Modelling and design of digital DC-DC converters

Master thesis performed in datorteknik

by

Hiwa Mobaraz

LiTH-ISY-EX--16/4942--SE
Linköping 2016
Modelling and design of digital DC-DC converters

Master thesis in datorteknik
at Linköping institute of technology
by
Hiwa Mobarak
LiTH-ISY-EX--16/4942--SE
Linköping 2016

Supervisor:
David Karlsson
Syntronic AB
Gävle
davk@syntronic.com

Examiner
Kent Palmkvist
Linköpings Universitet
Linköping
kentp@isy.liu.se
**Summary**

Digital Switched mode power supplies are nowadays popular enough to be the obvious choice in many applications. Among all set-up and control techniques, the current mode DC-DC converter is often considered when performance and stability are of interest. This has also motivated all the "on chip" and ASIC implementations seen on the market, where current mode control technique is used. However, the development of FPGAs has created an important alternative to ASICs and DSPs. The flexibility and integration possibility is two important advantages among others. In this thesis report, an FPGA-based current mode buck/boost DC-DC converter is built in a stepwise manner, starting from the mathematical model. The goal is a simulation model which creates a basis for discussion about the advantages and disadvantages of current mode DC-DC converters, implemented in FPGAs.

**Keywords**

DC-DC, Buck, Boost, Digital, Converter, Design, Comparision.
ABSTRACT

Digital Switched mode power supplies are nowadays popular enough to be the obvious choice in many applications. Among all set-up and control techniques, the current mode DC-DC converter is often considered when performance and stability are of interest. This has also motivated all the “on chip” and ASIC implementations seen on the market, where current mode control technique is used. However, the development of FPGAs has created an important alternative to ASICs and DSPs. The flexibility and integration possibility is two important advantages among others. In this thesis report, an FPGA-based current mode buck/boost DC-DC converter is built in a stepwise manner, starting from the mathematical model. The goal is a simulation model which creates a basis for discussion about the advantages and disadvantages of current mode DC-DC converters, implemented in FPGAs.
# TABLE OF CONTENTS

1 Introduction......................................................................................................................1  
  1.1 Motivation....................................................................................................................1  
  1.2 Purpose........................................................................................................................2  
  1.3 Problem statements......................................................................................................2  
  1.4 Tools............................................................................................................................3  
  1.5 Limitations...................................................................................................................3  

2 Background..........................................................................................................................5  
  2.1 Introduction..................................................................................................................5  
  2.2 Requirements................................................................................................................6  

3 Theory................................................................................................................................8  
  3.1 Introduction..................................................................................................................8  
  3.2 Basic theory..................................................................................................................8  
  3.3 Theory of operation......................................................................................................11  
  3.4 Transfer function of Buck converter............................................................................13  
  3.5 Transfer function of Boost converter..........................................................................16  
  3.6 Digital converter.........................................................................................................19  
  3.7 Subharmonic oscillation...............................................................................................22  
  3.8 Slope compensation.....................................................................................................24  

4 Method................................................................................................................................27  
  4.1 Introduction..................................................................................................................27  
  4.2 Used Design method.....................................................................................................28  
  4.3 Used measurement method.........................................................................................28  
  4.4 Implementation.............................................................................................................30  

5 Results................................................................................................................................33  
  5.1 Introduction..................................................................................................................33  
  5.2 Design and Measurement results.................................................................................33  
  5.3 Used Subsystems.........................................................................................................56  

6 Discussion............................................................................................................................61  
  6.1 Results..........................................................................................................................61  
  6.2 Bits needed....................................................................................................................63  
  6.3 Method..........................................................................................................................63  

7 Conclusions..........................................................................................................................67  
  7.1 Performance and complexity.......................................................................................67  
  7.2 Future work...................................................................................................................68  

Bibliography............................................................................................................................70  

Appendix................................................................................................................................73
LIST OF FIGURES

Figure 2.2.1: Buck and boost mode set-ups ................................................................. 6
Figure 3.2.1: Buck converter during ON and OFF states of the switch ....................... 9
Figure 3.2.2: Boost converter during ON and OFF states of the switch ...................... 10
Figure 3.3.1: Operation of Current mode Buck converter ........................................... 11
Figure 3.3.2: Buck converter current rising/falling through the inductor ...................... 11
Figure 3.4.1: System set-up for current mode control .................................................. 14
Figure 3.5.1: System set-up for current mode control .................................................. 17
Figure 3.6.1: Digital implementation of the controller .................................................. 21
Figure 3.7.1: Subharmonic oscillation ...................................................................... 22
Figure 3.8.1: Compensated system ...................................................................... 24
Figure 3.8.2: Slope compensation ....................................................................... 25
Figure 4.2.1: Method and work flow ..................................................................... 28
Figure 4.3.1: Combined step response .................................................................... 29
Figure 5.2.1: Frequency response of Buck converter model 1 ..................................... 35
Figure 5.2.2: Model of the Plant, using Simulink components ..................................... 36
Figure 5.2.3: Model of the Controller using Simulink components ............................. 37
Figure 5.2.4: Model of the Error Amp using Simulink components (Transfer Fcn block) .............................................................................................................. 37
Figure 5.2.5: Combined step response for Buck converter model 2 ............................. 38
Figure 5.2.6: Model of the Plant, using Simulink components ..................................... 39
Figure 5.2.7: Model of the Controller using Simulink components ............................. 40
Figure 5.2.8: Model of the Error Amp using Simulink components ............................. 40
Figure 5.2.9: Combined step response for Buck converter model 3 ............................. 41
Figure 5.2.10: Model of the Plant, using Simulink components ................................... 42
Figure 5.2.11: Model of the Controller using Altera DSP builder components (VHDL) .............................................................................................................. 43
Figure 5.2.12: Combined step response for Buck converter model 4 ............................. 44
Figure 5.2.13: Frequency response of Boost converter model 1 ................................... 46
Figure 5.2.14: Model of the Plant, using Simulink components ................................... 47
Figure 5.2.15: Model of the Controller using Simulink components ........................... 48
Figure 5.2.16: Model of the Error Amp using Simulink components (Transfer Fcn block). .............................................................................................................. 48
Figure 5.2.17: Combined step response for Boost converter model 2 ........................... 49
Figure 5.2.18: Model of the Plant, using Simulink components ................................... 50
Figure 5.2.19: Model of the Controller using Simulink components ............................ 51
Figure 5.2.20: Model of the Error Amp using Simulink components ............................ 51
Figure 5.2.21: Combined step response for Boost converter model 3 ........................... 52
Figure 5.2.22: Model of the Plant, using Simulink components ................................... 53
Figure 5.2.23: Model of the Controller using Altera DSP builder components (VHDL) .............................................................................................................. 54
Figure 5.2.24: Combined step response for Boost converter model 4 ............................ 55
Figure 5.3.1: Components inside the Switch blocks .................................................... 56
Figure 5.3.2: Components inside the Single Pulse blocks ........................................... 56
Figure 5.3.3: Components inside the ADC blocks ...................................................... 57
Figure 5.3.4: Components inside the DAC blocks ...................................................... 58
Figure 5.3.5: Components inside the Voltage Divider blocks ...................................... 59
Figure 5.3.6: Components inside the BUCK/BOOST SWITCH block .......................... 59
Figure 6.3.1: Example design of simple circuit ............................................................ 64
Figure 7.1.1: Digital Slope compensation .................................................................... 67
# LIST OF TABLES

