Gezamenlijke opleiding UHasselt en KU Leuven

2021•2022
Faculteit Industriële Ingenieurswetenschappen
master in de industriële wetenschappen: elektronica-ICT

# Masterthesis

Design of an integrated digital controller for a buck converter

PROMOTOR :
Prof. dr. ir. Nele MENTENS

BEGELEIDER :
dr. ing. Jo VLIEGEN

Luke Geelen

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

2021•2022
Faculteit Industriële Ingenieurswetenschappen
**master in de industriële wetenschappen: elektronica-ICT**

**Masterthesis**

# Design of an integrated digital controller for a buck converter

**PROMOTOR :**
Prof. dr. ir. Nele MENTENS

**BEGELEIDER :**
dr. ing. Jo VLIEGEN

**Luke Geelen**

**Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT**

# Acknowledgements

In this first part, I would like to express my gratitude towards the people that allowed me to realise this thesis. First and foremost, I would like to thank my promotors, prof. dr Ir. Mentens and dr. Ing. Vliegen for the guidance during this project and drs. Biesmans for analysing the maximum clock speed of the PWM components and determining the area of the ASIC implementations.
Finally, I would like to thank my family for supporting me during these challenging times.

# Contents

# Table of figures

# Table of tables

# Table of abbreviations

| | |
|---|---|
| ADC | Analog Digital Converter |
| ASIC | Application Specific Integrated Circuit |
| CoDiCApp | Co-Design and Integration in power electronics Converter for high power density Applications |
| DC | Direct Current |
| DPWM | Digital Pulse width modulator |
| FCS-MPC | Finite control set model predictive control |
| FLS | Frequency Loop Shaping |
| FO | First order |
| FPGA | Field Programmable Gate Array |
| HDL | Hardware description language |
| LCO | Limit Cycle Oscillation |
| LPF | Low Pass Filter |
| LSB | Least Significant Bit |
| LUT | LookUp Table |
| MIMO | Multiple input multiple output |
| MPC | Model Predictive Control |
| MSB | Most significant Bit |
| PWM | Pulse Width Modulation |
| SO | Second order |
| TF | Transfer function |
| VHDL | VHSIC Hardware Description Language |
| VHSIC | Very High Speed Integrated Circuit |

# Abstract

The CoDiCApp project aims to design a power converter integrated into a package with a high power density. It is preferred to control this converter digitally. This thesis aims to design a controller that can be converted into an ASIC to control this power converter.

The challenge in designing a controller for this converter can be split into three key points. First, as the goal is to design a power converter as small as possible, the ASIC implementation should be made as compact as possible, giving a preference to use few resources. Secondly, the controller should function well under different loads, even though a change in load alters the behaviour of the converter system. Finally, the power converter needs to be controlled by a Pulse Width Modulation (PWM) signal with a frequency of 1 MHz while maintaining a reasonable control resolution.

To keep the ASIC compact, a relatively simple implementation of a PID controller was chosen as the base design of the controller. Implementing a well-tuned PID controller also allowed for operation under various loads. Finally, attempts were made to use high-frequency techniques to improve the resolution given the available clock frequency. However, the converter's switching frequency and filtering capabilities proved not high enough to integrate a Σ–Δ generator or dither generator, resulting in a simple sawtooth comparator DPWM generator.

# Abstract in Dutch

Het CoDiCapp project richt zich tot het ontwerpen van een power converter geïntegreerd in een package met een hoge vermogensdichtheid en digitale sturing. Het doel van deze thesis is een controller te ontwerpen die omgevormd kan worden tot ASIC om de power converter aan te sturen.

De uitdaging in het ontwerpen deze controller kan in drie kernpunten opgesplitst worden. Allereerst, aangezien het doel is een zo compact mogelijke power converter te ontwerpen is ook bij voorkeur de ASIC implementatie van de aansturende controller zo compact mogelijk. Ten tweede dient de converter correct aangestuurd te kunnen worden onder verschillende belastingen. Ten derde dient de power converter aangestuurd te worden door een Pulse Width Modulation (PWM) signaal met een frequentie van 1 MHz met een redelijke stuur-resolutie.

Om de ASIC compact te houden is er voor een relatief simpele implementatie van het PID stuur-algoritme geopteerd als basisontwerp voor de controller. De implementatie van een goed afgestemde PID controller zorgt meteen ook voor het stabiel functioneren van de controller onder verschillende belastingen. Ten slotte is geprobeerd om hoge frequentie technieken toe te passen om de aanstuur-resolutie onder gegeven frequentie beperkingen te verhogen, namelijk een Σ–Δ generator en dither-generator. Echter bleek de schakelsnelheid en filterend vermogen van de converter niet genoeg om deze technieken toe te passen. Hierdoor is er geopteerd om een simpele zaagtand comparator DPWM-generator te gebruiken.

# 1 Introduction

Currently, most DC-DC power converters exist as discrete components with a separate controller. The control block of most DC-DC power converters is often still analogue; however digital converters can offer several advantages, offering better scalability and portability, and integrability with other digital systems [1].

The CoDiCApp (Co-Design and Integration in power electronic Converter for high power density Application) project aims to design a functional power converter combining all components and control logic into a single package.
This project is divided into four sections. The first is the design of the power converter with passive components. The second section will consider the design of the pre-driver. The third section will be the design and development of the controller, and finally, the fourth section will be the integration of all these sub-sections into one power converter in a package. This thesis will focus on the third section of this project: the design of the controller.

The topology chosen for the DC-DC power converter is that of the synchronous buck converter, as shown in Fig. 1.1. It should be noted that this figure does not display all inputs of the controller and pre-driver.

A digital feedback-control system will be designed to maintain a stable output voltage. This feedback control circuit will be responsible for interpreting ADC measurements of the output- voltage, calculating the duty cycle for the next period, and output a PWM signal with this duty cycle and repeating these processes.

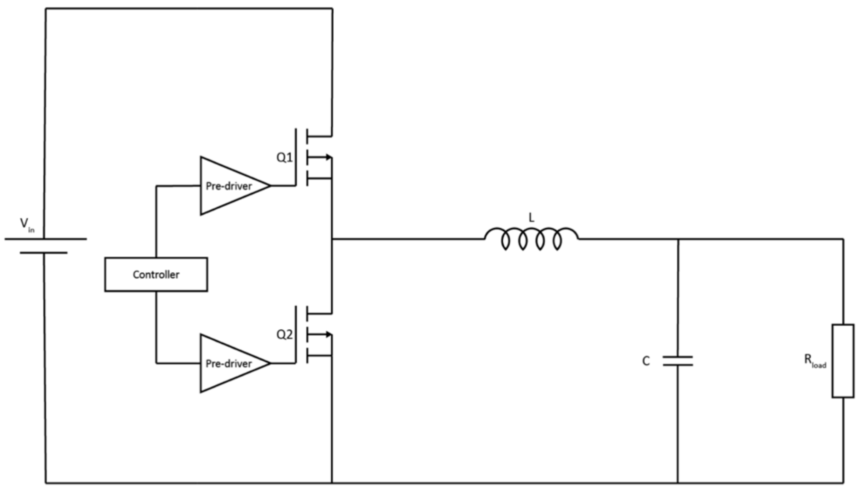


*Fig. 1.1 Synchronous Buck DC-DC converter with pre-driver and controller*

## 1.1 Project Goal

This project aims to design a VHDL implementation of a digital controller for the buck converter. This implementation will then serve as a prototype for the ASIC integration that is considered outside this master thesis's scope.

The controller should have the following requirements:

- Control the output voltage of the synchronous buck converter.
- The design should use as few resources as necessary, eventually resulting in a compact ASIC.
- The controller should generate a control directive using PWM. As a synchronous buck-converter is the topology of choice, there should both be a high-side and low-side driving signal, which should never overlap. A frequency of approximately 1 MHz has been requested for this PWM signal.

## 1.2 Outline of the thesis

Chapter 2 of this thesis introduces the scientific and theoretical backgrounds of concepts and techniques, which formed the basis of this thesis. Here the basic control architectures are presented with potential tuning methods and DPWM requirements and architectures. Chapter 3 gives an overview of the materials and software used during the project. In Chapter 4, the steps taken to obtain the actual controller are described. Chapter 5 evaluates the obtained controller performance. A consideration in the next steps of developing the controller ASIC is given in chapter 6, and a conclusion is made in chapter 7.

# 2 Literature study

## 2.1 Control structures

To select and design a control approach for the controller of the DC-DC Buck converter, an analysis of control structures that have previously been used for this kind of plant is given. The analysed digital control strategies can be split up into three categories. First, there are the digital PID controllers. Secondly, a dual control loop structure exists that controls the voltage via a current control loop. And finally, due to increasing on-chip processing capabilities, MPC (model process control) became a competitor of the techniques mentioned above.

### 2.1.1 Voltage mode digital PID controller using multiplication

A PID controller combines a classical P, I and D-controller, respectively applying a proportional gain, integration of the error signal and a derivation of the error signal. This is a feedback control loop mechanism. A PID controller can be implemented either in a serial or parallel design, slightly altering the formula of the controller. Although the serial- and parallel-controller have the same effect, the gain or time constants of the I- and D-terms would be slightly different.

One way to implement a PID controller on an FPGA, or eventually on an ASIC, is by using multipliers. In order to obtain a formula that can be converted to an FPGA architecture the derivation from [2] can be used: first the time-domain formula of a serial PID controller is taken:

$$u(t) = K_p \left\{ e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right\}$$

In this form variable ‘u’ symbolises the output (or control) signal whilst e symbolises the error signal, $K_p$ equals the proportional gain, $T_i$ the integral time constant and $T_d$ the derivative time constant. This formula in continuous time can be sampled into discrete time using sampling time $T_s$:

$$u(k) = K_p \left\{ e(k) + \frac{T_s}{T_i} \sum_{i=0}^{k-1} e(i) + \frac{T_d}{T_s} [e(k) - e(k-1)] \right\}$$

This form is not optimal for a hardware implementation as all previously recorded errors would need to be stored to perform the summation. This can be solved by using the recursive PID algorithm. First, we find the control output u(k-1):

$$u(k-1) = K_p \left\{ e(k-1) + \frac{T_s}{T_i} \sum_{i=0}^{k-2} e(i) + \frac{T_d}{T_s} [e(k-1) - e(k-2)] \right\}$$

The difference between u(k) and u(k-1) is:

$$u(k) - u(k-1) = K_p \left\{ e(k) - e(k-1) + \frac{T_s}{T_i} e(k-1) + \frac{T_d}{T_s} [e(k) - 2e(k-1) - e(k-2)] \right\}$$

The recursive control output u(k) can be written as:

$$u(k) = u(k-1) + a_0 e(k) + a_1 e(k-1) + a_2 e(k-2)$$

With:

$$a_0 = K_p \left(1 + \frac{T_d}{T_s}\right) \quad a_1 = -K_p \left(1 - \frac{T_s}{T_i} + 2\frac{T_d}{T_s}\right) \quad a_2 = K_p \frac{T_d}{T_s}$$

This form can be translated into an FPGA design using multipliers, adders, and registers. Fig. 2.1 shows how this PID controller would be integrated into a closed-loop design.
Besides a control component, there is also an input and an output interface needed to form the closed-loop. As an input interface, an ADC can measure the voltage over the output of the power converter and translate this into a digital signal as needed by the digital controller.
A possible output interface can be seen in Fig. 2.2 consisting of an up-and-down counter (replaceable by a saw-tooth generator) and a comparator. This component could generate 2 PWM signals, one to control the high-side driver and one to control the low-side driver.

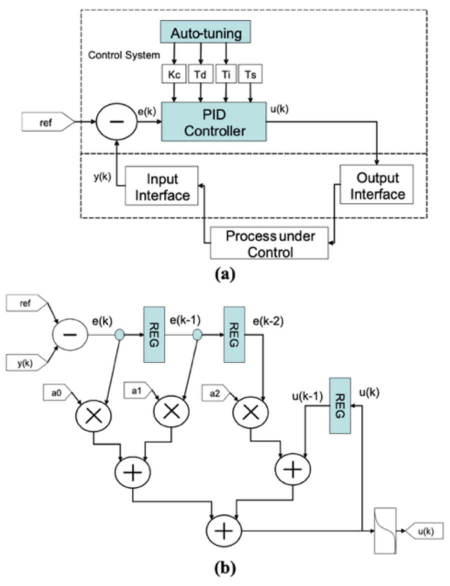


*Fig. 2.1 a) basic block scheme of the closed-loop system with optional autotuning b) logic design of the PID controller without auto-tuning [3]*

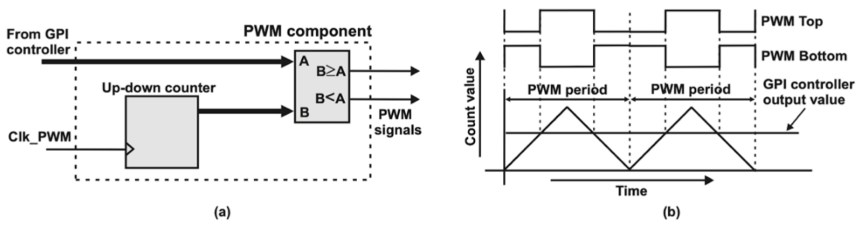


*Fig. 2.2 Potential PWM modulator output interface [4]*

### 2.1.2 Voltage mode digital PID controller using lookup tables

It is possible to replace the multipliers from the previous architecture with lookup tables. This involves pre-calculating all the possible values and storing them. [5] suggests two different architectures for implementing this concept.
The trade-offs between the multiplier and LUT scheme are as follows: The LUT design requires less logical resources and power whilst requiring more clock cycles to obtain a result and a storage mechanism to store the lookup tables.
The first architecture from [5] requires only 13% of the logical resources of a comparative multiplier design. However, it requires 14 clock cycles to obtain a single control directive compared to 1 for the multiplier scheme. Furthermore, the power consumption of this architecture is 40% less than from the multiplier design.
The second architecture suggested in [5] is more resource optimised, requiring only 4% of logic resources needed by the multiplier design whilst having similar power requirements as the first architecture.

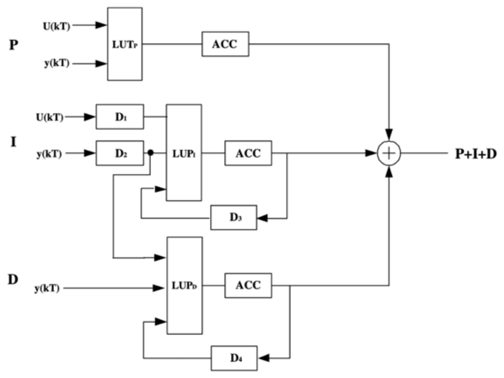


Fig. 2.3 Suggested PID LUT architecture 1 [5]

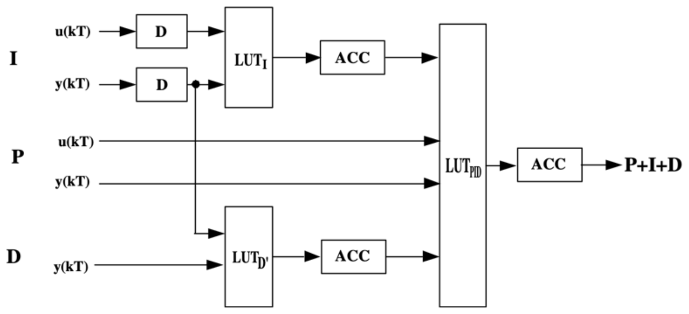


Fig. 2.4 Suggested PID LUT architecture 2 [5]

## 2.1.3 Voltage and current mode control

Another potential approach is described in [6]. This architecture uses a multi-loop feedback control structure based on two variables: the current injected into the inductor and the voltage measured over the output capacitor. An overview of this architecture can be seen in Fig. 2.5.

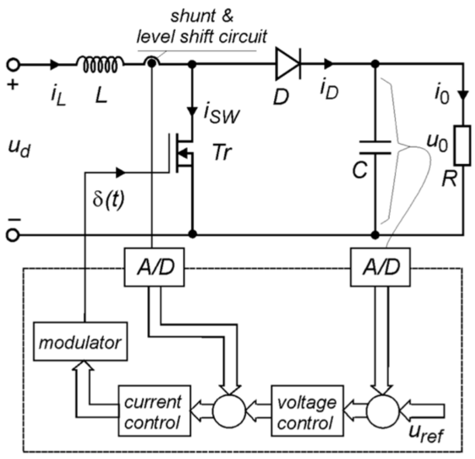


*Fig. 2.5 Overview of the multi-loop feedback control structure suggested in [6]*

This architecture's outer control loop (voltage control) is a PI regulator. This PI regulator generates the reference current signal for the current control loop. The logical implementation of the voltage control loop architecture can be seen in Fig. 2.6.

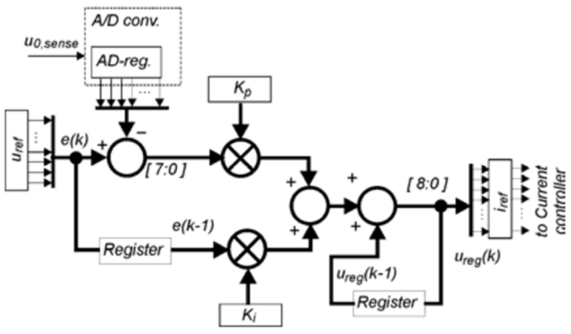


*Fig. 2.6 Logical implementation of the outer voltage control loop [6]*

The inner current control loop contains an interpolating predictive control algorithm. After measuring the current twice with a known time interval, the algorithm will calculate the $T_{on}$ time required to meet the current reference. [6] also makes a side-by-side comparison of two different methods of compensating for current variations introduced by the measuring circuit. Fig. 2.7 shows the logical implementation of the inner control loop with extra switches to switch between these compensating slopes. The data from [6] show that the dynamic compensating slope provides the best performance.

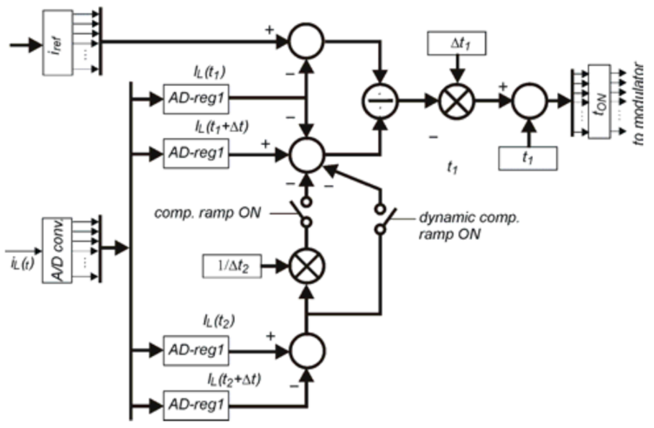


*Fig. 2.7 Logical implementation of the outer current control loop [6]*

Finally, the obtained control signal is sent to a modulator that converts this to a PWM signal. This modulator can be seen in Fig. 2.8.

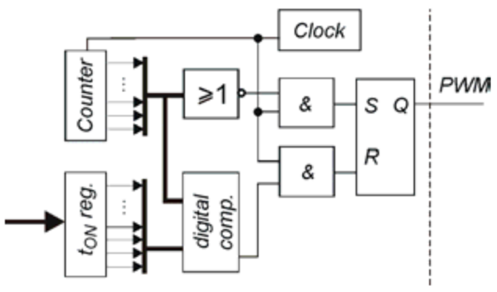


*Fig. 2.8 Logical implementation of the PWM-Modulator [6]*

Unfortunately, [6] shows little comparative data concerning resources and power used on the FPGA, making it difficult to compare this control method to the others mentioned in this document. An advantage of this controller is that it can be easily altered to function as a current supply source by manually feeding a reference current instead of feeding the output of the outer voltage PI Loop.

According to [7], current-mode control can improve control speed compared to voltage mode control but introduce some new challenges. First, the existence of two feedback loops makes the analysis of the circuit more difficult. Second, slope compensation is needed to prevent an unstable control loop. Third, the leading-edge current spike caused by the transformer winding capacitance is a new noise source to be dealt with.

### 2.1.4 MPC

Model Predictive Control (MPC) offers an alternative control method. As described in [8], an MPC approach considers a model of the system that is being controlled. This is used to predict future behaviour over time. These predictions are evaluated using a cost function. A sequence with minimal cost is chosen. This sequence contains the future control actions. An example of such a sequence with predictions can be seen in Fig. 2.9. The first control action from this sequence is applied, the new plant state is measured, and then the entire algorithm is calculated again; this is done every sampling period.

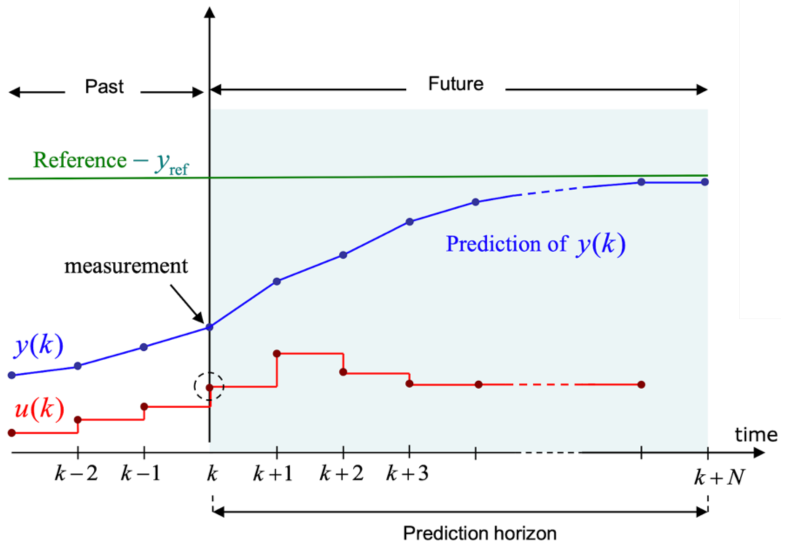


*Fig. 2.9 MPC prediction horizon [9]*

MPC has several advantages over PID control as it can be easier to deal with nonlinear plants and can function as a multiple-input and multiple-output (MIMO) system.
Whilst MPC can result in accurate control actions, the primary disadvantage is the speed, as the entire predictive horizon needs to be calculated to obtain the following control action. This optimisation problem is often solved online using a more powerful computer. However, some approaches manage to integrate MPC into an FPGA implementation.
Since power converters only have a finite number of switching states, it is possible to simplify the optimisation problem of the system behaviour for only those possible states. Then each prediction can be used to evaluate the cost or decision function. The state with minimum cost is selected. This approach is known as Finite Control Set MPC, or FCS-MPC.
The prediction horizon is the next consideration for altering the MPC algorithm for use on an FPGA. A short horizon is often chosen with prediction horizon N varying between 1 and 3 for several FPGA-MPC implementations to reduce the complexity.

Due to the complexity of these systems, it is often opted to use high-level tools to design the architecture. For example, [10] uses the “System Generator toolbox for Simulink/MATLAB from Xilinx”, and [11] uses the HDL coder tool from MATLAB’s Simulink.
In [12], an overview of FPGA-MPC implementations is given. However, it should be noted that these are still relatively slow and have a complicated architecture, requiring a high amount of resources.

### 2.1.5 Conclusion

Careful analysis of the listed control strategies has resulted in a choice for the voltage mode digital PID approach for the following reasons. The digital PID controller is the least resource-consuming integration as the voltage and current mode control methods require a second loop. Although voltage and current mode control can be slightly faster, it is not optimal, and according to [7], for systems with a wide input line and load variations or applications where the complexity of a dual feedback loop or slope compensation should rather be avoided, voltage-mode is the better pick. Due to the way the prediction-based current control loop works, it is required for the DPWM to be operated synchronously with the control mechanism, preventing the use of multiple clock domains. That would make it nearly impossible to obtain the switching frequency required for this project. MPC requires significantly more resources and time, as for each control directive, the control horizon needs to be recalculated. This can, however, result in slightly higher performance. However, according to [13], the results of a PID controller are generally better than more complex control structures, like MPC, when enough effort is put into tuning the PID controller right.

Although the PID controller using LUTs appears to use fewer resources, it limits the controller's flexibility, making it less easy to incorporate multiple tunings, which may be needed to control different buck converters. This has made the multiplier PID approach the preferred implementation.

The decision to opt for a control logic using voltage mode control is further backed by several other high-frequency implementations using the same control strategy [14], [15] and several implementations opting for voltage mode control with PID as controller logic [4], [16], [17].

## 2.2 Optimisation methods for PID controllers

While PID controllers can offer very robust performance under different circumstances, some suggestions are made in an attempt to improve this performance.

### 2.2.1 Nonlinear PID control

As is suggested in [17] and [18], it is possible to change the system’s behaviour depending on the absolute value of the error signal. Although described differently, both methods take a similar approach. When the error gets very large, the controller should saturate, either outputting the maximum or minimum duty cycle. [17] obtains this effect by forcing the control signal to this duty-cycle whilst [18] opts for using a significantly large gain parameter in a serial PID controller.
The size of the error signal is also suggested to influence the gain parameter. [17] divides the total range into three fields: saturation, strong control and common control, whilst [18] uses five ranges: saturation and four different control strengths.

Both papers report a decreased settling time on the step response and after a load change. Based on this data, this control method could help increase the controller's performance in design.

### 2.2.2 Adding a low pass filter to the derivative component

The transfer function of a PID controller can be augmented with a first-order low pass filter in series with the derivative component as done in [19]. The transfer function of such a controller is as follows:

$$\frac{U(s)}{e(s)} = K_p \left( 1 + \frac{1}{s \cdot T_i} + s \cdot \frac{s \cdot T_d}{1 + s \cdot \frac{T_d}{N}} \right)$$

The effect of this low pass filter is a limited overshoot of the regulator response and limited amplification of noise at the controller's input.

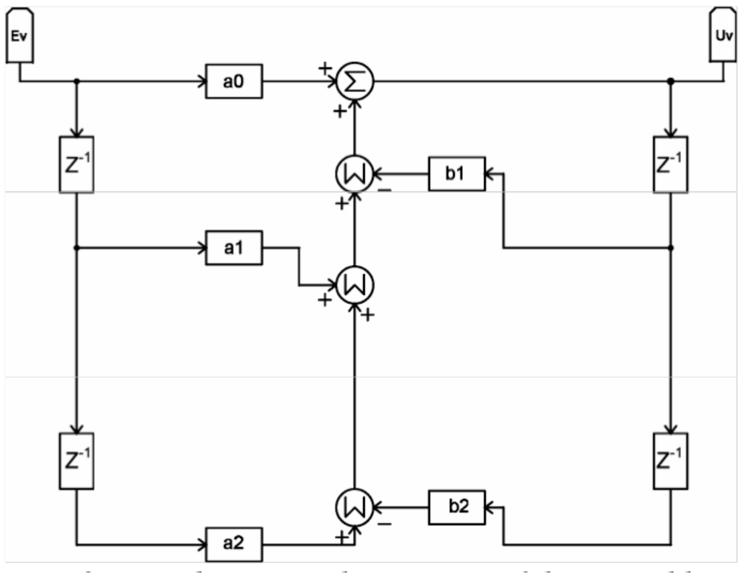


*Fig. 2.10 Suggested logical implementation of the filtered transfer function [19]*

Fig. 2.10 shows the suggested implementation for an FPGA [19]. This indicates that this design is not much more complex when compared to that of Fig. 2.1, only requiring two more multipliers, an additional register, and an additional adder.
The relatively low complexity increase with promising results for potential performance increase makes this a promising possible improvement.

### 2.2.3 Setpoint filter

Another method to potentially decrease or prevent overshoot on the step response is to avoid rapid changes to the setpoint. The idea is that the largest error signal is most likely to occur during the power converter's start-up. A more aggressive tuning can be used by slowly ramping up the setpoint whilst limiting the total overshoot.

[20] suggests two different setpoint filters for PID control. The simplest, seen in Fig. 2.11a, functions as a rate limiter, which will increase the effective setpoint linearly until the target setpoint is obtained. The maximum value of the saturator will determine the speed of the ramp.
Fig. 2.11b shows a setpoint jump and rate limiter. The output follows the input for a small change in the input signal. The output will follow the input with a limited rate for large setpoint changes.
Both filters have a low-pass character and cause delays.

The advantage of the jump and rate filter in a system with varying setpoint would be that the setpoint is better followed for small changes in setpoint and only use the rate limiter for larger changes in setpoint. As the implementation of the controller will be expected to function at a fixed setpoint, the jump and rate limiters offer no advantage over the rate limiter whilst requiring more resources

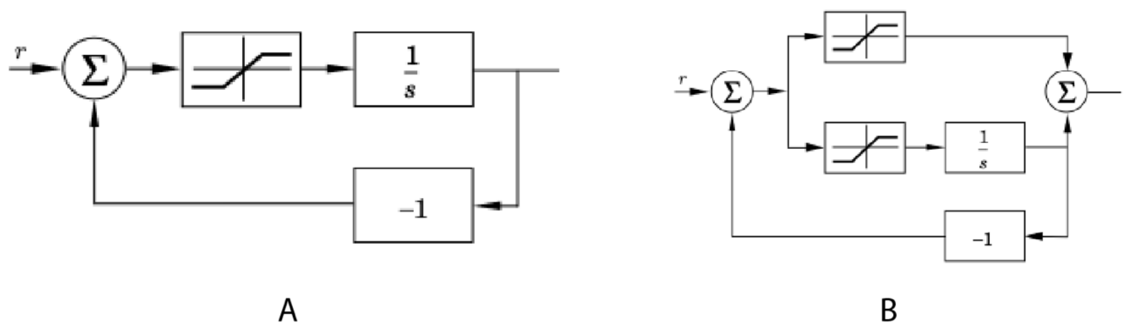


*Fig. 2.11 a) Setpoint rate limiter filter b) setpoint jump and rate limiter [20]*

## 2.3 Methods of tuning PID Controllers

PID controllers are often used in industry, and whilst there are many different approaches to tuning the control parameters ($K_p$, $K_i$ and $K_d$), poorly tuned PID controllers are often still found in industry [21], [22].

### 2.3.1 SIMC- and K-SIMC tuning method

The SIMC tuning method, introduced and clarified in [13], [22], offers an analytic tuning method to systematically find the PID tuning parameters. According to [22], the SIMC method aims to improve upon the classic method by Ziegler and Nichols [23], the IMC PID method by Rivera et al. [24] and the related direct synthesis tuning rules by Smith and Corripio [25].

The SIMC methods suggest a two-step procedure to obtain a potential tuning. First, the plant that should be controlled is approximated by either a first- or second-order plus delay model. A half-rule is proposed to determine the effective delay used in this approximation. Second, the tuning parameters can be derived. This will be a PI tuning when a first-order plus delay model is used or a PID tuning when a second-order plus delay model is used.
The SIMC tuning method works well for integrating and time delay processes, for both setpoint changes and load disturbances [22].

**First-order approximation**

To obtain the first order plus delay approximation of the plant, the original model should be written in the following form [22]:

$$\frac{\prod_j\left(-T_{j0}^{inv}+1\right)}{\prod_i \tau_{i0}s+1}e^{-\theta_0 s}$$

In this form the time constants $\tau_{i0}$ are ordered according to magnitude. From this form, the first order magnitudes can be derived using the following formulas [22]:

$$\tau_1 = \tau_{1,0}+\frac{\tau_{2,0}}{2}\ ;\ \theta = \theta_0+\frac{\tau_{2,0}}{2}+\sum_{i\geq 3}\tau_{i,0}+\sum_j T_{j0}^{inv}+\frac{h}{2}$$

In the case of digital implementations of the PID controller, let h be equal to the sampling frequency. These parameters are to be filled in into the following equation to obtain the first-order model [13]:

$$G(s) = k\frac{e^{-\theta s}}{\tau_1 s+1}$$

**Second-order approximation**

To obtain the plant's second-order plus delay approximation, the original model should be written in the same form as for the first-order approximation. The following formulas can then be used to obtain the parameters for the second-order approximation [22]:

$$\tau_1 = \tau_{1,0}\ ;\ \tau_2 = \tau_{2,0} + \frac{\tau_{3,0}}{2}\ ;\ \theta = \theta_0 + \frac{\tau_{3,0}}{2} + \sum_{i\geq 4} \tau_{i,0} + \sum_j T_{j0}^{inv} + \frac{h}{2}$$

From these parameters, the following second-order plus delay can be derived [22]:

$$G(s) = \frac{k}{(\tau_1 s + 1)(\tau_2 s + 1)} e^{-\theta s}$$

**Calculation of the PI parameters from the first order plus delay approximation**

When the first order plus delay approximation of the model is obtained, the parameters for the PI tuning can be achieved using simple formulas [22]:

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c + \theta}\ ;\ \tau_i = \min\left(\tau_1, 4(\tau_c + \theta)\right)$$

A tuning parameter $\tau_c$ is introduced. This tuning parameter should uphold $-\theta < \tau_c < \infty$ to get a positive and non-zero gain. $\tau_c$ is essentially a trade-off between response time (favoured by small $\tau_c$ ) and stability (favoured by large $\tau_c$). A recommendation for fast response with good robustness of $\tau_c = \theta$ is suggested in [22].

