reason it is important to ensure that the device only responds to high frequency and short-term current spikes. Although subtracting the PLL reference voltage from the measured voltage removes a significant part of the fundamental frequency, further filtering is necessary to remove all low frequency components. As the device is still under development, there is no strict definition on the cutoff frequency. For this reason, a flexible system is to be designed that is easily configurable. For the design of the control system, a cutoff frequency of 100 Hz was used. This is suitable as the frequency components of inrush current or fast transient events are in the range of 1 - 10 kHz.

- **PWM & sampling frequency**

  The PWM frequency that the inverter should be driven by is not defined yet. However, the maximum frequency that the controller has to support was decided to be 1 MHz . During simulation and verification, a frequency of 10 kHz was used. This is however easily adjusted as similar to the cutoff frequency, the PWM frequency is parameterised.

The maximum effective frequency at which the control loop can operate is mainly determined by the grid voltage tracker subsystem. The built-in Analog to Digital converter (ADC) of the used FPGA has a sampling frequency of 1 MS/s [6]. For minimal latency, this sampling frequency is maintained throughout the design of the control system.

2.2 Tooling

2.2.1 Hardware

During the design of the controller the CMOD A7 FPGA board has been used. This board is equipped with a XC7A35T FPGA, a 1 MS/s ADC and a 12 MHz oscillator [6]. We chose to use this board as it was readily available, compact and also contains a builtin 1 MS/s ADC. The XC7A35T FPGA should also contains sufficient logic blocks to fit the complete design of the system.

2.2.2 Software workflow overview

For designing control systems, Simulink is a useful tool. It contains many drag & drop blocks making it easy to quickly create a control system and simulate the design. The project consisted of several sub-packages completed by different individuals. The modular work-packages created by each project member was planned to be integrated into one functional system. Therefore Simulink was chosen for the high-level design, ease of integration and simulation effectiveness. In order to use the designed model on the FPGA, several Matlab extensions are required. These include HDL Coder, HDL Verifier and their dependencies. With the HDL Workflow advisor that is part of the HDL Coder extension it is possible to progressively convert the Simulink model to VHDL code and insert this into a Vivado project. After this conversion, the Vivado project can be opened and the bitstream generation can be performed. In the following sections, this workflow is explained in greater detail.

2.2.3 Simulink modelling

Before the model is designed, it is useful to use the command 'hdllib'. This limits the blocks shown in the Simulink library to only those that are compatible with HDL coder and synthesisable on the FPGA. The Simulink model is created by placing blocks and connecting these with virtual wires. In order to use the design in the next step, the system is placed in a subsystem block. For every input/output (I/O) the final system on the FPGA should have, the subsystem should have a port. These ports should be linked inside the subsystem to the different blocks of the controller. By defining these ports, HDL Coder is able to define the necessary I/O pins for VHDL generation. For efficient implementation of the design on a FPGA, it is necessary to make the Simulink model synchronous. Even though asynchronous designs are possible on FPGA's, this is not advised [7].

2.2.4 Vivado

In order to use HDL Coder to generate a Vivado project, a reference board design has to be created. This consists of a Vivado 'Block Design' and defines the different components that should be used in the FPGA and their external connections. These components are called IP Cores and can be drag & dropped into the design. IP-Cores are used to create clock lines, (a)synchronous resets, AXI-busses and other interconnecting signals. As the FPGA is highly configurable, care must be taken when setting up the design. The settings in the different blocks allow for a lot of customization and are therefore difficult to configure.

2.2.5 HDL Coder

Before continuing with the workflow, it is important to have registered the FPGA board and the reference design files in Matlab. This ensures that HDL Coder recognizes the target FPGA board. This procedure can be relatively time-consuming. The details of this process are not further explained in this report, but can be found online [8]. With the Simulink model completed, it is necessary to check the HDL compatibility using the 'check subsystem compatibility' option. This is to make sure that the blocks used in the model are compilable and synthesisable on the FPGA. After the compatibility has been verified, HDL Coder's ' HDL Workflow Advisor' can be used. This is a step-by-step process that guides the user through different settings and configurations. During this process the user can specify where the I/O ports created in the Simulink model should connect to on the FPGA. The process generates an IP Core and places this core in the reference design. The result is saved in a Vivado project. The workflow also allows automated bitstream generation by calling Vivado in the background. If more configuration and customization is required, the generated project can be opened in Vivado. After the bitstream is generated, the FPGA board can be programmed. This can either be done from Vivado or the HDL Workflow Advisor.

2.3 Fixed-Point Designer

Even though HDL Coder and the FPGA have support for native floating point arithmetic, certain Simulink blocks that are used in this design are not compatible [9]. Therefore, the floating point arithmetic should be converted to fixed-point. Fixed-point notation is useful to represent fractional numbers with fixed precision, allowing the use of efficient and fast integer arithmetic. In Simulink it is possible to design and simulate systems with fixed-point arithmetic. As converting floating point to fixed-point manually can be difficult and time consuming, a Matlab tool is available that helps the user through this process. This tool, called 'Fixed-Point Tool', is part of the Fixed-Point Designer Matlab extension. In this design this tool is used in the fixed-point conversion of the filter, as this allows for optimizations by reducing the number of bits. The tool calculates the value range of every signal and in combination with user-specified signal specific precision, determines the optimal word & fraction length.

Chapter 3

Design & Implementation

The following chapter further describes the design, implementation and sub-package integration process. Design considerations and trade-offs are discussed in detail.

3.1 Simulink model design

With the requirements defined, it is possible to create a Simulink model of the controller. In figure 3.1 an overview is given of the designed model. The controller can be subdivided into several segments, which are explained in the sections below.

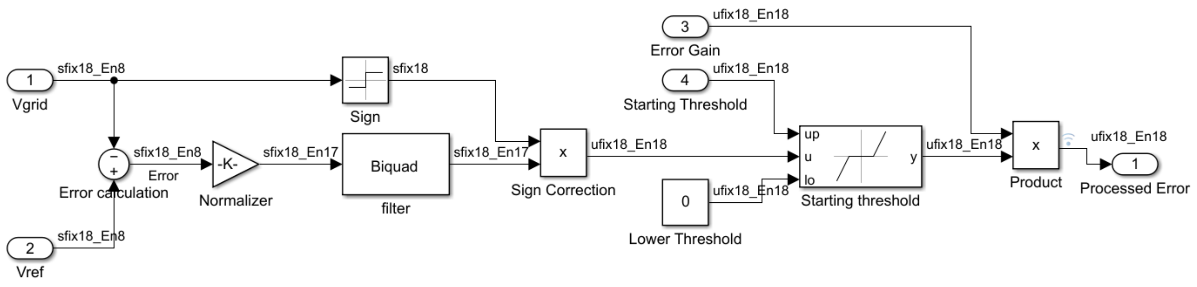


FIGURE 3.1: The designed simulink model

3.1.1 Error calculation

The difference between the 'clean' sinusoidal signal from the GVT subsystem and the measured grid voltage creates an error which can be used in further processing. In an ideal case, the PLL inside the GVT is able to track the grid voltage perfectly, resulting in an error signal that does not contain the grid frequency. In practice however, this is not achievable.