Table 1: List of acronyms........................................................................................................viii
Table 4.1: Voltages and currents used during the measurements........................................29
Table 5.1: List of coefficients for Buck converter model 1..................................................34
Table 5.2: Coefficients used inside the Plant in Buck converter model 2..........................36
Table 5.3: Coefficients used inside the Controller in Buck converter model 2..................37
Table 5.4: Coefficients used inside the Plant in Buck converter model 3..........................39
Table 5.5: Coefficients used inside the Controller in Buck converter model 3..................40
Table 5.6: Coefficients used inside the Controller and Plant in Buck converter model 4....43
Table 5.7: List of coefficients for Boost converter model 1..............................................45
Table 5.8: Coefficients used inside the Plant in Boost converter model 2.........................47
Table 5.9: Coefficients used inside the Controller in Boost converter model 2...............48
Table 5.10: Coefficients used inside the Plant in Boost converter model 3......................50
Table 5.11: Coefficients used inside the Controller in Boost converter model 3..............51
Table 5.12: Coefficients used inside the Controller and Plant in Boost converter model 4.54
## LIST OF ACRONYMS

<table>
<thead>
<tr>
<th>Abbreviation/ Acronym</th>
<th>Meaning</th>
<th>Explanation</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISY</td>
<td>A department in the University of Linköping</td>
<td>One of the departments of the Linköping institute of technology. Department of electrical engineering in English.</td>
<td>Mentioned in the front pages</td>
</tr>
<tr>
<td>SMPS</td>
<td>Switched mode power supply</td>
<td>Abbreviation of Switched mode power supply</td>
<td>Used throughout the work.</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse width modulation</td>
<td>A technique to modulate the width of a pulse.</td>
<td>Used throughout the work.</td>
</tr>
<tr>
<td>Plant</td>
<td>Part of the whole buck/boost system</td>
<td>Part of the Buck/boost system that consists of resistor, inductor, capacitor and load.</td>
<td>Used throughout the work.</td>
</tr>
<tr>
<td>Controller</td>
<td>Part of the whole buck/boost system</td>
<td>Part of the Buck/boost system that controls the Plant</td>
<td>Used throughout the work.</td>
</tr>
<tr>
<td>Duty cycle</td>
<td>The percentage of a digital signal that is a logical one.</td>
<td>The percentage of a digital signal that is a logical one, during one whole signal cycle.</td>
<td>Used throughout the work.</td>
</tr>
<tr>
<td>Buck Converter</td>
<td>Step down converter</td>
<td>Voltage converter with higher input voltage than output voltage</td>
<td>Used throughout the work.</td>
</tr>
<tr>
<td>Boost Converter</td>
<td>Step up converter</td>
<td>Voltage converter with higher output voltage than input voltage</td>
<td>Used throughout the work.</td>
</tr>
<tr>
<td>ESR</td>
<td>Equivalent series resistance</td>
<td>The Equivalent series resistance of an analog component</td>
<td>Used throughout the work.</td>
</tr>
</tbody>
</table>

Table 1: List of acronyms
1 INTRODUCTION

Switched mode power supplies offer several advantages over the alternatives. One of those is the efficiency factor where efficiencies up to 92% have been reported [1]. They can be implemented either with analog components or with a digital chip such as a microcontroller, DSP or FPGA. One advantage of digital controllers is that analog components are replaced by coefficients in a digital calculation. This gives room for flexibility since coefficients can be changed whenever needed, without any change in hardware. There is also dedicated DSP-chips on the market with special compilers, only intended for DC-DC converters. Although they provide simplicity and abstraction, FPGAs has become more and more popular in this field. This increase of popularity has many reasons and one of them is “fast prototyping” tools, as seen here [2]. In this thesis report, an FPGA based current mode buck/boost DC-DC converter is built in a stepwise manner, starting from the mathematical model. The goal is a simulation model which creates a basis for discussion about the advantages and disadvantages of current mode DC-DC converters, implemented in FPGAs.

1.1 Motivation

The popularity of FPGA-based DC-DC converters can be explained by many reasons as discussed above. Especially when it comes to voltage-mode DC-DC converters, since an ADC is the only link between the analog and digital parts [3]. However, this is not the case with current mode control where the current has to be sensed as well, which is described in the theory chapter. By increasing the signals that have to be sensed, the components needed will usually increase as well. Increasing the components in an implementation can have a bad impact on both the price and the performance. This is not only because it takes the implementation further away from the “fast prototyping approach”, but also because of making the system more sensitive to noise. Subharmonic oscillation is another drawback that current-mode control suffers from, as described in the theory chapter. All this makes FPGA-based current mode control not an obvious choice, compared to the alternatives. This is also the motivation behind this work.
1.2 Purpose

The purpose of this thesis is to:

- Build a Buck/Boost converter system beginning from the mathematical models.
- Discuss the complexity, costs and performance.
- Evaluate the results

1.3 Problem statements

Since a digital controller is going to be built, there will be several multiplications and additions inside the controller. Most probably, the digital controller itself will fit inside an FPGA without any bigger problems. But the choice of an FPGA partly comes from the idea of integrating the controller, with another (and usually bigger) system. It is of interest to leave room for other systems inside the FPGA, as much as possible. Therefore, it is interesting to get a lower bound on the amount of bits needed for each operation. The speed of calculation is also of interest. For an FPGA, speed is not a big problem since calculations are done in parallel. But in this case, the speed is more related to the DAC and ADC. This is not only because the data transfer is usually done in a serial manner, but also because of minimum settle times of the voltages. The following problem statements below is then to be answered during this work, creating a basis for discussion.

- How many bits are needed at the input of the multiplication and arithmetic blocks, to maintain functionality?
- What is the minimum clock frequency needed, compared to a given sample rate?
1.4 Tools

Only software tools are used in this work. This is because simulation models and simulation results are the only results that will be presented in the result chapter. The tools used in this work are:

- **Matlab**: For mathematical calculations and function/script generation.
- **Simulink**: For simulations of both the ideal and less ideal models.
- Inside Simulink, two important packages are used:
  1. **Altera DSP builder**: Used to generate the digital blocks inside the controller.
  2. **Simscape**: Used to simulate/model the analog components in the plant.

1.5 Limitations

The thesis will start at a theoretical level and a system will be built, starting from mathematical functions and end up with simulation models. Since this work includes an implementation, it is always of interest to compare theory with reality. A big limitation is therefore the lack of hardware, that would give feedback and validity to the results obtained in the simulations. A solution to this is to build up simulation models where the component behaviors are as realistic as possible. Parasitics, delays and voltage drops are introduced to the analog components in the plant and calculation delays are introduced to the digital components inside the controller. Another limitation that has an important impact on both the work and results are time. The tools used as described above has a student licensing, which will run out after 2 months.
2 BACKGROUND

2.1 Introduction

In this chapter, a description will be given about the given system requirements. The background of this thesis is a growing interest of DC-DC converters implemented in FPGAs at Syntronic (company name). This thesis is thus part of a project with the goal to investigate whether the advantages of current-mode DC-DC converters can outscore its price and complexity, when implemented into an FPGA. One can wonder why FPGAs has to be involved since there is a market providing both analog and digital ASICS with proven functionality and performance. The answer is all those advantages that usually come with FPGAs, no matter where they are implemented:

- **Flexibility**: Easy to change the design in the same chip, without changing the hardware. This allows "step by step" designs where the performance can be improved with each iteration.

- **Integration**: Possibility to integrate the DC-DC converters into bigger systems such as voltage control over the internet.

- **Parallelism**: powerful parallelism easily achieved.

- **Powerful tools**: The development of software tools and synthesizers has become far more helpful today. “Fast prototyping” tools and IP-blocks shortens time to market.

Another advantage is the amount of I/O pins, usually available on FPGAs. This creates a possibility to control multiple switches, with full synchronization. A good example is a multiphase DC-DC converter, simply achieved by extending a single phase system [4].
2.2 Requirements

Since there only will be simulations performed in this thesis, the interest of a hardware implementation still remains. The idea is thus to create the opportunity for other students to implement the system. Students usually belong to a less experienced group. There are thus some requirements that need to be taken into account.

1. **Safe voltage ranges:** The input voltages should be the same for both buck and boost mode. The maximum output voltage, however, can not be the same. The goal is to have an output voltage that corresponds to a duty-cycle of more than 50 %. The reason for this can be found in the theory chapter (Slope compensation).
   - \( V_{in} = 5 \, V \) for both buck and boost converter.
   - \( V_{\text{out max}} = 5 \, V \) for buck converter.
   - \( V_{\text{out max}} = 12 \, V \) for Boost converter.