**Calculation of the PID parameters from the second-order plus delay approximation**

When the second-order plus delay approximation of the model is obtained, the parameters for the PID tuning can be achieved using similar formulas [22]:

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c + \theta}\ ;\ \tau_i = \min\big(\tau_1, 4(\tau_c + \theta)\big)\, ; \tau_D = \tau_2$$

The influence of tuning parameter $\tau_c$ is the same as on the PI parameters used with the first-order approximation.

**Supposed Improvements in K-SIMC over SIMC**

The SIMC method has been adapted in [21]. A change to the model reduction technique and tuning rules have been proposed to permit the SIMC method to be applied more confidently to a wider variety of systems.

The formulas for determining the first order plus delay approximation parameters are as follows [21]:

$$\tau_1 = \tau_{1,0} + \frac{1}{2}\frac{{\tau_{2,0}}^2}{\tau_{1,0}} \;\; ; \; \theta = \theta_0 + \tau_{2,0}\left(1 - \frac{1}{2}\frac{\tau_{2,0}}{\tau_{1,0}}\right) + \sum_{i\geq 3}\tau_{i,0} + \sum_j T_{j0}^{inv} + \frac{h}{2}$$

The formula for the first-order plus delay approximation remains the same [21]:

$$G(s) = k\frac{e^{-\theta s}}{\tau_1 s + 1}$$

The formula for determining the second-order plus delay approximation are as follows [21]:

$$\tau_1 = \tau_{1,0} \;\; ; \;\; \tau_2 = \tau_{2,0} + \frac{1}{2}\frac{{\tau_{3,0}}^2}{\tau_{2,0}} \;\; ; \; \theta = \theta_0 + \tau_{3,0}\left(1 - \frac{1}{2}\frac{\tau_{3,0}}{\tau_{2,0}}\right) + \sum_{i\geq 4}\tau_{i,0} + \sum_j T_{j0}^{inv} + \frac{h}{2}$$

The formula for the second-order plus delay approximation, again, remains the same as using the SIMC method [21]:

$$G(s) = \frac{k}{(\tau_1 s + 1)(\tau_2 s + 1)}e^{-\theta s}$$

[21] also proposes altered formulas for determining the PI and PID parameters. The PI parameters can be derived from the first-order approximation as follows:

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c + \theta} \;\; ; \quad \tau_i = \min\big(\tau_1, 5(\tau_c)\big) \;\; ; \; F_R(s) = \frac{2.5\tau_c\left(1 + 5\frac{\tau_c}{\tau_1}\right)s + 1}{5\tau_c s + 1} \;\; if\; \tau_1 > 5\tau_c$$

A setpoint filter $F_R(s)$ is introduced to reduce overshoot should the resulting time constant $\tau_1$ become larger than five times the tuning time constant $\tau_c$.
Using the formulas proposed in [21], it is also possible to derive PID parameters from a first-order approximation:

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c + \theta} \;\; ; \; \tau_i = \min\big(\tau_1, 5(\tau_c)\big) \;\; ; \; \tau_D = \max\left(\frac{\theta - \tau_c}{2}, 0\right)$$

For the PID controller following from the first-order approximation, the same setpoint filter as with the PI controller is recommended when the criterion $\tau_1 > 5\tau_c$ is true.

Finally, [21] proposes a set of formulas to obtain a PID controller from the second-order approximation using the listed formulas:

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c + \theta} \;\; ; \; \tau_i = \min\big(\tau_1, 5(\tau_c)\big) \;\; ; \; \tau_D = \tau_2 + \max\left(\frac{\theta - \tau_c}{2}, 0\right)$$

Again, the setpoint filter as mentioned above is recommended when $\tau_1 > 5\tau_c$

**Overview**

An overview of the (k-)SIMC formulas used for the model reduction can be found in Table 2.1. An overview of the formulas used to tune a PID controller given the first (FO) or second-order (SO) plus delay approximation can be seen in Table 2.2.

*Table 2.1 Overview of (K-)SIMC formulas for first- and second-order plus delay approximation [13], [21], [22]*

| | | $\tau_1$ | $\tau_2$ | $\theta$ |
|---|---|---|---|---|
| SIMC | FO | $\tau_{1,0} + \frac{\tau_{2,0}}{2}$ | - | $\theta_0 + \frac{\tau_{2,0}}{2} + \sum_{i\geq 3}\tau_{i,0} + \sum_j T_{j0}^{inv} + \frac{h}{2}$ |
| | SO | $\tau_{1,0}$ | $\tau_{2,0} + \frac{\tau_{3,0}}{2}$ | $\theta_0 + \frac{\tau_{3,0}}{2} + \sum_{i\geq 4}\tau_{i,0} + \sum_j T_{j0}^{inv} + \frac{h}{2}$ |
| K-SIMC | FO | $\tau_{1,0} + \frac{1}{2}\frac{\tau_{2,0}{}^2}{\tau_{1,0}}$ | - | $\theta_0 + \tau_{2,0}\left(1 - \frac{1}{2}\frac{\tau_{2,0}}{\tau_{1,0}}\right) + \sum_{i\geq 3}\tau_{i,0} + \sum_j T_{j0}^{inv} + \frac{h}{2}$ |
| | SO | $\tau_{1,0}$ | $\tau_{2,0} + \frac{1}{2}\frac{\tau_{3,0}{}^2}{\tau_{2,0}}$ | $\theta_0 + \tau_{3,0}\left(1 - \frac{1}{2}\frac{\tau_{3,0}}{\tau_{2,0}}\right) + \sum_{i\geq 4}\tau_{i,0} + \sum_j T_{j0}^{inv} + \frac{h}{2}$ |

*Table 2.2 Overview of (K-)SIMC PI(D) tuning formulas [13], [21], [22]*

| | | $K_c$ | $\tau_i$ | $\tau_D$ | $F_R(s)$ |
|---|---|---|---|---|---|
| SIMC | FO | $\frac{1}{k}\frac{\tau_1}{\tau_c + \theta}$ | $\min(\tau_1, 4(\tau_c + \theta))$ | - | - |
| | SO | $\frac{1}{k}\frac{\tau_1}{\tau_c + \theta}$ | $\min\big(\tau_1, 4(\tau_c + \theta)\big)$ | $\tau_2$ | - |
| K-SIMC | FO | $\frac{1}{k}\frac{\tau_1}{\tau_c + \theta}$ | $\min\big(\tau_1, 5(\tau_c)\big)$ | $\max\left(\frac{\theta - \tau_c}{2}, 0\right)$ | $if\ \tau_1 > 5\tau_c$ |
| | SO | $\frac{1}{k}\frac{\tau_1}{\tau_c + \theta}$ | $\min\big(\tau_1, 5(\tau_c)\big)$ | $\tau_2 + \max\left(\frac{\theta - \tau_c}{2}, 0\right)$ | $if\ \tau_1 > 5\tau_c$ |

## 2.3.2 Frequency loop shaping

Another approach to tuning a system is to use control system theory to derive what the open-loop behaviour of the system should be. This method, known as Frequency Loop Shaping (FLS), is explained in [26] and applied in [27], [28].

**Concept**

To derive what the open-loop behaviour of the system should be like, a base model is given in Fig. 2.12, inspired from lecture [29].

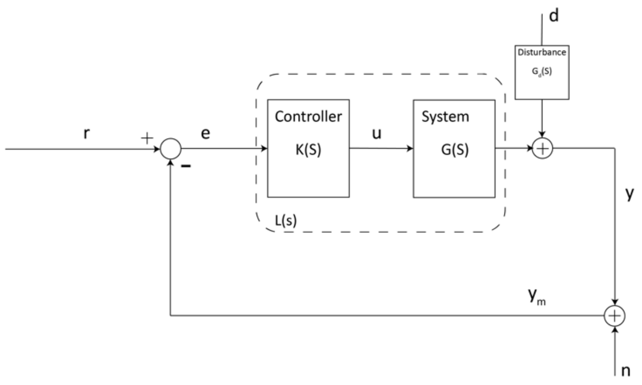


*Fig. 2.12 Base model for deriving closed-loop behaviour*

From the model in Fig. 2.11, the following parameters exist:

- r: The reference or set point
- e: The error signal
- u: The control signal
- d: Disturbance
- y: The effective output of the plant
- n: measurement noise
- $y_m$: The value of the plant after measurement noise occurs

And the following systems exist:

- K(s): The controller we aim to tune
- G(s): The plant under control, in this case, the DC-DC Buck controller
- $G_d$(S): optional transfer function of a disturbance, according to [29], this transfer function can be similar to the transfer function of the system

For this system, it can be written that:

$$y = G_d \cdot d + G \cdot K(r - y - n)$$

By bringing the y-term over to the left-hand side and rewriting the right-hand side:

$$(1 + GK)y = GKr + P_d d - PKn$$

Calculating for y:

$$y = (1 + GK)^{-1}GKr + (1 + GK)^{-1}G_d d - (1 + GK)^{-1}GKn$$

The sensitivity S and complementary sensitivity T can now be defined as follows:

$$S = (1 + GK)^{-1} = (1 + L)^{-1}$$

$$T = (1 + GK)^{-1}GK = (1 + L)^{-1}L$$

By substituting these equations in the equation for y, the meaning of the term sensitivity and complementary sensitivity can be derived:

$$y = T \cdot r + S \cdot G_d d - T \cdot n$$

It can be seen now that sensitivity S describes how sensitive the total system is to disturbances. The complementary sensitivity describes how sensitive the complete system is to noise and following the reference.

The feedback controller should be tuned to obtain the following qualities [29]:

- Stability
- Uncertainty compensation
- Disturbance rejection
- Noise attenuation

To provide stability and uncertainty compensation, the error function should always be as low as possible:

$$e = Sr - SG_d + Tn$$

For error function e to be low, the absolute value of S is preferred to be low for low frequencies. This allows for tracking the reference accurately and rejecting disturbances at low frequencies. As noise occurs on higher frequencies, it is preferred for T to be low for high frequencies. From the formulas defining the sensitivity and complementary sensitivity, it can be concluded that the sum of T and S is always 1. This means a trade-off needs to be made for keeping S small at low frequencies and keeping T small at high frequencies. This trade-off can be seen in Fig. 2.13, the point where the values of S and T cross is the cut-off frequency $\omega_c$. The controller and plant should act upon signals with a frequency lower than the cut-off frequency to track the reference and reject disturbances. Signals with a frequency higher than the cut-off frequency should be attenuated to reduce the effect of noise. The transfer function of an integrator ($\omega_c$/s) provides this trade-off. Therefore [26] suggests using the transfer function of an integrator as a target transfer function for the control loop (L).

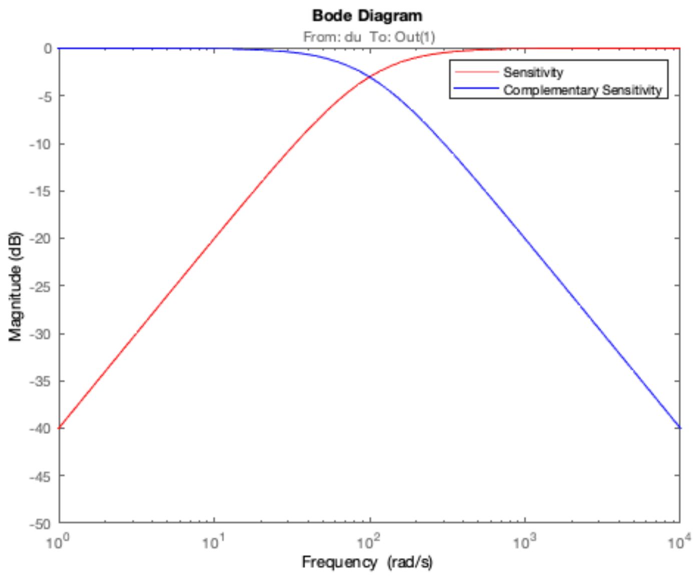


*Fig. 2.13 Sensitivity S vs complementary sensitivity T for an integrator with $\omega_c$ = $10^2$ rad/s*

## Choosing the transfer function

The choice of the target transfer function is crucial as it determines the frequency response of the open-loop system as a whole (L). A target transfer function of an integrator has already been suggested for having desirable sensitivity and complementary sensitivity functions:

$$L_0(S) = \frac{\omega_c}{s}$$

Besides the integrator [26], the following transfer function is suggested to be used when the plant under control has a slow pole with a time constant larger than that of the desired closed-loop system as using the integrator would result in a PD-like compensator:

$$L_0(S) = \frac{\omega_s(S+a)}{s(s+e)}$$

In the above equation, 'e' represents the small pole of the plant, and a is a tuning parameter balancing settling-time and overshoot properties, a default value of $0.25\omega_s$ is suggested.
In both suggested transfer functions, $\omega_s$ should be chosen to keep in mind two limitations. First, stability requirements and expected noise frequency provide an upper bound constraint. Second, maximum settling time to step response and disturbances can form a lower bound constraint.

**Calculating control function**

When a target transfer function is obtained, the tuning of the controller can be derived. [27] suggest writing the transfer function of the PID controller with a low pass filter in the derivative in the following form:

$$C(s) = \frac{K_1 s^2 + k_2 s + k_3}{s(\tau s + 1)}$$

In this formula, $\tau$ is the prior selected time-constant of the low pass filter whilst the k parameters relate to the PID controller parameters in the following fashion:

$$K_p = K_2 - K_3\tau \ ; \ K_i = K_3 \ ; \ K_d = K_1 - K_2\tau + K_3\tau^2$$

Now using this control formula, the following optimisation formula can be proposed

$$\min_{k1,k2,k3} \|W_1 \cdot S_0 \cdot (GC - L)\|_{\mathcal{L}\infty}$$

This formula introduces a weighing parameter $W_1$ that can be used to emphasise the approximation around the crossover frequency [26] suggests the following formula for this weighing parameter:

$$W_1 = \frac{s}{s + \omega_s{}'}$$

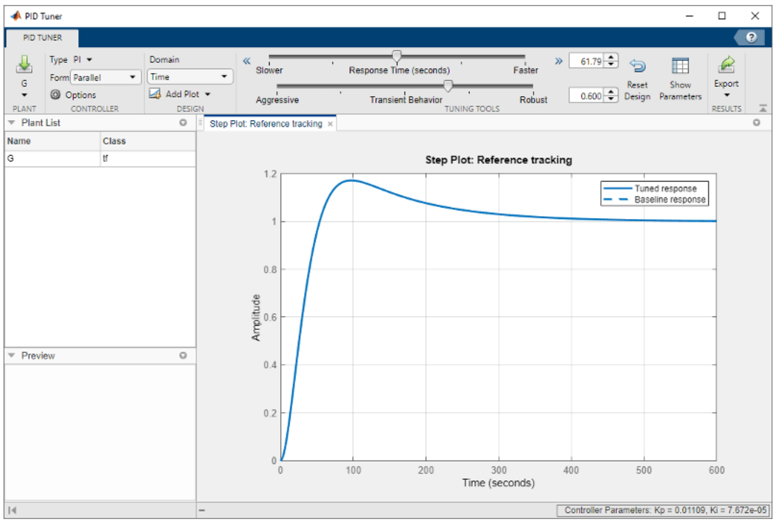
[26] Suggests using $\omega_s{}' = 0.1\omega_s$ to define the weighing function.
This optimisation problem can be further defined by adding the PID parameters' positivity as a constraint.
When the minimum of the optimisation problem results in a value close to zero, the resulting loop is close to the selected target loop and thus inherits its robustness properties. [26] suggest a minimum objective value of < 0.2 should mean the properties are approximately preserved whilst larger values (higher than 0.5) allow for unacceptable performance deterioration and recommends retrying with a smaller cut-off frequency.

### 2.3.3 MATLAB PID tuning applications

MATLAB's control system toolbox [30] offers two tools that can be used to tune a PID controller. Unfortunately, the algorithms used in this process are proprietary to MATLAB, and little information about this is made publicly available.

**Transfer function tuner (PID tuner app)**

The PID tuner app from MATLAB allows tuning controllers via two parameters: the response time (in seconds) and the transient behaviour on a scale from aggressive to robust.
The PID tuner allows for several plots to see the controller's influence on the plant. These include but are not limited to the step response for reference tracking, the step response for disturbance rejection and bode plots. The PID tuner application can be seen in Fig. 2.14.



G

Type PI

Form Parallel

Domain Time

Response Time: 61.79

Transient Behavior: 0.600

| Name | Class |
|---|---|
| G | tf |

Controller Parameters: Kp = 0.01109, Ki = 7.672e-05

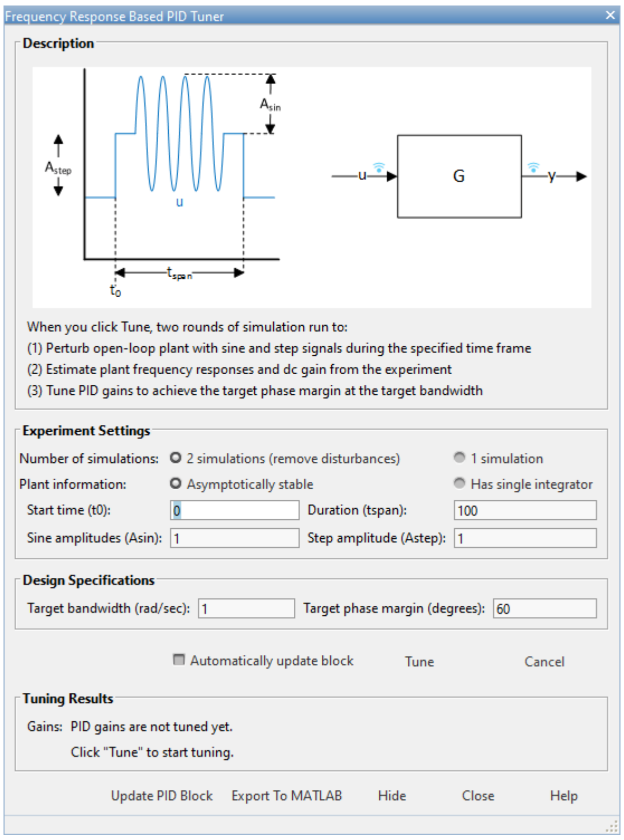
*Fig. 2.14 PID tuner application from MATLAB*

**Frequency-response based tuner**

The frequency response-based tuner from MATLAB allows tuning a PID controller by setting a target bandwidth and target phase margin. According to [31], the tuner performs the following steps:

- It breaks the feedback loop at the controller output and simulates the model. A sinusoidal perturbation signal is applied to the plant.
- The response of the perturbation signal is measured at the input of the PID controller.
- The data obtained from this experiment is used to estimate the plant frequency response. For stable plants, the tuner will also use the results to estimate the DC gain of the plant.
- The estimated frequency response is used to compute PID gains that should balance performance and robustness

The frequency response-based PID tuner can be seen in Fig. 2.15.



When you click Tune, two rounds of simulation run to:
(1) Perturb open-loop plant with sine and step signals during the specified time frame
(2) Estimate plant frequency responses and dc gain from the experiment
(3) Tune PID gains to achieve the target phase margin at the target bandwidth

Experiment Settings

Number of simulations: [✓ 2 simulations (remove disturbances)] 1 simulation
Plant information: [✓ Asymptotically stable] Has single integrator
Start time (t0): 0 Duration (tspan): 100
Sine amplitudes (Asin): 1 Step amplitude (Astep): 1

Design Specifications

Target bandwidth (rad/sec): 1 Target phase margin (degrees): 60

Tuning Results

Gains: PID gains are not tuned yet.
Click "Tune" to start tuning.

*Fig. 2.15 MATLAB Frequency-response based PID tuner [31]*

## 2.4 Limit cycle oscillations

As [32] explains, LCO (Limit Cycle Oscillation) is a problem associated with digitally controlled converters due to the quantisation effects of the ADC and the PWM modulator.

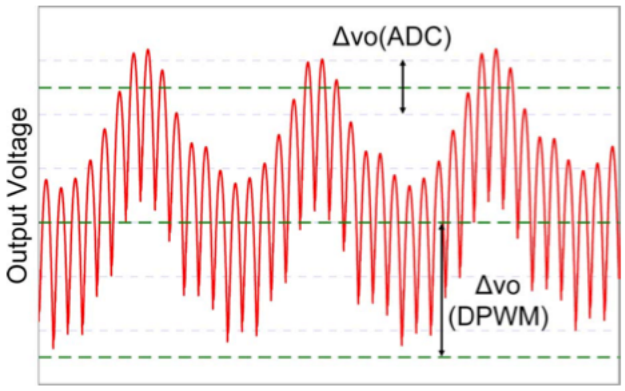


*Fig. 2.16 Simulated example of LCO due to too coarse DPWM [32]*

Fig. 2.16 shows LCO as a result of too coarse DPWM. The controller will, in this case, continuously switch between two control states. Yet, due to the quantisation of the DPWM, the state needed to measure no error can never be obtained, resulting in oscillation.
According to [33], LCO can become significant when the integrator gain is set to high or when the quantisation step of the DPWM model is too coarse. An inequality that must be true to prevent LCO is suggested as follows:

$$\Delta d_{LSB}\, V_{IN} < \Delta V_{Out\_LSB}$$

This equation states that the value of the least significant bit (LSB) of the duty cycle multiplied with the input voltage should be lower than the LSB value of the ADC signal measuring the output voltage. However, it should be noted that complying with this inequation can result in a very high DPWM resolution. Using a high DPWM resolution with a traditional DPWM module will require very high clock frequencies that might not be attainable. Careful consideration between LCO and clock speed will need to be made during this project.

## 2.5 High-frequency DPWM

To prevent LCO, the resolution of the digital pulse width modulator should be higher than the measuring resolution of the ADC. This section will consider several digital pulse width modulator designs and evaluate the best fit for this project.

### 2.5.1 Sawtooth comparator DPWM

A sawtooth comparator DPWM, shown in Fig. 2.17, works by implementing a counter and a comparator.

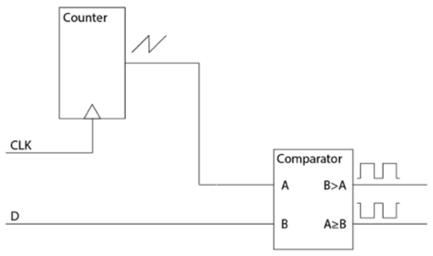


*Fig. 2.17 Sawtooth comparator digital pulse width modulator*

This implementation requires little hardware but requires a high clock frequency when a large resolution is needed.
The clock frequency ($f_{clk}$) required to generate a PWM signal with a set switching frequency ($f_{switch}$) can be calculated with the following formula:

$$f_{clk} = f_{switch} \cdot 2^{resolution(D)}$$

For example, a duty-cycle resolution of 8 (fractional) bits, making for a control resolution of 390mV, would result in a clock frequency of 256 MHz. This example shows that a relatively high clock frequency is required for a relatively low control resolution of 390 mV. This method can quickly result in required clock frequencies into the GHz range making this output stage a potential bottleneck for control accuracy. This effect can be seen in Fig. 2.18.

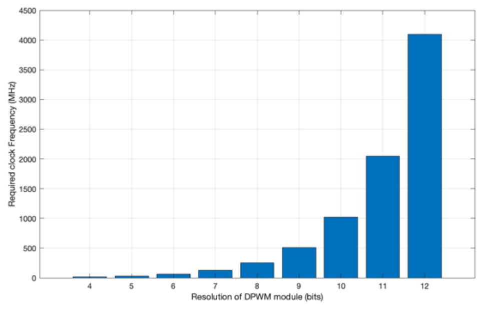


*Fig. 2.18 Required clock speed in function of the control resolution in bits*

### 2.5.2 Multibit Σ–Δ generator

An attempt to solve the need for extreme clock frequencies is made by the multibit Σ–Δ generator suggested for use in switching power converters [16].

**Principle of operation**

The concept of the Σ–Δ generator is explained by using a first-order Σ–Δ generator, as seen in Fig. 2.19.

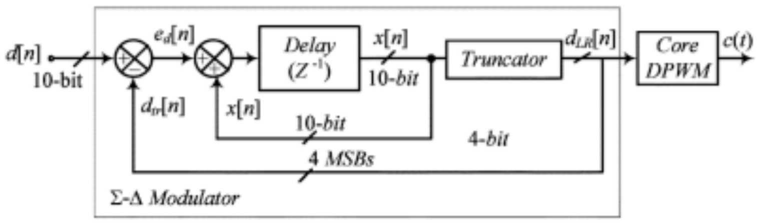


*Fig. 2.19 First order multibit Σ–Δ generator [16]*

The first order Σ–Δ generator needs to be paired with another method of DPWM generation, assuming the sawtooth generator is used. The Σ–Δ modulator takes a higher resolution duty-cycle input and outputs a lower resolution duty-cycle output called the effective resolution. This effective resolution output is forwarded to the traditional sawtooth-comparator DPWM. However, as the duty cycle resolution of the signal fed into the DPWM is lower, the clock cycle can be lower.
The Σ–Δ modulator varies the least significant bit of the effective resolution over several switching cycles to obtain an average value equal to the higher resolution input signal.
For example, assume a Σ–Δ generator that translates a 3-bit input resolution into a 2-bit effective resolution: a binary input signal of 0.101 is translated into a continuous variation between 0.10 and 0.11, ensuring the average output equals the input signal. This is done by recursively feeding the difference until the sum of the differences is as large as 1 LSB of the effective resolution.
According to [16], no additional hardware is required to average the output signal as the filtering components in the power stage will have this effect.
The hardware cost of integrating a first-order multibit Σ–Δ generator is relatively low as the modulator only consists of two adders and a register.

The first order multibit Σ–Δ generator can be modelled as a control loop as seen in Fig. 2.20.

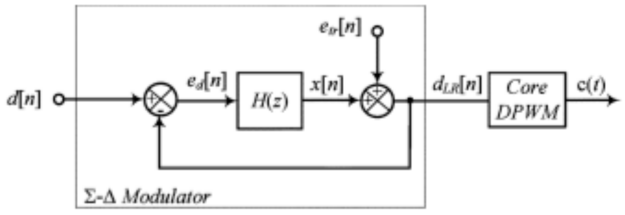


*Fig. 2.20 multibit Σ–Δ generator equivalent model [16]*

The control loop consists of a transfer function from the generator added together with truncation noise $e_{tr}$. The difference between the target duty cycle d(n) and the effective resolution duty cycle $d_{LR}(n)$ is noted as $e_d(n)$.
The transfer function of the generator is as follows [16]:

$$H(z) = \frac{X(z)}{E_d(Z)} = \frac{Z^{-1}}{1 - Z^{-1}}$$

The integrating effect of the inner loop can be noted from the transfer function, forcing the value of $e_d(n)$ to 0.

The first order Σ–Δ generator allows increasing the resolution of the controller but still requires a core DPWM module with a relatively high effective resolution. Moreover it suffers from low-frequency noise at the converter output and has slow convergence towards the target average [16], [34]. The slow convergence of the first order Σ–Δ generator limits the update frequency of the target duty-cycle, resulting in a significant limitation for the bandwidth.

### Second-order multibit Σ–Δ generator

To limit the negative effects of the first order Σ–Δ  generator, a second-order multibit Σ–Δ generator is suggested in [16]. It shows that a second order Σ–Δ strongly suppresses the low-frequency tones caused in a first-order Σ–Δ generator and offers faster convergence. The schematic for a second-order Σ–Δ generator can be seen in Fig. 2.21. This generator consists of two registers, three adders, and a multiplication by two (which can be realised as a bit shift).

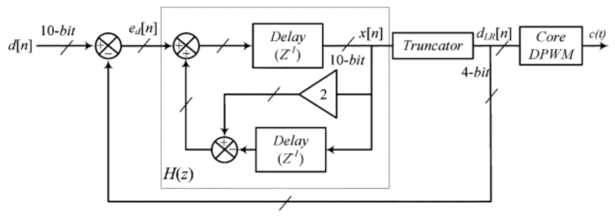


*Fig. 2.21 Second-order multibit Σ–Δ generator*

The second-order multibit Σ–Δ can also be modelled as a transfer function [16]:

$$H(z) = \frac{z^{-1}}{(1 - z^{-1})^2}$$

Although the second-order multibit Σ–Δ generator converges much faster than the first order Σ–Δ generator, there is still some delay for the controller to reach the correct averaged output. The following formula, derived from noise-shaping analysis, allows calculating the frequency the PID controller should function at for proper averaging to take place [16]:

$$N_{DPWM} \approx N_{core} + 2.5 \log_2 \left( \frac{f_{sw}}{f_{update}} \right)$$

In this formula $N_{DPWM}$ represents the total input resolution, $N_{core}$ the effective resolution, $f_{sw}$ the clock frequency of the PID controller and $f_{sw}$ the switching frequency. Although the second-order generator allows for much higher control frequencies than the first order generator, it would still have a significant negative impact on the dynamic performance of the controller.

**Dual-mode compensator**

To prevent the negative impact on the dynamic performance, introduced by a limited updating frequency, [16] suggests using a dual-mode compensator to solve this issue. A steady-state mode with clock speed calculated by the above formula will cause very accurate control using the Σ–Δ generator. However, when the absolute value of the error signal exceeds a threshold, the converter switches to dynamic mode with a clock frequency equal to the switching frequency, allowing for a better transient response. [16] also suggest only changing back to steady-state when the absolute value of the error has been lower than the threshold for a set duration of time as switching immediately to steady-state when the error value is within a specific error band was found to cause stability issues. A finite state machine implementing this can be seen in Fig. 2.22.

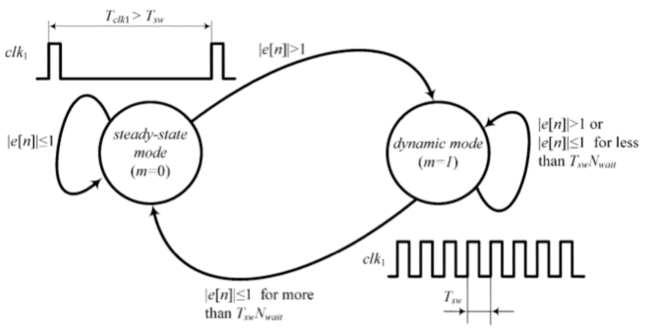

$T_{clk1} > T_{sw}$
$clk_1$
$|e[n]|>1$
$|e[n]|\leq 1$
steady-state mode ($m=0$)
dynamic mode ($m=1$)
$|e[n]|>1$ or $|e[n]|\leq 1$ for less than $T_{sw}N_{wait}$
$clk_1$
$|e[n]|\leq 1$ for more than $T_{sw}N_{wait}$
$T_{sw}$

*Fig. 2.22 Finite State Machine representation of dual-mode compensator [16]*

## 2.5.3 LUT Dither generator

An LUT Dither generator achieves the same effect as the Σ–Δ generator: it modulates the least significant bit to create a virtually higher resolution.
However, instead of mathematically determining the modulation, a LUT, fed by the virtual resolution bits and a clock, is used to determine the modulation needed to become an average signal equal to the virtual resolution [32].
This method yields the same requirements and limitations as the Σ–Δ generator. Some cycles are needed to obtain a correct average signal, reducing the bandwidth. The converter also needs to be capable of averaging the signal sufficiently to compensate for the variations created by the dither generator.

When the virtual resolution is relatively low (e.g. one or two bits), the LUT generator could prove a more efficient implementation to increase the total resolution. However, as the Σ–Δ generator can be used with a broader range of virtual resolutions, this implementation proves more versatile for testing modulating PWM generators and evaluating the averaging capabilities of the system.

## 2.6 Clock domain crossing

The PWM generator will need to function at a significantly higher clock resolution than the PID logic to obtain the required switching frequency. The controller under design will most likely need to use two clock domains to realise the controller.

When transmitting data from one clock domain to another, metastability or data loss could occur. Take for example, two flip flops clocked in a different clock domain, as shown in Fig. 2.23a.
When the rising edge of clock 2 occurs closely after the rising edge of clock 1, the output of the first flip flop (Q1) could still be in the process of transferring to its next state whilst the second flip flop attempts to sample this signal. This would make it unpredictable what the output of the second flip flop (Q2) would be. It could either settle to the new or the old value. This state of uncertainty is called metastability. The described procedure can be seen in Fig. 2.23b.