3.1.2 Normalization

When assuming the input signals from the grid voltage tracker are in Volts, the input is scaled down to ensure that the signal fits within the duty cycle range of 0 to 1.

3.1.3 Filtering

As explained before, low frequency components need to be removed from the signal. This is achieved by using a high-pass (HP) filter. Digital filters can be implemented using different structures but are mainly categorized by the presence of recursion. Infinite impulse response (IIR) filters contain feedback paths inside the filter while finite impulse response

(FIR) filters do not. In this design, it was chosen to use an IIR filter. These filters have the advantage of achieving the same filtering performance as FIR filters with a lower order. This is important for minimizing the group delay (latency) but also for implementation on a FPGA. Using a lower order filter reduces the number of taps and therefore also the hardware needed. One of the disadvantages of an IIR filter is the difficulty in designing a stable filter. Due to the feedback inside the filter, it is not inherently stable such as FIR filters.

Filter type

There exists many types of filters. In this design, an inverted Chebyshev filter was chosen. This filter has a relatively steep roll-off and will therefore result in a lower order filter than would be possible with for example a Butterworth filter. Chebyshev filters do cause ripple in the stopband but in this case this does not have a significant negative impact. The error signal that is filtered contains a measurable but not significant grid frequency component, therefore the filter only requires light attenuation in the stopband.

Filter Designer tool

The filter is designed using the 'Filter Designer' tool in Simulink. With this tool a minimum order Chebyshev IIR filter can be designed. For minimal group delay, the transition band is chosen relatively high. The bandstop and bandpass frequencies are 80Hz and 200Hz respectively. This ensures that the 50Hz component is removed completely. The stopband attenuation is set to 40dB, but this requires tuning when tests can be performed with the integrated system.

With these requirements a minimum order filter with an order of 4 was designed. The magnitude response and group delay plots are shown in figure 3.3 and figure 3.4 respectively. In these figures the frequency range has been reduced to 0 to 500Hz to show the transition band more clearly. For higher frequencies the attenuation is almost 0dB while the group delay asymptotically approaches 0ms.

The filter is implemented as a direct form 2 biquad filter. Higher order IIR filters are efficiently implemented using several cascaded second order filters. This reduces the sensitivity to coefficient quantization instability [10]. The structure of a direct form 2 biquad filter is shown in figure 3.2.

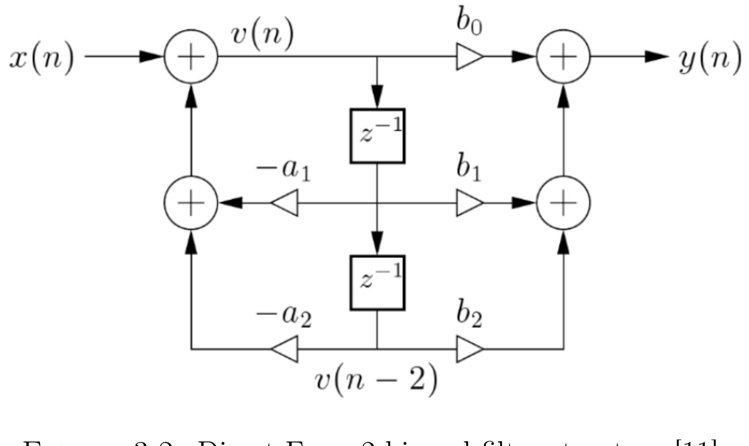


FIGURE 3.2: Direct Form 2 biquad filter structure [11]