2. **General solution:** Both Buck and Boost converter is desired to be designed. This shall not be confused with a standard buck/boost converter system.

3. **Maximum of two switches:** A maximum of two switches (N-MOS transistors) shall be used. This is not only because of reducing nonlinear components and price, but also to reduce complexity and resistance.

4. **Same plant:** With the same analog circuit (plant), the system shall be capable of acting both as a buck and a boost converter. Depending on to which end the source and the load are connected, the system should change its mode. This is illustrated in the figure below. The theory behind this can be found in the theory chapter.

5. **Switch Frequency:** A switching frequency of 200 kHz shall be used.

![Figure 2.2.1: Buck and boost mode set-ups](image)
3 THEORY

3.1 Introduction

In this chapter, the theory behind both buck and boost converters will be given. At first, basic and general theory with simple set-ups will be presented to introduce the reader. Later on, a more detailed explanation will be given, with the goal to specifically describe current-mode DC-DC converters. When current mode control is understood, a deeper mathematical model and analysis is given. Another interesting phenomenon is subharmonic oscillation. The problem and solution for this phenomenon are found in this chapter as well. Current mode control suffers from Subharmonic oscillation and a standard solution is called “slope compensation”. Even if slope compensation is not implemented in any of the models in this work, the theory behind it is still of interest. This is because it will be an important part of the discussion chapter.

3.2 Basic theory

The absolute easiest way of describing a Buck converter is done with the concept of pulse width modulation (PWM). By switching a transistor on and off, the level of the output voltage will directly be related to the duty cycle. However, switching a transistor will generate an output voltage that contains high-frequency components. This is due to the relatively sharp edges, when toggling the switch. A solution to this is thus a low-pass filter, connected to the switch. In the following pages, the relations for both buck and boost converters will be derived.
Buck converter relations

To begin simple, steady state mode is assumed during this derivation. Steady state means that the current in the inductor never fall to zero. The picture below shows a buck converter set-up where the switch is either closed or open. When the switch is closed, the voltage across the inductor is given by equation (1). When the switch is open, the voltage across the inductor is given by equation (2). (Note: The voltage drop across the diode is ignored for simplicity).

\[ V_L = V_i - V_o \]  \hspace{1cm} (1)

\[ V_L = -V_o \]  \hspace{1cm} (2)

The voltage across an inductor is given as

\[ V_L = L \frac{dI_L}{dt} \]  \hspace{1cm} (3)

This gives the charging current:

\[ \nabla I_{L(ON)} = \int_0^D \frac{V_L}{L} dt = \int_0^D \frac{V_i - V_o}{L} dt = \frac{V_i - V_o}{L} \cdot D \cdot T \]  \hspace{1cm} (4)

And the discharge current:

\[ \nabla I_{L(OFF)} = \int_D^T \frac{V_L}{L} dt = -\frac{V_o}{L} \cdot (1-D) \cdot T \]  \hspace{1cm} (5)

Since steady state condition is assumed, the following relations can be used:

\[ \nabla I_{L(ON)} + \nabla I_{L(OFF)} = 0 \iff D \cdot T \cdot \frac{V_i - V_o}{L} - \frac{V_o}{L} \cdot (1-D) \cdot T = 0 \Rightarrow D \cdot V_i = V_o \]  \hspace{1cm} (6)

This gives the well known relation

\[ D = \frac{V_o}{V_i} \]  \hspace{1cm} (7)

Figure 3.2.1: Buck converter during ON and OFF states of the switch
Boost converter relations

Like the basics for the buck converter, **steady state mode** is assumed here as well. The picture below shows a boost converter set-up where the switch is either closed or open. When the switch is closed, the voltage across the inductor is given by equation (8). When the switch is open, the voltage across the inductor is given by equation (9). (Note: The voltage drop across the diode is ignored for simplicity).

\[ V_L = V_i \] \hspace{1cm} (8)

\[ V_L = V_i - V_o \] \hspace{1cm} (9)

The voltage across an inductor is given as

\[ V_L = L \frac{dI_L}{dt} \] \hspace{1cm} (10)

This gives the charging current:

\[ \nabla I_{L(ON)} = \int_0^{\frac{D \cdot T}{L}} V_i dt = \int_0^{\frac{D \cdot T}{L}} \frac{V_i}{L} dt = \frac{V_i}{L} D \cdot T \] \hspace{1cm} (11)

And the discharge current:

\[ \nabla I_{L(OFF)} = \int_{\frac{D \cdot T}{L}}^{\frac{T}{L}} V_i dt = \frac{V_i}{L} \left(1 - D\right) \cdot T \] \hspace{1cm} (12)

Since steady state condition is assumed, the following relations can be used:

\[ \nabla I_{L(ON)} + \nabla I_{L(OFF)} = 0 \quad \Leftrightarrow \quad D \cdot T \cdot \frac{V_i}{L} + \frac{V_i - V_o}{L} \left(1 - D\right) \cdot T = 0 \quad \Rightarrow \quad \frac{1}{1 - D} = \frac{V_o}{V_i} \] \hspace{1cm} (13)

This gives the duty cycle

\[ D = 1 - \frac{V_i}{V_o} \] \hspace{1cm} (14)

![Figure 3.2.2: Boost converter during ON and OFF states of the switch](image)
3.3 Theory of operation

The pictures below show the relation between the components and the rising/falling currents. An explanation of each step is given on the next page. The example illustrates a buck converter but by replacing the diode and N-MOS transistor with each other, a boost converter can be explained with the same principles.

Figure 3.3.1: Operation of Current mode Buck converter

Figure 3.3.2: Buck converter current rising/falling through the inductor
The pictures on the previous page show an overview of each step in an operation cycle. For simplicity, one can assume the voltage $V_c$ (control voltage) to be constant.

1. Clock trigs the SR-latch. This turns the N-MOS transistor on. When the Transistor is on, the inductor current starts to rise.

2. The voltage over $R_i$ increases, which is amplified by the OP-Amp. The output of this amplifier has now a voltage that is greater than $V_c$. This trigs the comparator which resets the SR-latch. The inductor current starts to discharge.

3. The clock trigs the SR-latch again and a new cycle is repeated.

**Error amplifier**

In a real application, the load resistance can vary. Changes in load resistance will also affect the voltage over it. As seen in figure 3.3.1, any change in the output voltage compared to a given reference voltage will affect the error amplifier. The error amplifier will in its turn change the control voltage $V_c$ such that the desired output voltage (= reference voltage) is maintained.

**Notes:**

- The graph represents currents but is represented as voltages in hardware.
- The goal is to keep an average current ($I_a$) such that $V_{out} = I_a \cdot R_L$.
- The resistor $R_i$ needs to have a low resistance. An amplifier is needed to sense the small voltage.
- The clock is a sample clock which determines the sample rates and switch frequency.
3.4 Transfer function of Buck converter

Plant

According to Freescale Semiconductor [5], the control to output function (function for the plant) is a combination of 3 terms:

\[
H(s)_{\text{plant}} = \frac{V_{\text{out}}(s)}{V_{\text{err}}(s)} = F_H(s) \cdot H_B(s) \cdot H_{\text{DC}}(s) \tag{15}
\]

\[
F_H = \frac{1}{1 + \frac{s}{\omega_n} + \frac{s^2}{\omega_n^2}} \quad H_B = \frac{1 + \frac{s}{\omega_{\text{er}}}}{1 + \frac{s}{\omega_{\text{op}}}} \quad H_{\text{DC}} = \frac{R_L}{R_i} \times \frac{1}{\left[1 + \frac{R_i T}{L_o} \frac{1}{\pi}\right]}
\]