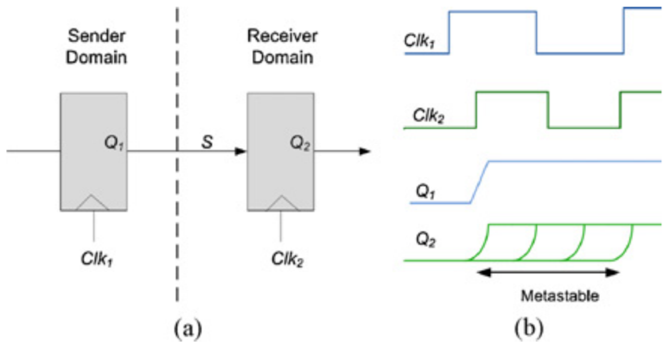


*Fig. 2.23 Example of metastability (a) Example system (b) Metastability on Q2 [35]*

This metastable state can be prevented by using a synchroniser in the destination clock domain. The second synchroniser samples Q2 after it has been stabilised, allowing a stable output out of the synchroniser. An implementation of such a synchroniser can be seen in Fig. 2.24.

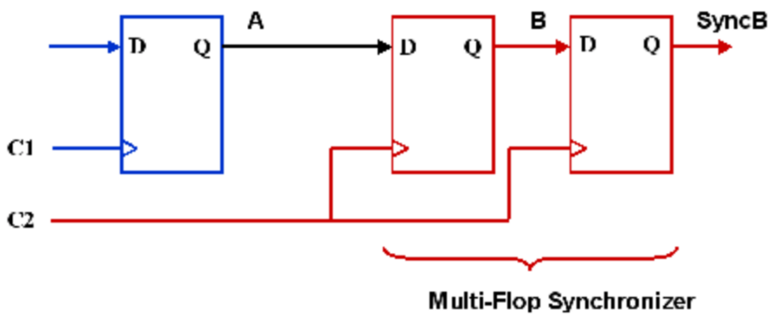


*Fig. 2.24 Clock-domain-crossing synchroniser [36]*

# 3 Materials and methods

## 3.1 MATLAB Simulink

MATLAB is a programming and numeric computing platform from MathWorks. It enables the user to calculate and automate calculations using various built-in functions.
Simulink is another tool from MathWorks that is integrated into MATLAB. Simulink functions as a platform for model-based design, allowing simulation of control logic and other components without the need for programming. An example block schematic from a Simulink project can be seen in Fig. 3.1.

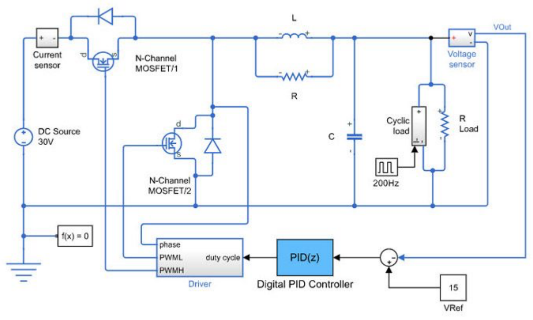


*Fig. 3.1 Simulink example schematic [37]*

During the development of the controller, Simulink was used to simulate the DC-DC buck-converter and to validate potential control strategies. Simulink allows for analysing the resulting system using virtual scopes with measurement tools to measure a variety of benchmarks (e.g., rise-time, settling time, etc.).
Simulink can also be used to verify the functionality of HDL blocks by either FPGA-in-the-loop (FIL [38] or by co-simulating the VHDL program in a simulator like ModelSim [39].
In the first setup, Simulink feeds data to an FPGA that processes it and sends it back to Simulink. The FPGA component can then be incorporated within a Simulink schematic. This could test the controller on a simulated buck converter as the physical converter was still being designed. In this setup, Simulink parses the output voltage signals to the FPGA, calculating a control directive and modulating a PWM signal that is sent back to Simulink.
In the second setup, MATLAB's Simulink will connect to an HDL simulator and then function in a similar method as described above: the output voltage is transmitted to the HDL simulator, which processes this, generates a PWM signal which is then sent back to the Simulink simulation.

## 3.2 Xilinx Vivado

Vivado Design Suite is a software suite from Xilinx that synthesises HDL (Hardware Description Language) designs, implement the design for a specific FPGA and allows downloading a bitstream to the FPGA. Vivado also allows behavioural verification of designed HDL components by designing a testbench and simulating the designs with specified input/stimulations as Field [40] described. An example of such a simulation can be seen in Fig. 3.2.

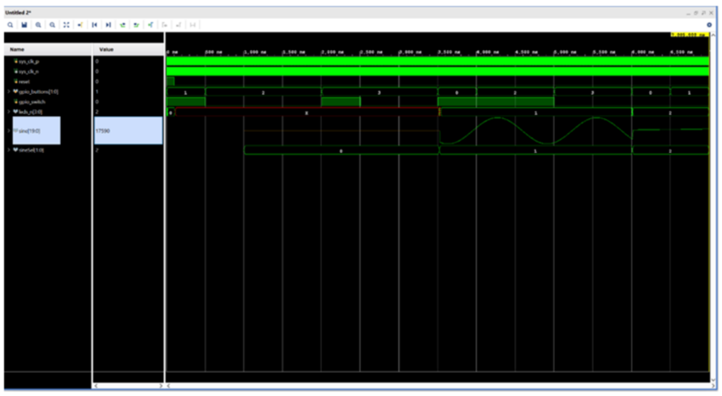


*Fig. 3.2 Xilinx Vivado Simulation [40]*

Vivado was used to develop the HDL-code for the controller and subsequently to analyse if this implementation behaved as expected on a signal level.

## 3.3 Mentor Graphics ModelSim

ModelSim is a software program from Mentor Graphics that allows for the simulation of hardware description designs like VHDL.
ModelSim was used in this thesis for co-simulation with MATLAB Simulink, allowing for verifying the controller’s VHDL design.

It is important to use a version of ModelSim that is compatible with the HDL-verifier toolbox from MATLAB [41]. The version of ModelSim used in this thesis is ModelSim SE-64 2020.4.

# 4 Experimental and results

This chapter will provide insight into how the design of the controller was tested and optimised. As the DC-DC buck converter is still under development, provisional parameters will be used during this project. The provisional parameters are obtained from [42] and are listed in Table 4.1.

*Table 4.1 DC-DC buck converters parameters*

| | L (µH) | C (µC) | Target power (W) |
|---|---|---|---|
| 100V to 48V | 32.8 | 0.39 | ~ 100 W |
| 100V to 24V | 22 | 0.47 | ~ 100 W |

Although these parameters might still be subject to change, the design of the controller will be made based on these parameters. Should a significant change occur, then the methodology of this thesis can be used to re-design the controller accordingly. A detailed description of what steps need to be repeated when a parameter should change can be found in chapter 6: future work.

## 4.1 MATLAB Simulink setup

To be able to test different designs, a simulation environment was created. This environment must simulate the behaviour of the buck converter as the converter was still being designed at the moment of writing the thesis. First, two methods of simulating the buck converter are provided, then three methods of simulating the controller design are described.

### 4.1.1 DC-DC converter simulation

**SimScape electrical**

The SimScape electrical plugin for Simulink models the DC-DC buck converter as an electrical schematic. The schematic used can be seen in Fig. 4.1.

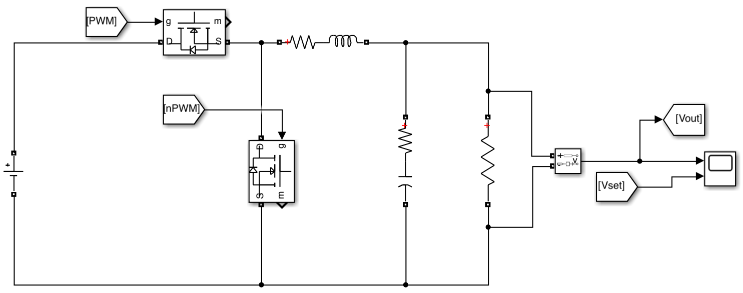


*Fig. 4.1 Simulink SimScape electrical schematic for simulating the DC-DC buck converter*

This setup allows for accurate simulation of the behaviour of the buck converter. The inputs this model expects are a PWM signal and an inverted PWM signal. The inputs expected are either a logical high or a logical low. This model outputs the resulting voltage to be used by the controller.

The advantages of this model are that it allows for accurate simulation and that it expects an actual PWM signal as input. The drawbacks of this method are that this model allows for little mathematical analysis and has as a side effect that some of MATLAB's automated tools cannot be used. Another minor issue is that simulating this schematic with a high switching frequency makes the simulation relatively slow.

**Transfer function**

It is also possible to reduce the electrical schematic to a transfer function using a time averaging model, as described in [43]. The result of such derivation can be found in [44] and gives the following transfer function as a system that outputs a voltage when a duty cycle is applied:

$$\frac{\Delta V_{out}}{\Delta D} = \frac{V_{in}}{LCs^2 + \frac{L}{R}s + 1}$$

When filling in the 48V parameters from Table 4.1, the following transfer function is obtained:

$$\frac{\Delta V_{out}}{\Delta D} = \frac{100}{32.8 \cdot 10^{-6} \cdot 0.39 \cdot 10^{-6} s^2 + \frac{32.8 \cdot 10^{-6}}{R} s + 1} = \frac{100}{1.2792 \cdot 10^{-11} s^2 + \frac{32.8 \cdot 10^{-6}}{R} s + 1}$$

This approximation can be verified by simulating the step response of the transfer function and comparing it to the schematic's step response. A schematic with which this can be achieved is shown in Fig. 4.2.

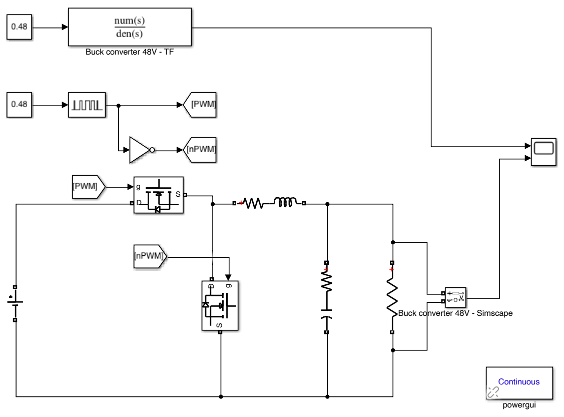


*Fig. 4.2 MATLAB Simulink schematic used for verifying TF-approximation*

In Fig. 4.2, a unit step with an amplitude of 0.48 is applied to both systems with a load resistance of 23.04 Ohm (P=100W). In Fig. 4.3, the step response of both systems is shown.

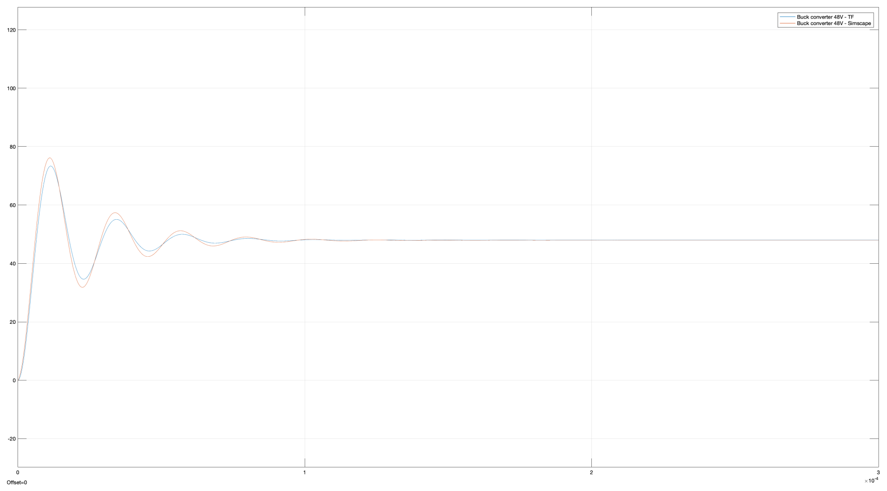


*Fig. 4.3 Step response of DC-DC Buck converter compared to step response of TF-approximation*

From Fig. 4.3, we can conclude that there are slight differences between the step response of the DC-DC buck converter and the transfer function approximation. These differences can be attributed to simplifications in the mathematical model. For example, TF-approximation does not account for the inductor's serial resistance or the capacitor's resistance.

It should also be noted that the transfer function does not consider the physical limitations of the buck converter, meaning that if an input higher than one (translating to a duty cycle higher than 100%, which is physically impossible) is applied, then the output (translating to the output voltage of the converter) would be higher than the input voltage.
To prevent this, a saturation module is applied before the transfer function, as shown in Fig. 4.4.

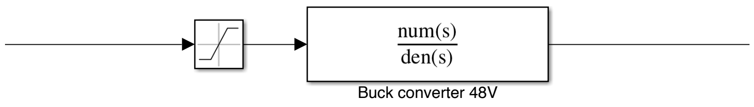


*Fig. 4.4 Saturation module in series with the TF of the buck converter*

A buck converter cannot function at a 100% duty cycle as this would mean the output voltage of the buck converter would equal the input voltage. As some switching losses occur, this is impossible. The maximum duty cycle of the DC-DC buck converter can be calculated with the following formula from [45]:

$$D_{max} = \frac{V_{OUT,max}}{V_{IN}} \text{ and } V_{OUT,max} = V_{in,max} \cdot \eta$$

However, as the efficiency (η) is not known at the moment of writing this thesis, [45] suggests using 90%. This would result in a duty-cycle range of [0; 0.9]. Hence this is the range the saturation block in Fig. 4.4 is configured to use.

### 4.1.2 PID controller simulation

**PID control block**

The most obvious way of simulating the PID controller is using the MATLAB Simulink discrete PID block. The sampling frequency of this block can be manually set, and using this block allows the use of the PID tuning applications. This implementation can be seen in Fig. 4.5

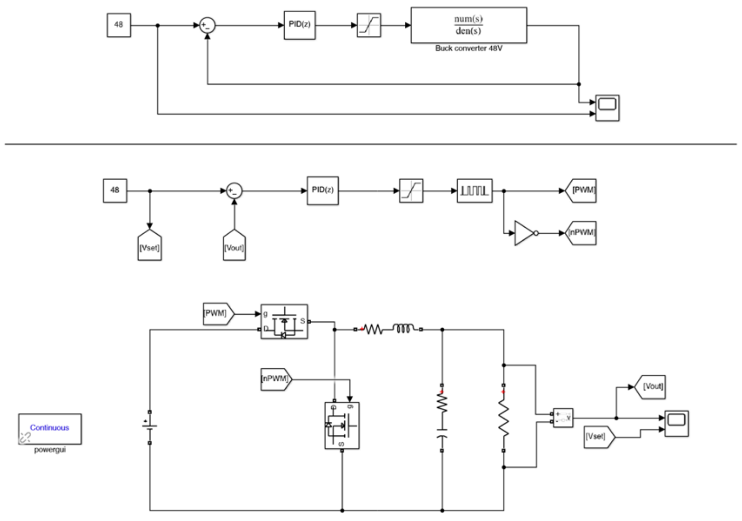


*Fig. 4.5 MATLAB Simulink schematics with PID control block a) using buck converter TF (above) b) using buck converter SimScape schematic (below)*

This theoretical discrete PID block allows for testing tuning parameters at fixed sampling frequencies, functioning as a reference for comparing the VHDL PID.

**MATLAB SIMULINK function block**

A MATLAB function block was written to test the implementation of a PID controller that will eventually be converted into a VHDL implementation. The code contained in this function block can be found in appendix A. A schematic using this control block in a control loop can be seen in Fig. 4.6. It should be noted that the parameters are not yet adequately tuned.

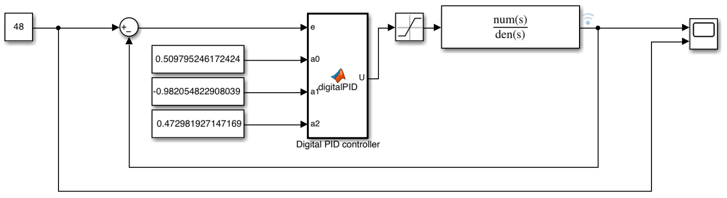


*Fig. 4.6 MATLAB Simulink schematic control loop using the MATLAB function block*

The MATLAB function block allows the verification of the PID implementation described in section 2.1.1. When plotting the system's closed-loop response in Fig. 4.6 compared to the system in Fig.

4.5a using the same tuning parameters and a resistance R = 23.04 Ohm, the graph shown in Fig. 4.7 is obtained.

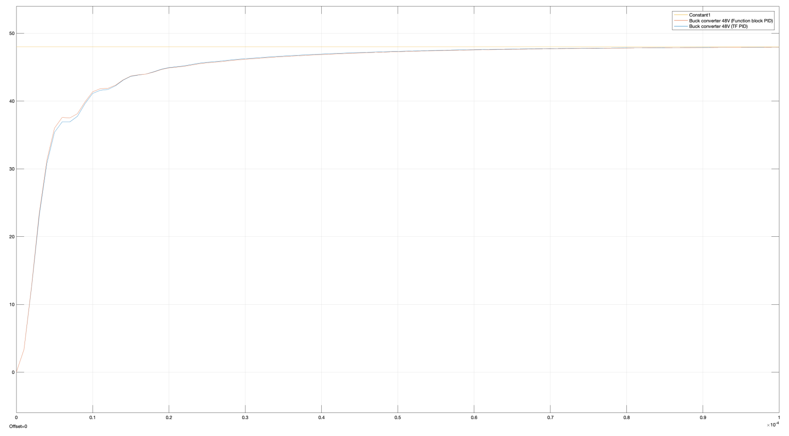


*Fig. 4.7 Closed-loop step response of Simulink PID block compared to MATLAB function block*

Fig. 4.7 shows that the step response of the function block and the PID block are practically the same. The minor difference can be assigned to accuracy limits in calculating the parameters. This confirms that the model discussed in section 2.1.1 is a usable and correct implementation of the PID model.
Besides verifying the implementation, the MATLAB function block can be used to easily test potential improvements to the controller by altering the MATLAB code in the function block.

### Separate component model

The final model used to simulate the PID controller in this thesis is the separate component model. This model describes the entire functioning of the PID controller as basic Simulink blocks. The implementation of this can be seen in Fig. 4.8.

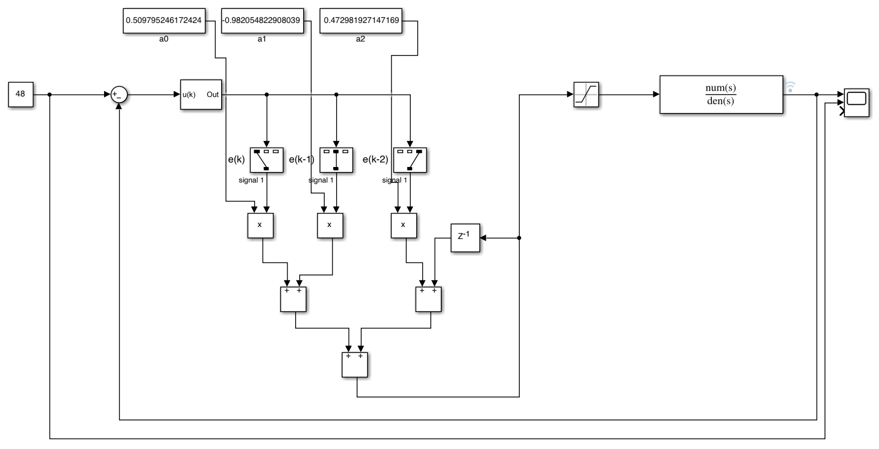


*Fig. 4.8 MATLAB Simulink PID control loop using the separate component model*

This setup, as shown in Fig. 4.8, makes it easy to analyse the minima and maxima of each of the signal lines and helps to analyse the effects of quantisation on the controller.

## 4.2 Controller tuning

Next, the controller should have optimised tuning parameters, as tuning can improve the controller's behaviour without requiring additional resources. The three methods from the literature study ([K-]SIMC, Frequency loop shaping, and MATLAB autotune) are implemented and compared to select a tuning that yields optimal results. When analysing the tuning methods, some trade-offs between reference following and noise attenuation have to be made. [28] suggests opting for a relatively high bandwidth when tuning a controller for a DC-DC Buck converter to give an advantage for reference following over noise attenuation. The tuning methods are explored using the parameters of the 48V DC-DC buck converter as described in Table 4.1. A viable tuning must be found to control the converter stably under various load resistances. For this analysis, we consider the load resistance for a total power range of 100W (the maximum target wattage of the controller) down to 1nW.
Then, when an optimal tuning method is found, this method will also be applied for the 24V DC-DC buck converter.

### 4.2.1 (K)-SIMC Controller tuning

First, the SIMC and K-SIMC methods are considered for tuning the controller using the process described in section 2.2.1. First, a first- or second-order plus delay approximation of the plant will be determined. Next, the PID gain parameters will be derived from this model approximation.
For all the SIMC sub-methods, the plant system needs to be written into the following form:

$$G(s) = \frac{\prod_j \left(-T_{j0}^{inv} + 1\right)}{\prod_i \tau_{i0} s + 1} e^{-\theta_0 s}$$

The maximum resistance for our plant to be rewritten to this form is 4.585 Ω as when the resistance increases, there are no longer real zeros to factor in the denominator. Using this resistance, the following transfer function is obtained for the plant:

$$G(s) = \frac{V_{in}}{L \cdot C \cdot s^2 + \frac{L}{R} s + 1} = \frac{100}{32.8 \cdot 10^{-6} \cdot 0.39 \cdot 10^{-6} \cdot s^2 + \frac{32.8 \cdot 10^{-6}}{4.585} s + 1}$$

After factoring the denominator, the following transfer function is obtained:

$$G(s) = \frac{100}{1.2792 \cdot 10^{-11} (s + 276054.3487)(s + 283182.7176)}$$
$$= \frac{100}{\left(\frac{1}{276054.3487} s + 1\right) \left(\frac{1}{283182.7176} s + 1\right)}$$

From this form, the following time constants are derived:

$$\tau_{1,0} = \frac{1}{276054.3487}$$

$$\tau_{2,0} = \frac{1}{283182.7176}$$

## A) SIMC

### First-order

Using the time-constants that are derived above, the first order plus delay approximation of the SIMC method can be obtained using the formulas from Table 2.1**:**

$$\tau_1 = \tau_{1,0} + \frac{\tau_{2,0}}{2} = \frac{1}{276054.3487} + \frac{1}{2 \cdot 283182.7176} = 5.3881 \cdot 10^{-6}$$

$$\theta = \theta_0 + \frac{\tau_{2,0}}{2} + \sum_{i \geq 3} \tau_{i,0} + \sum_j T_{j0}^{inv} + \frac{h}{2} = \frac{1}{2 \cdot 283182.7176} = 1.7656 \cdot 10^{-6}$$

This forms the first order plus delay approximation:

$$G_a(s) = k \frac{e^{-\theta s}}{\tau_1 s + 1} = 100 \frac{e^{-1.7656 \cdot 10^{-6} s}}{5.3881 \cdot 10^{-6} s + 1}$$

This approximated transfer function can be verified by comparing the open-loop step response of the approximation and the original plant, as shown in Fig. 4.9.

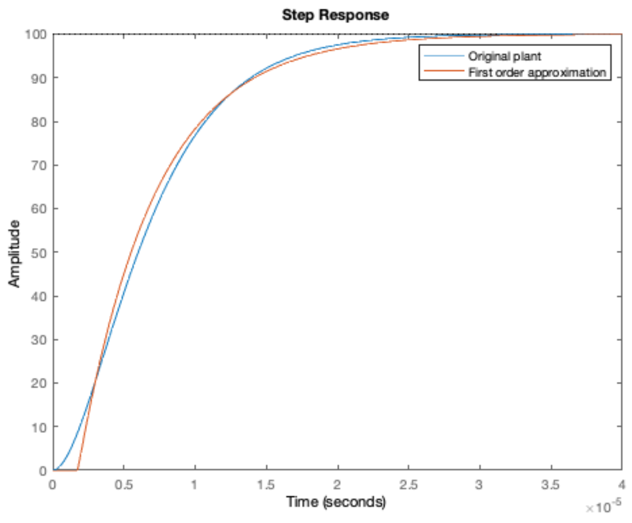


*Fig. 4.9 Comparison of open-loop step response of the original plant vs the first order plus delay approximation*

Fig. 4.9 shows that the open-loop step responses of the first order plus delay approximation is similar to that of the original plant. This confirms that the approximation has been made correctly.

Using the SIMC method, only a PI controller can be derived from a first-order approximation. This can be done using the formulas from Table 2.2:

$$K_C = \frac{1}{k} \frac{\tau_1}{\tau_c + \theta} = \frac{1}{100} \frac{5.3881 \cdot 10^{-6}}{\tau_c + \theta}$$

$$\tau_i = \min(\tau_1, 4(\tau_c + \theta)) = (5.3881 \cdot 10^{-6}, 4(\tau_c + 1.7656 \cdot 10^{-6})$$

[22] suggested using $\tau_c = \theta$. However, this resulted in significant overshoot, $\tau_c = \theta * 2.5$ provided a more viable alternative. The step response of the closed-loop control system using $\tau_c = \theta$ and $\tau_c = \theta * 2.5$ are shown in Fig. 4.10.

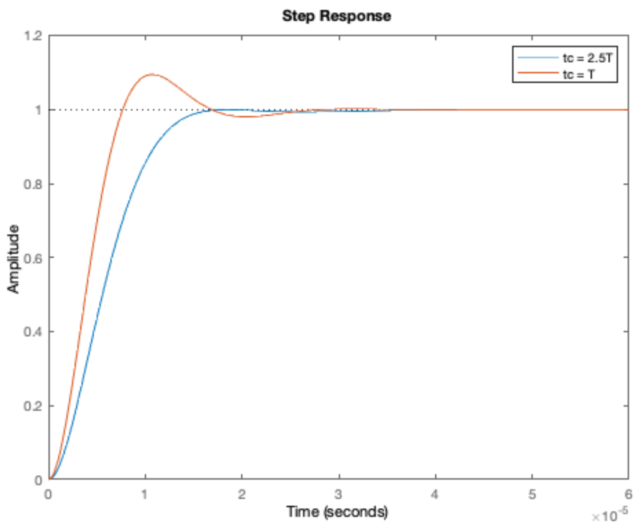


*Fig. 4.10 closed-loop step response of PI controller using $\tau_c = \theta$ and $\tau_c = \theta * 2.5$*

The closed-loop step response shown in Fig. 4.10 shows that using a larger tuning-time constant results in significantly less overshoot, a slightly lower settling time but a somewhat higher rise time. The exact benchmarks of the closed-loop step responses can be found in Table 4.2

*Table 4.2 Closed-loop benchmarks of SIMC FO PI controller*

| | $\tau_c = \theta$ | $\tau_c = \theta * 2.5$ |
|---|---|---|
| Rise time (s) | 5.1418e-06 | 9.0600e-06 |
| Settling time (s) | 1.5469e-05 | 1.4303e-05 |
| Overshoot (%) | 9.43 | 0 |

The information from Fig. 4.10 and Table 4.2 makes the PI controller with $\tau_c = \theta * 2.5$ a more interesting choice due to the lack of overshoot and a faster settling time.

When the system's resistance is altered so that the power consumption of the DC-DC converter equals 1W (R = 2304Ω), the system behaves as can be seen in Fig. 4.11.

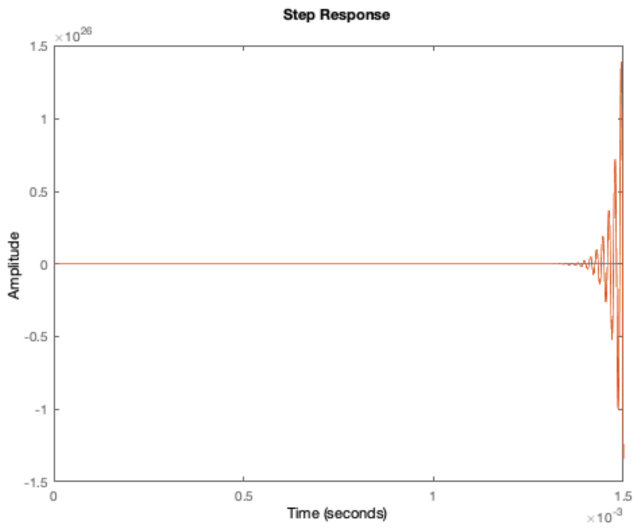


*Fig. 4.11 Closed-loop step response of PI controller when a larger resistance is applied ($\tau_c = \theta * 2.5, R = 2304\ Ohm$)*

Fig. 4.11 shows that the closed-loop step response of the PI controller is unstable. This makes the obtained tuning unfavourable for the design.

**Second-order**

Using the time-constants that are derived above the second order plus delay approximation of the SIMC method can be obtained using the formula's form Table 2.1**:**

$$\tau_1 = \tau_{1,0} = \frac{1}{276054.3487} = 3.6225 \cdot 10^{-6}$$

$$\tau_2 = \tau_{2,0} + \frac{\tau_{3,0}}{2} = \frac{1}{283182.7176} + \frac{0}{2} = 3.5313 \cdot 10^{-6}$$

$$\theta = \theta_0 + \frac{\tau_{3,0}}{2} + \sum_{i\geq 4} \tau_{i,0} + \sum_j T_{j0}^{inv} + \frac{h}{2} = \frac{0}{2} = 0 + \frac{1 \cdot 10^{-6}}{2} = 0.5 \cdot 10^{-6}$$

By filling in the obtained parameters in the formula of the second-order approximation, the following transfer function is found, which is essentially the same as the original:

$$G(s) = \frac{k}{(\tau_1 s + 1)(\tau_2 s + 1)} e^{-\theta s} = \frac{100}{(3.6225 \cdot 10^{-6} s + 1)(3.5313 \cdot 10^{-6} + 1)} e^{-0.5 \cdot 10^{-6} s}$$

The only difference between this transfer function and that of the original plant is that the approximation allows for adding the sampling time of the controller as a dead time to provide more accurate digital control.
The approximation can be verified by comparing the open-loop step response with the original plant. This comparison can be seen in Fig. 4.12.

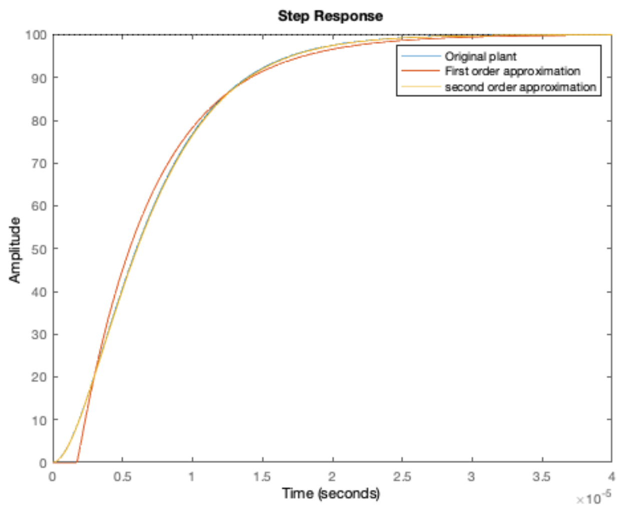


*Fig. 4.12 comparison of the original plant, first- and second-order plus delay approximation*

Fig. 4.12 shows that nearly no distinction between the second-order plus delay approximation and the original plant can be made, making it a more accurate approximation than the first order. Now that the second-order plus delay approximation of the plant is obtained, the PID control parameters can be derived from the model using the formulas from Table 2.2:

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c + \theta} = \frac{1}{100}\frac{3.6225 \cdot 10^{-6}}{\tau_c + 0.5 \cdot 10^{-6}}$$

$$\tau_i = \min\big(\tau_1, 4(\tau_c + \theta)\big) = \min\,(3.6225 \cdot 10^{-6}, 4(0.5 \cdot 10^{-6} + \tau_c)$$

$$\tau_d \;=\; \tau_2 = 3.5313 \cdot 10^{-6}$$

The continuous-time step response shows promising results, as shown in Fig. 4.13. The system is stable for both the resistance the system is tuned to and the reference resistance of 2304 Ohm, equal to 1 W. $\tau_c = \theta$ yielded the most balanced results and was therefore used.