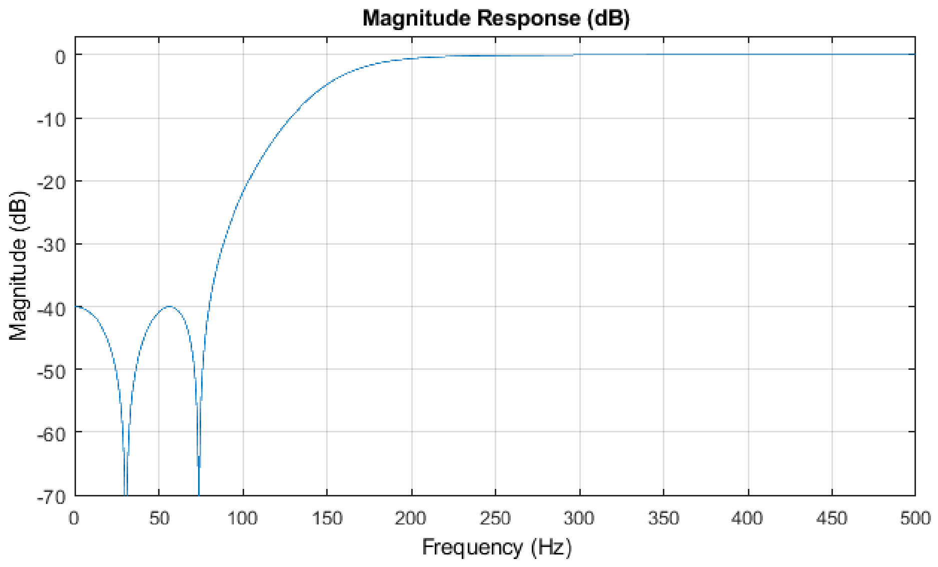


FIGURE 3.3: Magnitude response of the designed filter

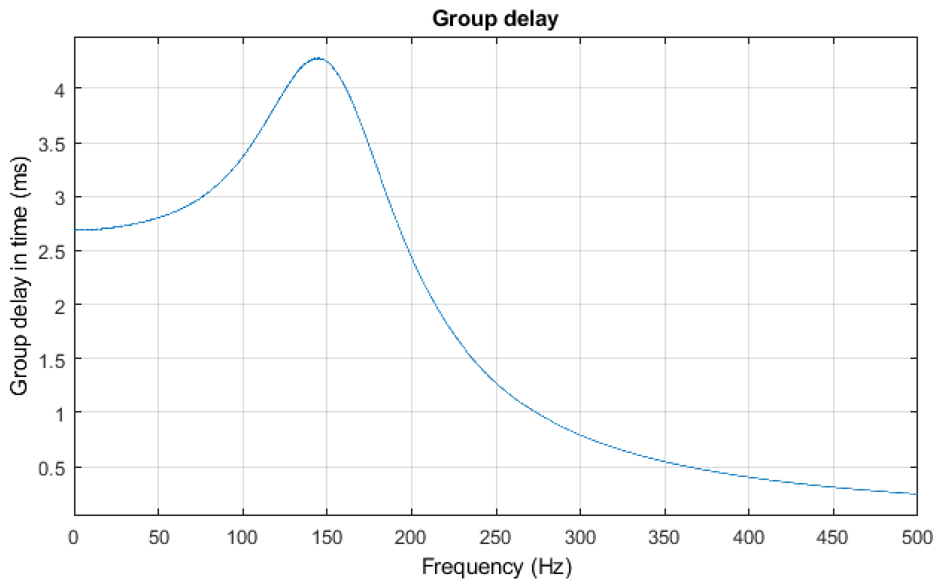


FIGURE 3.4: Group delay of the designed filter

3.1.4 Sign correction

The error signal that has been calculated is based on the difference between the PLL reference and the measured voltage. In the positive half cycle, the subtraction of the measured voltage from the reference voltage means that a positive signal respresents a voltage sag. During the negative half cycle this is reversed; a voltage sag is represented by a negative error. To correct for this, the filtered signal is multiplied by the sign of the measured voltage. By multiplying the signal after it is filtered, discontinuities caused by this correction do not affect the filtering.

3.1.5 Further signal processing

To ensure that the signal is bounded between 0 and 1, the Simulink block 'Saturation' is used. After this a 'Dead Zone' block is applied. This acts as a threshold to ensure that the inverter is only enabled for significant voltage sags. The last step consists of a gain block. This determines the gain applied to the control signal. Both the gain and the threshold are parameterized to allow for adjustment and tuning later on.

The functional model is now completed. Verification was performed to ensure that the model behaves as expected. The verification method is further explained in section 3.1.7.

3.1.6 Fixed-point conversion

Before implementing the design on a FPGA it is necessary to convert the floating point arithmetic to fixed-point.

Data types & ranges

In order to convert the model to fixed-point arithmetic, it is necessary to determine the signal value ranges. In general, the design consists of 3 different signal types:

- **Voltages**
  The input signals from the GVT subsystem are the grid voltage and reference voltage, V_{meas} and V_{ref} respectively, and are assumed to be in Volts. As this device is designed for low-voltage applications, a range of 0 to 230Vrms is to be expected. In order to cover voltage overshoots, a 40% margin is applied. This results in a maximum amplitude of $230 \times \sqrt{2} \times 1.4 \approx 456$. To be able to represent this in binary fixed-point notation, a minimum of $\lceil \log_2(456) \rceil = 9$ bits are required for the notation of the integer part. The voltage can either be positive or negative, thus signed notation is required. This results in a minimum of 10 bits needed for complete representation. Any additional bits improve the precision by adding to the fractional length.

- **Normalized voltages**
  For normalized voltages it is assumed the signal is within -1 to 1. This requires no dedicated bits for representing the integer part. However, 1 bit is still required for signed notation. To avoid loss of precision, the total word length of this data type is chosen to be equal to the voltage data type.

- **Duty Cycle**
  In this design the duty cycle is represented with a range from 0 to 1. To represent this in binary fixed-point notation, all bits can be dedicated to the fractional part. Following the same reasoning as mentioned above, the total word length is kept equal to the other data types.

The total word length was decided to be 18 bits. This was chosen for two reason: The digital signal processing (DSP) slices on the Artix 7 FPGA's have dedicated 18x25 bit multipliers and can therefore be efficiently utilized [12]. Also, the ADC that is used in the grid voltage tracker subsystem has a maximum resolution of 12 bits [6]. By using 18 bits the full precision of the ADC is utilized while also leaving headroom for the use of higher-precision ADC's. The final fixed-point data types are shown in table 3.1.

Number of bits per data type	Wordlength	Fraction	Sign
Voltages	18	8	1
Normalized Voltages	18	17	1
Duty Cycle	18	18	0

TABLE 3.1: The three different fixed-point data types that were chosen

Filter

As was mentioned before, stable IIR filters can be difficult to design. To assess the stability of the designed filter one can use a pole-zero plot. Poles and zeros are characteristics of the filter's transfer function (TF) and indicate when the TF converges to infinity and zero respectively. When one of the poles has a magnitude of more than 1 (i.e. outside the unit circle) the impulse response of the filter diverges, resulting in an unstable system. From figure 3.5 it can be seen that the poles are located inside but very close to the unit circle. When representing these poles in fixed-point notation, it is important that there are no significant quantization errors. This could cause the poles to move outside the unit circle, destabilizing the filter. The behaviour of the filter can also change significantly, therefore verification must be performed to ensure that the filter behaves as expected after fixed-point conversion.

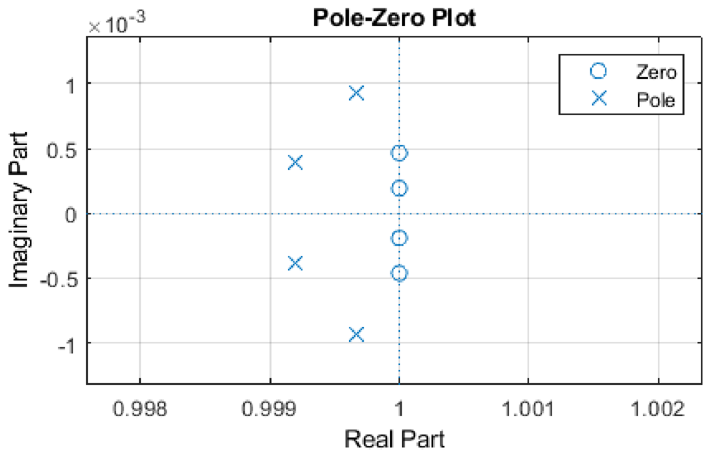


FIGURE 3.5: Pole-zero plot of the designed filter

As there are many different components to the filter, fixed-point conversion can be difficult. To start, a high number of bits was chosen. This ensured that the filter was stable and operated as normal. Afterwards, optimizations are performed to reduce the number of bits without compromising the behaviour. By decreasing the number of bits less hardware is needed and therefore higher operating frequencies can be achieved. Saving hardware might also be necessary in order to fit the integrated system onto the FPGA.

An initial word length of 96 bits was chosen for the coefficient representation. For the I/O of the filter the Normalized voltage data type was chosen, as this is the data type that the filter should operate with. Internal filter components such as the accumulators and multipliers are set to a word length of 64 bits and a fraction length of 32 bits. By comparing the magnitude and group-delay plots with the floating point based filter, it was observed that the deviation is insignificant.