Where \( F_H \) is the high-frequency term, \( H_B(s) \) is the small signal model of the power stage and \( H_{\text{DC}}(s) \) is the DC gain. So totally, the relation below can be used:

\[
H(s)_{\text{plant}} = \frac{R_L}{R_i} \times \frac{1}{\left[1 + \frac{R_L T}{L} \frac{1}{\pi}\right]} \times \frac{1 + \frac{s}{\omega_{\text{er}}}}{1 + \frac{s}{\omega_{\text{op}}}} \times \frac{1}{1 + \frac{s}{\omega_n} + \frac{s^2}{\omega_n^2}} \tag{16}
\]

Where

\[
\omega_n = \frac{\pi}{T} \quad \omega_{\text{er}} = \frac{1}{R_c C} \quad \omega_{\text{op}} = \frac{1}{R_c C} + \frac{T}{L C \pi}
\]

Controller

A compensator is needed to control the voltage. This is the model of the controller [6]. The values of the coefficients are given on the next page.

\[
H(s)_{\text{Controller}} = \frac{\omega_{\text{c}}^2}{s} \times \frac{1 + \frac{\omega_{\text{c}}}{\omega_{\text{er}}} s}{1 + \frac{\omega_{\text{c}}}{\omega_{\text{op}}} s} \tag{17}
\]
Whole System

The system transfer function is thus

\[ H(s)_\text{system} = \frac{H_{\text{plant}} \times H_{\text{controller}}}{1 + H_{\text{plant}} \times H_{\text{controller}}} \]  \hspace{1cm} (18)

To obtain a stable system, \( H_{\text{controller}} \) needs to be based on \( H_{\text{plant}} \). According to [7], this is done by

1. \( \omega_{cp1} \) is chosen to cancel out the ESR zero in \( H_{\text{plant}} \). \( \omega_{cp1} = \omega_{esr} = \frac{1}{R_c C} \)

2. Choose a certain crossover frequency \( f_x \) such that \( f_x \frac{f_s}{10 + \alpha} \). \( f_s \) is the sample rate.

3. The zero \( \omega_{cz1} \) is chosen to 20% of the crossover frequency \( f_x \). \( \omega_{cz1} = \frac{f_x}{5 \pi} \)

\[
1.23 \cdot f_x \cdot R_i \cdot (L + 0.32 \cdot R_L) \cdot \sqrt{(1 - 4 \cdot f_x^2 \cdot T^2 + 16 \cdot f_x^4 \cdot T^4) \cdot \left[1 + \frac{39.48 \cdot C_0^2 \cdot f_x^2 \cdot L^2 \cdot R_L^2}{(L + 0.32 \cdot R_L \cdot T)^2}\right]}
\]

4. \( \omega_{cp0} = \frac{1.23 \cdot f_x \cdot R_i \cdot (L + 0.32 \cdot R_L) \cdot \sqrt{(1 - 4 \cdot f_x^2 \cdot T^2 + 16 \cdot f_x^4 \cdot T^4) \cdot \left[1 + \frac{39.48 \cdot C_0^2 \cdot f_x^2 \cdot L^2 \cdot R_L^2}{(L + 0.32 \cdot R_L \cdot T)^2}\right]}}{L \cdot R_L} \)

Notes:
- \( R_L \) is the load resistance.
- \( L \) is the Inductance of the coil.
- \( T = 1/f_s \) is the switching period.
- \( R_i \) is the current sense gain (see Figure 3.3.1)
- \( R_c \) is the ESR (equivalent series resistance) of the capacitor.
- \( \alpha \) is a margin constant used for conservative design.

Figure 3.4.1: System set-up for current mode control
Derivation of $\omega_{cp0}$

To achieve the desired crossover frequency of $f_x$, the gain of the controller $\omega_{cp0}$ needs to be chosen based on $f_x$. The crossover frequency occurs when

$$|H(i \cdot f_x)_{plant} \cdot H(i \cdot f_x)_{controller}| = 1 \iff |H_{DC}(i \cdot f_x) \cdot H(i \cdot f_x)_{controller}| = 1$$

(19)

By inserting equation (16) and (17) into (19), the result will be:

$$\frac{R_L}{R_i} \times \frac{1}{1 + \frac{R_L T}{L} \frac{1}{\pi}} \times \frac{1 + i \cdot f_x}{\frac{i \cdot f_x}{\omega_{esr}}} \times \frac{1}{1 + \frac{i \cdot f_x}{\omega_{op}}} \times \frac{\omega_{cp0}}{\omega_{cz1}} = 1$$

(20)

Since $\omega_{cp1}$ is chosen to cancel out $\omega_{esr}$, solving out $\omega_{cp0}$ gives

$$\omega_{cp0} = \frac{R_i \cdot f_x}{R_L} \times \left[1 + \frac{R_L T}{L} \frac{1}{\pi}\right] \times \left[1 + \frac{i \cdot f_x}{\omega_{op}}\right] \times \left[1 + \frac{\omega_{op}}{\omega_n}ight] \times \frac{1}{1 + \frac{i \cdot f_x}{\omega_{cz1}}}$$

(21)

Inserting the values of $R_L$, $R_i$, $L$, $T$, $R_c$, $\omega_{cz1}$, $\omega_n$, $\omega_{op}$ and the desired crossover frequency $f_x$ gives

$$\omega_{cp0} = \frac{1.23 \cdot f_x \cdot R_i \cdot (L + 0.32 \cdot R_L) \cdot \sqrt{1 - 4 \cdot f_x^2 \cdot T^2 + 16 \cdot f_x^4 \cdot T^4} \cdot \sqrt{1 + 39.48 \cdot C_0^2 \cdot f_x^2 \cdot L^2 \cdot R_L^2}}{L \cdot R_L}$$

(22)
3.5 Transfer function of Boost converter

Plant

According to TI [8], the control to output function (function for the plant) is given by

\[ H(s)_{\text{plant}} = \frac{V_{\text{out}}(s)}{V_{\text{err}}(s)} = F_H(s) \cdot H_B(s) \cdot H_{DC}(s) \]  

(23)

\[ F_H = \frac{1}{1 + \frac{s}{\omega_n} + \frac{s^2}{\omega_n^2}} \]

\[ H_B = \frac{(1 + \frac{s}{\omega_{esr}}) \times (1 - \frac{s}{\omega_{rhp}})}{1 + \frac{s}{\omega_p}} \]

\[ H_{DC} = \frac{R_L}{R_i} \times (1 - D) \]

Where \( F_H \) is the high frequency term, \( H_B(s) \) is the small signal model of the power stage and \( H_{DC}(s) \) is the DC gain. So totally:

\[ H_{\text{plant}} = \frac{R_L}{R_i} \times (1 - D) \times \frac{(1 + \frac{s}{\omega_{esr}}) \times (1 - \frac{s}{\omega_{rhp}})}{1 + \frac{s}{\omega_p}} \times \frac{1}{1 + \frac{s}{\omega_n} + \frac{s^2}{\omega_n^2}} \]  

(24)

Where

\[ \omega_n = \frac{\pi}{T} \]

\[ \omega_{esr} = \frac{1}{R_c \cdot C} \]

\[ \omega_p = \frac{2}{C \cdot R_L} \]

\[ \omega_{rhp} = \frac{R_i}{L} \]

Controller

A compensator is needed to control the voltage. This is the model of the controller [9]. The values of the coefficients are given on the next page.

\[ H(s)_{\text{Controller}} = \frac{\omega_{cp0}}{s} \cdot \frac{1 + \frac{s}{\omega_{cz1}}}{1 + \frac{s}{\omega_{ep1}}} \]  

(25)
Whole System

The system transfer function is thus

\[ H(s)_{\text{system}} = \frac{H_{\text{plant}} \times H_{\text{controller}}}{1 + H_{\text{plant}} \times H_{\text{controller}}} \]  \hfill (26)

Figure 3.5.1: System set-up for current mode control

To obtain a stable system, \( H_{\text{controller}} \) needs to be based on \( H_{\text{plant}} \). According to [8], this is done by