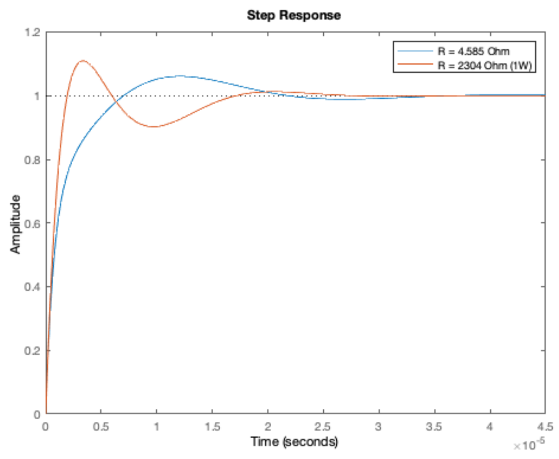


*Fig. 4.13 continuous-time closed-loop step response of SIMC PID controller at different loads*

The performance benchmark of this tuning method can be found in Table 4.3. It shows that for the controller's usable range, the response time is approximately 15 µS, with an overshoot varying between 5% and 11%. An attempt to reduce the overshoot by increasing the tuning parameter $\tau_c$ as suggested by [22] was made. Unfortunately, this yielded larger overshoot for larger loads whilst decreasing $\tau_c$ had a similar effect.

*Table 4.3 Performance benchmarks of the SIMC PID tuning under different loads*

| | R = 4.585 Ω | R = 23.04 Ω (100W) | R=2304 Ω (1W) | R=2304MΩ (1nW) |
|---|---|---|---|---|
| Rise time (s) | 4.0605e-06 | 1.6216e-06 | 1.4135e-06 | 1.4117e-06 |
| Settling time (s) | 1.8682e-05 | 1.5108e-05 | 1.5351e-05 | 1.5349e-05 |
| Overshoot (%) | 6.075 | 4.9917 | 10.948 | 11.0148 |

**B) K-SIMC**

**First-order**

Using the K-SIMC formulas from Table 2.1, the following first-order plus delay approximation can be obtained:

$$\tau_1 = \tau_{1,0} + \frac{1}{2}\frac{{\tau_{2,0}}^2}{\tau_{1,0}} = \frac{1}{276054.3487} + \frac{1}{2}\frac{\left(\frac{1}{283182.7176}\right)^2}{\frac{1}{276054.3487}} = 5.3437 \cdot 10^{-6}$$

$$\theta = \tau_{2,0}\left(1 - \frac{1}{2}\frac{\tau_{2,0}}{\tau_{1,0}}\right) + \sum_{i\geq 3}\tau_{i,0} + \sum_j T_{j0}^{inv} + \frac{h}{2} = \frac{1}{283182.7176}\left(1 - \frac{1}{2}\frac{\frac{1}{283182.7176}}{\frac{1}{276054.3487}}\right) + \frac{10^{-6}}{2}$$
$$= 1.8101 \cdot 10^{-6}$$

$$G(s) = \frac{1}{\tau_1 s + 1}e^{-\theta s} = \frac{1}{5.3437 \cdot 10^{-6}s + 1}e^{-1.8101\cdot 10^{-6}s}$$

From this approximation, the PID parameters can be derived in the following fashion, using formulas from Table 2.2:

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c + \theta} = \frac{1}{100}\frac{5.3437 \cdot 10^{-6}}{\tau_c + 1.8101 \cdot 10^{-6}}$$

$$\tau_i = \min(\tau_1, 5(\tau_c)) = \min\,(5.3437 \cdot 10^{-6}, 5 \cdot \tau_c)$$

$$\tau_D = \max\left(\frac{\theta - \tau_c}{2}, 0\right) = \max\left(\frac{1.8101 \cdot 10^{-6} - \tau_c}{2}, 0\right)$$

Fig. 4.14 shows the closed-loop step response for a load of 4.858 Ohm and 2304 Ohm (= 1W) using $\tau_c = \theta$. Unfortunately, this system becomes unstable when the resistance increases, making this not a viable tuning method.

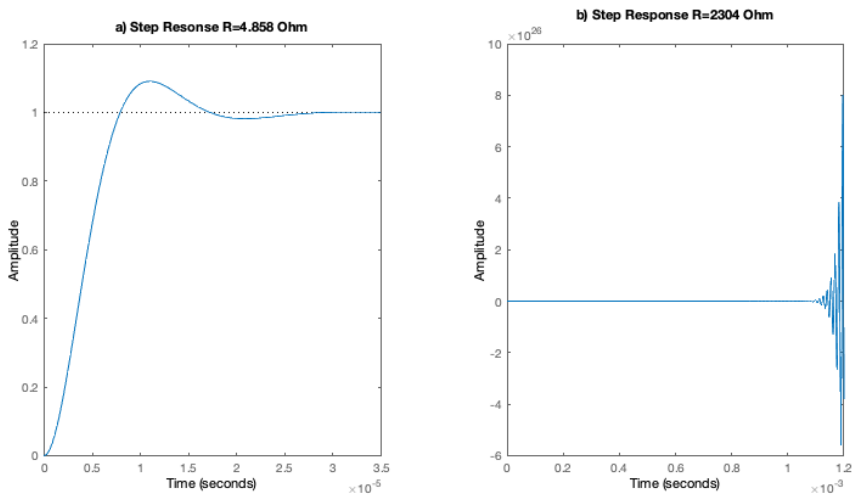


*Fig. 4.14 Closed-loop step response of K-SIMC FO PID tuning a) for R=4.858 Ohm B) for R=2304 Ohm*

**Second-order**

Again, using formulas from Table 2.1 allows the derivation of the K-SIMC second-order plus delay approximation:

$$\tau_1 = \tau_{1,0} = \frac{1}{276054.3487} = 3.6225 \cdot 10^{-6}$$

$$\tau_2 = \tau_{2,0} + \frac{1}{2}\frac{{\tau_{3,0}}^2}{\tau_{2,0}} = \frac{1}{283182.7176} + \frac{1}{2}\frac{0^2}{\frac{1}{283182.7176}} = 3.5313 \cdot 10^{-6}$$

$$\theta = \theta_0 + \tau_{3,0}\left(1 - \frac{1}{2}\frac{\tau_{3,0}}{\tau_{2,0}}\right) + \sum_{i\geq 4}\tau_{i,0} + \sum_{j} T_{j0}^{inv} + \frac{h}{2} = 0 + \frac{1 \cdot 10^{-6}}{2} = 0.5 \cdot 10^{-6}$$

$$G(s) = \frac{100}{(3.6225 \cdot 10^{-6}s + 1)(3.5313 \cdot 10^{-6}s + 1)} e^{-0.5\cdot 10^{-6}s}$$

From this approximation, the PID parameters can be derived using the formulas from Table 2.2:

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c + \theta} = \frac{1}{100}\frac{3.6225 \cdot 10^{-6}}{\tau_c + 0.5 \cdot 10^{-6}}$$

$$\tau_i = \min\big(\tau_1, 5(\tau_c)\big) = \min\big(3.6225 \cdot 10^{-6}, 5(\tau_c)\big)$$

$$\tau_D = \tau_2 + \max\left(\frac{\theta - \tau_c}{2}, 0\right) = 3.5313 \cdot 10^{-6} + \max\,(\frac{0.5 \cdot 10^{-6} - \tau_c}{2}, 0)$$

Fig. 4.15 shows the step response from the obtained tuning parameters. This system is stable for a variety of load resistances. This tuning, like the second-order SIMC tuning, has some overshoot. Increasing the tuning parameter $\tau_c$ reduces the overshoot slightly. However, it results in more undershoot when the resistance tends to be higher. A value of $\tau_c = 1.5\,\theta$ was selected to balance the effect.

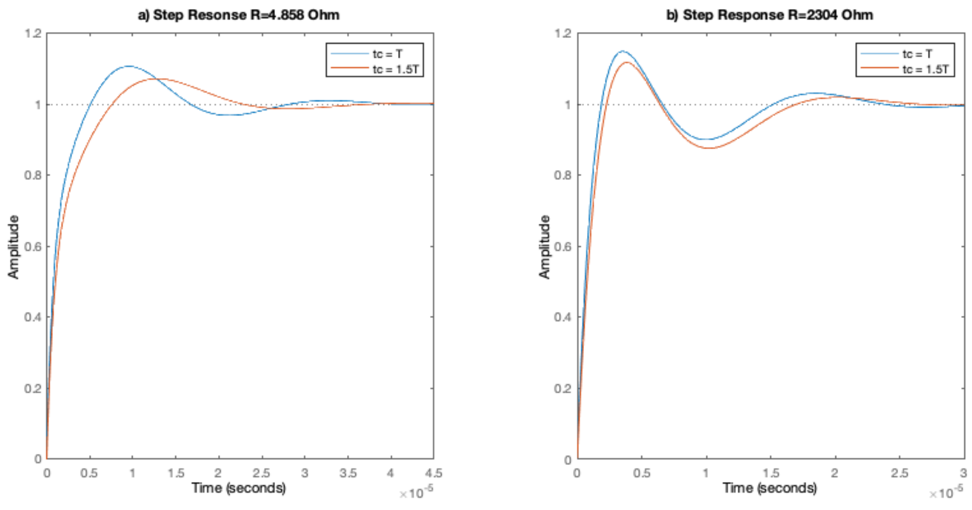


*Fig. 4.15 Closed-loop step response of PID control tuned by SO SIMC with a) resistance of 4.858 Ohm b) resistance of 2304 Ohm*

A detailed overview of the step-performance benchmarks can be found in Table 4.4.

*Table 4.4 Performance benchmarks of the second-order K-SIMC PID tuning under different loads*

| | R = 4.585 Ω | R = 23.04 Ω (100W) | R=2304 Ω (1W) | R=2304MΩ (1nW) |
|---|---|---|---|---|
| Rise time (s) | 4.8657e-06 | 1.9598e-06 | 1.6804e-06 | 1.6780e-06 |
| Settling time (s) | 2.0005e-05 | 1.5932e-05 | 1.5761e-05 | 1.5755e-05 |
| Overshoot (%) | 7.1397 | 4.8864 | 11.6948 | 11.7711 |

**Conclusion**

The controllers obtained from the (K-)SIMC methods can be split up into two categories: first, the PI and PID controller from the first order SIMC and K-SIMC methods, respectively. Both controllers showed an unstable response when the load-resistance of the converter model was increased, making these methods unviable. Next, the two PID controllers resulting from the second-order SIMC and K-SIMC tuning methods. When comparing the results in Table 4.3 and Table 4.4, it can be concluded that the SIMC-method results in a slightly better tuning as the settling time is smaller in all four cases whilst the overshoot is marginally smaller in 3 of 4 cases (with the plant of 23.04 Ohm being the exception).

### 4.2.2 Frequency loop shaping

The frequency loop shaping technique allows tuning a controller based on a desired cut-off frequency. [28] suggests using a high value for the cut-off frequency, using a frequency in the order of MHz. It is important to analyse if this magnitude of cut-off frequency is realistic as the minimisation objective should preferably be smaller than 0.2 and no larger than 0.5 to keep a similar behaviour to the target open-loop transfer function. This open-loop target TF is described as L(s).

**Integrator open-loop transfer function**

The first tuning attempt uses the transfer function of an integrator as open-loop target transfer function:

$$L_0(S) = \frac{\omega_c}{s}$$

The tuning method requires a time constant for the low pass filter to be set before the tuning process can begin. This low pass filter prevents the derivative term of the PID controller from responding strongly to noise. Decreasing the constant reduces the maximum cut-off frequency for which an objective lower than 0.5 or 0.2 can be obtained.

Using the MATLAB function as shown in appendix B, the minimisation function can be computed, constrained by the positivity of the PID gain parameters.

Using an LPF time constant of $10^{-2}$ resulted in a maximum usable cut-off angular velocity of $2.86 \cdot 10^3\ rad/s$ or $445.18\ Hz$. Comparing these results to the results of [28] it can be noted that the achieved maximum obtained cut-off angular velocity of $2.86 \cdot 10^3\ rad/s$ is significantly lower than [28]'s $1.19 \cdot 10^6\ rad/s$. The use of different parameters for the electrical components and a different theoretical model for the converter can be attributed to these differences.
Using the minimisation solution, the following closed-loop step response is obtained. A slight difference between the ideal target transfer function and the result of the minimisation is found, resulting from a limited degree of freedom from the transfer function of the PID controller, as can be seen in Fig. 4.16.

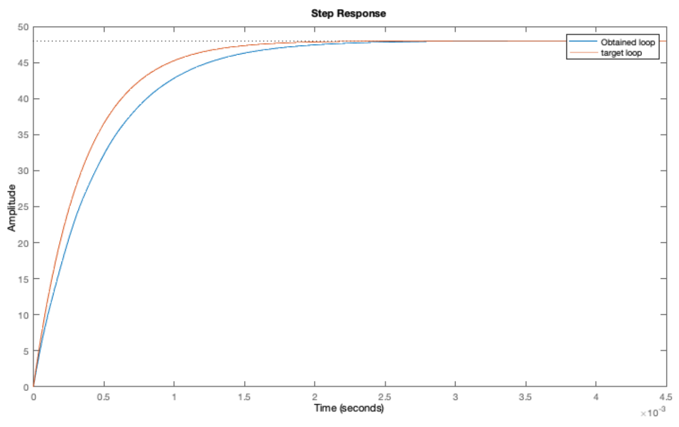


*Fig. 4.16 closed-loop step response of target transfer function versus obtained control loop from minimisation*

The performance benchmarks of this tuning method can be found in Table 4.5. It can be seen that the settling time is a lot higher than the most optimised SIMC method (factor ~120) but offers no overshoot.

*Table 4.5 Performance benchmarks of FLS Method using integrator as target*

| | R = 23.04 Ω (100W) | R=2304 Ω (1W) | R=2304MΩ (1nW) |
|---|---|---|---|
| Rise time (s) | 9.8768e-04 | 9.9037e-04 | 9.8768e-04 |
| Settling time (s) | 0.0018 | 0.0018 | 0.0018 |
| Overshoot (%) | 0 | 0 | 0 |

**Second-degree open-loop transfer function**

Besides targeting a transfer function of an integrator [26], suggest targeting the following transfer function:

$$L(s) = \frac{\omega_c(s+a)}{s(s+e)}$$

From which parameters “a” and “e” are introduced as discussed in chapter 2 of the thesis. “a” was a tuning parameter making a trade-off between settling time and overshoot, whilst “e” compensated for a slow pole in the system. The goal of this transfer function is to compensate for one slow pole, as the slowest pole in the system is from the LPF (the buck converter does not have a slow pole). This function would compensate for the effect of the low pass filter. This fact makes this transfer function not compatible for use on the plant.

### 4.2.3 MATLAB autotune software

Since MATLAB's tune software can directly tune digital controllers, these two subsections will use a digital PID control block with a sampling frequency of 1 MHz as the duty cycle command can only be altered once per switching period. The saturation of the plant control can also already be considered, limiting the duty cycle between 0 and 90%, as discussed in chapter 2. Tuning is performed using the largest resistance, 2306 MOhm, as this resulted in the highest overshoot.

**Transfer function tuner**

The MATLAB PID tune application allows tuning a controller given a plant. The following method was used for obtaining a tuning. Two tunings were derived: one with no overshoot and one with some overshoot (<5%). For these tunings, the transient behaviour was set to 0.8, and the response time was tuned to obtain the desired overshoot. The results for tuning towards no overshoot can be found in Table 4.6, and the results for tuning towards some overshoot can be found in Table 4.7

*Table 4.6 Performance benchmarks of MATLAB TF tuned towards no overshoot*

| | R = 23.04 Ω (100W) | R=2304 Ω (1W) | R=2304MΩ (1nW) |
|---|---|---|---|
| Rise time (s) | 1.7421e-05 | 4.6563e-06 | 4.6257e-06 |
| Settling time (s) | 5.3891e-05 | 6.4726e-05 | 6.4845e-05 |
| Overshoot (%) | 0 | 0 | 0 |

*Table 4.7 Performance benchmarks of MATLAB Frequency response tuner tuned towards some overshoot*

| | R = 23.04 Ω (100W) | R=2304 Ω (1W) | R=2304MΩ (1nW) |
|---|---|---|---|
| Rise time (s) | 8.1879e-06 | 3.6924e-06 | 3.6805e-06 |
| Settling time (s) | 1.7146e-05 | 4.1139e-05 | 4.1204e-05 |
| Overshoot (%) | 0.4860 | 2.5624 | 2.9046 |

It can be concluded from Table 4.6 and Table 4.7 that tuning for some overshoot has a significant positive effect on settling time compared to tuning towards no overshoot. Although overshoot would rather be prevented, it is worth considering testing with overshoot-reducing improvements that will be analysed in the next section.

**Frequency response tuner**

The final method for tuning the PID controller is the MATLAB frequency response tuner. The frequency response tuner allows tuning a target bandwidth up to the sampling frequency of the digital PID block divided by 0.3. When tuning for high bandwidth, as suggested in [28] (The maximum possible using a sampling frequency of 1 MHz), the results obtained can be seen in Fig. 4.17.

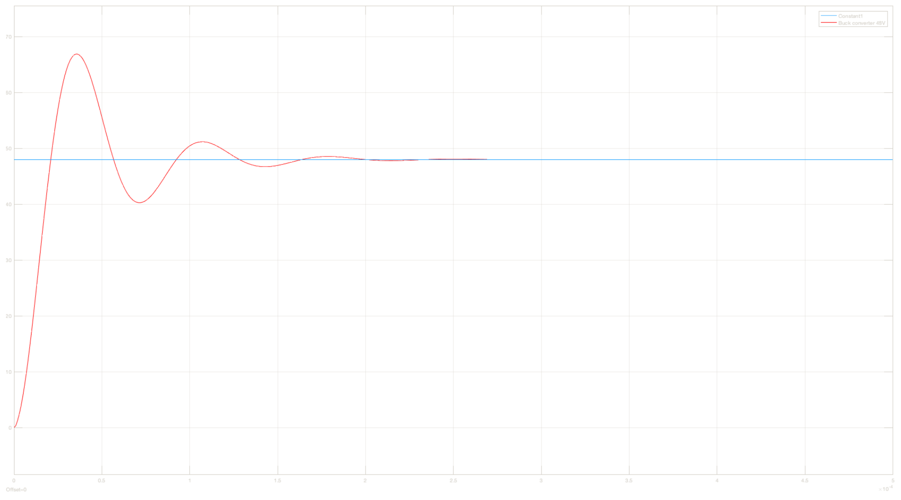


*Fig. 4.17 Step response of MATLAB frequency tuner tuned controller (R=23.04 Ohm)*

Fig. 4.17 shows the step response using the frequency response tuning method. For this step response, a load resistance of 23.04 Ohm was used. A significant overshoot and oscillation can be observed. The performance benchmarks from this tuning can be found in Table 4.8, showing that this method is stable for various loads and offers a reasonable settling time.

*Table 4.8 performance benchmarks of MATLAB frequency response tuned controller*

| | R = 23.04 Ω (100W) | R=2304 Ω (1W) | R=2304MΩ (1nW) |
|---|---|---|---|
| Rise time (s) | 1.4544e-05 | 1.4126e-05 | 1.4122e-05 |
| Settling time (s) | 1.5159e-04 | 1.4723e-04 | 1.4718e-04 |
| Overshoot (%) | 39.3231 | 39.7766 | 39.7807 |

As observed from Table 4.8 and Fig. 4.17, this method yields a large overshoot. The overshoot as is, is considered unacceptable. However, several potential improvements will be discussed further in this thesis. When comparing this data to the transfer function tune method data in Table 4.7 and 4.6, the transfer tune method results in tunings with a lower settling and rise time whilst producing less overshoot. This makes the frequency response tune method less favourable.

### 4.2.4 Conclusion

We can compare the step response data from previous experiments and conclude that the SIMC method appears to offer a more favourable step-response based on rise time, settling time, and overshoot. However, converting the design to a digital PID controller with a sample rate of 1 MHz shows the design becomes unstable. Simulation shows that for sampling frequencies up to 1 GHz, the system does not return to a stable state for realistic values of $\tau_c$, making an implementation of this tuning near impossible.
Comparing the result tables of the FLS and MATLAB autotune methods shows that MATLAB's transfer-function tuner gives the best results. The tuning with no overshoot will be considered as base-tuning. However, the improvements in the next section may allow the implementation of the tuning designed for some overshoot.

The obtained base-tuning can be verified against the SimScape model to ensure the behaviour is similar, as shown in Fig. 4.18. It can be concluded from this image that the behaviour is similar.

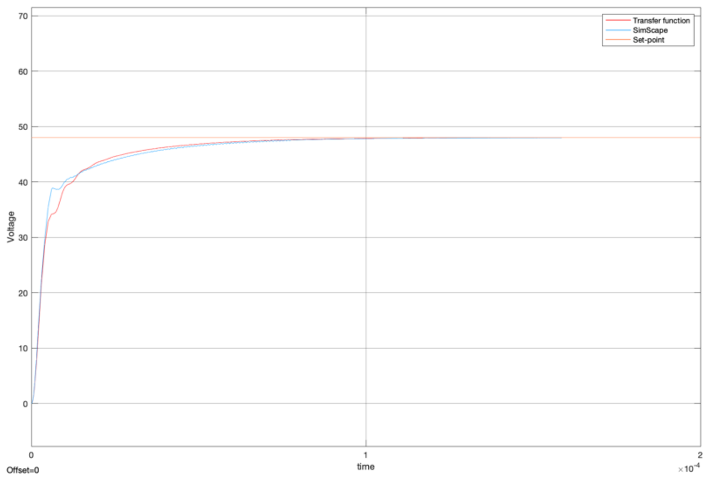


*Fig. 4.18 Compare PID controller with base tuning to transfer function as plant and SimScape simulation as plant*

### 4.2.5 100V to 24V Converter

Previous calculations were done using the 100V to 48V controller, as the 100V to 24V uses a different capacitance and inductance value, the transfer function differs from that of the 48V model. The same procedure can be followed for tuning the 24V controller. Using MATLAB transfer function tuner to tune for no overshoot, the results from Table 4.9 are obtained.

*Table 4.9 Benchmarks of tuning 24V controller using the MATLAB Transfer function tuner for no overshoot*

| | R = 5.76 Ω (100W) | R=576 Ω (1W) | R=576MΩ (1nW) |
|---|---|---|---|
| Rise time (s) | 4.6731e-05 | 4.0111e-06 | 3.9459e-06 |
| Settling time (s) | 8.7268e-05 | 9.4894e-05 | 9.4941e-05 |
| Overshoot (%) | 0 | 0 | 0 |

When comparing the data from Table 4.6 to Table 4.9, the tuned system for the 24V controller behaves similar to that of the 48V controller when the same tuning method is applied. A method with overshoot can also be found. It should be stated that as the final duty cycle is a lot smaller (~0.24 under nominal conditions), higher overshoot occurs due to the proportional response of the PID controller, but the next section will consider a solution to reduce this. The result of a controller tuned ignoring the overshoot on the step response can be seen in Table 4.10.

*Table 4.10 Benchmarks of tuning 24V controller using the MATLAB Transfer function tuner for overshoot*

| | R = 5.76 Ω (100W) | R=576 Ω (1W) | R=576MΩ (1nW) |
|---|---|---|---|
| Rise time (s) | 1.2307e-06 | 1.0403e-06 | 1.0388e-06 |
| Settling time (s) | 6.4252e-05 | 6.8881e-05 | 6.8864e-05 |
| Overshoot (%) | 31.35 | 70.47 | 70.96 |

## 4.3 Potential control improvements

In chapter 2, several potential improvements to PID control were suggested. In this section, these methods will be tested and evaluated whether the improvement these changes offer can justify the increase in complexity and resources used.

### 4.3.1 Nonlinear PID control

A new MATLAB function block was designed to test the suggested nonlinear PID control method. The code for this function block can be found in appendix C. The schematic used to test this setup can be seen in Fig. 4.19

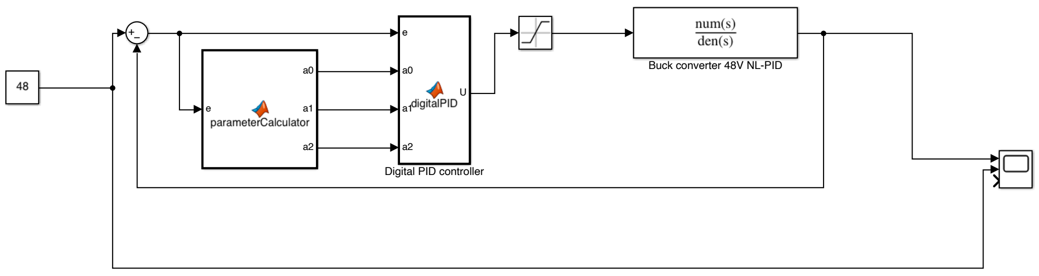


*Fig. 4.19 Simulink schematic for testing Non-Linear PID*

Similar results are obtained when testing for several different error-band thresholds, as can be seen in Fig. 4.20.

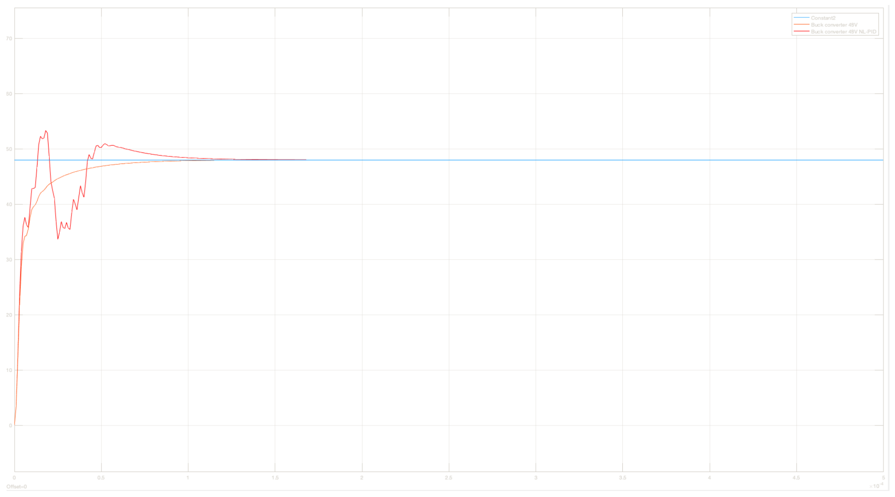


*Fig. 4.20 Step response of 48V Converter controlled by PID vs Non-Linear PID*

Fig. 4.20 shows the step response of the 48V buck-converter controlled by a regular PID controller vs controlled by the proposed nonlinear PID. Even when using different thresholds, the following symptoms are observed. First, the system rises at a rate equal to the rise rate of the regular PID block. Next, the Nonlinear block either overshoots or undershoots depending on the thresholds, creating an irregular signal. The settling time of the nonlinear circuit tends to be longer due to recovering from this irregular period.

The lack of performance increase whilst applying this schematic can be explained. The goal of the circuit is to push the controller in saturation during the start-up period. However, the obtained tuning already puts the controller into saturation. Further increasing the gain has no positive effect and introduces irregularities when changing between tuning parameters due to changing between error bands.

As there is no gain in performance, this circuit is not recommended for use in the control circuit of this specific DC-DC Converter.

### 4.3.2 Low pass filter derivative component

Using MATLAB's Simulink Filtered PID block with a transfer function similar to that described in section 2.2.2 with various values for filter coefficient N (attempts ranging from N=$10^{-6}$ to N=$10^{6}$) makes the closed-loop system unstable.
An attempt to retune the controller for this setup was made using MATLAB's transfer function PID tuner and frequency response tune. However, neither obtained a stable tuning.
Due to a lack of stable tuning, this optimisation method was dismissed.

### 4.3.3 Setpoint filter

The setpoint filter aims to reduce overshoot in the step response. As the opted base tuning does not result in overshoot, an attempt was made to reduce the overshoot in the MATLAB transfer function tuning with some overshoot (Table 4.7).
The design of the filter can be made in Simulink and can be seen in Fig. 4.21.

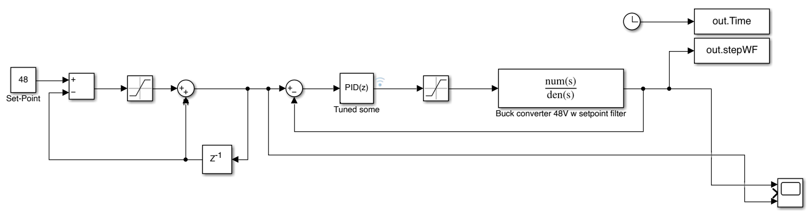


*Fig. 4.21 Simulink schematic for testing setpoint filter*

The maximum setpoint rate change value was determined experimentally to reduce the overshoot to 0%. This led to a maximum setpoint rate of 4V/µs. Using the filter with this rate limitation led to the results shown in Table 4.11.

*Table 4.11 Results using setpoint filter at 4V/µs with PID tuning from Table 4.7*

| | R = 23.04 Ω (100W) | R=2304 Ω (1W) | R=2304MΩ (1nW) |
|---|---|---|---|
| Rise time (s) | 1.0551e-05 | 1.0144e-05 | 1.0125e-05 |
| Settling time (s) | 5.9976e-05 | 6.1525e-05 | 6.1534e-05 |
| Overshoot (%) | 0 | 0 | 0 |

The results from Table 4.11 show that the setpoint filter successfully reduces or removes overshoot. However, comparing the settling times from this method to the settling times of the MATLAB transfer-function tuned without overshoot shows that the two systems behave similarly.

The tuning of the controller does not only affect the step response, but It also affects its ability to reject disturbances. Both responses can be seen in Fig. 4.22.

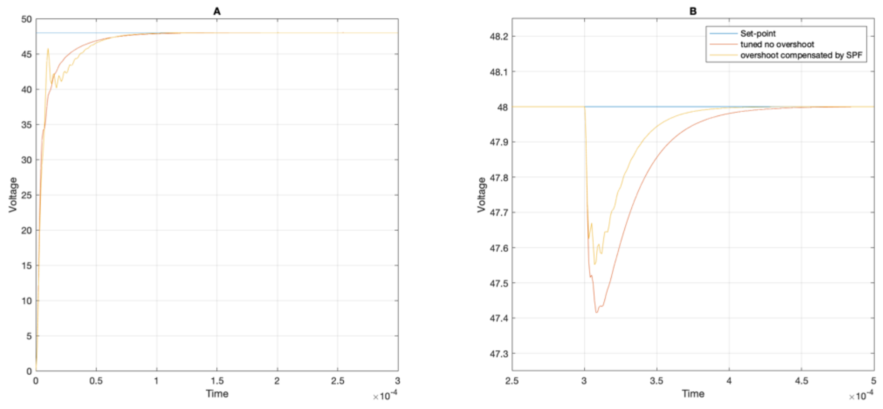


*Fig. 4.22 a) step response b) 5V voltage drop disturbance response of PID control using setpoint filter vs base tuning*

Fig. 4.22 shows that the more aggressively tuned controller with a setpoint filter responded slightly faster to a disturbance. In this case, a voltage drop of 5V applied to the input. The more aggressively tuned controller with setpoint filter needed 52µs to return to 47.95V, whilst the controller tuned for no overshoot needed 76µs. Another notable difference is that the more aggressive tuned controller results in a lower undershoot in case of a voltage drop, dropping the output to 47.55V rather than 47.42V.

Similar effects can be obtained using the 24V controller. Starting from a tuning that normally results in overshoot and removing the overshoot by using a setpoint filter positively affects the disturbance rejection rate. A setpoint rate limit of 5V/µs was used in this test. The results can be seen in Fig. 4.23, having a similar effect but in a larger order of magnitude compared to the impact on the 48V controller. In this case, the filter also significantly improved the step response, showing much fewer oscillations.

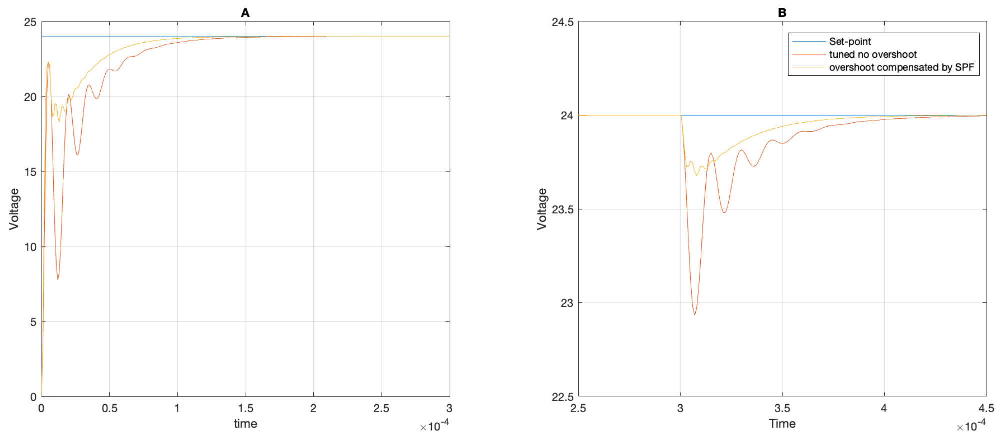


*Fig. 4.23 a) step response b) 5V voltage drop disturbance response of PID control using setpoint filter vs base tuning on 24V controller*

Even though this method does not improve the step response of the 48V converter significantly, using this filter in combination with a slightly more aggressive tuning does allow for significantly improved disturbance response, allowing the converter to recover from voltage drops or load changes more quickly with less hard drops on the output voltage. It also contributes significantly to improving the step response of the 24V controller. This contributes considerably to a better performing controller and hence is worth the relatively low addition of complexity.