In order to reduce the number of bits the Fixed-Point Tool (FPT) is used. In this tool, a tolerance was specified which determines the maximum signal deviation the optimized filter can have with respect to the initial filter. This tolerance was set to 10^{-4}. After the tool was run, several improvements were suggested. The word length reduction suggested by FPT can be seen in table 3.2. These optimization were applied and a verification run was done to ensure correct behaviour. It was found that the suggested adjustments were causing the filter to behave significantly different and incorrect. This is possibly caused by the lower word length of the coefficients causing significant quantization errors, but this is not further investigated in this report.

For the final design, the word and fraction length of the filter components were chosen as the initial verified fixed-point values. These values can be seen in table 3.2.

Number of bits per data type	Word length	Fraction	Sign	FPT word length difference
Coefficients	96	94	1	-2
Accumulators	64	32	1	-12
Multipliers	64	32	1	-12
States	64	32	1	-13
I/O	18	17	1	0 (locked)

Table 3.2: Filter Fixed-point conversion word and fraction lengths

3.1.7 Simulation

In order to verify the behaviour of the designed model, we used simulations. This is achieved by applying realistic input signals to the model and observing the output. Data has been provided in which measurements have been performed on voltage sags during inrush currents. As there is no model of the voltage tracker subsystem available, the reference voltage was obtained by using a Simulink PLL block. Additional scaling was applied to the output of the PLL to achieve similar amplitude to the measured voltage. The Simulink simulation model is visible in figure 3.6.

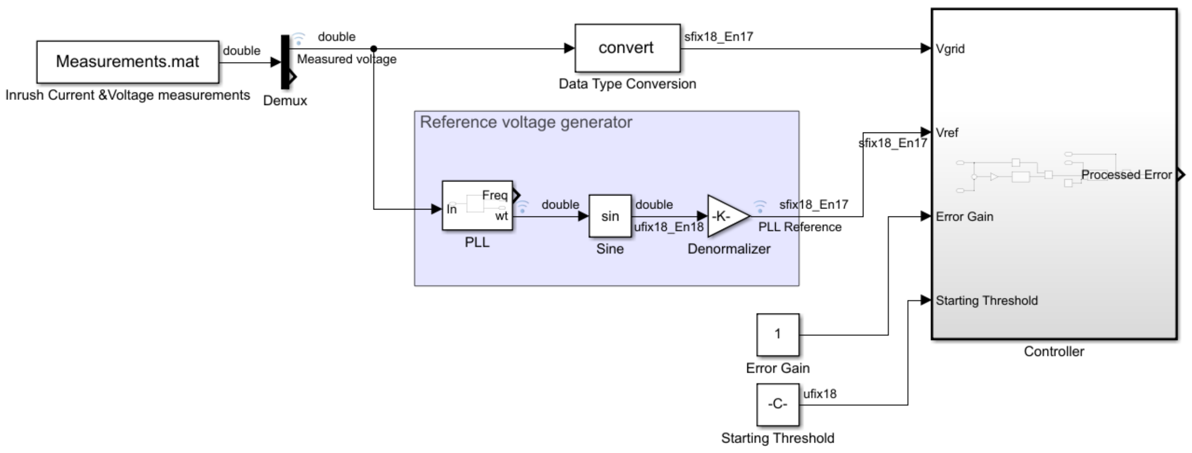


Figure 3.6: Simulink simulation model used for the verification of the design

3.2 Vivado reference board design

With the Simulink model finished, we can proceed with the design by creating a reference board design in Vivado. As explained in section 2.2.4, this reference board design is necessary for creating a Vivado project using HDL Coder.

The created model is based on an existing design developed by a project team member. From earlier work this reference design was verified to perform well. This design is shown in figure 3.7.

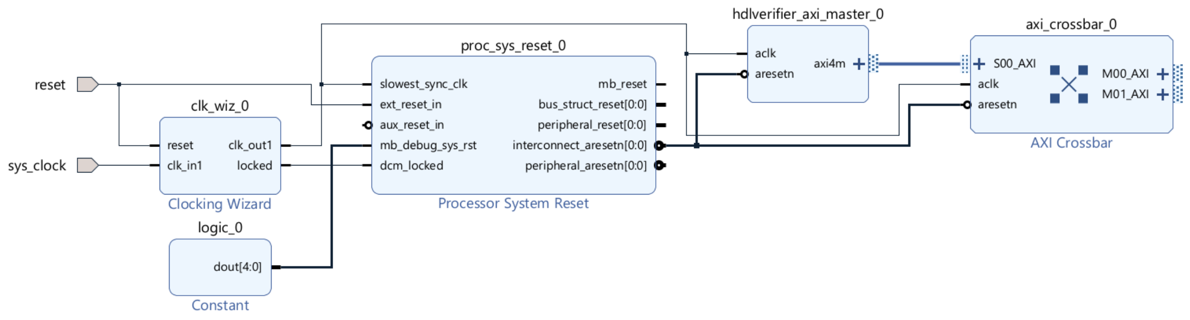


Figure 3.7: Provided Vivado reference board design

The design consists of three parts: clock generation, reset managing and Matlab interfacing.

Clock generation

The CMOD A7 FPGA Board that is used in this project provides a single clock of 12 MHz that is driven by an internal oscillator [6]. For lower latency and a better resolution, we chose a main clock frequency of 100MHz. This provides 100 clock cycles in between the PWM pulses (at 1MHz), resulting in a duty cycle resolution of 1%. To generate arbitrary clocks in Vivado, the IP Core 'Clocking Wizard' is provided. We can configure this block to generate a single 100MHz clock from the 12MHz clock signal.

Reset managing

The board is resettable using a push button. As the push button is a mechanical switch, it is desirable to add debouncing. With the system being synchronous, additional synchronisation is also necessary to use this asynchronous reset. With the IP Core 'Processor System Reset' these requirements can be implemented.

Matlab interfacing

The existing reference board design also contained IP Cores necessary to allow interfacing between Matlab and the FPGA. These are used in the final integration stage where different parameters of the system can be changed while the model is running. This greatly improves debugging speed as the model does not have to go through the compilation and synthesis workflow after a parameter change. The blocks involved for this feature are the 'HDL Verifier AXI Master' and 'AXI Crossbar'.

As the provided reference design is sufficient for our system, it was decided to use this. This has the advantage that it is verified to work correctly. When all subsystems of the SD are ready to be integrated, a new reference design has to be made that also includes an ADC IP-Core.

3.3 HDL Coder

The final step in the implementation of the system is to run through the HDL Workflow Advisor. As explained in section 2.2.5, this consists of several semi-automated steps that convert the Simulink system into VHDL and finally a Vivado project. However, before this we confirm the HDL Coder compatibility of the system with the function 'Check subsystem compatibility'. The Simulink model was verified to be compatible with HDL Coder.