1. \( \omega_{cp1} = \min(\omega_{esr}, \omega_{rhp}) \)
2. \( f_x = \min\left(\frac{f_s}{10 + \alpha}, \frac{\omega_{rhp}}{2 \cdot \pi}, \frac{1}{5 + \beta}\right) \)
3. \( \omega_{cz1} = \frac{f_x \cdot 2 \cdot \pi}{5} \)
4. \[ \omega_{cp0} = \frac{1.23 \cdot f_x \cdot R_i}{R(1-D)} \cdot \frac{\sqrt{1 - 4 f_x^2 \cdot T^2 + 16 f_x^4 \cdot T^4} \cdot \sqrt{1 + (\pi \cdot f_x \cdot C \cdot R)^2} \cdot \sqrt{1 + \left(\frac{2 \pi \cdot f_x}{\omega_{cp1}}\right)^2}}{\sqrt{1 + (2 \pi \cdot f_x \cdot C \cdot R)^2} \cdot \sqrt{1 + \left(\frac{2 \pi \cdot f_x \cdot L}{R \cdot (1 - D)^2}\right)^2}} \]

Notes:
- The order of the steps 1 to 4 must be kept because of coefficient dependencies.
- \( L \) is the coil inductance, \( R_i \) the load resistance and \( C \) the capacitance.
- \( T = 1/f_s \) is the switching period.
- \( R_i \) is the current sense gain (see Figure 3.3.1).
- \( R_c \) is the ESR (equivalent series resistance) of the capacitor.
- \( \alpha \) and \( \beta \) is margin constants used for conservative design. \( D \) is the duty cycle.
Derivation of $\omega_{cp0}$

To achieve the desired crossover frequency of $f_x$, the gain of the controller $\omega_{cp0}$ needs to be chosen based on $f_x$. The crossover frequency occurs when

$$|H(i \cdot f_x)_{plant} \cdot H(i \cdot f_x)_{controller}| = 1 \iff |H_{DC}(i \cdot f_x) \cdot H_0(i \cdot f_x) \cdot F(i \cdot f_x) \cdot H(i \cdot f_x)_{controller}| = 1$$

(27)

By inserting equation (24) and (25) into (27), the result will be:

$$\frac{R_L}{R_i} \times (1 - D) \times \frac{1 + (i \cdot f_x)}{\omega_{esr}} \times \frac{1 + (i \cdot f_x)^2}{\omega_{op}^2} \times \frac{1 + (i \cdot f_x)}{1 + \frac{i \cdot f_x}{\omega_{rhp}}} \times \frac{1 + (i \cdot f_x^2)}{\omega_{cz1}} \times \frac{1 + (i \cdot f_x^2)}{\omega_{rhp}} = 1$$

(28)

Solving out $\omega_{cp0}$ gives

$$\omega_{cp0} = \frac{R_i \times f_x}{R_L (1 - D)} \times \frac{1 + (i \cdot f_x^2)}{\omega_{op}^2} \times \frac{1 + (i \cdot f_x)}{\omega_{esr}} \times \frac{1 + (i \cdot f_x^2)}{\omega_{cz1}} \times \frac{1 + (i \cdot f_x)}{1 + \frac{i \cdot f_x}{\omega_{rhp}}}$$

(29)

Inserting the values of $R_L$, $R_i$, $L$, $T$, $R_c$, $\omega_{cz1}$, $\omega_n$, $\omega_{op}$, $\omega_{rhp}$ and the desired crossover frequency $f_x$ gives

$$\omega_{cp0} = \frac{1.23 \cdot f_x \cdot R_i}{R (1 - D)} \times \frac{\sqrt{1 - 4 f_x^2 \cdot T^2 + 16 f_x^4 \cdot T^4}}{\sqrt{1 + (\pi \cdot f_x \cdot C \cdot R)^2}} \times \frac{1 + (\frac{2 \pi \cdot f_x}{\omega_{cp1}})^2}{1 + \frac{2 \pi \cdot f_x \cdot L}{R (1 - D)^2}}$$

(30)

OBS:

Depending on what value $\omega_{cp1}$ gets, $\frac{1 + (\frac{i \cdot f_x}{\omega_{cp1}})}{1 + (\frac{i \cdot f_x}{\omega_{esr}})}$ will cancel out either $\frac{1 + (\frac{i \cdot f_x}{\omega_{cp1}})}{1 + (\frac{i \cdot f_x}{\omega_{rhp}})}$ or $\frac{1 + (\frac{i \cdot f_x}{\omega_{cz1}})}{1 + (\frac{i \cdot f_x}{\omega_{rhp}})}$.
3.6 Digital converter

So far, the model of both the plant and the controller are extracted. The model of the controller is the most interesting part. But since the controller depends on the plant, both models need to be considered. For the analog designer, the controller coefficients \( \omega_{cp0} \), \( \omega_{cp1} \), \( \omega_{cz1} \) can be directly related to the capacitor and resistor values inside an analog controller [10]. For a digital designer, a conversion is needed to the discrete time domain. This is done by 2 conversions:

1. **Bilinear transform**  
   This is done to move from the “s-plain” to “z-plain”

2. **Inverse discrete Fourier transform**  
   This is done to move to the discrete time domain.

### Bilinear transform

According to [11], \( s \) can be approximated by:

\[
s \approx \frac{2}{T} \cdot \frac{z-1}{z+1}
\]  

(31)

Inserting this into the controller will give

\[
H(s)_{Controller} = \omega_{cp0} \cdot \frac{1 + \frac{s}{\omega_{cz1}}}{s} \Rightarrow H\left(\frac{2}{T} \cdot \frac{z-1}{z+1}\right)_{Controller} = \frac{\omega_{cp0}}{1 + \frac{2}{T} \cdot \frac{z-1}{z+1}}
\]  

(32)

Writing both he numerator and denominator on quadratic form with respect to \( Z \) gives

\[
H_{controller} = \frac{Z^2 + \frac{T \cdot \omega_{cp0} \cdot \omega_{cp1} \cdot (2 + T \cdot \omega_{cz1})}{2(2 + T \cdot \omega_{cp1}) \cdot \omega_{cz1}} + Z \cdot \frac{T \cdot \omega_{cp0} \cdot \omega_{cp1} \cdot T^2}{2(2 + T \cdot \omega_{cp1})} + \frac{T \cdot \omega_{cp0} \cdot \omega_{cp1} \cdot (T \cdot \omega_{cz1} - 2)}{2(2 + T \cdot \omega_{cp1}) \cdot \omega_{cz1}}}{Z^2 - \frac{4}{(2 + T \cdot \omega_{cp1})} - \frac{T \cdot \omega_{cp1} - 2}{(2 + T \cdot \omega_{cp1})}}
\]  

(33)
Inverse discrete fourier transform

By replacing the coefficients multiplied with $Z$, a standard discrete two-pole-two-zero controller is given:

$$H[Z] = \frac{B_0 \cdot Z^2 + B_1 \cdot Z + B_2}{Z^2 - Z \cdot A_1 - A_2}$$

(34)

where

$$B_0 = \frac{T \omega_{cp0} \omega_{cp1} (2 + T \omega_{cz1})}{2(2 + T \omega_{cp1}) \omega_{cz1}}$$

$$B_1 = \frac{\omega_{cp0} \omega_{cp1} \cdot T^2}{2(2 + T \omega_{cp1})}$$

$$B_2 = \frac{T \omega_{cp0} \omega_{cp1} (T \omega_{cz1} - 2)}{2(2 + T \omega_{cp1}) \omega_{cz1}}$$

$$A_1 = \frac{4}{(2 + T \omega_{cp1})}$$

$$A_2 = \frac{T \omega_{cp1} - 2}{(2 + T \omega_{cp1})}$$

By knowing that

$$H[Z] = \frac{Y[Z]}{X[Z]} = \frac{B_0 \cdot Z^2 + B_1 \cdot Z + B_2}{Z^2 - Z \cdot A_1 - A_2} = \frac{B_2 \cdot Z^{-2} + B_1 \cdot Z^{-1} + B_0}{-A_2 \cdot Z^{-2} - A_1 \cdot Z^{-1} \cdot + 1} \iff$$

$$\iff Y[Z](-A_2 \cdot Z^{-2} - A_1 \cdot Z^{-1} \cdot + 1) = X[Z](B_2 \cdot Z^{-2} + B_1 \cdot Z^{-1} + B_0)$$