It should be noted that using this filter has an effect should variable setpoints be introduced. However, as the DC-DC converter is designed to function at a fixed target voltage, the impact of this filter is not considered for this scenario.

## 4.4 Design of DPWM output stage

The DPWM is critical in the controller design as it is responsible for controlling the converter directly. Its resolution will impact the control resolution of the entire controller and could be accountable for causing limit cycle oscillations (LCO), as described in chapter 2. This section describes the design process of the output stage, comparing a setup with a Σ–Δ generator to a setup with only a sawtooth comparator DPWM generator.

### 4.4.1 Σ–Δ Generator

**MATLAB Simulink model and verification**

The first step in the design process was to obtain a functioning MATLAB model of a second-order Σ–Δ generator. During this step, the Σ–Δ generator presented in [16] with a 4-bit effective core resolution and 10-bit input resolution was recreated.
Recreating the Σ–Δ generator in MATLAB required separating a fixed set of MSBs and a fixed set of LSBs. Due to the method of notation, it was easiest to convert the fractional notation of the duty cycle to an entire integer notation by shifting left the duty-cycle word at the input of the Σ–Δ generator and shifting the word back to the fractional domain at the end of the Σ–Δ generator. This implementation of the second-order Σ–Δ generator can be seen in Fig. 4.24. Besides extra bit shifts, several data-conversion blocks are added. These blocks force MATLAB to use the correct fixed-point notation types as would happen on an FPGA, or eventually an ASIC, making it possible to simulate quantisation effects. The registers, simulated as discrete delay, are clocked at the target switching frequency of 1 MHz.

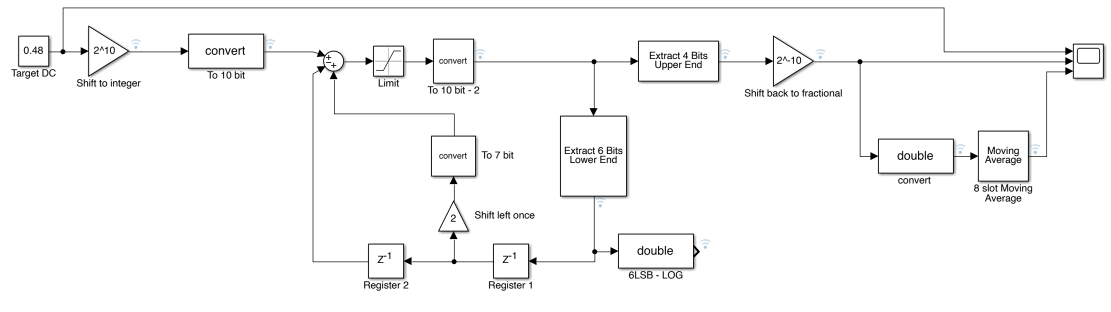


*Fig. 4.24 MATLAB Simulink implementation of multibit second-order Σ–Δ generator*

The implementation can be verified by analysing the output of the simulation and comparing the average signal to the setpoint.

The Simulink simulation takes a moving average of the output of the Σ–Δ generator with a window length of 8 slots. This averaging will eventually be done by the filtering effect of the power stage of the buck-converter. The result of this simulation can be seen in Fig. 4.25.

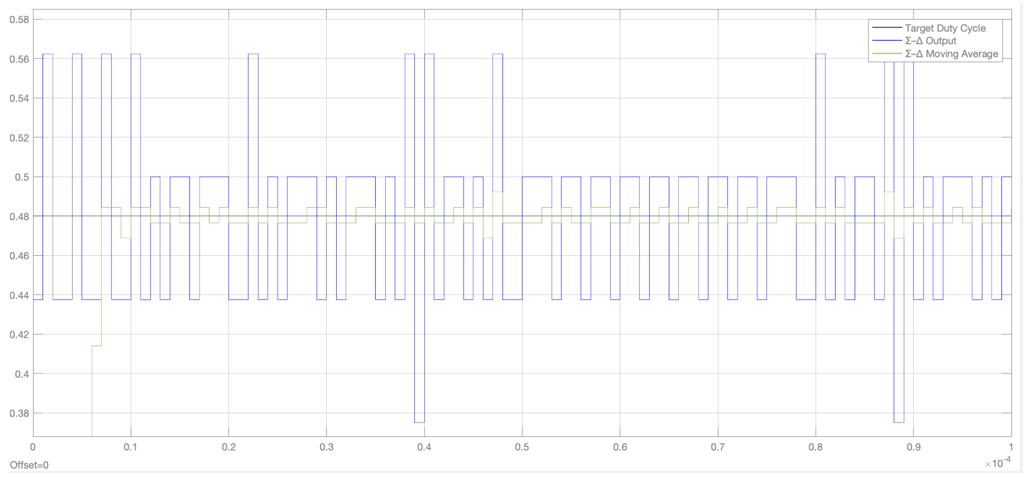


*Fig. 4.25 MATLAB Simulink verification of second-order multibit Σ–Δ generator implementation*

The graph from Fig. 4.25 shows that, although only a limited control resolution of 4 bits, a control accuracy of 0.0625 ($2^{-4}$), an average of the supplied signal of 0.48 can be attained by continuously switching in steps of two LSBs. This graph confirms the correct function of the Simulink implementation presented in Fig. 4.24.

**Design**

To design an output stage with a Σ–Δ generator, some design choices will have to be made first: control accuracy and measurement resolution. From [42] can be concluded that the ripple size of the controller equals approximately 0.2V. The following formula is suggested for choosing the minimal measurement resolution:

$$\Delta V_{LSB} < \frac{V_{Ripple}}{4}$$

Choosing a measurement resolution finer than the ripple might seem like a waste of resources at first. However, the voltage measurement always happens at the same time-point in the switching period (in this MATLAB simulation). This will result in a steady-state error with a maximum size of the ripple voltage. Choosing a control resolution finer than the ripple voltage allows for an offset in the target voltage smaller than the ripple voltage to compensate for this effect.
Using this logic, a minimum resolution of 0.05V is found. Combining this with the knowledge that the output voltage can never exceed the input voltage of 100V, the measured output voltage can be represented as a fixed-point number with 7 integral bits and 5 fractional bits ($2^{-5}$ = 0.03125 < 0.05).

Knowing the measurement resolution, the minimal control resolution required to prevent LCO can be determined:

$$\Delta d_{LSB}\, V_{IN} < \Delta V_{Out\_LSB}$$

$$\Delta d_{LSB}\ 100 < 0.03125$$

The smallest binary notation of the duty-cycle word ‘d’ for which this inequality would be true is a fixed-point notation with 12 fractional bits, as:

$$2^{-12} * 100\ =\ 0.02441\ <\ 0.0312$$

A 9-bit Core DPWM generator was used as it was considered the highest accuracy attainable given the switching frequency. Later analysis concluded that the available clock frequency would be too low for a 9-bit DPWM. However, the 9-bit DPWM will still function well for testing the averaging capabilities of the converter.

When using an input resolution of 12 bits with an effective resolution of 9 bits requires the Σ–Δ generator to add a virtual resolution of 3 bits. As both the effective resolution and input resolution of the Σ–Δ generator is known, the sampling frequency of the steady-state PID controller can be determined:

$$N_{DPWM} \approx N_{core} + 2.5\log_2\left(\frac{f_{sw}}{f_{update}}\right)$$

$$12 \approx 9 + 2.5\log_2\left(\frac{1\ MHz}{f_{update}}\right)$$

$$f_{update,max} \approx 435275\ Hz = 0.435275\ MHz$$

A PID update frequency will be chosen as a factor of the switching frequency, allowing the clock signal for steady-state PID control to be created using a prescaler. A factor of ¼ allows for an update frequency of 0.25 MHz as some margin is recommended in [16].

A practical implementation of a Σ–Δ generator with the aforementioned effective and input resolution can be seen in Fig. 4.26.

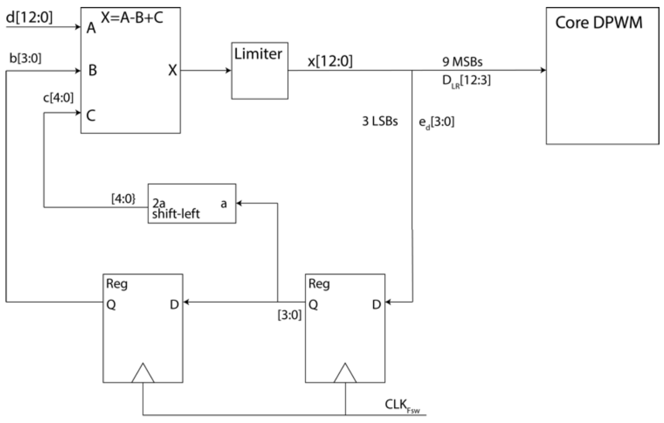


*Fig. 4.26 Practical implementation of the second-order multibit Σ–Δ generator*

**Testing averaging capability of the system**

The Σ–Δ generator varies the output to the core DPWM by 1*LSB or 2*LSB to obtain an average value equal to the input of the Σ–Δ generator. As the design of the output stage has been determined, the averaging capabilities of the system can be tested by feeding a fixed duty cycle through the Σ–Δ generator and analysing the variations measured in the (simulated) buck converter. To filter out the ripple of the buck converter, the scope sampling time was set equal to a switching period of the DPWM signal. This way, the signal is sampled at the same time point in each switching period. Although this measurement might introduce some steady-state error, this works well to analyse variations in the signal whilst filtering out the ripple of the buck converter. This test was performed using a duty-cycle word of 0.48.

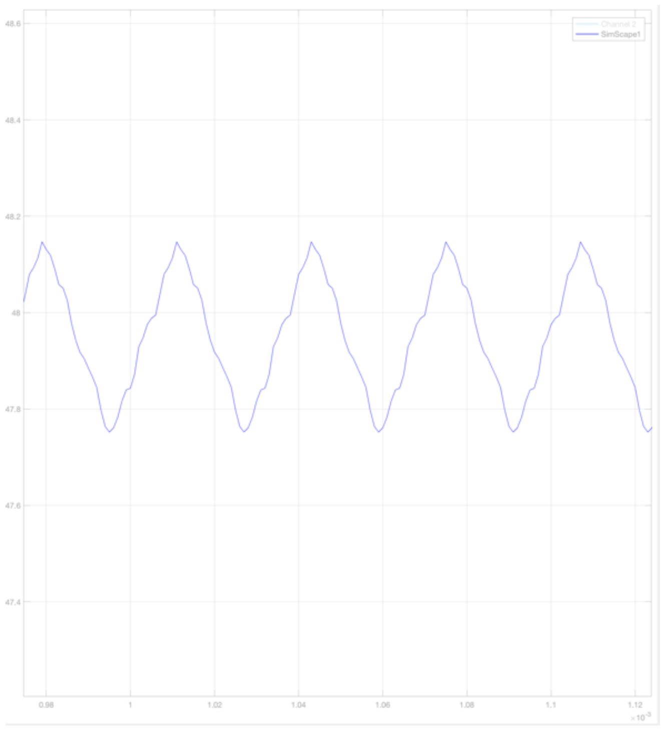


*Fig. 4.27 Steady-state variation caused by second-order multibit Σ–Δ generator on 48V Buck converter*

The variations caused by the designed second-order multibit Σ–Δ generator are displayed in Fig. 4.27. From this graph, it can be read that the Σ–Δ generator introduces a relatively large variation of approximately 0.4V peak to peak.
Comparing this to the LCO effect of using the same measurement resolution as with the multibit Σ–Δ generator and using the same 9-bit core DPWM generator, shown in Fig. 4.28.

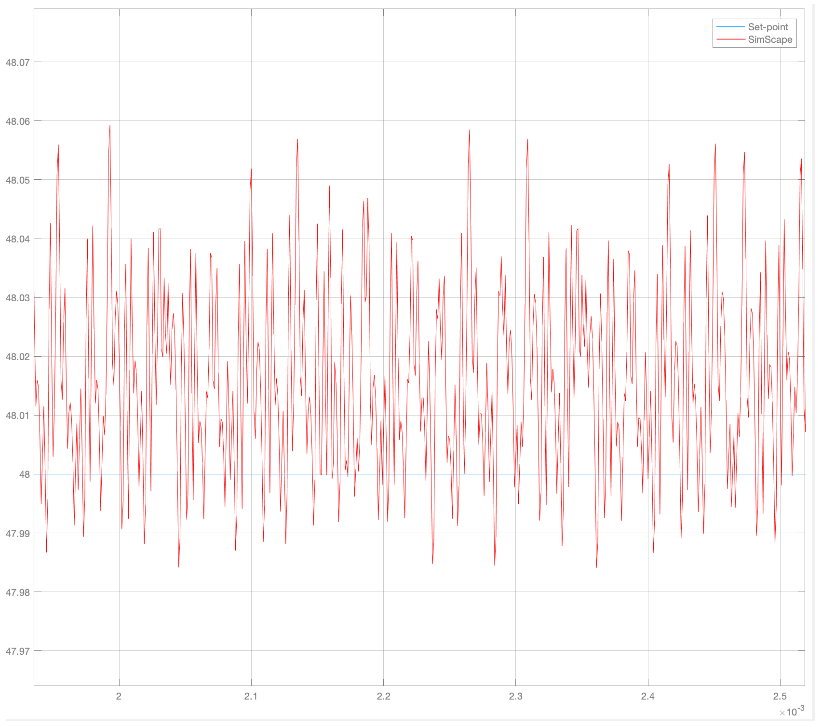


*Fig. 4.28 Steady-state variation caused by limit cycle oscillation*

Analysing the variation from Fig. 4.28 shows that the peak-to-peak variation is 0.068V. Comparing the variation of using the Σ–Δ generator or suffering the effects of LCO shows that the impact of LCO causes less variation than the implementation of the Σ–Δ generator, which was implemented with reducing the effects of LCO. The variation caused by the Σ–Δ generator indicates that the converter design at the switching frequency of 1 MHz cannot average/filter the small variations the Σ–Δ generator causes.
The amplitude of the variations occurring with the Σ–Δ generator can be confirmed as the second-order Σ–Δ generator varies the control signal by a maximum of 2*LSB. Translating this to terms of voltage would result in variations of $2^{-9}$ *100*2, which equals 0.39V.
This difference in findings between [16] and the above experiment can be attributed to a significant lower switching frequency. The designed controller has a switching frequency of 1 MHz, whilst the controller from [16] has a switching frequency of 10 MHz. Besides this, a difference can occur by using different inductance and capacitance for the components of the buck converter.
This analysis makes the Σ–Δ generator unfeasible for integration in the control circuit of the DC-DC Buck converter as either accepting the effects of LCO or lowering the control resolution yields better results with less hardware. It should also be noted that the dither generator, which has a similar effect on the input of the core DPWM, would be infeasible too for the same reasons the Σ–Δ generator is.

### 4.4.2 Sawtooth comparator DPWM output

The variations from dither-based solutions cause the system to oscillate more than the effects of LCO, which it was supposed to prevent, due to the lack of averaging capabilities of the buck converter. A trade-off should be made between steady-state oscillation, measurement-resolution and setpoint-resolution. It is still possible to design a controller that should show no signs of LCO by maintaining the following inequation with the known maximum control resolution of 9 bits:

$$\Delta d_{LSB}\, V_{IN} < \Delta V_{Out_{LSB}}$$

$$2^{-9} \cdot 100 < \Delta V_{Out_{LSB}} \quad or\ 0.195313 < \Delta V_{Out_{LSB}}$$

A measured voltage representation of 7 bits integer and 2 bits fractional can be derived, resulting in a measurement and setpoint resolution of 0.25V.
Although this resolution does not allow for compensating the offset of measuring at the same moment of time in the switching period, some control of this offset can be obtained by managing the time point ADC takes its measurement. However, this is considered outside the scope of this thesis.
The steady-state response is obtained using the above-calculated resolution, as shown in Fig. 4.29. The variations in this setup have a peak-to-peak amplitude of 0.01V, significantly smaller than previous attempts and negligible compared to the ripple voltage of the buck converter.

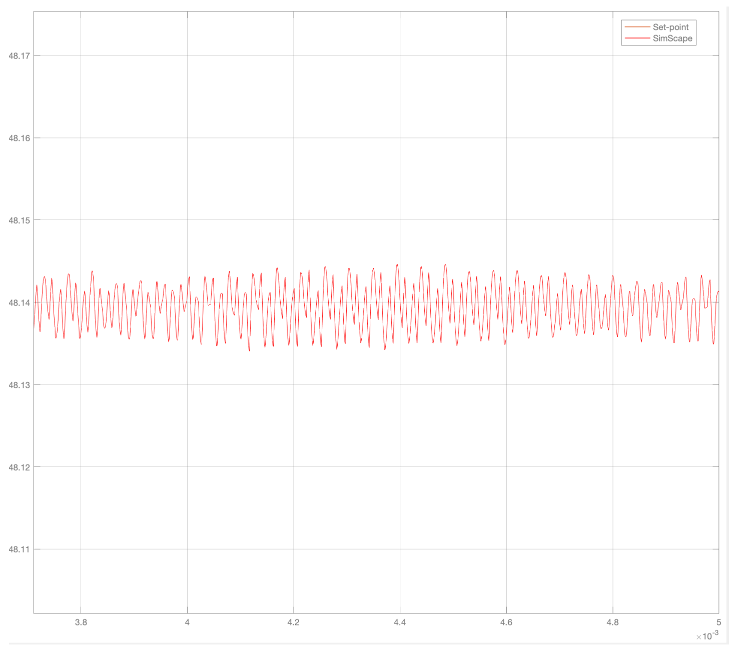


*Fig. 4.29 Steady-state variation using sawtooth comparator DPWM*

Further analysis turned out that the maximum clock frequency for the DPWM generator using the 180nm BCD technique from the TSMC foundry is 444.44 MHz or 2.25ns. This makes counting to 512 (the value required for a 9-bit DPWM generator) with a frequency of 1 MHz (switching frequency) impossible. However, instead of resorting to an 8 bit DPWM generator and having to decrease the

measurement voltage as a result. A counter that counts to 444 could be implemented. The input signal should still be 9 bits but it should be scaled down to a factor of 444/512 as otherwise the controller's gain would be increased.
Instead of multiplying the duty-cycle with this factor, the following changes could be made to obtain the same result without requiring extra hardware:

- Multiply the tuning parameters by 444/512 before calculating the parameters a0, a1, and a2, lowering the controller's gain back to the original gain.
- Decrease the duty-cycle saturator maximum by a factor of 444/512, preventing a duty cycle higher than the efficiency of the buck converter.

Verification can be made to determine if the measurement voltage resolution of 2 bits can be maintained:

$$\frac{1}{N_{steps}} V_{IN} < \Delta V_{Out_{LSB}}$$

$$\frac{1}{444} * 100 = 0.225\,V < 0.25\,V$$

This implementation yields a measurement resolution of 0.25V and a control resolution of 0.225V.

A logical implementation using this strategy can be seen in Fig. 4.30. This implementation also uses two shift registers to allow the negative PWM signal to have a dead time before and after each switch of the positive PWM signal. Should it be requested to alter the dead-time of the converter, then this can be arranged in steps of 2.25ns by installing a prescaler and clocking the registers with this slower clock.

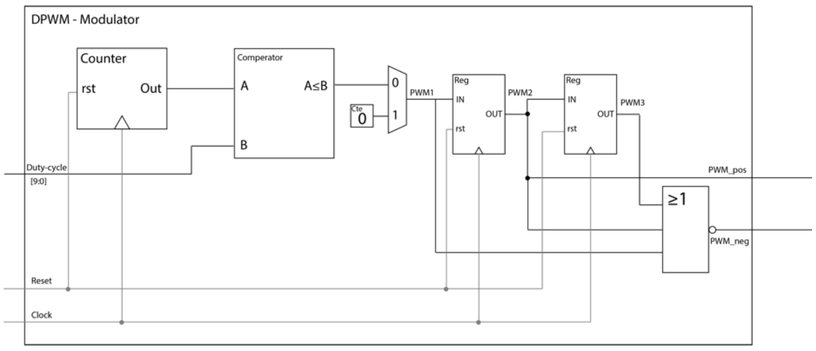


*Fig. 4.30 Implementation of DPWM generator*

## 4.5 Converting to fixed-point representation

To implement the obtained control logic in VHDL, a representation method for the numeric values in the calculation process needs to be chosen. It was opted to go with fixed-point representation rather than floating-point implementation as it can be implemented easier on FPGA and ASIC, needing fewer resources.
To convert the existing system to fixed-point representation, a bit size for the integral and fractional parts must be chosen.

### 4.5.1 Integral lengths

To determine the size of the integral length, an analysis can be made to determine the maximum and minimum value each signal must be able to represent. The input limits are listed in Table 4.12.

*Table 4.12 Input maxima & minima with fitting integral representation*

|  | Min | Max | Representation | Required integer bits |
|---|---|---|---|---|
| Ref | 0 | 90 | Unsigned | 7 |
| V | 0 | 105 | Unsigned | 7 |
| A0 | 0.13 | 0.19 | Unsigned | 0 |
| A1 | -0.34 | -0.23 | Signed | 1 sign + 0 |
| A2 | 0.10 | 0.15 | Unsigned | 0 |

The limits listed in Table 4.12 can be derived as follows. The maximum reference voltage cannot be higher than the efficiency of the DC-DC buck converter multiplied by the input voltage. As the controller's efficiency is not yet known, a suggested value of 90% is taken. Obtaining a negative reference is impossible using a switching converter making the minimum reference 0.
The measured voltage of the controller cannot exceed the input voltage of the controller, allowing an additional 5% ensures stable operation in case of a higher input voltage under maximum duty cycle. The measured voltage cannot be negative using a switching converter.
The maxima and minima from the PID parameters A0, A1 and A2 are derived from Appendix D, specifically, the two tunings using overshoot for the 48V and 24V controller scaled by 444/512.

Simple calculations can be used to derive the integral size constraints from these input values. In these calculations, the signals will be named as shown in Fig. 4.31.

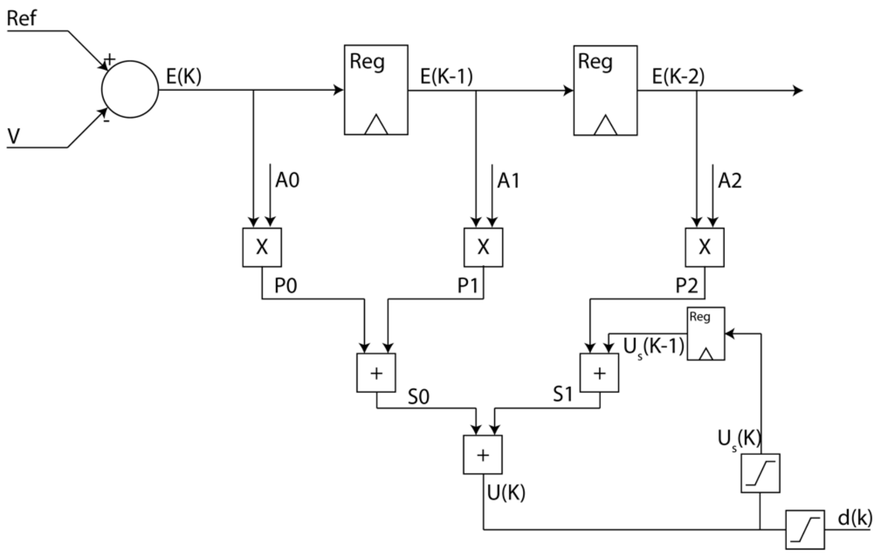


*Fig. 4.31 Control schematic naming the signals of the PID controller*

Using the input limitations, the value range of all signals up to S1 can be calculated. S1 uses a feedback signal formed from the result, meaning no mathematical limit based on the input signals can be used to determine the maximum range U and hence for S1. This can be solved by saturating the U(k) signals into $U_s(k)$. The saturation value should be high enough not to be met during regular operation yet low enough to prevent too many resources from being wasted.
Adding this saturation brings a secondary advantage: should (due to some error) the input voltage be lower than the setpoint voltage, an uncorrectable error is introduced. The PID circuit without saturation will keep integrating the mistake, causing a very large value of U(K). Running in an unconstrained environment would result in a long recovery time when the input voltage is restored as the overshoot beyond the maximum duty cycle would need to be "integrated"-away by the new error signal. Running in a constrained environment would eventually result in an overflow.
Saturation on the return path will prevent both: the size of the accumulated error is limited, as is the time to recover after the input voltage recovers, and the use of saturation prevents overflow.
By simulating the step and disturbance response in MATLAB, we can find the maximum value of U(k) by logging the signal. A maximum of 7.99 is found for the 48V controller and a maximum of 0.68 is found using the 24V controller.
A safety factor of 2 is applied for this calculation, allowing for some margin should the physical system experience a more significant value (due to slower step response or other disturbances/noise), making the maximum for the saturated value 15.98 or rounded up to 16. The value of U can become negative, using the same method of logging a minimum of -0.8 was found using the 48V controller, using the same safety factor of 2 results in a minimum of -1.6.
Using these values for maxima and minima of $U_s$ and performing a maxima and minima analysis on the circuit using the input constraints, the maxima and minima with integral representation can be obtained, as shown in Table 4.13.

*Table 4.13 Calculated maxima & minima with fitting integral representation*

| | Min | Max | Representation | Required integer bits |
|---|---|---|---|---|
| E | -105 | 90 | Signed | 1 sign + 7 |
| P0 | -19.95 | 17.1 | Signed | 1 sign + 5 |
| P1 | -30.6 | 35.7 | Signed | 1 sign + 6 |
| P2 | -15.75 | 13.5 | Signed | 1 sign + 4 |
| S0 | -50.55 | 52.8 | Signed | 1 sign + 6 |
| Us | -1.6 | 16 | Signed | 1 sign + 4 |
| S1 | -17.35 | 29.5 | Signed | 1 sign + 5 |
| U | -67.9 | 82.3 | Signed | 1 sign + 7 |
| d | 0 | 0.9 | Unsigned | 0 |

Tables 4.12 and 4.13 show that some values can be represented unsigned. It would require conversion to signed notation to perform mathematical operations with other signed values. Therefore, all values will be noted as signed in the VHDL implementation.

### 4.5.2 Fractional lengths

To complete the conversion to fixed-point representation, a length for the fractional bits needs to be chosen as well. [3] suggests using MATLAB's fixed point tool to determine what the minimal fractional length is for which the behaviour of the PID controller remains within a certain error-bound of floating-point representation, giving extra care not to create a steady-state error.
The fixed-point notation of the reference voltage and voltage measurement has already been determined in section 4.4.2.
A notation for the implementations PID parameters (A0, A1 and A2) can be determined using MATLAB's fixed point tool. When these representations are determined, the representation of the data lines resulting from calculations with known representations can be calculated using the rules of the IEEE fixed point package explained in [46].

First, the representation of the parameters is determined using the fixed-point tool. An error band of 5% is accepted for dynamic response and a steady-state error of 0.01V. For this test, the setup of using both the setpoint filter with the PID controller is used. The measured voltage signal is not represented in fixed-point notation to prevent cumulative quantisation errors during this test. A stricter tolerance for the steady-state fault is chosen as it is more important that the controller reaches a correct steady-state than following the exact dynamic behaviour as the ideal, non-quantised PID controller. These tests were performed using both the 48V and 24V controller with a 5V voltage drop to analyse the effect of quantisation on the step response and the disturbance response.

Using the 48V controller, it was concluded that an 11-bit fractional representation for the parameter suffices to reach less than 5% deviation from the non-quantised form. For the 24V controller, it was concluded that at least a 10-bit fractional representation is needed to obtain the same result. The graph showing the deviation caused by fixed point representation using the 48V controller with 11-fractional bit parameters to floating-point parameter representation can be seen in Fig. 4.32, clearly showing it remaining in the 5% error bound and having no significant steady-state error. Although it would be possible to design a different control ASIC for the 24V converter and the 48V converter, it would make more sense to design a single controller with different input parameters to keep complexity low and only manufacture a single ASIC. Keeping this in mind, the representation needed to meet the 5% target on both converters is the 11-bit fractional representation.

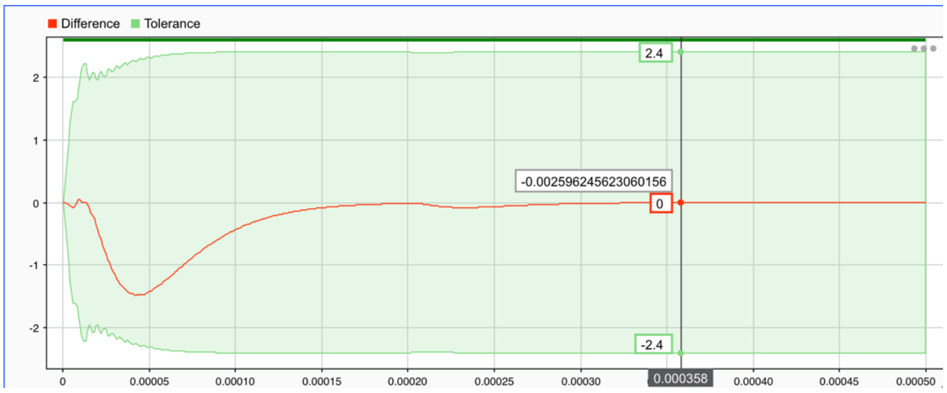


*Fig. 4.32 Difference signal of 48V controller step and disturbance response with parameters represented as floating-point with 11 fractional bits*

The following sizing rules from [46] can be used to determine the fractional representation as a result of calculations with the parameters and error signals:

- A + B
  Integral bits : Max(A'Integral, B'Integral)
  Fractional bits : Min(A'fractional, B'fractional)

- A * B
  Integral bits : A'Integral + B'Integral
  Fractional bits : A'fractional + B'fractional

Applying these rules to all signals that form as a result from calculations with either the parameters, with a 15-bit fractional notation and the reference and voltage signal with a 2-bit fractional notation, the notation of all signals can be derived.

*Table 4.14 Fixed point fractional notation of signals in PID controller*

| Signal | Result of | Nr of fractional bits |
|---|---|---|
| Ref | - | 2 |
| V | - | 2 |
| E | Ref-V | 2 |
| P0 | A0*E(k) | 13 |
| P1 | A1*E(k-1) | 13 |
| P2 | A2*E(k-1) | 13 |
| S0 | P0 + P1 | 13 |
| Us | S0 + S1 | 13 |
| S1 | P2 + Us | 13 |
| U | S0+S1 | 13 |
| d | Limit(U) | 9 |

### 4.5.3 Overview

Combining the information from Table 4.13 and 4.14, the following fixed-point notations for all signals can be found in Table 4.15.

*Table 4.15 Overview of fixed-point notations in the PID controller*

| Signal | Sign bit | Integral bits | Fractional bits |
|---|---|---|---|
| Ref | 1 | 7 | 2 |
| V | 1 | 7 | 2 |
| A0 | 1 | 0 | 11 |
| A1 | 1 | 0 | 11 |
| A2 | 1 | 0 | 11 |
| E | 1 | 7 | 2 |
| P0 | 1 | 5 | 13 |
| P1 | 1 | 6 | 13 |
| P2 | 1 | 4 | 13 |
| S0 | 1 | 6 | 13 |
| Us | 1 | 4 | 13 |
| S1 | 1 | 5 | 13 |
| U | 1 | 7 | 13 |
| d | 0 | 0 | 9 |

## 4.6 Implementing and verifying VHDL design

### 4.6.1 Implementation

Now that the controller's design, tuning, and fixed-point representation are known, this design can be converted to VHDL. To make a successful conversion, a functional diagram is created first. A diagram of the control loop can be seen in Fig. 4.33, an enlarged version of the functional diagrams of the control loop and all subcomponents can be found in appendix E.
This shows the introduction of multiplexers, allowing for selecting the control mode. A logical zero is applied to make the controller function in 24V mode with appropriate tuning, rate limit and reference. A logical one is applied to make the controller function in 48V with proper tuning, rate limit and 48V reference signal.