Settings

The first step in the workflow advisor consists of selecting the target board (CMOD A7-35T) and the reference board design that has been created. Secondly, the I/O is configured. The created ports in the Simulink model can be connected to the available pins and components of the FPGA. During this design, the PWM Output is connected to LED LD1. This makes it easy to quickly verify if the system is operating. The inputs of the system, the reference voltage and the measured voltage, are unconnected. When the other subsystems are ready for integration, these inputs are to be connected to the grid voltage tracking subsystem. The other intermediate steps are unimportant for this report and are not further discussed here.

Code & Vivado project generation

The last steps of the HDL Workflow Advisor consist of generating the VHDL Code, wrapping this in an IP Core and embedding this in a Vivado project. After these steps, a summary is given on the high level resource consumption and information about the critical path.

3.3.1 Programming the device

We can use the generated Vivado project to program the FPGA. This is achieved in three steps [13]. Firstly, synthesis is performed; This transforms the generated VHDL code into a gate-level representation. Secondly, the implementation process is executed; This places and routes the gate-level representation onto the hardware of the FPGA. Thirdly, Bitstream generation is performed. After this, the bitstream can be directly uploaded to the FPGA.

3.4 Integration

This report is focused on the controller subsystem of the inverter. However, the integration of the subsystems can provide a good insight into the performance of the controller. Although the grid voltage tracker (GVT) subsystem was finished in time to allow for integration, the H-bridge driver was not. The GVT subsystem was combined with the controller in Simulink and Simulations have been performed that show the behaviour of this integrated system. Using the workflow described in section 2.2.2 the model was also tested on the FPGA.

Chapter 4

Evaluation

In this chapter the results are shown and discussed. For complete analysis, the controller should be analysed in a closed-loop system where it forms a part of the final inverter. This allows assessment of the stability and performance of the complete system and therefore verifies the behavior of the controller. However, not all subsystems of the inverter are fully functional at the moment of writing, the model can only be analysed by verifying it's open-loop behaviour. The first verification that has been performed is with a Simulink simulation. After this, the system is synthesised and programmed onto the FPGA and tested in a FPGA-in-the-loop (FIL) configuration. Finally, the system is combined with the GVT subsystem and tested on the FPGA.

4.1 Open-loop Simulation

As explained in section 3.1.7, a realistic data set is applied to the system. This allows verification whether the output is as expected but also if this behavior is desired. For these test, the threshold voltage and the processed error gain parameters are adjusted to observe their effect on the performance of the system. The initial simulation was performed with a threshold voltage of 15V and a processed error gain of 10. The result of this can be seen in figure 4.1.

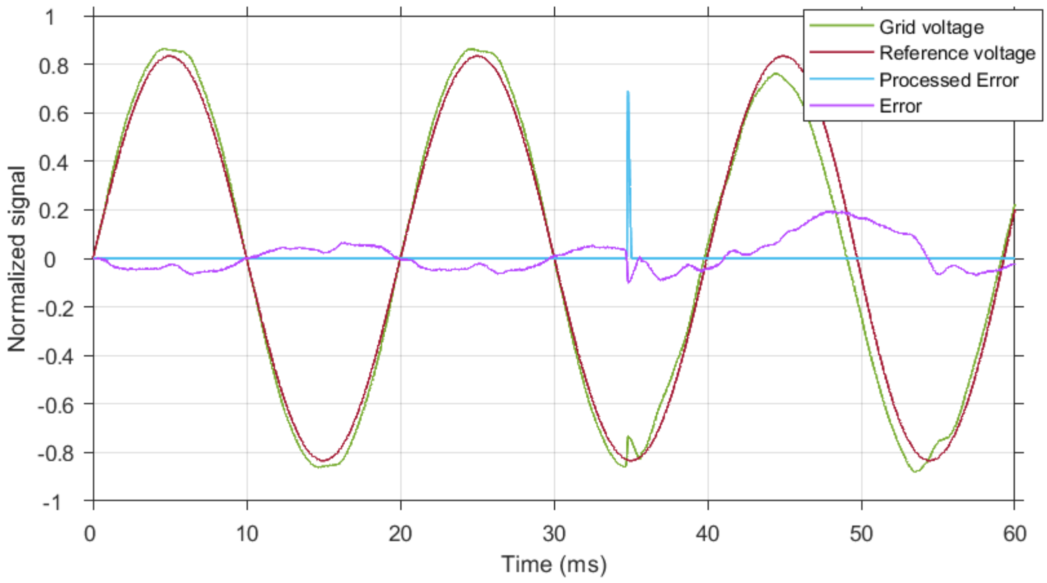


FIGURE 4.1: The simulink simulation showing the input and output signals with the applied data set. In red and green are shown the reference voltage and the grid voltage respectively, in blue the processed error as the output of the system while in purple the error as the difference between the reference and the grid voltage. All signals are normalized.

From this figure it is visible that the processed error is mostly zero, during these moments the inverter is idle. However, it is also visible that around 35ms a voltage sag due to an inrush current occurs. The processed error signal peaks to 0.7 but quickly returns to zero. By having a closer look at the voltage sag the behavior of the system can be analysed more effectively. This plot is shown in figure 4.2.

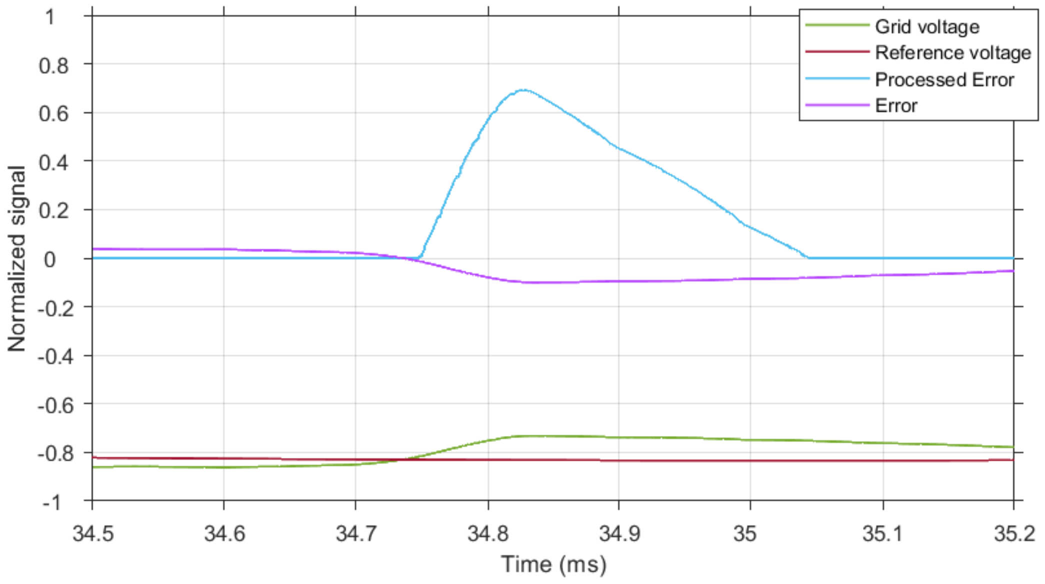


FIGURE 4.2: Zoomed-in figure of 4.1

It can be seen that there is minimal delay in the controller. When the voltage sag is defined as the moment the grid voltage subceeds the reference voltage, there is a delay of 10 µs before the processed error increases. However, it can also be observed that the processed error decays faster than the error signal.