An inverse discrete fourier transform would lead to an LDE (linear difference equation).

$$-A_2 \cdot y[n-2] - A_1 \cdot y[n-1] + y[n] = B_2 \cdot x[n-2] + B_1 \cdot x[n-1] + B_0 \cdot x[n] \iff$$

$$y[n] = B_2 \cdot x[n-2] + B_1 \cdot x[n-1] + B_0 \cdot x[n] + A_2 \cdot y[n-2] + A_1 \cdot y[n-1]$$

Notes:

- $x[n]$ denotes the error input to the controller
- $y[n]$ denotes the output of the controller
- $x[n-1]$ / $y[n-1]$ denotes the input/output 1 sample ago
- $x[n-1]$ / $y[n-1]$ denotes the input/output 2 samples ago

Implementation
The result of the transformations and derivations on previous page gave the following equation

\[ y[n] = B_2 x[n-2] + B_1 x[n-1] + B_0 x[n] + A_2 y[n-2] + A_1 y[n-1] \]  

(35)

Since \( y[n] \) (output of the controller) only depends on previous inputs and outputs of itself multiplied with coefficients, a digital implementation is possible.

\[ y[n] = y_1[n] + B_0 x[n] \]

(36)

Where \( y_1[n] \) is calculated in-between the samples and \( y[n] \) is calculated whenever \( x[n] \) is given.

This reduces the calculations to 1 addition and 1 multiplication. The other operations are calculated before each sample and thus ready to use.
3.7 Subharmonic oscillation

A drawback with Current mode control is that it suffers from subharmonic oscillation [12]. To describe the problem in detail, an example is given below.

Geometric description

Figure 3.7.1: Subharmonic oscillation

Suppose the system is started for the first time. As seen in the figure, the current (red line) rises towards its maximum limit ($I_{\text{control}}$). The green line is an ideal reference current and does not exist in real implementations. The goal is to align the real current with the ideal reference current. By inspecting the reference current, one can see that it always starts to rise at $I_0$ and stops at $I_{\text{control}}$. The real current however, has and will have an error ($e_0$, $e_1$). The current does not have a fixed minimum current level and subharmonic oscillation occurs. This is since the current does not start from steady state initially in reality. Instead, the minimum current level ($I_0$) varies with time. When $I_0$ varies, the average level of the current (not depicted) will vary as well. Since the output voltage is directly related to the average level of the current ($V_{\text{out}} = I_a R_L$), the voltage varies with time as a consequence. The variations are often small enough to be classed as noise, but is instead a fundamental problem of current mode set-up.
**Algebraic description**

By looking at figure 3.7.1, one can see the following relations:

1. \( I_{\text{control}} = I_0 + m_1 T_1 \Leftrightarrow T_1 = \frac{I_{\text{control}} - I_0}{m_1} \)

2. \( I_{\text{control}} - m_2 T_2 = I_0 \Leftrightarrow T_2 = \frac{I_{\text{control}} - I_0}{m_2} \)

3. \( I_{\text{control}} = I_0 + e_0 + m_1 T_{1r} \Leftrightarrow T_{1r} = \frac{I_{\text{control}} - I_0 - e_0}{m_1} \)

4. \( i - m_2 T_{2r} = I_0 + e_1 \Leftrightarrow T_{2r} = \frac{I_{\text{control}} - I_0 - e_2}{m_2} \)

5. \( T_s = T_{1r} + T_{2r} = T_1 + T_2 \)

Inserting (1), (2), (3), (4) into (5) gives

\[
\frac{I_{\text{control}} - I_0 - e_0}{m_1} + \frac{I_{\text{control}} - I_0 - e_2}{m_2} = \frac{I_{\text{control}} - I_0}{m_1} + \frac{I_{\text{control}} - I_0}{m_2} \Leftrightarrow \frac{e_0}{m_1} = \frac{-e_1}{m_2} \Leftrightarrow \]

\[
\quad \Leftrightarrow e_1 = -\frac{m_1}{m_2} e_0
\]

This can be generalized to

\[
e_n = \left( -\frac{m_1}{m_2} \right)^n e_0
\]

(37)

For steady state condition, the relation below can be used:

\[
\frac{m_2}{m_1} = \frac{D}{1 - D}
\]

(38)

It can be seen that when \( D < 50 \% \), then \( m_1/m_2 \) will decrease towards zero. This will cancel out the error \( e_0 \), which will lead to an inductor current that settles. For \( D > 50 \% \), subharmonic oscillation occurs.
3.8 Slope compensation

A solution to subharmonic oscillation is called slope compensation [13]. Just as in the case of the problem, a geometrical description is given followed by algebraic relations.

Suppose the system is started for the first time. As seen in the figure, the real current (red line) rises towards its maximum limit $I_{\text{control}}$. The green line is only an ideal reference current, just like the case of previous geometrical description. The goal is to align the real current with the ideal reference current here as well. The difference is that the maximum current limit $I_{\text{control}}$ (black line) is varying with time. Note that whenever the real current reaches the maximum current limit, it starts to decrease. By inspecting the figure, one can see that the real current aligns itself with the ideal reference current. This type of slope compensation is linear, which can be seen in the figure. There are other types such as quadratic slope compensation. For simplicity, only linear slope compensation is discussed.

Notes:

- The real current settles within one switching cycle. This is called dead beat.
- The average value of the compensation ramp is the same as $I_{\text{control}}$ would be without slope compensation.
**Algebraic description**

By looking at the figure below, one can see the following relations:

\[ e_0 = m_1 T_d + m_c T_d \]

\[ e_1 = -(m_2 T_d - m_c T_d) \]

\[ \frac{e_1}{e_0} = \frac{m_2 - m_c}{m_1 + m_c} \iff e_1 = -\frac{m_2 - m_c}{m_1 + m_c} e_0 \]

This can be generalized to \( e_n = \left(-\frac{m_2 - m_c}{m_1 + m_c}\right)^n e_0 \)

**Notes:**

- If \( m_c = m_2 \), the perturbed current will align itself with the steady state current within one switching cycle. This is called dead beat.
- If \( m_c > m_2 \) such that \( 0 < m_c - m_2 < 1 \), the perturbed current will align itself but after more than 1 switching cycle.
- If \( m_c > m_2 \) such that \( m_c - m_2 > 1 \), the perturbed current will not align itself and subharmonic oscillation occurs.
- The average value of the compensation ramp should be the same as the control voltage.
4  METHOD

4.1 Introduction

The way this thesis is performed can be seen as an iterative process. The work starts at a theoretical level and the first goal is a mathematical model that describes the system. In later steps, a simulation model will be built and improved for each step. Between these steps, measurements will take place. The results of all measurements can be found in the result chapter. Since the simulation models are a result of the work, they will be presented in the results chapter as well. In this chapter, the methods used are described. This chapter can also be seen as a "step by step" guide, for the interested readers to perform the same work and measurements. Totally, there will be 4 implementation iterations. Each iteration has a

- **Work method**: Method used to create the simulation models, presented in the result chapter.

- **Measurement method**: A description of how the measurements are performed on the simulation models.

- **Goal**: The goal of each implementation iteration is described here. This is to help the reader maintain an overview of the work done.

- **Comments**: If needed, comments are given to clarify and avoid misunderstandings.
4.2 Used Design method

Component values of the plant will be considered first. The theory chapter and other designs found in the literature with similar input/output voltages will be used. When the component values of the plant are found, the controller will be designed. The theory chapter will be used during the design of the controller.