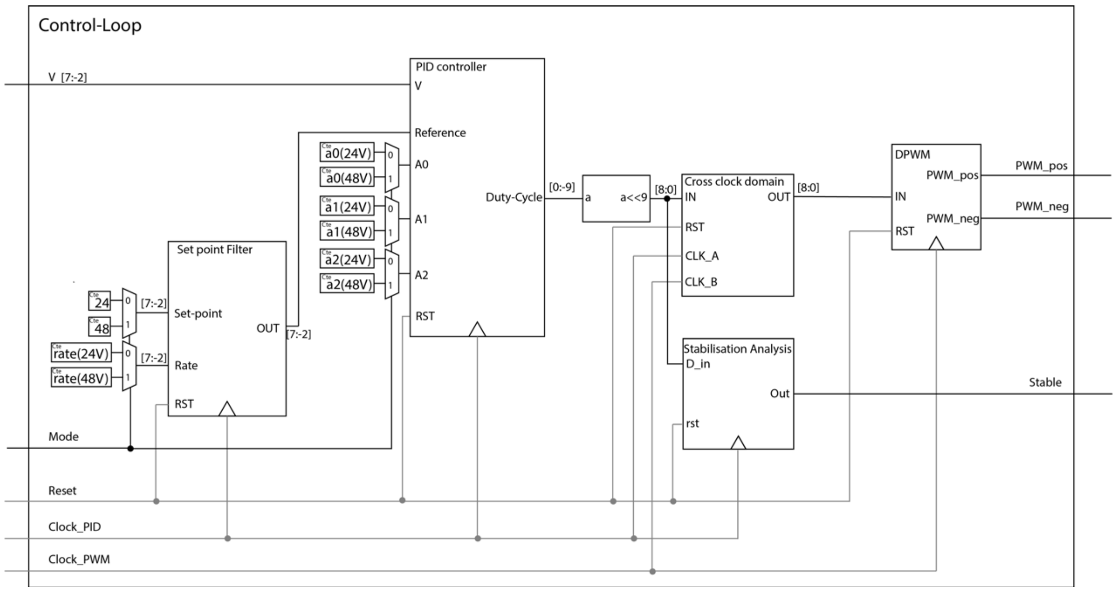


*Fig. 4.33 Functional implementation of the control loop*

Besides the introduction of the multiplexers, Fig. 4.33 shows another new component: The stabilisation analysis. This module analyses the duty-cycle word fed into the DPWM generator and signals when the controller has fully stabilised, in other words, when the duty cycle no longer changes. The logical implementation of the module can be seen in Fig. 4.34.

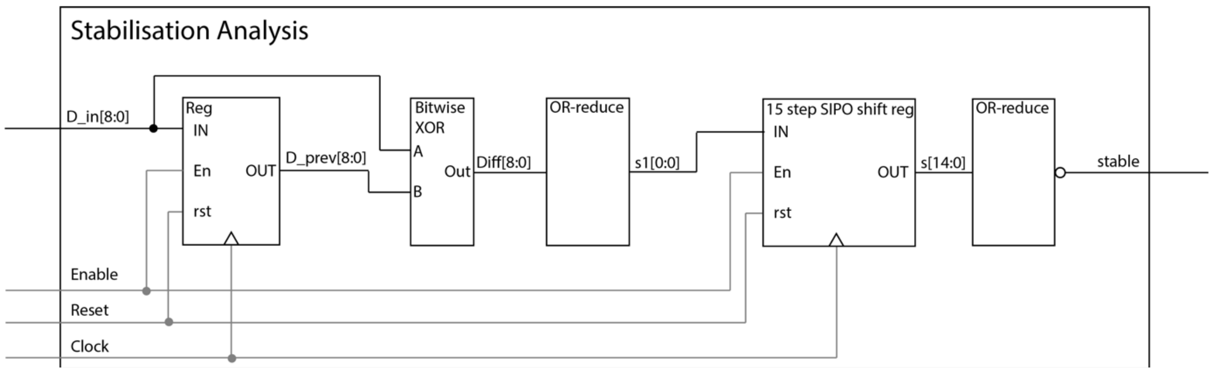


*Fig. 4.34 Functional implementation of the stabilisation indicator*

This module works by taking the binary bitwise differences, or XOR, of two subsequent duty-cycle commands and taking the reduced or form of these. If all bits are zero, meaning the two duty cycles are equal, a zero is shifted into a 15-step shift register. Finally, the “or”-form of all the different steps in the shift register is taken and inverted. If this signal is high, then the duty-cycle word had been the same for 15-cycles, indicating that the controller is fully stable. An example of this signal in action can be seen in Fig. 4.35, where the controller stabilises from its start-up sequence and is then destabilised twice by a voltage drop and a voltage jump at approximately 0.70 ms and 1.22 ms.

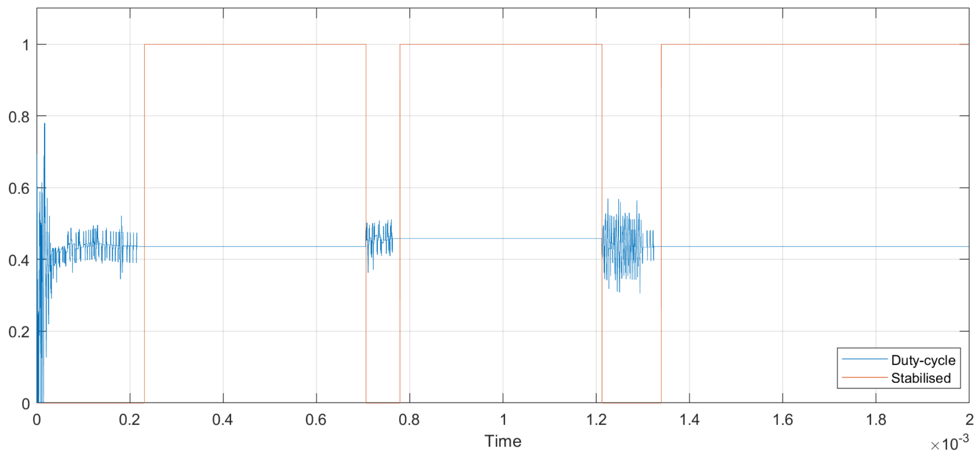


*Fig. 4.35 Output of the stabilisation indicator compared to the duty-cycle*

This design was implemented in VHDL. Initial testing of the VHDL design occurred by designing a testbench in Vivado and stimulating it with the voltage values recorded from the step response of the quantised MATLAB Simulink model. The design was corrected until no significant differences between the control signals generated from MATLAB Simulink and the testbench were found. Although working well to verify the basic operation of the control loop, this test method only provided data for very limited time intervals and did not form an actual control loop, meaning that small compounding errors would not be noted.

To thoroughly test the designed control loop, it needs to be paired with the simulation of the buck converter. This was done by pairing the ModelSim VHDL simulator to the MATLAB Simulink simulation using the co-simulation feature of the MATLAB’s HDL-verifier toolbox.

The schematic to perform this simulation can be seen in Fig. 4.36. It can also be noted that the VHDL control-loop block has two additional outputs: “Ud” and “V”. These were temporarily placed for debug purposes and allowed measuring the generator duty-cycle and measured voltage.

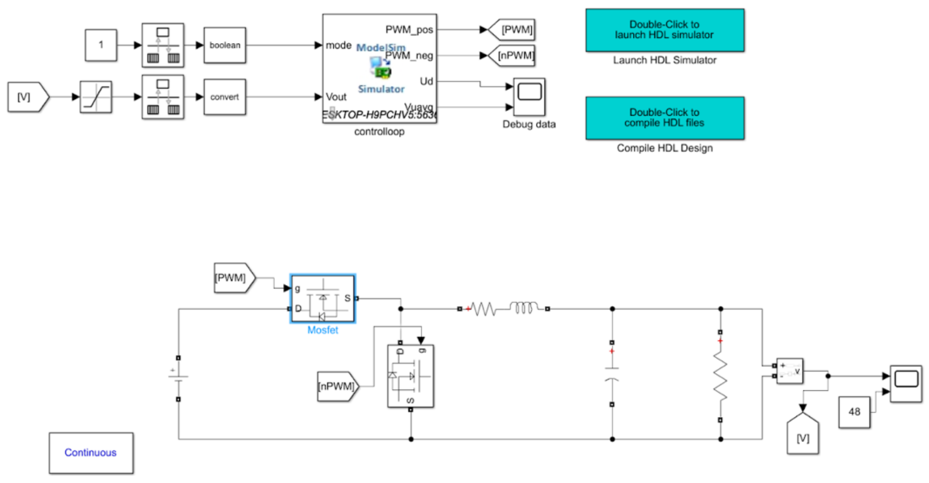


*Fig. 4.36 MATLAB Simulink schematic for VHDL Co-simulation*

During initial testing, some overshoot was found due to the simulated converter responding slightly differently than the transfer function. The rate limit of the setpoint filter was reduced experimentally to 3.75V/µs for the 48V controller and 2.5V/µs for the 24V controller to minimise the overshoot to zero.

### 4.6.2 Verification

Using the described testing method, the proper functioning of the VHDL descriptions can be verified. The results for simulating with this method are shown in Fig. 4.37.

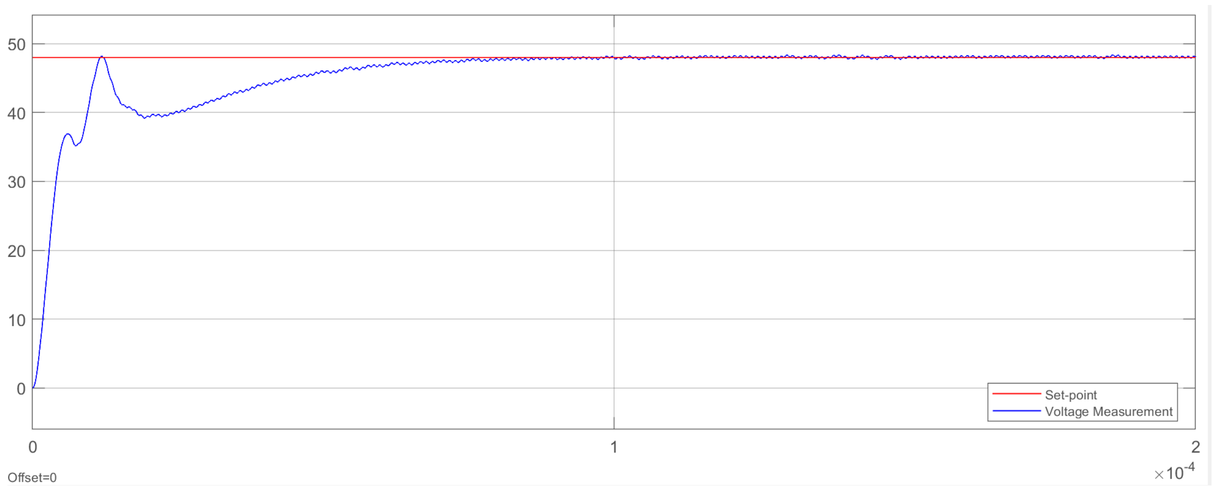


*Fig. 4.37 Step response of 48V Buck converter controlled by VHDL-simulated controller*

From Fig. 4.37 it can be concluded that the VHDL controller appears to function reasonably well. However, some steady-state oscillations occur. A close-up of these oscillations can be found in Fig. 4.38.

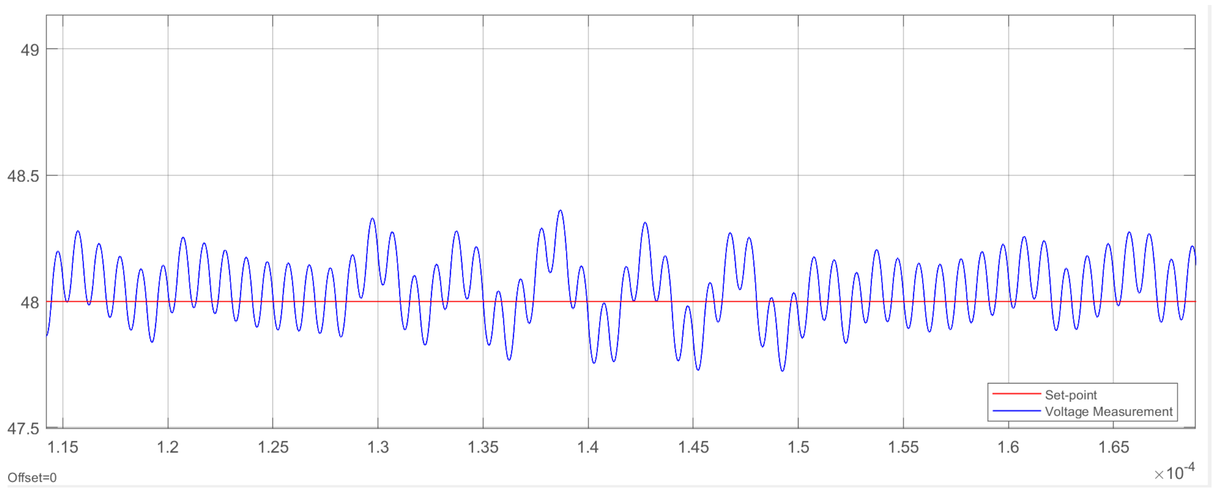


*Fig. 4.38 Oscillation close-up of 48V buck converter controlled by VHDL-simulated controller*

The oscillations shown in Fig. 4.38 introduce a new issue: previously, the clock of the PWM generator was perfectly in sync with the clock of the PID controller and thus the sampling occurred at the same moment in a switching period. Due to ModelSim Simulator resolution, these clocks are not perfectly in sync, meaning that the voltage measurement will not always occur at the same moment in a switching cycle. As a result, sometimes a higher value is measured (when measuring near the end of the charging period), or a lower value is measured (when measuring near the end of the discharge period).
The controller compensating for these measurements is precisely what causes these oscillations.

The problem is that the sampling frequency is 1 MHz, whilst the highest frequency occurring in the system is also 1 MHz. This is not conforming to the Nyquist–Shannon sampling theorem, which states that the sampling frequency should be at least twice as high as the highest frequency occurring in the observed system.

The solution to this problem is to increase the sampling frequency. However, the frequency of the PID controller should not be raised as a control value should be offered to the PWM-module for one switching period. By measuring the voltage four times per switching period and averaging these four measurements, the PID controller can still function at 1 MHz without the aliasing effects that occurred before. Taking four samples effectively samples the output voltage at a frequency of 4 MHz. this was opted over sampling at 2 MHz to make sure the Nyquist–Shannon sampling theorem is still met should the clock frequency of the PWM module be slightly higher than expected or the clock frequency of the PID module slightly lower then expected.
The implementation of the new sampling block can be seen in Fig. 4.39**.** This does change the required PID clock speed from 1 MHz to 4 MHz. The sample block generates an enable signal once every four clock cycles to ensure the PID logic is still functioning at 1 MHz. A verification to whether the clock speed was attainable has been made and showed that using the 180nm BCD technique from the TSMC foundry, the minimum clock period was 21ns, resulting in a maximum frequency of approximately 47.6 MHz, a factor 10 larger than the clock frequency needed by this new implementation.
The new complete control loop with sample averaging can be seen in Fig. 4.40. This shows that due to this change, the resolution of the input voltage has increased from 2 bit fractional to 4 bit fractional. This prevents compounding quantisation noise from significantly affecting the average value measured. However, it is very important that the output signal of the average module only has 2 fractional bits to uphold the equation to prevent LCO.

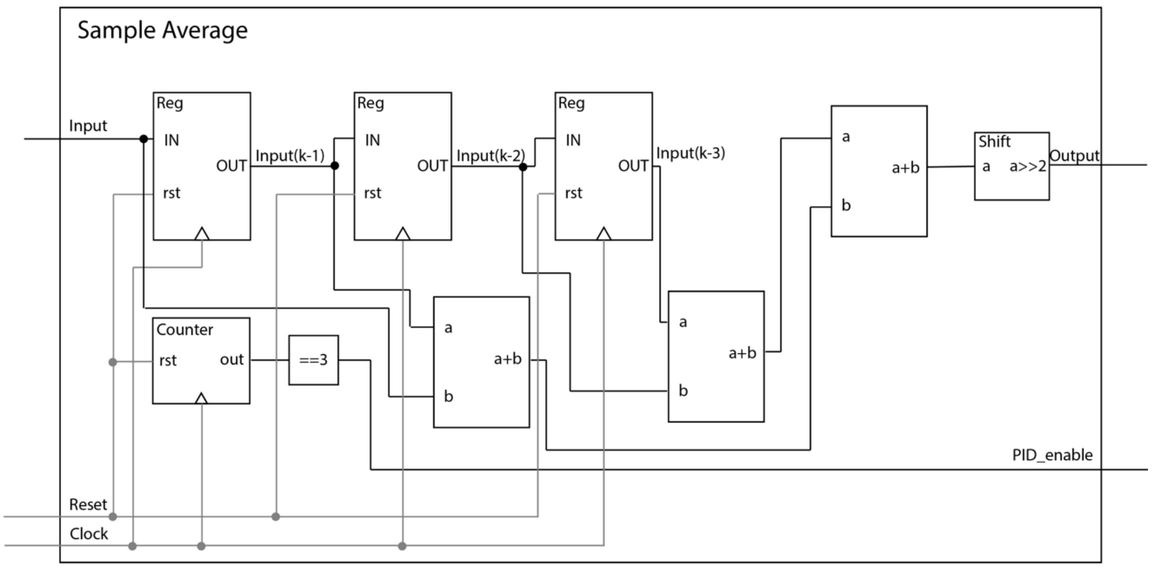


*Fig. 4.39 Functional implementation of the sample averaging module*

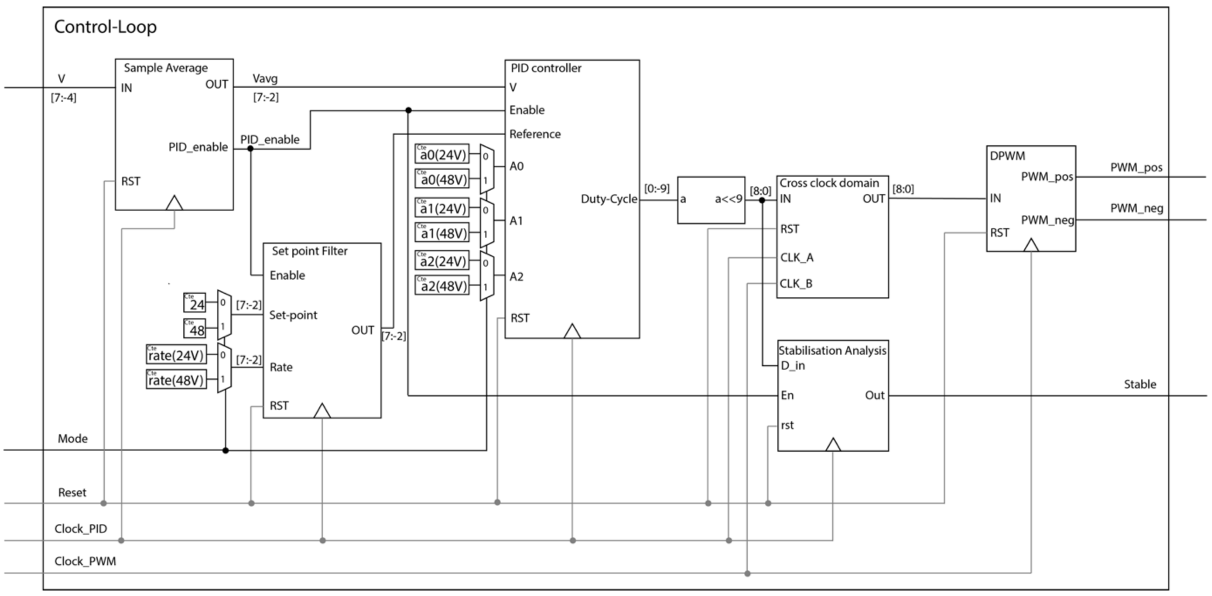


*Fig. 4.40 Functional implementation of the control loop with sample averaging*

The steady-state response shown in Fig. 4.41 is obtained using this method, showing none of the variations seen in Fig. 4.38.

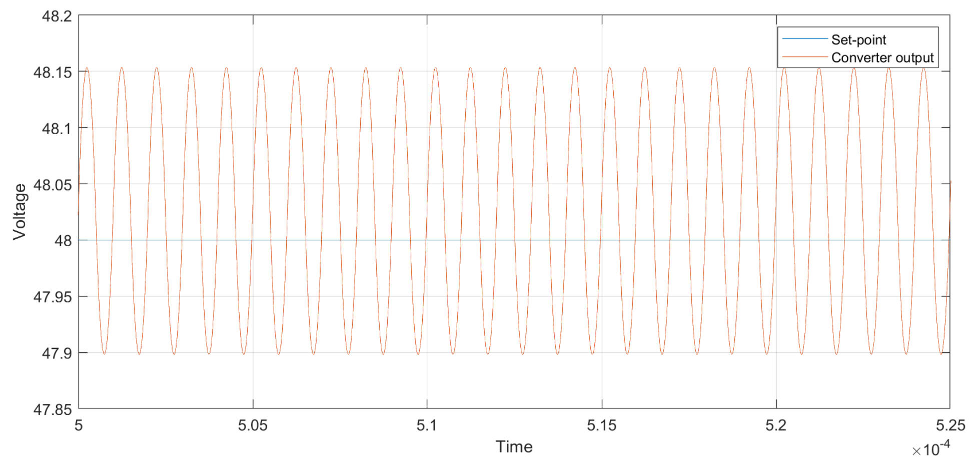


*Fig. 4.41 Oscillation close-up of 48V buck converter controlled by VHDL-simulated controller with 4-sample averaging*

## 4.7 Conclusions

This section will give a short overview of all options attempted in chapter 4 and clarify why these were or were not integrated into the project.

### 4.7.1 Tuning

Three tuning methods were considered to determine the PID controller's gain parameters. The tuning obtained from the SIMC method did not allow to be converted into a digital controller with a realistic sampling frequency. The Frequency loop shaping method did not result in a good tuning, being considerably slower than both MATLAB’s transfer function tuner and frequency response tuner’s results.
The MATLAB transfer function tuner did outperform the frequency response tuner and is, therefore, the method of choice for this thesis.

### 4.7.2 Controller improvements

Three potential improvements were considered: a nonlinear PID controller, the implementation of a low pass filter in the controller's derivative component, and a setpoint filter.

The nonlinear PID controller did not improve performance over the regular PID controller whilst requiring extra resources and is therefore not used.
Implementing a low pass filter required retuning the controller. However, no stable tuning was found. This caused experiments with this alteration to be aborted.
The setpoint filter allowed to reduce the overshoot on a slightly faster tuning making this tuning viable. Using this other tuning allowed for better disturbance rejection on both converters and a better step response on the 24V controller. These advantages, bundled with the fact that this implementation required little resources, caused the setpoint filter to be implemented in this project.

### 4.7.3 DPWM output stage

A comparison between two DPWM output stages has been made: A sawtooth comparator with Σ–Δ generator and a standalone sawtooth comparator.
Testing a simulated Σ–Δ generator in Simulink led to the conclusion that the power converter does not have enough averaging capabilities to function with such a generator. This causes more variations than the effect of LCO would have. Therefore, it was opted to use the standalone sawtooth comparator. The output stage must be altered to the maximum clock frequency of 444 MHz. This involves adjusting the controller’s gain and verifying that no LCO occurs.

### 4.7.4 Fixed point representation

By carefully selecting both the integral and fractional representations of the controller make it possible to obtain a fixed-point implementation of the PID control algorithm that behaves very similar to a more accurate floating-point implementation whilst using significantly fewer resources.

### 4.7.5 Sampling theorem

The initial design of the controller relied on the fact that the clock controlling the PID-module was perfectly in sync with the clock controlling the PWM-module. When this turned out not to be the case in co-simulation, issues arose due to not meeting the sampling theorem.
The sampling frequency of the controller was increased to 4 MHz instead of 1 MHz to prevent this issue. A new sampling module will take 4 samples, calculate the average and feed this to a PID controller that still functions at 1 MHz, as each duty cycle control word need to be offered to the PWM module for one switching period.

### 4.7.6 Overview

A short overview of the design decisions listed above is made in Table 4.16, together with the primary reason for integrating or not integrating the listed technology.

*Table 4.16 Overview of design decisions*

| | Tuning | | | | Improvements | | | Output stage | |
|---|---|---|---|---|---|---|---|---|---|
| | SIMC | FLS | MATLAB Transfer function tuner | MATLAB Frequency response tuner | NL-PID | Low pass filter | Setpoint filter | Σ–Δ generator + counter | Counter |
| Implemented | | | Best perfor-mance | | | | Perfor-mance gain at low resource cost | | Stable |
| Not implemented | Not convertible to discrete time domain | Bad perfor-mance | | Out-performed | No perfor-mance gain | No stable tuning found | | Causes variations | |

# 5 Evaluation

The design of the controller and its VHDL implementation according to the schematics in appendix E has been tested by co-simulation of MATLAB Simulink and ModelSim using the schematic previously seen in Fig. 4.36. During these tests, the following benchmarks will be used:

| | |
|---|---|
| Rise time | The time required for the system to rise from 10% of the final value to 90% of the final value. For the 24V system, this means the time between reaching 2.4V to 21.6V. For the 48V system, this means the time between reaching 4.8V to 43.2V. |
| Settling time | The time required for the system to reach and maintain within an error band of 2% of the final value. For the 24V system, this means the time until the system remains within an error band of [23.52V ; 24.48V]. For the 24V system, this means the time until the system remains within an error band of [47.04V ; 48.96V]. |
| Overshoot | The amount of overshoot occurred as a result of the PID controller steering the controller toward the setpoint during the step response. Expressed as a percentage of the setpoint value. |
| Maximum variation | The maximum deviation from the setpoint after settling, during the stabilising phase. Expressed as voltage. |
| Stabilisation time | The time needed for the controller to fully stabilise |

These tests will be performed on various loads to prove the correct functioning under different load conditions. The load will be determined by power, making measurements for 100 W, 10 W, 1 W and 1 nW.

## 5.1 48V DC-DC converter

Testing the 48V DC-DC buck converter, controlled by a VHDL simulation of the controller over the power values mentioned above, requires testing with the load values shown in Table 5.1.

*Table 5.1 Resistances used for testing the 48V buck converter*

| Load (W) | Required resistance (Ω) |
|---|---|
| 100 | 23.04 |
| 10 | 230.4 |
| 1 | 2304 |
| $1 * 10^{-9}$ | $2304 * 10^{6}$ |

### 5.1.1 Step and disturbance response

Testing under the loads from Table 5.1 results in the step-responses as shown in Fig. 5.1.

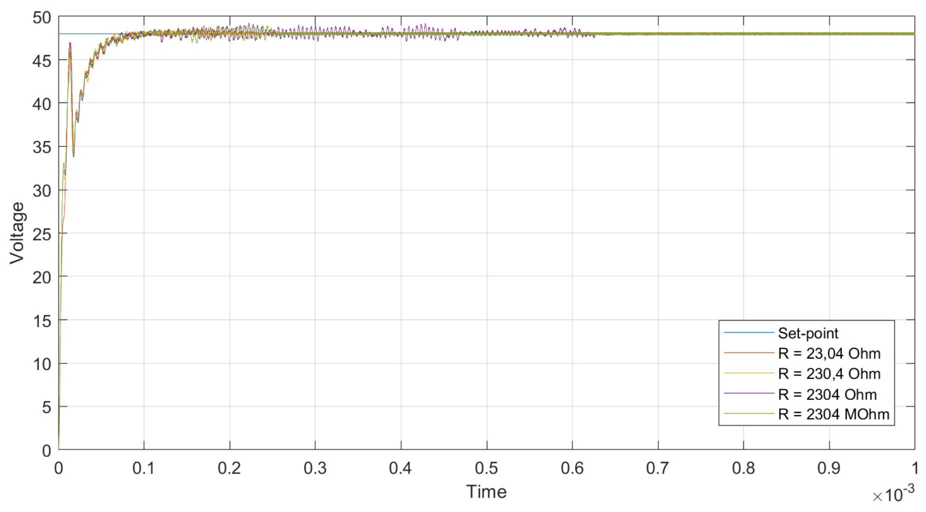


*Fig. 5.1 Step-responses of 48V buck converter controlled by VHDL-simulated controller under various loads*

Fig. 5.1 shows that the step responses for different loads are very similar, showing that the effect of load on the closed-loop response is limited when using a well-tuned controller. It also indicates that the controller successfully obtains the target value of 48V in a relatively short time frame. The process the controlled systems go through can be split into three stages: first, the controlled system rises to the setpoint value of 48V. Then the controller slightly oscillates around the setpoint in a similar fashion to LCO. This oscillation results from the control resolution being very close to the measurement resolution and could be reduced by increasing the control resolution. However, the switching frequency and maximum clock frequency limit the control resolution for the chosen implementation. After the oscillation in the second phase, the controller reaches a third, fully stable phase. The exact benchmarks of the controller can be found in Table 5.2.

*Table 5.2 Performance benchmark of the 48V buck converter controlled by VHDL-simulated controller*

| | 23.04Ω (100W) | 230.4Ω (10W) | 2304Ω (1W) | 2304MΩ (1nW) |
|---|---|---|---|---|
| Rise time (µS) | 11.060 | 10.331 | 10.200 | 10.264 |
| Settling time (µS) | 55.323 | 54.373 | 58.314 | 68.315 |
| Overshoot (%) | 0 | 0 | 0 | 0 |
| Max variation (V) | 0.5986 | 0.4737 | 1.2100 | 0.9514 |
| Stabilization time (µS) | 216.3 | 189.3 | 628.2 | 255.1 |

The controller's performance, as described in Table 5.2, shows similar performance to that of MATLAB's simulation of the control architecture.
It also shows how the stabilisation time takes longer than for others. This is related to how close the measurement of the stable state is to the subsequent measurement resolution step. Should this be close, then the inertia of oscillation of the buck converter itself could "push" the measurement to the next value and have the controller respond to this. If multiple control options within one measurement window were available, this effect would be much less pronounced. This confirms that increasing the control resolution would decrease this problem.

Fig. 5.2 shows the system's oscillations after the controller has successfully stabilised.

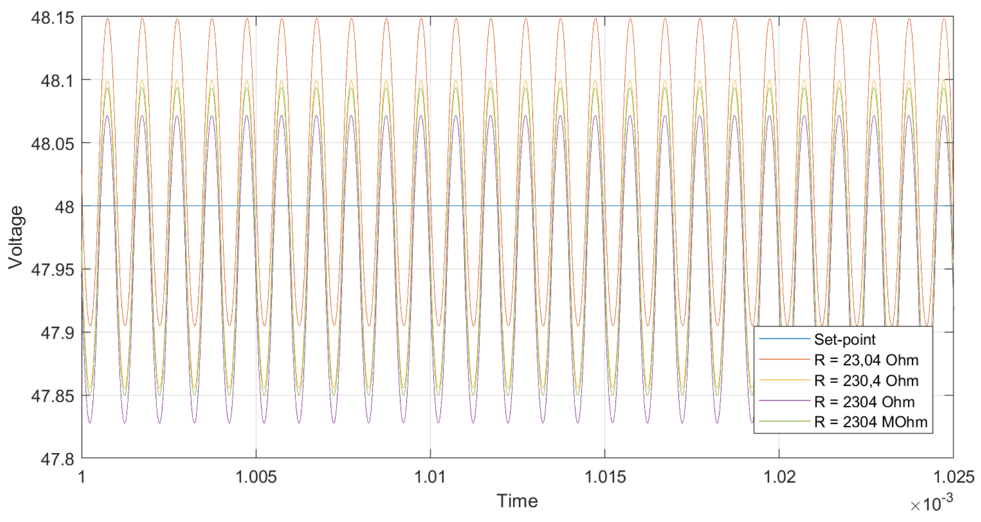


*Fig. 5.2 Steady-state oscillation of 48V buck converter controlled by VHDL-simulated controller under various loads*

The fast oscillations seen in Fig. 5.2 are the ripple of the buck converter with a frequency of approximately 1 MHz.

Besides a good start procedure, the controller also needs to recover from a disturbance. Fig. 5.3 shows the effect of the controller's response when the input voltage drops suddenly by 5V. The maximum output-voltage drop is less than 1V, and the controller quickly brings the voltage back into the 48V range but needs a little more time to recover completely. The time required to re-attain 48V is approximately 30 µs, whilst the time required to fully re-stabilise is approximately 140µs.
A similar effect can be found when analysing the recovery of a sudden increase of input voltage by 5V, shown in Fig. 5.4. The maximum output-voltage rise is again less than 1V, and the controller requires approximately 120 µs to recover and re-stabilise fully.