By using a threshold voltage of 0V, some observations can be made about the behavior of the controller in situations where the inverter should idle. The simulation result of this is shown in figure 4.3.

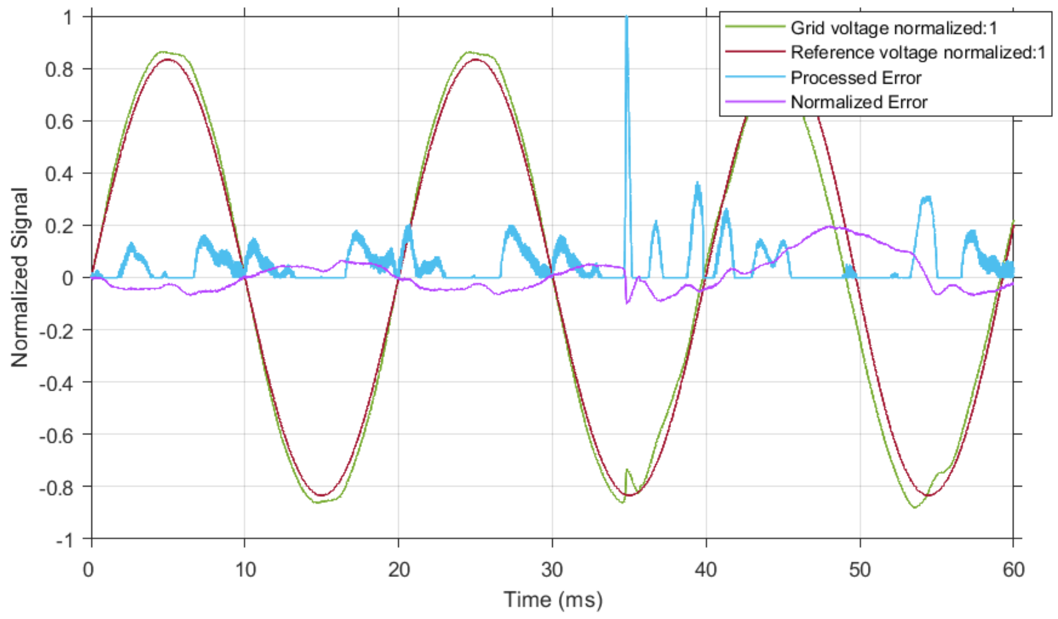


FIGURE 4.3: Inputs and outputs of the controller with a threshold voltage of 0V. Similar to the first simulation, a gain of 10 is still applied to the processed error

From this figure it is visible that some ringing occurs in the processed error after the voltage sag at 35 ms. This is also visible in the error signal and is thus expected. Another observation that can be made is the repeating pattern in the processed signal around the first three zero crossings. This is probably caused by the distorted grid voltage, as before the inrush current occurs the grid already has a deviation from the reference voltage. This distortion could be caused by the non-ideal grid emulator that was used to generate the grid voltage.

4.2 FPGA verification

Now that the model simulation has been analysed, the system needs to be verified that it also works correctly on the FPGA. The HDL Verifier extension provides several methods to check this; We chose to use a FPGA-in-the-loop (FIL) analysis. In such an analysis, the model is programmed onto the FPGA and ran in parallel to the simulation [14]. The same input is applied to both models and after retrieving the data from the FGPA, the results can directly be compared. As the FPGA runs much faster than the simulation, the clock of the FPGA is halted in between samples. Thus, the results of a FIL analysis are still representative of an actual deployed model. In figure 4.4 the result of this analysis is shown, figure 4.5 shows the result zoomed into the area of interest.

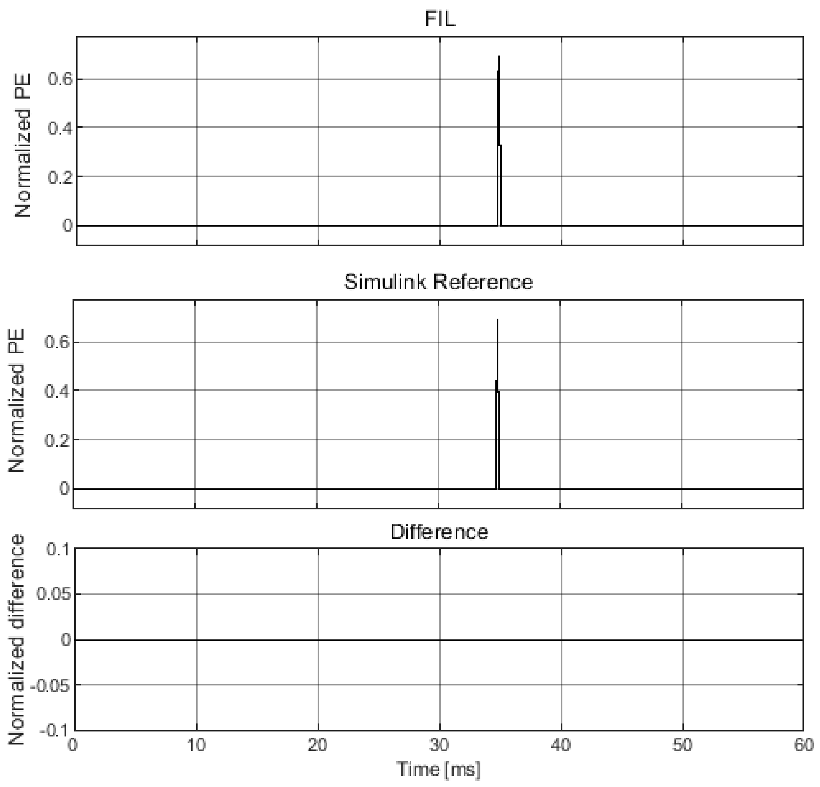


Figure 4.4: Processed error (PE) comparison between the simulated Simulink reference and the measured FIL signal