4.3 Used measurement method

Two types of measurements are of interest, the frequency response and the “combined step response”

- **Frequency response:**
  The frequency response of the open-loop system is of interest. This is the same as the frequency response of both the plant and the controller combined. The open-loop frequency response gives the phase-margin, gain-margin and crossover frequency. These parameters tell how fast and stable the system is.

- **Combined step response:**
  The response of a step in both voltage and current are of interest. First, the desired output voltage is increased (Vref to the controller), to see how much time and overshoot it takes to reach the desired voltage. Then, the required load current is increased, to see how much time and voltage drop it takes for the system to maintain the desired output voltage.
As seen in the picture above, 2 voltage levels (voltage steps) are of interest. The first voltage corresponds to a duty cycle less than 50 % and the higher voltage corresponds to a duty cycle higher than 50 %. For each voltage level, a step load is performed. This is done by reducing (or increasing) the load, increasing the need of more current, to maintain the same output voltage. The picture above will also act as a reference response, since it is illustrating the ideal case (no delays in steps, no voltage drops etc.). Vout and Iout for buck and boost mode are given in the list below. The reason of the chosen voltages/currents to test with is discussed in the background chapter.

<table>
<thead>
<tr>
<th>Vin</th>
<th>Vout1</th>
<th>Vout2</th>
<th>Iout1</th>
<th>Iout2</th>
<th>Iout3</th>
<th>Iout4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buck</td>
<td>5 V</td>
<td>1 V</td>
<td>4.5 V</td>
<td>0.1 A</td>
<td>0.2 A</td>
<td>0.45 A</td>
</tr>
<tr>
<td>Boost</td>
<td>5 V</td>
<td>6 v</td>
<td>12 v</td>
<td>0.5 A</td>
<td>1 A</td>
<td>1 A</td>
</tr>
</tbody>
</table>

Table 4.1: Voltages and currents used during the measurements.
4.4 Implementation

Totally there are 4 work iterations. Each iteration is described here in detail.

Work iteration 1

- **Working method**: Use the theory chapter and the literature to create a system model with the same system behavior as described in the Background chapter.

- **Measurement method**: Measure the open loop frequency response when the model is done. A detailed description of open-loop frequency response is given under “Used measurement method”.

- **Goal**: The goal is a mathematical model of both the plant and the controller.

- **Comments**: The work during this step is rather a research than implementation.

Work iteration 2

- **Working method**: Change the mathematical model of the plant to a Simulink model with components from the Simulink package Simscape (resistor, capacitor, inductor and load). Use the same mathematical model of the controller as used in the previous step. During the conversion to a Simulink model, use the theory chapter.

- **Measurement method**: Measure the “combined step response” when the model is done. A detailed description of “combined step response” is given under “Used measurement method”.

- **Goal**: The goal is a model where the plant is a Simulink model and the controller is a mathematical model.

Work iteration 3

- **Working method**: Change the mathematical model of the controller to a Simulink model with components from the Simulink package “commonly used blocks”. Use the same Simulink model for the plant as used in the previous step. During the conversion to a Simulink model, use the theory chapter.

- **Measurement method**: Measure the “combined step response” when the model is done. A detailed description of “combined step response” is given under “Used measurement method”.

- **Goal**: The goal is to create a model where both the plant and controller consists of Simulink components (system without mathematical parts).
Work iteration 4

- **Working method:**
  Increase the plant model with the following components:

  1. ADC with data transfer delay and a bit accuracy of 10 bits. An Appropriate Vref voltage should be used such that functionality is maintained.

  2. DAC with data transfer delay and a bit accuracy of 10 bits. An Appropriate Vref voltage should be used such that functionality is maintained.

  3. Resistive divider, which is needed at the input of the ADC.

  4. Introduce delays to the switches \((10^{-8} \text{ s})\)

Increase/change the model of the controller with the following components:

  1. Softstart block, such that instability is avoided

  2. Voltage dividers needed because of the DAC, ADC and resistive divider.

  3. Change all blocks from Simulink blocks to “ALTERA DSP builder” blocks (VHDL blocks).

  4. Introduce calculation delays of 1 clock cycles in the controller.

- **Measurement method:**
  Simulate the model until

  - Minimum bit accuracy (needed for the controller) is found

  - Minimum clock frequency of the system if found

- **Goal:** The goal is to create a model that behaves as realistic as possible.

- **Comments:** There are no components such as ADC or DAC in Simulink. Thus, a model that behaves as an ADC/DAC is to be built. The “data transfer delays” are introduced since the ADC/DAC data is transferred in serial. “An appropriate Vref voltage” means a Vref voltage that maintains functionality but is still realistic. A bit accuracy of 10 bits is chosen since it is considered to be realistic and common. Switch and calculation delays are predicted to be close to the given values, if implemented in real hardware.
5  RESULTS

5.1 Introduction

In this chapter, the result of all designs and measurement are given. Each design has its own measurements as described in the method chapter. Totally there are eight design results with a corresponding measurement result. At the end of this chapter, subsystems used in the designs are given. Since the controller inside model 4 (both Buck and Boost designs) is containing VHDL blocks, only the block names are given. The VHDL code for all blocks can be found in APPENDIX.

5.2 Design and Measurement results

In the following pages, both design and measurement results are given. The first four results are given for the Buck converter and the following four is Boost converter results. Each result contains the following information:

1. Model of the Plant
2. Model of the controller.
3. Coefficients used in the models.
4. Measurement result (frequency response or combined step response)
Buck converter model 1

The obtained mathematical model of the Buck converter is given below. The values of each coefficient are given in the table below. The Bode plot (frequency response) is given on the next page.

\[
H_{Plant} = \frac{R_L}{R_i} \times \frac{1}{1 + \frac{R_i T \cdot 1}{L_o}} \times \frac{1 + \frac{s}{\omega_{ESR}}}{1 + \frac{s}{\omega_{op}}} \times \frac{1}{1 + \frac{s}{\omega_n} + \frac{s^2}{\omega_n^2}}
\]  

(40)

\[
H_{Controller} = \frac{\omega_{cp0}}{\omega_{cz1}} \cdot \frac{1 + \frac{s}{\omega_{cz1}}}{s} \times \frac{1 + \frac{s}{\omega_{cp1}}}{s}
\]  

(41)

\[
H(s)_{system} = \frac{H_{plant} \times H_{controller}}{1 + H_{plant} \times H_{controller}}
\]  

(42)

<table>
<thead>
<tr>
<th>$R_L$</th>
<th>$10 \Omega$</th>
<th>$\omega_{op} = 3.590482 \cdot 10^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_i$</td>
<td>$0.05 \Omega$</td>
<td>$\omega_n = 6.2831853 \cdot 10^5$</td>
</tr>
<tr>
<td>$T$</td>
<td>$5 \cdot 10^{-6}$ s</td>
<td>$\omega_{cp0} = 4.12725 \cdot 10^4$</td>
</tr>
<tr>
<td>$L_o$</td>
<td>$2.2 \cdot 10^{-5}$ H</td>
<td>$\omega_{cz1} = 1.88495 \cdot 10^4$</td>
</tr>
<tr>
<td>$\omega_{ESR}$</td>
<td>$2.604166 \cdot 10^5$</td>
<td>$\omega_{cp1} = 2.60416 \cdot 10^5$</td>
</tr>
</tbody>
</table>

table 5.1: List of coefficients for Buck converter model 1
• Phase Margin (deg): 70.1812
• Gain Margin (dB): 16.4972
• Crossover frequency (HZ): 14975.5755
Buck converter model 2

The model of the Plant is given in the figure. The coefficients are given in the table.