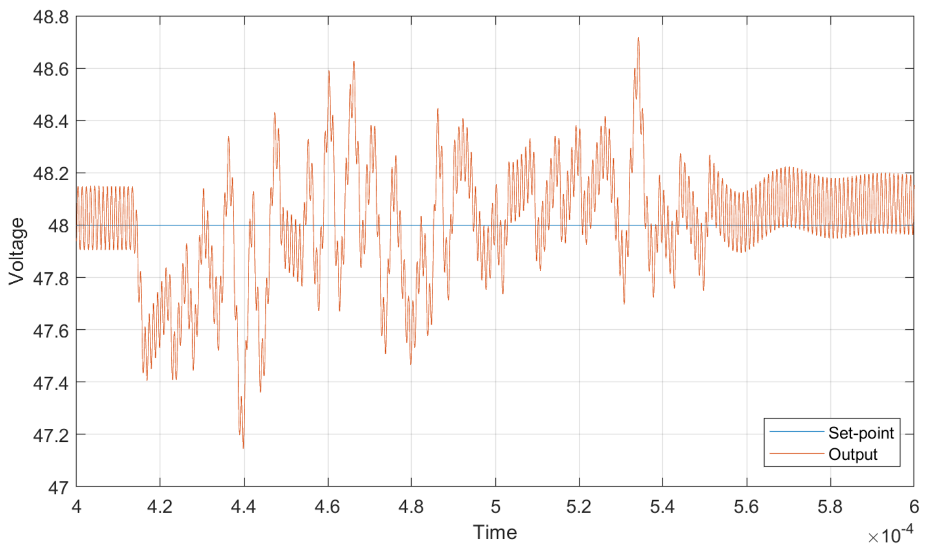


*Fig. 5.3 Recovery after 5V drop of 48V buck converter controlled by VHDL-simulated controller*

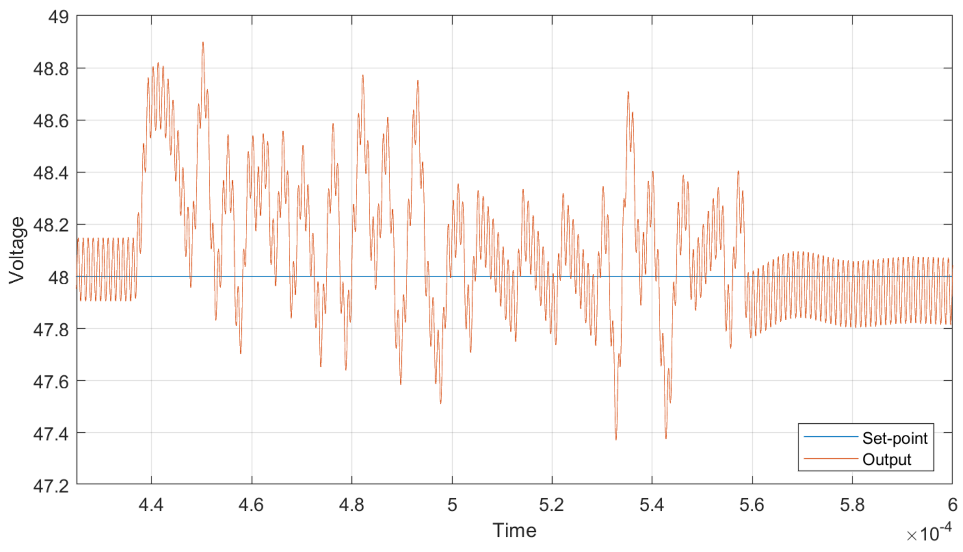


*Fig. 5.4 Recovery after 5V jump of 48V buck converter controlled by VHDL-simulated controller*

### 5.1.2 Limitations on input voltage

The effect of input voltage can also be analysed for the obtained controller design.
The minimum input voltage can be determined based on the controller's efficiency, as the maximum duty cycle equals the efficiency. These calculations are done using a provisional efficiency of 90%, as the converter's efficiency was not known at the moment of writing this thesis.

$$V_{in} \cdot Dutycycle = V_{out}$$

$$V_{in,min} \cdot \eta \geq V_{out}$$

$$V_{in,min} \cdot 0.90 \geq 48$$

The smallest input voltage for this inequality to be true is 53.33V.
This minimum results from the physical limitation of the buck-converter being controlled and cannot be improved upon by changing the controller design.

The maximum voltage is limited by both the gain margin of the PID controller with converter and the equation to inhibit LCO.
Using the currently used resolutions, LCO was a larger inhibitor than the amplitude margin.
The maximum voltage can be calculated as follows:

$$V_{in} * \frac{1}{N_{pwm\ steps}} < \Delta V_{LSB}$$

$$V_{in} \cdot \frac{1}{444} < 2^{-2}$$

The largest value of input voltage for this inequality to be true is 111V. It should be noted that as the voltage gets closer to this 111V limitation, the controller will have more trouble stabilising fully, requiring more time to do so.
This maximum could be increased by either decreasing the measurement resolution (which will have as a side effect that the potential steady-state error increases) or increasing the control resolution.

The confirmation of the converter's stable behaviour in this voltage range can be verified, as shown in Fig. 5.5**.** It also shows that the converter's step response is slightly slower for lower voltages as the control loop gain is lower. It also indicates that the controller needs longer to stabilise for voltages approaching the maximum voltage.

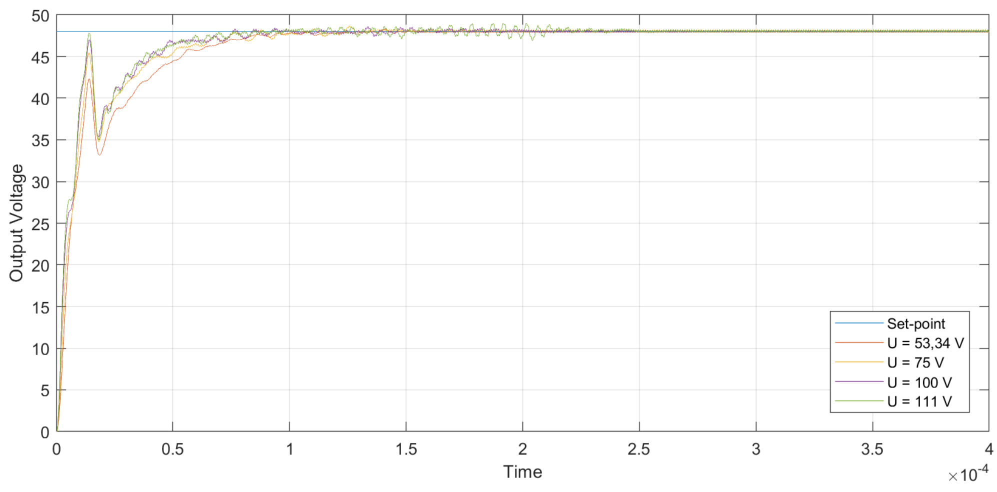


*Fig. 5.5 Influence of input voltage within the operating range on the 48V buck converter*

It can also be confirmed that the controller does not function properly when the voltages are outside this operating range, as shown in Fig. 5.6. This Fig. shows that for voltages lower than the minimum, the controller doesn't obtain the correct value, the output voltage remains significantly lower. When the output voltage is higher than the calculated maximum, the controller will never stabilise due to the effect of LCO.

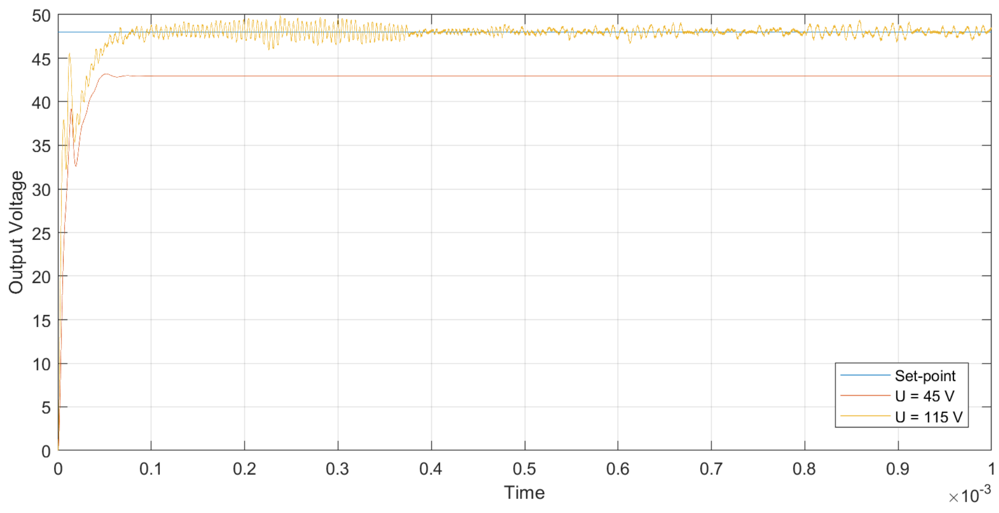


*Fig. 5.6 Influence of input voltage outside the operating range on the 48V buck converter*

## 5.2 24V DC-DC converter

Testing the 48V DC-DC buck converter, controlled by a VHDL simulation of the controller over the power values mentioned above, requires testing with the load values shown in Table 5.3.

*Table 5.3 Resistances used for testing the 24V buck converter*

| Load (W) | Required resistance (Ω) |
|---|---|
| 100 | 5.76 |
| 10 | 57.6 |
| 1 | 576 |
| 1 * $10^{-9}$ | 576 * $10^{6}$ |

### 5.2.1 Step and disturbance response

Testing under the loads from Table 5.3 loads results in the step-responses as shown in Fig. 5.5.

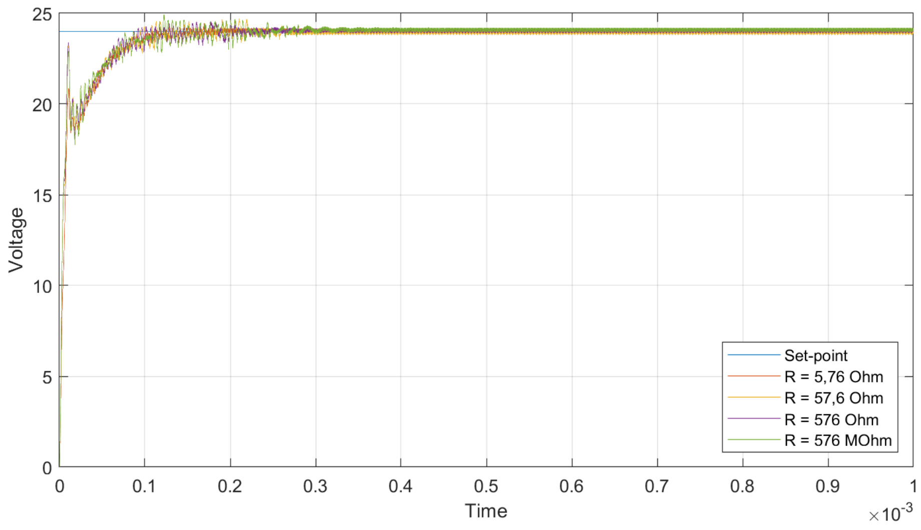


*Fig. 5.7 Step-responses of 24V buck converter controlled by VHDL-simulated controller under various loads*

The step responses from Fig. 5.7, show again that the effect of load on the buck converter is minimal, obtaining a steady-state of 24V for each of the loads. A stabilisation period can be seen again. The exact benchmarks of the controller can be found in Table 5.4.

*Table 5.4 Performance benchmark of the 24V buck converter controlled by VHDL-simulated controller*

| | 5.76Ω (100W) | 57.6Ω (10W) | 576Ω (1W) | 576MΩ (1nW) |
|---|---|---|---|---|
| Rise time (μS) | 9.3814 | 8.7952 | 8.5827 | 8.7545 |
| Settling time (μS) | 95.077 | 111.31 | 168.07 | 203.24 |
| Overshoot (%) | 0 | 0 | 0 | 0 |
| Max variation (V) | 0.4240 | 0.7020 | 0.5511 | 0.9243 |
| Stabilization time (μS) | 179.0 | 261.1 | 237.1 | 282.0 |

Once the controller has fully stabilised, no further oscillations, besides the ripple of the buck converter, can be seen as displayed in Fig. 5.8.

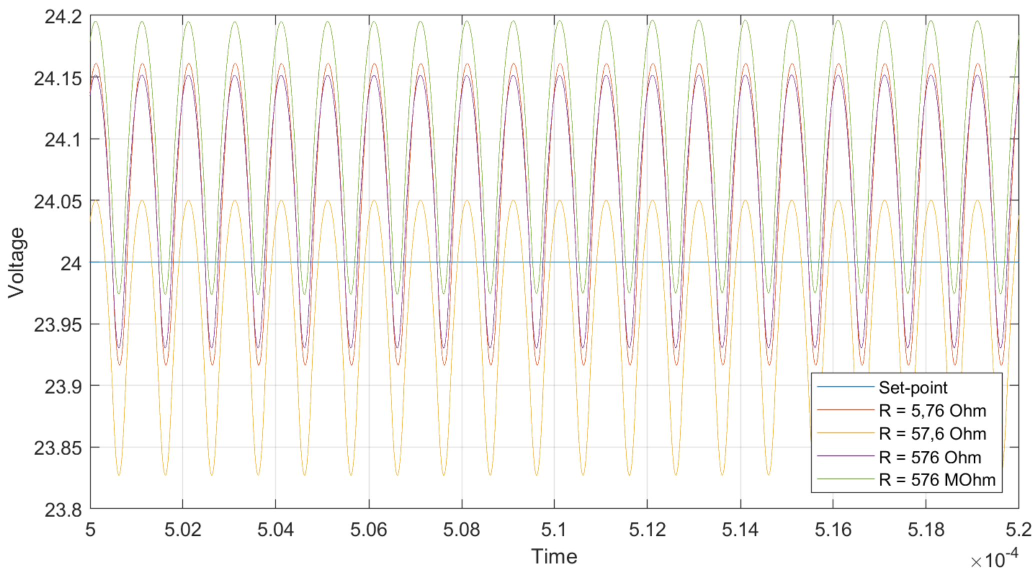


*Fig. 5.8 Steady-state oscillation of 48V buck converter controlled by VHDL-simulated controller under various loads*

Analysing the recovery from a 5V drop and 5V jump, as shown in Fig. 5.9 and Fig. 5.10, respectively. The maximum deviation that occurs from the 5V drop is approximately 0.5V, and the total re-stabilisation takes approximately 70 µs. The maximum deviation that occurs from the 5V jump is 0.55V, and the time needed for complete stabilisation is approximately 80 µs showing similar results as the 48V buck converter.

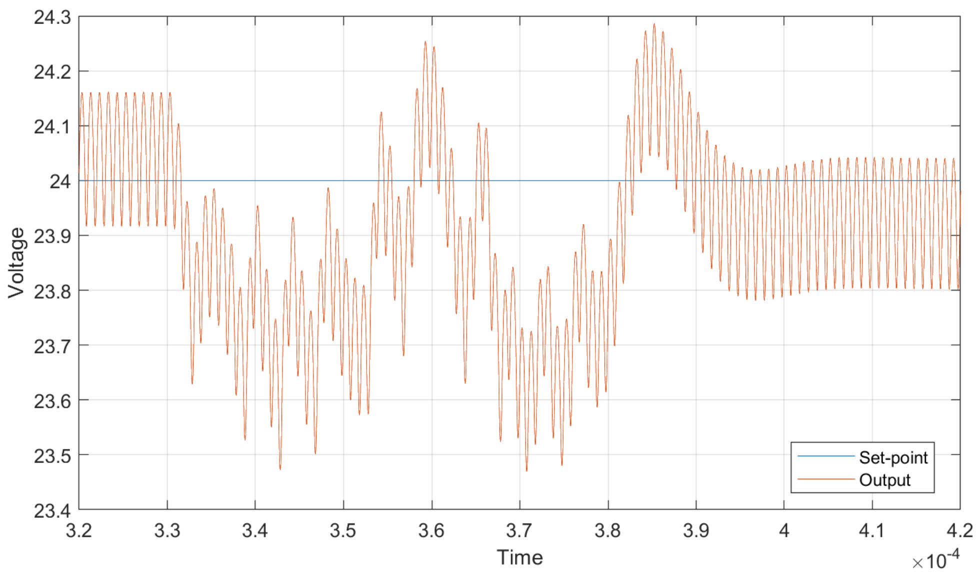


*Fig. 5.9 Recovery after 5V drop of 48V buck converter controlled by VHDL-simulated controller*

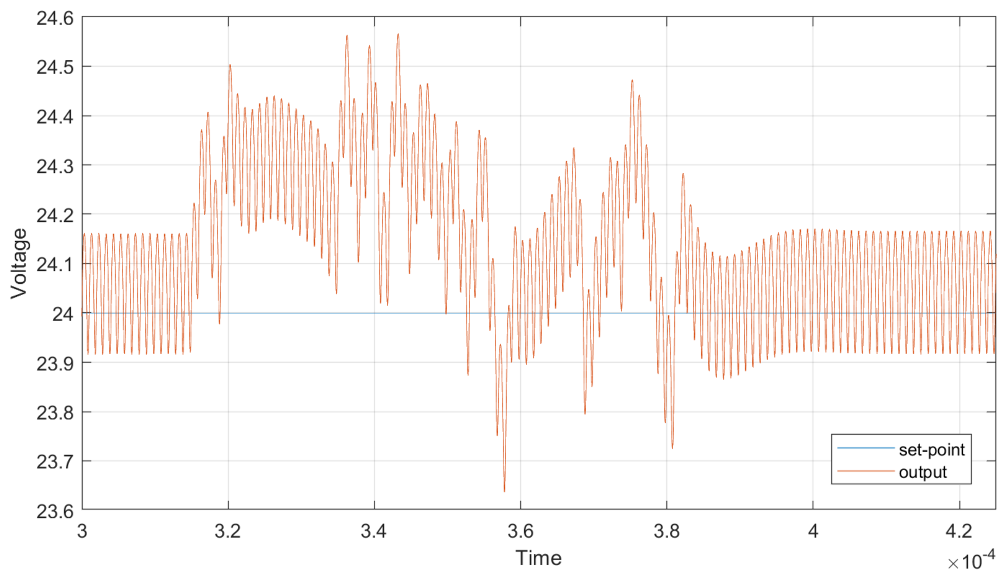


*Fig. 5.10 Recovery after 5V jump of 48V buck converter controlled by VHDL-simulated controller*

### 5.2.2 Limitations on input voltage

A similar analysis on input voltage can be made as done in section 1.1.2.

First, the minimum voltage can be determined based on the controller's efficiency. Again, these calculations will be made using a provisional efficiency of 90% as the converter's efficiency has not yet been determined.

$$V_{in,min} \cdot \eta \geq V_{out}$$

$$V_{in,min} \cdot 0.90 \geq 24$$

The minimum value for the equation to uphold is 26.67V.

The calculation of maximum voltage remains the same as the resolutions are the same for the 24V and 48V controller, meaning the maximum input voltage is 111V.

The stable operation for voltages inside the operating parameters is verified, as shown in Fig. 5.11. Indication a similar influence of input voltage on the control loop. For lower values, the controller takes longer to obtain a value of 24 V, whilst for values close to the maximum voltage, the controller needs longer to stabilise.

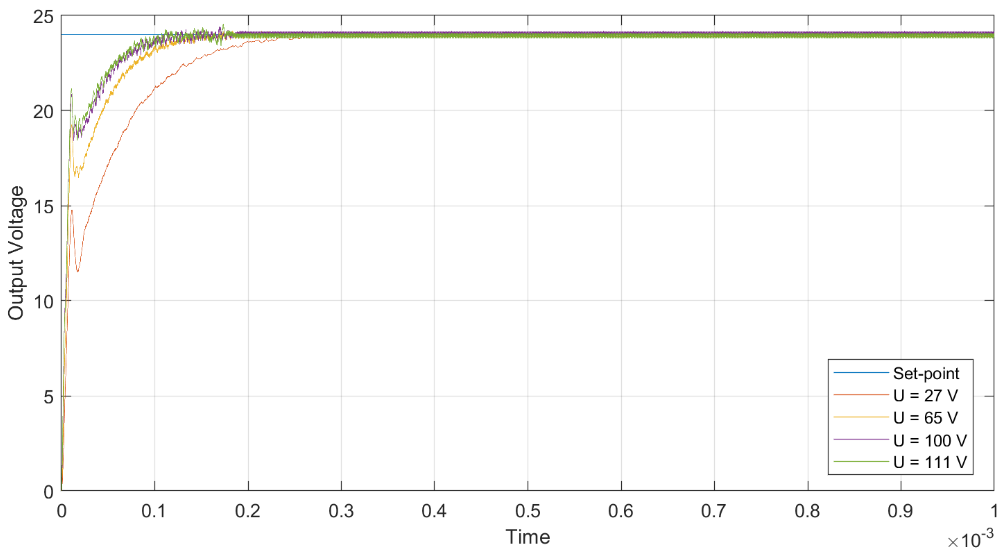


*Fig. 5.11 Influence of input voltage within the operating range on the 24V buck converter*

Faulty behaviour of the controller can be confirmed for voltages outside the operating range, as seen in Fig. 5.12. Again, this shows that the controller cannot reach the correct voltage value for values lower than the input voltage minimum and suffers from LCO for input voltage values higher than the maximum.

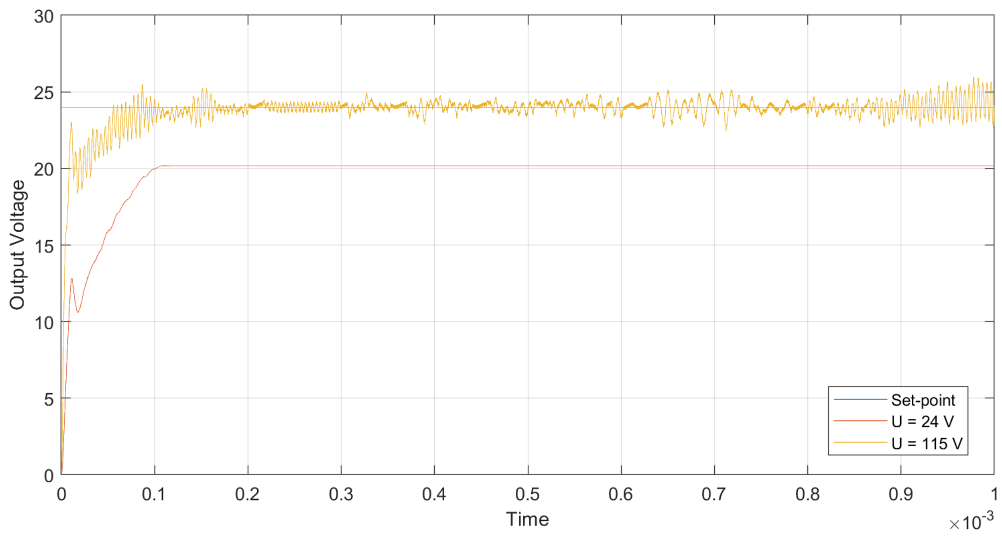


*Fig. 5.12 Influence of input voltage outside the operating range on the 24V buck converter*

## 5.3 Area and resources

The VHDL files based on the designs in appendix E can be converted to an ASIC implementation. Doing this synthesis for the 180nm BCD technique from the TSMC foundry results in an implementation requiring 1012 cells and needing a total area of 74,727.402 $\mu m^2$. This area was determined using a 4 MHz frequency for the PID clock and a 435 MHz frequency for the PWM clock. For reference, the smallest NAND-gate in this technology requires a surface of 18.816 $\mu m^2$.

# 6 Future work

This thesis provided the first step in designing the controller needed for the power converter in a package. A short consideration into improvements that could still be integrated and the next steps in the development process are given.

## 6.1 Potential improvements

As discussed in chapter 5, the controller's performance evaluation shows that the controller successfully stabilises, both the 48V and 24V variants, to the respective setpoints and manages to do so relatively quickly. One of the more unfortunate side effects from both the step and disturbance response is that, although the controller quickly reasserts the converter to a value near the setpoint, some LCO-like oscillations occur. These oscillations result from the measurement resolution being almost as high as the control resolution. Decreasing the measurement resolution would be one potential solution to this problem but would increase the amplitude of the oscillations, nonetheless it would shorten the duration. A better solution to this problem would be to increase the control resolution. Tests using a 512 MHz DPWM rather than a 444 MHz DPWM yields faster stabilisation. However, this would require the DPWM generator to be redesigned to meet a minimum clock period of 1.95 ns rather than the current limitation of 2.25 ns.
When increasing the resolution of the controller it might also be worthwhile to analyse the effect of decreasing the gain of the controller. The full potential of more accurate control might get wasted when the control-action over the minimal measurable error is much larger than the controller’s new resolution.

## 6.2 Further development

### 6.2.1 Converter parameter change

Should the parameters of the 48V or 24V buck converter change or an additional converter be introduced, the design of the controller should be adapted accordingly. The following steps should be repeated, as described in chapter 4 of this thesis, to obtain an altered controller design:

- The PID controller should be re-tuned
- The maximum rate of the setpoint filter should be redetermined to have no overshoot
- The maximum value of the saturator should be determined
  - For the feedback signal internally in the PID controller by simulating in MATLAB, determining the maximum value, and applying a safety margin
  - For the duty-cycle output by taking the efficiency of the controller and multiplying this by the DPWM correction rate (444/512)
- The bit widths should be redetermined
- The obtained design should be verified by co-simulation

Should the parameters of the converter change significantly, it might be feasible to re-analyse if the implementation of a Σ–Δ generator in the output stage could be used to improve the control resolution of the controller.

### 6.2.2 Testing and implementation

Should the performance of the controller be deemed sufficient for integrating into the CoDiCApp project, the next step would be to test the design of the controller in the real world, rather than simulation, by implementing the design on an FPGA and connecting this to a buck-converter with same capacitance and inductance as the buck converter's final design.

The obtained design can be converted into an ASIC should these tests yield positive results. Once this has been thoroughly tested, this ASIC could be integrated into the power converter in a package.

# 7 Conclusion

This thesis aimed to design a VHDL implementation of a controller that could control the 24V and 48V buck converter developed parallel in time with this project.
A VHDL implementation of a PID controller was developed and verified.
Although the final testing method was done through co-simulation rather than real-world testing using an FPGA and a physical buck converter due to lack of time and access to a similar buck converter, this still proved a suitable method of validating the obtained VHDL design.

Several tuning methods were tried and tested to analyse what method would yield the best methodology in designing a digital PID controller for a DC-DC buck converter. After analysing the (K-)SIMC, Frequency loop shaping, and both MATLAB's transfer function and frequency response based tuning methods, the MATLAB's transfer function tuner yielded the best results.

Three improvements to a traditional PID controller were considered and tested. Firstly, varying the PID parameters based on the magnitude of the error (nonlinear PID) did not improve the controller's response. Secondly, no stable tuning was found to test a PID controller with LPF in the derivative term. Finally, tests using a rate limiter as a setpoint filter allow for a slightly more aggressive tuning, improving the disturbance rejection whilst improving the step-response for the 24V controller.

Two DPWM output stages were designed in MATLAB and analysed: a Σ–Δ generator with a PWM counter and a sole PWM counter. Although a Σ–Δ generator aims to reduce variations by allowing a higher measurement resolution without causing LCO, it showed that the buck converter did not have sufficient averaging capabilities and introduced a larger variation than what LCO would have. Due to this, it was opted to use the sole counter DPWM generator. This counter counts to 444 to fully utilise the available clock speed of 444 MHz whilst maintaining a switching frequency of 1 Mhz.

The above results were combined into a VHDL implementation and tested using MATLAB Simulink and ModelSim co-simulation. These results showed that the closed-loop step response with quantisation noise can be split into three phases: first, the controller follows the PID logic to obtain the setpoint value. Secondly, the controller oscillates slightly around the setpoint in a fashion similar to LCO. Finally, the controller reaches a stable state, and the only oscillation measured is the ripple of the buck-converter.

# Bibliography

[1] H. K. Krishnamurthy *et al.*, “Digital Control of Switching and Linear Integrated Voltage Regulators,” *Proc. Cust. Integr. Circuits Conf.*, vol. 2020-March, no. c, pp. 9–12, 2020, doi: 10.1109/CICC48029.2020.9075934.

[2] M. Kocur, S. Kozak, and B. Dvorscak, “Design and implementation of FPGA - Digital based PID controller,” *Proc. 2014 15th Int. Carpathian Control Conf. ICCC 2014*, pp. 233–236, 2014, doi: 10.1109/CarpathianCC.2014.6843603.

[3] J. Lima, R. Menotti, J. M. P. Cardoso, and E. Marques, “A methodology to design FPGA-based PID controllers,” *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.*, vol. 3, pp. 2577–2583, 2006, doi: 10.1109/ICSMC.2006.385252.

[4] E. William, J. Linares-Flores, E. Guzman-Ramirez, and H. Sira-Ramirez, “FPGA Implementation of PID Controller for the Stabilization of a DC-DC ‘Buck’ Converter,” *Front. Adv. Control Syst.*, 2012, doi: 10.5772/39083.

[5] Y. F. Chan, M. Moallem, and W. Wang, “Efficient implementation of PID control algorithm using FPGA technology,” *Proc. IEEE Conf. Decis. Control*, vol. 5, pp. 4885–4890, 2004, doi: 10.1109/cdc.2004.1429572.

[6] M. Truntic, M. Milanovic, E. Vidal-Idiarte, C. E. Carrejo, and C. Alonso, “Digital current mode and voltage control for the dc-dc step-up converter based on the FPGA technology,” Sep. 2010, doi: 10.1109/EPEPEMC.2010.5606547.

[7] R. Mammano, “Switching Power Supply Topology Voltage Mode vs. Current Mode,” *Unitrode*, pp. 1–4, 1994, [Online]. Available: Switching Power Supply Topology Voltage Mode vs. Current Mode,” Unitrode.

[8] J. Rodriguez, P. Cortes, R. Kennel, and M. P. Kazmierkowski, “Model predictive control - A simple and powerful method to control power converters,” *2009 IEEE 6th Int. Power Electron. Motion Control Conf. IPEMC ’09*, vol. 56, no. 6, pp. 41–49, 2009, doi: 10.1109/IPEMC.2009.5289335.

[9] O. M. A. Mañozca and B. De Moor, “Presentation - Introduction to Model Predictive Control (ESAT),” 2017, [Online]. Available: https://homes.esat.kuleuven.be/~maapc/static/files/CACSD/Slides/all_slides_mpc.pdf.

[10] T. J. Vyncke, S. Thielemans, and J. A. Melkebeek, “Finite-set model-based predictive control for flying-capacitor converters: Cost function design and efficient FPGA implementation,” *IEEE Trans. Ind. Informatics*, vol. 9, no. 2, pp. 1113–1121, 2013, doi: 10.1109/TII.2012.2223707.

[11] S. Wendel, A. Dietz, and R. Kennel, “FPGA based finite-set model predictive current control for small PMSM drives with efficient resource streaming,” in *2017 IEEE International Symposium on Predictive Control of Electrical Drives and Power Electronics (PRECEDE)*, Sep. 2017, pp. 66–71, doi: 10.1109/PRECEDE.2017.8071270.

[12] I. McInerney, G. A. Constantinides, and E. C. Kerrigan, “A Survey of the Implementation of Linear Model Predictive Control on FPGAs⋆,” *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 381–387, 2018, doi: 10.1016/j.ifacol.2018.11.063.

[13] S. Skogestad, “PID-Tuning Using the SIMC Rules-PPT,” 2017, [Online]. Available: https://intranet.ceautomatica.es/sites/default/files/upload/13/files/XVSimpIC17_SSkogestad1_NTNU.pdf.

[14] H. K. Krishnamurthy *et al.*, “A 500 MHz, 68% efficient, fully on-die digitally controlled buck Voltage Regulator on 22nm Tri-Gate CMOS,” *IEEE Symp. VLSI Circuits, Dig. Tech. Pap.*, 2014, doi: 10.1109/VLSIC.2014.6858438.

[15] H. K. Krishnamurthy *et al.*, “A Digitally Controlled Fully Integrated Voltage Regulator With On-Die Solenoid Inductor With Planar Magnetic Core in 14-nm Tri-Gate CMOS,” *IEEE J. Solid-State Circuits*, vol. 53, no. 1, pp. 8–19, Jan. 2018, doi: 10.1109/JSSC.2017.2759117.

[16] Z. Lukić, N. Rahman, and A. Prodić, “Multibit ΣΔ PWM digital controller IC for DC-DC converters operating at switching frequencies beyond 10 MHz,” *IEEE Trans. Power Electron.*, vol. 22, no. 5, pp. 1693–1707, 2007, doi: 10.1109/TPEL.2007.904199.