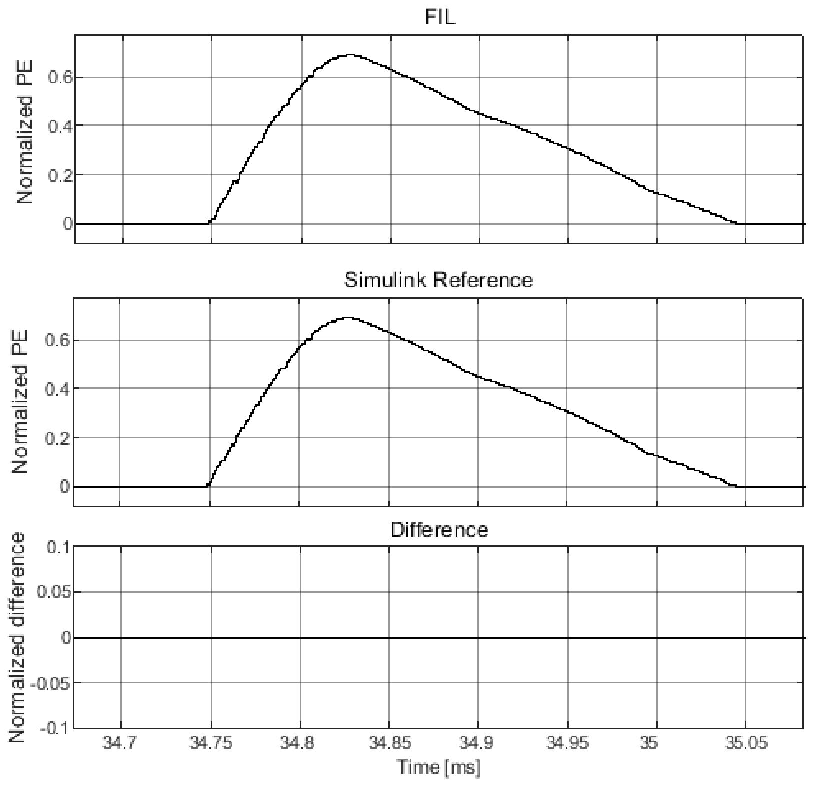


Figure 4.5: Processed error (PE) comparison between the simulated Simulink reference and the measured FIL signal, zoomed into the pulse at 35ms

As can be seen from these figures, the model running on the FPGA performs identical to the Simulink reference.

4.3 Integration

With the model verified to work correctly in Simulink as well as on the FPGA, the GVT subsystem and the controller subsystem can be integrated. As the focus of this report is not on this integration part, the listed results are limited to the model running on the FPGA. Using the signal generator of the PicoScope 4824, a grid voltage with distortion was emulated and applied to the ADC of the FPGA. With a limited number I/O pins available, it was not possible to output multiple internal signals as a vector of bits, therefore we chose to modulate several internal signals with PWM. These signals are captured and digitally filtered using the PicoScope 7 software. The result of this experiment is shown in figure 4.6.

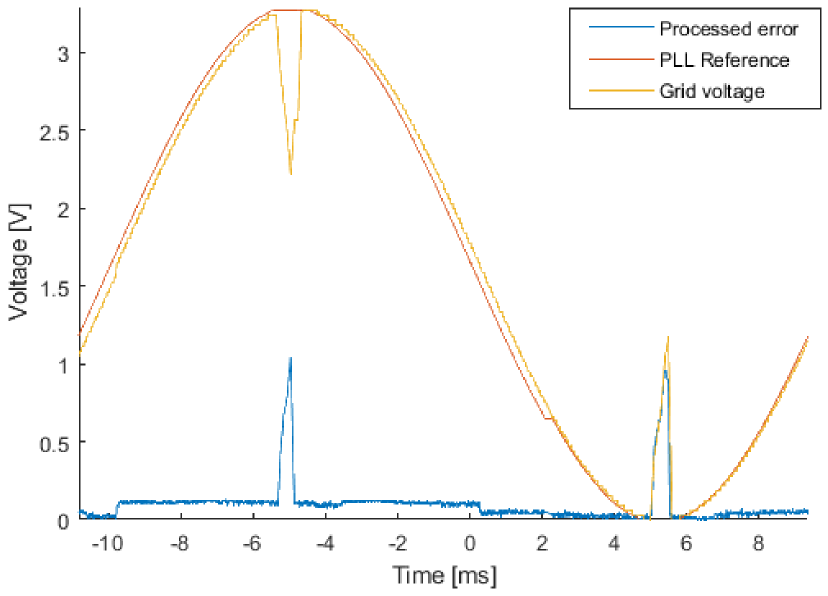


FIGURE 4.6: Measured signals with an error gain of 1 and a normalized threshold voltage of 0.05

From this figure we can observe that the PE responds quickly to any voltage sag and during this time behaves similar to the simulation. However, it is also visible that there is an offset to the PE during the positive half cycle and a smaller offset during the negative half cycle. This offset is not visible in the simulation. The timing constraints were not met after the bitstream was generated. Therefore this effect is probably caused by a propagation delay that is more than one clock cycle (negative slack), causing data integrity issues in the PWM generator block.

To solve this issue, a rate transition block was adjusted in Simulink. An option was set that ensures data integrity. The same test signals have been applied to this new model. The results can be seen in figure 4.7.

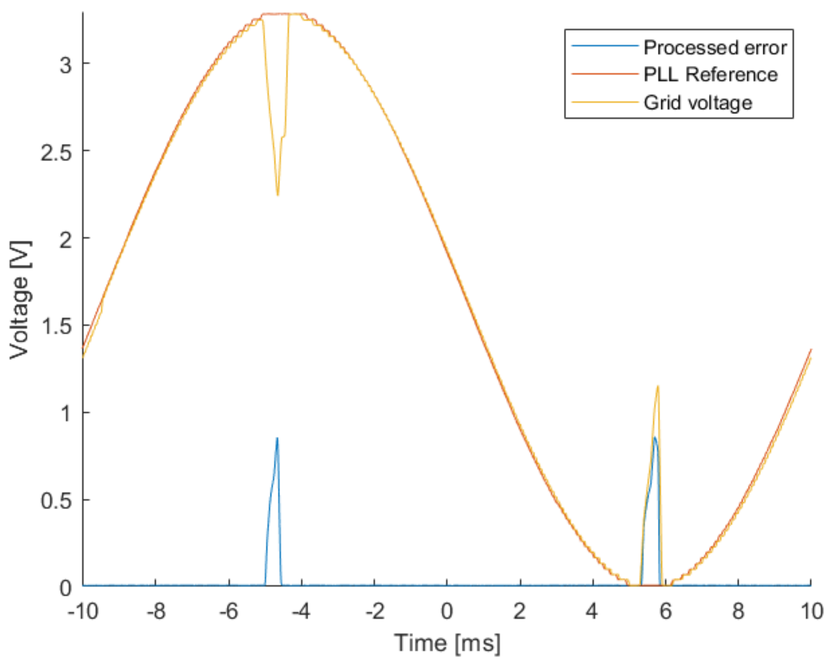


FIGURE 4.7: Measured signals of the integrated system without data integrity issues

As can be seen above, the error is completely removed and the results are as expected.

Chapter 5

Discussion

In this chapter, the results are further examined and several issues that were encountered with the tools are mentioned.

As the inverter is still under development and no strict requirements were specified, the performance of the controller was hard to evaluate. This also increased the difficulty in designing a high performance controller, as the different sections of the controller have to some extent be generic and parameterisable.

5.1 Filter behavior

By removing low frequency components from the error signal, Gibbs phenomenon can have a significant impact. For large transient voltage sags, the frequency spectrum does not only contain high frequency signals (above the cutoff frequency of the filter) but also lower frequencies and a DC component. By filtering the lower frequencies and DC signal from this transient, ringing can occur. Therefore it is important to correctly determine the threshold voltage and filter cutoff frequency that achieve the desired response. In this report, this optimization is not further explored, as this requires the full system to be functional.

5.2 Vivado

Significant issues and difficulties were encountered in Vivado. An earlier developed PWM generator design created by a project team member served as a starting reference for the controller. This system was synthesised and implemented in Vivado 2018.3, therefore this version was initially used. In this version there is a bug that can occur that throws an error during synthesis when it is run in 2022 or later. Finding the cause of this bug and solving it by migrating to Vivado 2022 wasted a lot of time.

Chapter 6

Conclusion

The goal of this project was to design a controller on a FPGA that is capable of responding to voltage sags with minimal delay. As this controller is part of a larger project, a parameterisable design has been made in Simulink that allows for adjustments when the complete system is functioning. As not all subsystems of the inverter are functional at the moment of writing, the evaluation of the controller's performance was limited to open-loop testing and verification on a FPGA. The controller has been integrated with the first subsystem, the grid voltage tracker, and verified to work correctly in simulation as well as on the FPGA. The synthesised model does not meet the timing requirements thus improvements need to be made in reducing the slack.

Chapter 7

Recommendations

The designed controller functions, but is not fully evaluated in this report. In future research, the complete integrated system should be analysed. This is necessary in order to guarantee closed-loop stability. By adjusting the threshold voltage, the error gain and the cut-off frequency of the filter the system can be tuned to perform optimally in the closed-loop system.

Before the complete system is to be implemented, the timing issues have to be resolved. A possible approach could be to implement pipelining. HDL Coder allows for automatic pipelining if this is enabled. Another option is to manually place delay's in the model, which ensures that the signal does not have to propagate through the complete path in one clock cycle. Although this decreases the maximum propagation delay, it also increases the latency of the controller.

In the simulink model of the controller, a dead-zone block has been used. This block acts as a threshold but also applies an offset to the signal. An alternative that could be explored is replacing this block with a switch statement and a comparator. The output can then be switched to zero if the signal is below the threshold or to the input if above.

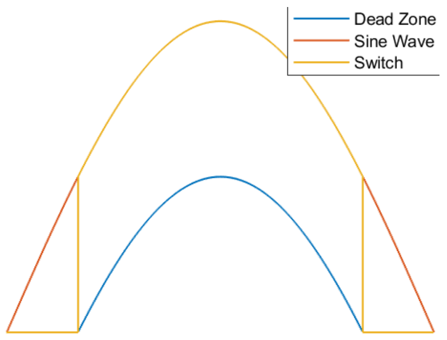


FIGURE 7.1: A possible alternative to explore: changing the dead-zone block to a switch and a comparator

Bibliography

[1] *Inrush current as a cause of failure and excessive wear and tear.* [Online]. Available: `https://hyteps.com/power-quality/inrush-current/`.

[2] R. Jaschke, "Efficiency and inrush current in low power equipment at AC-grid," *Proceedings - 2016 10th International Conference on Compatibility, Power Electronics and Power Engineering, CPE-POWERENG 2016*, pp. 49–52, Aug. 2016. DOI: `10.1109/CPE.2016.7544157`.

[3] A. Matthee, N. Moonen, and F. Leferink, "Micro-Grid Inrush Current Stability Analysis," *2021 Joint IEEE International Symposium on Electromagnetic Compatibility Signal and Power Integrity, and EMC Europe, EMC/SI/PI/EMC Europe 2021*, pp. 440–444, Jul. 2021. DOI: `10.1109/EMC/SI/PI/EMCEUROPE52599.2021.9559165`.

[4] H. Huang, O. J. Oghorada, L. Zhang, and B. V. Chong, "Active harmonic current elimination and reactive power compensation using modular multilevel cascaded converter," *2017 19th European Conference on Power Electronics and Applications, EPE 2017 ECCE Europe*, vol. 2017-January, Nov. 2017. DOI: `10.23919/EPE17ECCEEUROPE.2017.8098966`.

[5] B. Ihsan, A. Matthee, F. Leferink, T. D. Rachmilda, and D. Hamdani, "Active Transient EMI Stabilization," *Proceedings of the 2021 Asia-Pacific International Symposium on Electromagnetic Compatibility, APEMC 2021*, 2021. DOI: `10.1109/APEMC49932.2021.9596970`.

[6] *Cmod A7 Reference Manual - Digilent Reference.* [Online]. Available: `https://digilent.com/reference/programmable-logic/cmod-a7/reference-manual`.

[7] *State Control - Simulink - MathWorks.* [Online]. Available: `https://nl.mathworks.com/help/hdlcoder/ref/statecontrol.html`.

[8] *Board and Reference Design Registration System - MATLAB & Simulink - MathWorks.* [Online]. Available: `https://nl.mathworks.com/help/hdlcoder/ug/board-and-reference-design-system.html`.

[9] *Simulink Blocks Supported by Using Native Floating Point - MATLAB & Simulink - MathWorks.* [Online]. Available: `https://nl.mathworks.com/help/hdlcoder/ug/hdl-coder-support-for-native-floating-point-library-mapping.html`.

[10] R. G. Lyons, *Understanding digital signal processing.* Upper Saddle River, NJ : Prentice Hall, 2011, 1 online resource (xxiii, 954 pages) : ISBN: 0137027419.

[11] J. O. (O. Smith, *Introduction to digital filters : with audio applications.* [United States] : W3K, 2008, xviii, 460 pages : ISBN: 9780974560717.

[12] *Artix 7 Series DSP Slice User Guide.* [Online]. Available: `https://docs.xilinx.com/v/u/en-US/ug479_7Series_DSP48E1`.

[13] *Build a Vivado Project - Digilent Reference.* [Online]. Available: `https://digilent.com/reference/programmable-logic/guides/vivado-generate-bitstream`.

[14] *FPGA-in-the-Loop - MATLAB & Simulink - MathWorks Benelux.* [Online]. Available: `https://nl.mathworks.com/help/hdlverifier/fpga-in-the-loop.html?s_tid=CRUX_lftnav`.