[17] Y. Chen, C. Y. Chang, and Y. Yan, “FPGA-based expert PID controller for buck DC-DC converter,” *Appl. Mech. Mater.*, vol. 431, pp. 215–220, 2013, doi: 10.4028/www.scientific.net/AMM.431.215.

[18] M. He and J. Xu, “Nonlinear PID in Digital Controlled Buck Converters,” in *APEC 07 - Twenty-Second Annual IEEE Applied Power Electronics Conference and Exposition*, Feb. 2007, pp. 1461–1465, doi:

10.1109/APEX.2007.357709.

[19] C. Orian and D. Petreus, “the Design and Implementation of a Pid Controller in an Fpga Circuit,” *ACTA Tech. NAPOCENSIS Electron. Telecommun.*, vol. 56, no. 2, pp. 26–31, 2015.

[20] T. Hägglund, “Signal filtering in PID control,” *IFAC Proc. Vol.*, vol. 2, no. PART 1, pp. 1–10, 2012, doi: 10.3182/20120328-3-it-3014.00002.

[21] J. Lee, W. Cho, and T. F. Edgar, “Simple analytic PID controller tuning rules revisited,” *Ind. Eng. Chem. Res.*, vol. 53, no. 13, pp. 5038–5047, 2014, doi: 10.1021/ie4009919.

[22] S. Skogestad, “Simple analytic rules for model reduction and PID controller tuning,” *Model. Identif. Control*, vol. 25, no. 2, pp. 85–120, 2004, doi: 10.4173/mic.2004.2.2.

[23] J. G. Ziegler and N. B. Nichols, “Optimum settings for automatic controllers,” *J. Dyn. Syst. Meas. Control. Trans. ASME*, vol. 115, no. 2B, pp. 220–222, 1993, doi: 10.1115/1.2899060.

[24] D. E. Rivera, M. Morari, and S. Skogestad, “Internal model control: PID controller design,” *Ind. Eng. Chem. Process Des. Dev.*, vol. 25, no. 1, pp. 252–265, Jan. 1986, doi: 10.1021/i200032a041.

[25] C. A. Smith and A. B. Corripio, *Principles and Practice of Automatic Process Control*. New York: John Wiley & Sons, 1985.

[26] E. Grassi and K. Tsakalis, “PID controller tuning by frequency loop-shaping,” *Proc. IEEE Conf. Decis. Control*, vol. 4, no. December, pp. 4776–4781, 1996, doi: 10.1109/cdc.1996.577667.

[27] E. Grassi and K. Tsakalis, “PID controller tuning by frequency loop-shaping: Application to diffusion furnace temperature control,” *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 5, pp. 842–847, 2000, doi: 10.1109/87.865857.

[28] V. Serrano and K. Tsakalis, “A study on the on-line system identification and PID tuning of a buck converter,” *ICNSC 2016 - 13th IEEE Int. Conf. Networking, Sens. Control*, 2016, doi: 10.1109/ICNSC.2016.7479008.

[29] Steve Brunton, “Control Bootcamp: Sensitivity and Complementary Sensitivity - YouTube,” Mar. 08, 2017. https://www.youtube.com/watch?v=hTu36q5yx20.

[30] “Control System Toolbox - MATLAB.” https://nl.mathworks.com/products/control.html.

[31] “Frequency-Response Based Tuning - MATLAB & Simulink - MathWorks Benelux.” https://nl.mathworks.com/help/slcontrol/ug/frequency-response-based-tuning-basics.html.

[32] Y. F. Liu, E. Meyer, and X. Liu, “Recent developments in digital control strategies for DC/DC switching power converters,” *IEEE Trans. Power Electron.*, vol. 24, no. 11, pp. 2567–2577, 2009, doi: 10.1109/TPEL.2009.2030809.

[33] H. Peng, A. Prodić, E. Alarcón, and D. Maksimović, “Modeling of quantization effects in digitally controlled dc-dc converters,” *IEEE Trans. Power Electron.*, vol. 22, no. 1, pp. 208–215, 2007, doi: 10.1109/TPEL.2006.886602.

[34] Z. Lu, Z. Qian, Y. Zeng, W. Yao, G. Chen, and Y. Wang, “Reduction of digital PWM limit ring with novel control algorithm,” *Conf. Proc. - IEEE Appl. Power Electron. Conf. Expo. - APEC*, vol. 1, pp. 521–525, 2001, doi: 10.1109/apec.2001.911696.

[35] N. Karimi and K. Chakrabarty, “Detection, diagnosis, and recovery from Clock-Domain Crossing failures in multiclock SoCs,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 32, no. 9, pp. 1395–1408, 2013, doi: 10.1109/TCAD.2013.2255127.

[36] S. Verma, A. S. Dabare, and A, “EETimes - Understanding Clock Domain Crossing Issues,” 2007. https://www.eetimes.com/understanding-clock-domain-crossing-issues/.

[37] “Simulink - Simulation and Model-Based Design - MATLAB & Simulink.” https://nl.mathworks.com/products/simulink.html.

[38] “Verify HDL Implementation of PID Controller Using FPGA-in-the-Loop - MATLAB & Simulink - MathWorks Benelux.” https://nl.mathworks.com/help/hdlverifier/ug/verify-hdl-implementation-of-pid-controller-using-fpga-in-the-loop.html.

[39] “Simulink Cosimulation - MATLAB & Simulink - MathWorks Benelux.” https://nl.mathworks.com/help/hdlverifier/simulink-cosimulation.html?s_tid=CRUX_lftnav.

[40] “Vivado Design Suite Tutorial Logic Simulation,” 2021, [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documentation/sw_manuals/xilinx2021_2/ug937-vivado-design-suite-simulation-tutorial.pdf.

[41] “Supported EDA Tools and Hardware - MATLAB & Simulink - MathWorks Benelux.” https://nl.mathworks.com/help/hdlverifier/gs/supported-eda-tools.html.

[42] W.-R. Lin, “Converter design update.” 2021.

[43] E. Van Dijk, H. J. N. Spruijt, D. M. O. Sullivan, and J. Ben Klaassens, “PWM-switch modeling of DC-DC converters,” vol. 10, no. 6, pp. 659–665, 1995, doi: 10.1109/63.471285.

[44] “Example of Derivation for a Step-Down Converter,” Dec. 21, 2017. https://techweb.rohm.com/knowledge/dentatsu/s-dentatsu/s-dentatsu04/5762.
[45] B. Hauke, “Basic Calculation of a Buck Converter’s Power Stage,” Dec. 2011, [Online]. Available: https://www.ti.com/lit/an/slva477b/slva477b.pdf.
[46] D. Bishop, “Fixed point package user’s guide,” p. 15, 2008, [Online]. Available: www.vhdl.org/fphdl/vhdl.html.

# Appendix A: MATLAB Function block PID controller

```
function U  = digitalPID(e, a0, a1, a2)

persistent U1; %U(k-1)
if(isempty(U1))
    U1 = 0; %initial or reset value of all registers is zero
end

persistent e1; %e(k-1)
if(isempty(e1))
    e1 = 0;
end

persistent e2; %e(k)
if(isempty(e2))
    e2 = 0;
end

%U(k) = U(k-1) + a0*e(k) + a1*e(k-1) + a2 * e(k-2)
U = U1 + a0*e + a1*e1 + a2*e2;

%Simulate behaviour of registers
e2 = e1;
e1 = e;
U1 = U;

end
```

# Appendix B: MATLAB program calculate FLS targeting integrator TF

```
% Set nondefault solver options
options = optimoptions("fmincon","PlotFcn","optimplotfvalconstr");

%define DC-DC Buck converter
L = 32.8e-06;
C = 0.39e-06;
R = 230.4;
Vin = 100;
converterTF = tf([Vin], [L*C L/R 1]);

x0    = [1;1;1];      %define start point for optimization
lamda = 2.86e03;      %define cut-off frequency
T     = 1e-02;        %define LPF time constant


% Solve minimization problem
[solution,objectiveValue] = fmincon(@objectiveFcn,x0,[],[],[],[],...
    [],[],@constraintFcn,options);

if objectiveValue > 0.5
    disp("FAILED Objective deviation too large")
    return
end
% Clear variables
clearvars options

ks = solution
k1 = ks(1)
k2 = ks(2)
k3 = ks(3)

%Calculate PID parameters
Kp = k2 - k3*T
Ki = 1/k3
kd = k1 - k2*T + k3*T*T


C = tf([k1 k2 k3], [T 1 0]);
G = converterTF;

L = tf([lamda], [1 0]);

OL = C*G;

opt = stepDataOptions('StepAmplitude',48);
step(feedback(OL, 1), feedback(L, 1), opt);
legend("Obtained loop", "target loop")
[stepRespone, t] = step(feedback(OL, 1));
result = stepinfo(stepRespone, t,'SettlingTimeThreshold',0.000208)




function f = objectiveFcn(optimInput)
k1 = optimInput(1);
k2 = optimInput(2);
k3 = optimInput(3);

%Buck converter parameters
L = 32.8e-06;
C = 0.39e-06;
R = 230.4;
Vin = 100;

Tlpf = evalin("base", "T");

%Controlle parameters
```

```
lamda = evalin("base", "lamda");
lamdaa = 0.1*lamda;

L = tf([lamda], [1 0]);


%calculations
W1 = tf([1 0], [1 lamdaa]);
G = evalin("base", "converterTF");
C = tf([k1 k2 k3], [Tlpf 1 0]);
S0 = (1+L)^(-1);

Objective = W1*S0*(G*C-L)

[f,fpeak] = norm(Objective, Inf)
end


function [c,ceq] = constraintFcn(optimInput)

k1 = optimInput(1);
k2 = optimInput(2);
k3 = optimInput(3);
tau = evalin("base", "T");
c(1) = 0 - (k2 - k3*tau);             %Kp > 0
c(2) = 0 - (1/k3);                    %Ki > 0
c(3) = 0 - (k1 - k2*tau + k3*tau*tau); %Kd > 0
ceq = [ ];
end
```

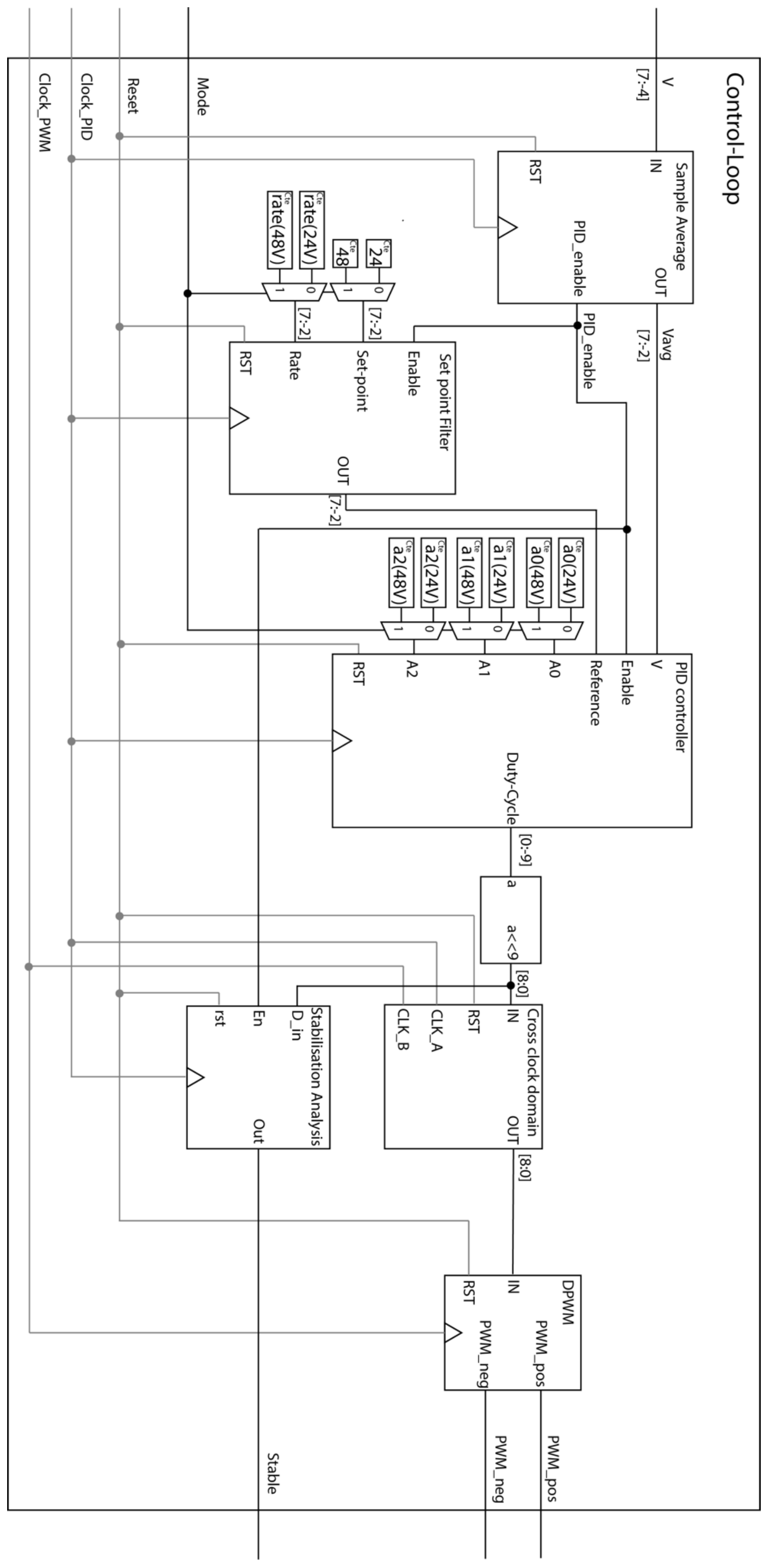
# Appendix C: MATLAB function block nonlinear PID

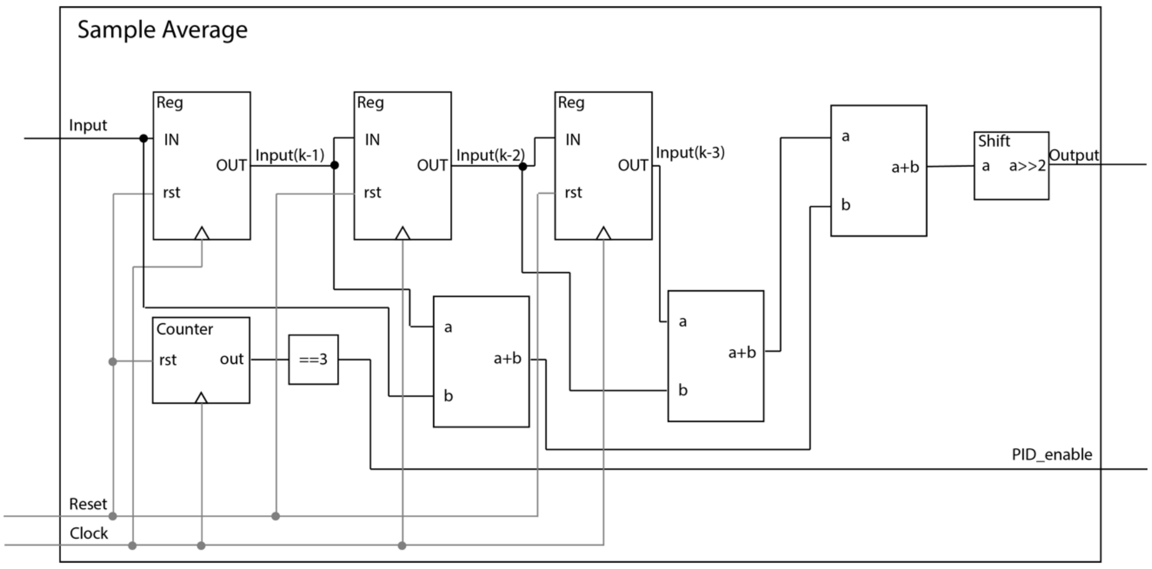
```
function [a0,a1,a2]= parameterCalculator(e)

%%%%%Define serial PID base tuning%%%%%%%%%%%
BKp = 0.0287727020453255;
BKi = 1434.95344169792/BKp;
BKd = 1.44232620887247e-07/BKp;
%%%%%end Define serial PID base tuning%%%%%%%%%%%

%%%%%define error bands%%%%%
if(abs(e)>20)
    errorband = 4
elseif(abs(e)>10)
    errorband = 3
elseif(abs(e)>5)
    errorband = 2
else
    errorband = 1
end

%%%%%%Calculate PID paramaters based on error size%%%%%%%%%
Kp = (errorband^2)*BKp;
Ki = BKi;
Kd = BKd;
%%%%%%
Td = Kd;
Ti = 1/(Ki);
Ts = 1e-06;

a0 = Kp*(1+Td/Ts);
a1 = -Kp*(1-(Ts/Ti)+2*(Td/Ts));
a2 = Kp*(Td / Ts);
```

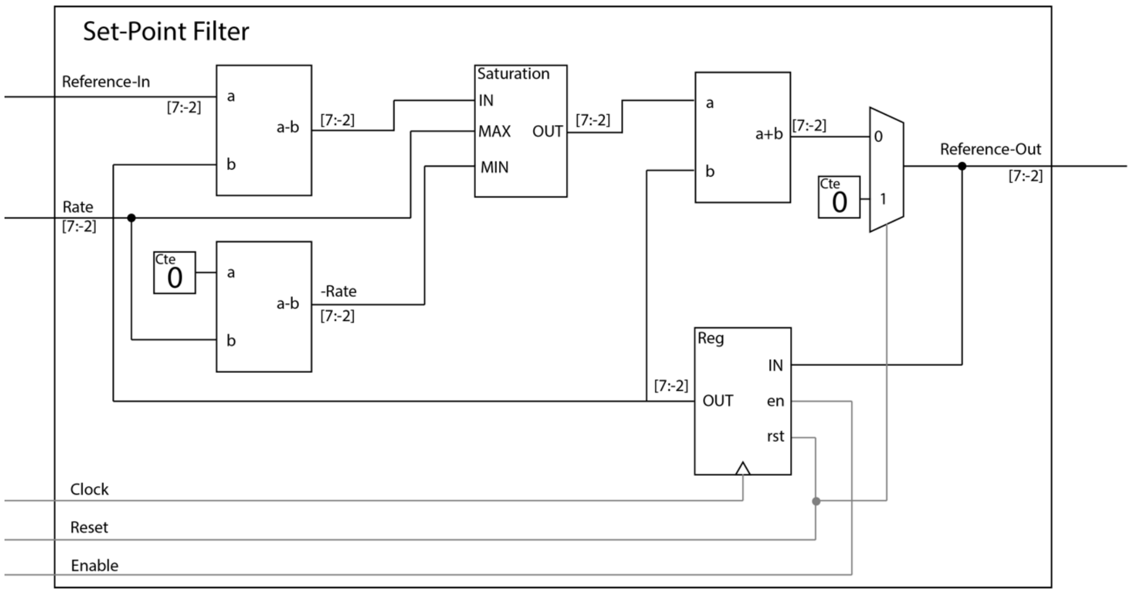
# Appendix D: PID tuning parameters

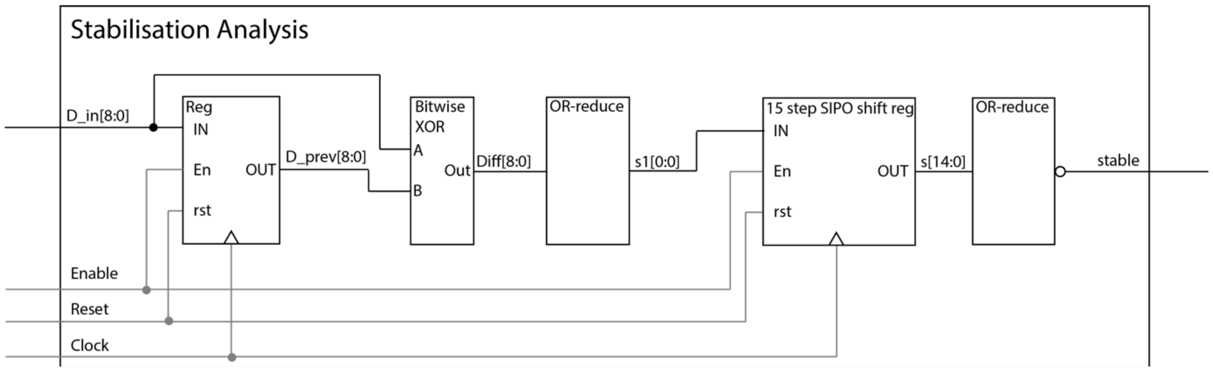
| Parallel PID parameters | 48V no overshoot | 48V overshoot | 48V overshoot - scaled (444/512) | | 24V no overshoot | 24V overshoot | 24V overshoot - scaled 444/512 |
|---|---|---|---|---|---|---|---|
| P | 0,025767138 | 0,038132836 | **0,033068318** | | 0,007039883 | 0,025329578 | **0,021965493** |
| I | 1178,713203 | 2071,43 | **1796,32** | | 580 | 1.272 | **1.103** |
| D | 1,41E-07 | 1,75E-07 | **1,52E-07** | | 2,14E-08 | 1,26E-07 | **1,09E-07** |
| Ts | 0,000001 | 0,000001 | **0,000001** | | 0,000001 | 1,00E-06 | **1,00E-06** |
| | | | | | | | |
| Serial PID paramers | | | | | | | |
| Kp | 0,025767138 | 0,038132836 | **0,033068318** | | 0,007039883 | 0,025329578 | **0,021965493** |
| Ki-s | 45744,82476 | 54321,44635 | **54321,44635** | | 82417,16503 | 50204,13408 | **50204,13408** |
| Kd-s | 5,47E-06 | 4,60E-06 | **4,60E-06** | | 3,03E-06 | 4,98E-06 | **4,98E-06** |
| | | | | | | | |
| Serial PID time constants | | | | | | | |
| Ti | 2,18604E-05 | 1,84089E-05 | **1,84089E-05** | | 1,21334E-05 | 1,99187E-05 | **1,99187E-05** |
| Td | 5,47E-06 | 4,60E-06 | **4,60E-06** | | 3,03E-06 | 4,98E-06 | **4,98E-06** |
| | | | | | | | |
| | | | | | | | |
| Circuit parameters | | | | | | | |
| A0 | 0,16658709899143 | 0,21362906242601 | **0,18525645257256** | | 0,02839430362539 | 0,15146250679374 | **0,13134639261020** |
| A1 | -0,30622834698965 | -0,38705385857365 | **-0,33564826798183** | | -0,04916851686064 | -0,27632378616668 | **-0,23962453331642** |
| A2 | 0,14081996120098 | 0,17549622692537 | **0,15218813428684** | | 0,02135442044877 | 0,12613292889776 | **0,10938089927853** |

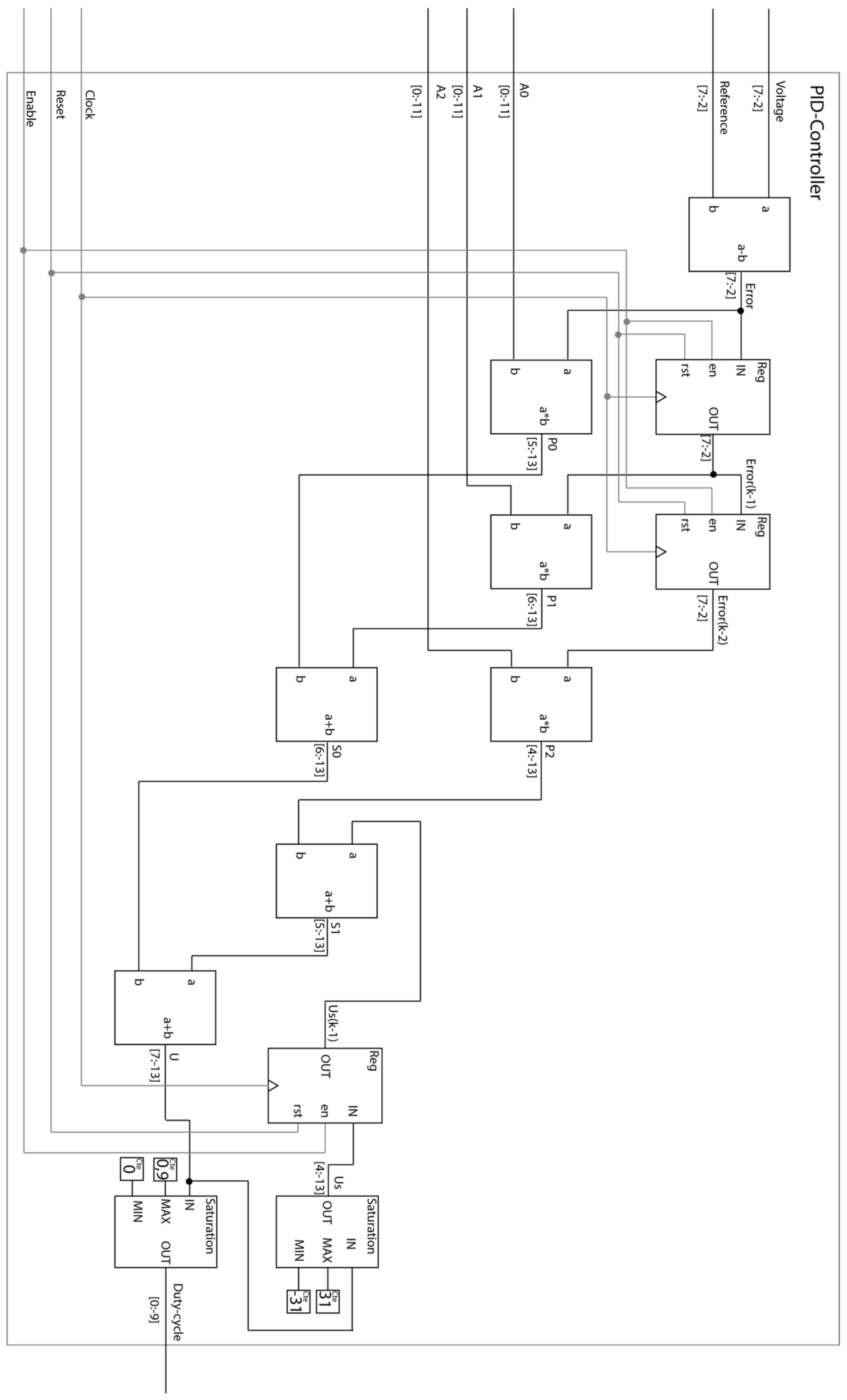
# Appendix E: Functional diagrams

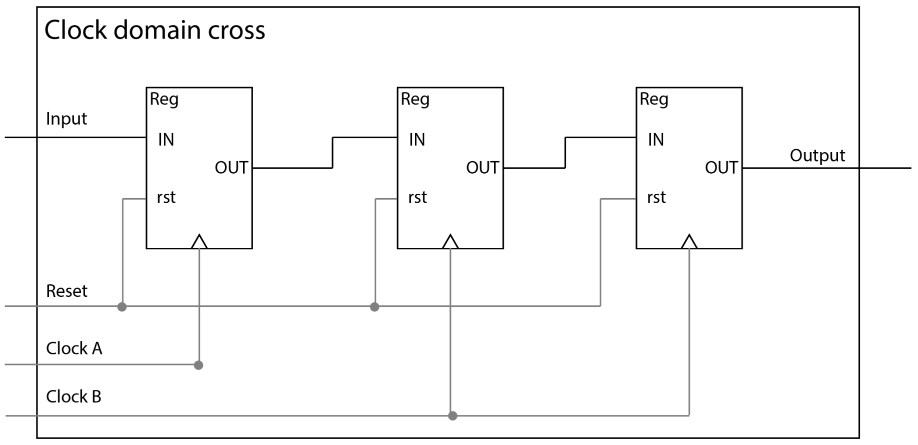


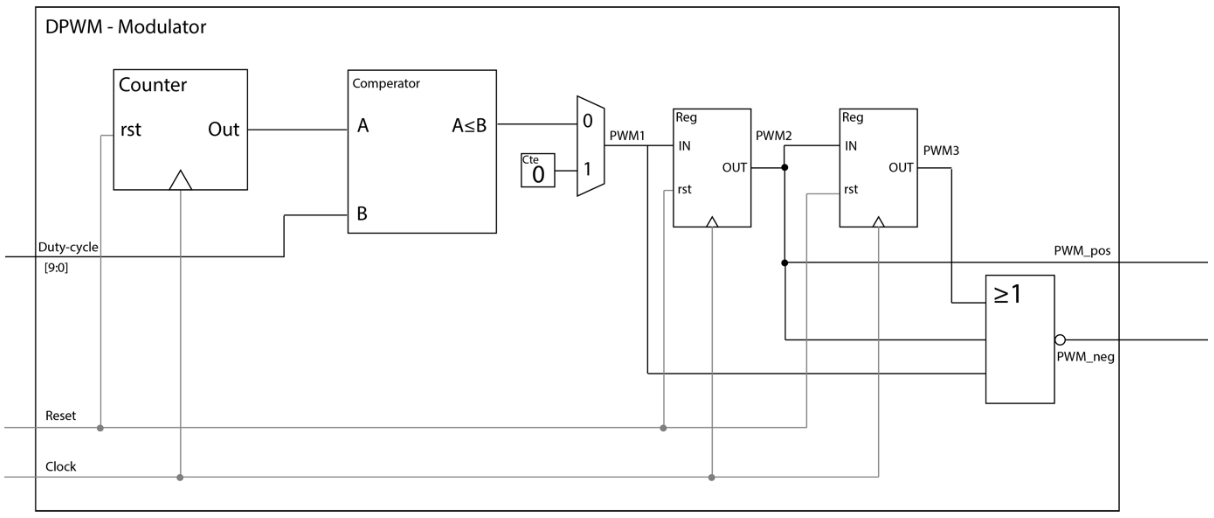

Sample Average
Input
Reg
IN
OUT
rst
Input(k-1)
Input(k-2)
Input(k-3)
Counter
rst
out
==3
a
b
a+b
Shift
a a>>2
Output
PID_enable
Reset
Clock

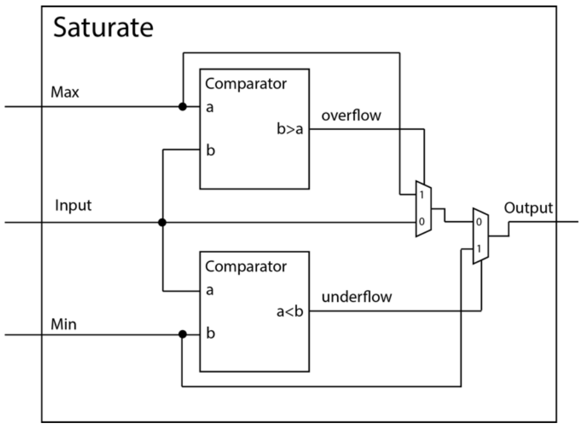

Set-Point Filter
Reference-In
[7:-2]
a
b
a-b
Saturation
IN
MAX
MIN
OUT
a+b
Reference-Out
Rate
Cte
0
-Rate
Reg
IN
OUT
en
rst
1
Clock
Reset
Enable


Stabilisation Analysis
D_in[8:0]
Reg
IN
En
rst
OUT
D_prev[8:0]
Bitwise XOR
A
B
Out
Diff[8:0]
OR-reduce
s1[0:0]
15 step SIPO shift reg
IN
En
rst
OUT
s[14:0]
OR-reduce
stable
Enable
Reset
Clock


PID-Controller
Voltage
[7:-2]
Reference
[7:-2]
A0
[0:-11]
A1
[0:-11]
A2
[0:-11]
Clock
Reset
Enable
a
b
a-b
Error
[7:-2]
Reg
IN
en
rst
OUT
[7:-2]
a*b
P0
[5:-13]
Error(k-1)
Reg
IN
en
rst
OUT
P1
[6:-13]
Error(k-2)
[7:-2]
a+b
S0
[6:-13]
a*b
P2
[4:-13]
a+b
S1
[5:-13]
a+b
U
[7:-13]
Us(k-1)
Reg
OUT
IN
en
rst
Us
[4:-13]
Saturation
IN
MAX
MIN
OUT
Cte
31
Cte
-31
Cte
0,9
Cte
0
Saturation
Duty-cycle
[0:-9]


Clock domain cross
Reg
IN
OUT
rst
Input
Output
Reset
Clock A
Clock B


DPWM - Modulator
Counter
rst
Out
Comperator
A
A≤B
B
0
1
Cte
PWM1
PWM2
PWM3
Reg
IN
OUT
rst
Duty-cycle
[9:0]
PWM_pos
≥1
PWM_neg
Reset
Clock


Saturate
Max
Comparator
a
b
b>a
overflow
Input
Output
Comparator
a
b
a<b
underflow
Min