![Model of the Plant, using Simulink components](image)

Table 5.2: Coefficients used inside the Plant in Buck converter model 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$2.2 \cdot 10^{-5} , H$</td>
</tr>
<tr>
<td>$T_{\text{Switch(ON)}}$</td>
<td>$0.001 , ns$</td>
</tr>
<tr>
<td>$R_i$</td>
<td>$0.05 , \Omega$</td>
</tr>
<tr>
<td>$C$</td>
<td>$4.8 \cdot 10^{-4} , F$</td>
</tr>
<tr>
<td>$L_{\text{ESR}}$</td>
<td>$0.1 , \Omega$</td>
</tr>
<tr>
<td>$R_{\text{Switch(ON)}}$</td>
<td>$0.1 , \Omega$</td>
</tr>
<tr>
<td>$R_{\text{switch(OFF)}}$</td>
<td>$10^8 , \Omega$</td>
</tr>
<tr>
<td>$C_{\text{ESR}}$</td>
<td>$8 , m\Omega$</td>
</tr>
</tbody>
</table>
The model of the Controller is given in the figure. The coefficients are given in the table.

\[
\omega_{cp_0} = 4.12725 \cdot 10^4 \\
\omega_{cz_1} = 1.88495 \cdot 10^4 \\
\omega_{cp_1} = 2.60416 \cdot 10^5
\]

Table 5.3: Coefficients used inside the Controller in Buck converter model 2.
Figure 5.2.5: Combined step response for Buck converter model 2
**Buck converter model 3**

The model of the Plant is given in the figure. The coefficients are given in the table.

---

**Figure 5.2.6: Model of the Plant, using Simulink components**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$2.2 \cdot 10^{-5} \text{H}$</td>
</tr>
<tr>
<td>$T_{\text{Switch}(ON)}$</td>
<td>$0.001 \text{ns}$</td>
</tr>
<tr>
<td>$R_i$</td>
<td>$0.05 \Omega$</td>
</tr>
<tr>
<td>$C$</td>
<td>$4.8 \cdot 10^{-4} \text{F}$</td>
</tr>
<tr>
<td>$L_{\text{ESR}}$</td>
<td>$0.1 \Omega$</td>
</tr>
<tr>
<td>$R_{\text{Switch}(ON)}$</td>
<td>$0.1 \Omega$</td>
</tr>
<tr>
<td>$R_{\text{Switch}(OFF)}$</td>
<td>$10^8 \Omega$</td>
</tr>
<tr>
<td>$C_{\text{ESR}}$</td>
<td>$8 \text{m} \Omega$</td>
</tr>
</tbody>
</table>

Table 5.4: Coefficients used inside the Plant in Buck converter model 3.
The model of the Controller is given in the figure. The coefficients are given in the table.

![Controller model](image)

Figure 5.2.7: Model of the Controller using Simulink components

![Error Amp model](image)

Figure 5.2.8: Model of the Error Amp using Simulink components

<table>
<thead>
<tr>
<th>B0</th>
<th>B1</th>
<th>B2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.90408</td>
<td>0.08137</td>
<td>-0.82271</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
<th>Z_{\text{delay}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.21135</td>
<td>-0.21135</td>
<td>5 \times 10^{-6} s</td>
</tr>
</tbody>
</table>

Table 5.5: Coefficients used inside the Controller in Buck converter model 3.
Figure 5.2.9: Combined step response for Buck converter model 3
Buck converter model 4

The model of the Plant is given in the figure. The coefficients are given in the table on next page.

Figure 5.2.10: Model of the Plant, using Simulink components
The model of the Controller is given in the figure. The coefficients for both the Plant and Controller are given in the table. The blocks Soft Start, Voltage Divider, ERROR AMP, Choose Bits and SET-RESET are described in detail in APPENDIX (VHDL).

![Diagram of Controller model](image)

**Figure 5.2.11:** Model of the Controller using Altera DSP builder components (VHDL)

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$2.2 \cdot 10^{-5}$</td>
</tr>
<tr>
<td>$T_{\text{Switch(OFF)}}$</td>
<td>$1 \text{ ns}$</td>
</tr>
<tr>
<td>$R_1$</td>
<td>$0.05 \Omega$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$1000 \Omega$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$1332 \Omega$</td>
</tr>
<tr>
<td>$R_4$</td>
<td>$2000 \Omega$</td>
</tr>
<tr>
<td>$R_{\text{ESR}}$</td>
<td>$0.1 \Omega$</td>
</tr>
<tr>
<td>$R_{\text{Switch(OFF)}}$</td>
<td>$0.1 \Omega$</td>
</tr>
<tr>
<td>$R_{\text{Switch(OFF)}}$</td>
<td>$10^8 \Omega$</td>
</tr>
<tr>
<td>$C_{\text{ESR}}$</td>
<td>$8 \text{ m}\Omega$</td>
</tr>
<tr>
<td>$L_{\text{ESR}}$</td>
<td>$0.1 \Omega$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$0.05 \Omega$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$1000 \Omega$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$1332 \Omega$</td>
</tr>
<tr>
<td>$R_4$</td>
<td>$2000 \Omega$</td>
</tr>
<tr>
<td>$R_{\text{ESR}}$</td>
<td>$0.1 \Omega$</td>
</tr>
<tr>
<td>$C_{\text{ESR}}$</td>
<td>$8 \text{ m}\Omega$</td>
</tr>
<tr>
<td>$L_{\text{ESR}}$</td>
<td>$0.1 \Omega$</td>
</tr>
</tbody>
</table>

Table 5.6: Coefficients used inside the Controller and Plant in Buck converter model 4.

- ERROR AMP is based on same coefficients as for Buck converter model 3
- Minimum bits needed inside controller: [Integer].[Fractional] = [3].[5]
- Minimum FPGA Clock frequency: 4 MHz
Figure 5.2.12: Combined step response for Buck converter model 4
Boost converter model 1

The obtained mathematical model of the Boost converter is given below. The values of each coefficient are given in the table below. The Bode plot (frequency response) is given on the next page.

\[
H_{\text{Plant}} = K_{dc} \times \frac{(1 + \frac{s}{\omega_{\text{ESR}}}) \times (1 - \frac{s}{\omega_{\text{RHP}}})}{1 + \frac{s}{\omega_p}} \times \frac{1}{1 + \frac{s}{\omega_n} + \frac{s^2}{\omega_n^2}} \tag{43}
\]

\[
H_{\text{Controller}} = \frac{\omega_{cp0}}{\omega_{cz1}} \times \frac{1 + \frac{s}{\omega_{cz1}}}{s} \tag{44}
\]

\[
H(s)_{\text{system}} = \frac{H_{\text{plant}} \times H_{\text{controller}}}{1 + H_{\text{plant}} \times H_{\text{controller}}} \tag{45}
\]

<table>
<thead>
<tr>
<th>(K_{dc})</th>
<th>(\omega_p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>6.94444 \times 10^2</td>
</tr>
<tr>
<td>(\omega_{\text{RHP}}) = 4.73484 \times 10^4</td>
<td>(\omega_n) = 6.28318 \times 10^5</td>
</tr>
<tr>
<td>(\omega_{cp1}) = 4.73484 \times 10^4</td>
<td>(\omega_{cp0}) = 100</td>
</tr>
<tr>
<td>(\omega_{ESR}) = 2.60416 \times 10^5</td>
<td>(\omega_{cz1}) = 1.51515 \times 10^3</td>
</tr>
</tbody>
</table>

Table 5.7: List of coefficients for Boost converter model 1
• Phase Margin (deg): 68.714
• Gain Margin (dB): 26.9616
• Crossover frequency (HZ): 408.804

Figure 5.2.13: Frequency response of Boost converter model 1
Boost converter model 2

The model of the Plant is given in the figure. The coefficients are given in the table.

\[
L = 2.2 \cdot 10^{-5} \text{ H} \quad T_{\text{Switch ON}} = 0.001 \text{ ns} \quad R_i = 0.05 \Omega \quad C = 4.8 \cdot 10^{-4} \text{ F}
\]

\[
L_{\text{ESR}} = 0.1 \Omega \quad R_{\text{Switch ON}} = 0.1 \Omega \quad R_{\text{Switch OFF}} = 10^8 \Omega \quad C_{\text{ESR}} = 8 \text{ m}\Omega
\]

Table 5.8: Coefficients used inside the Plant in Boost converter model 2.
The model of the Controller is given in the figure. The coefficients are given in the table.

\[ \omega_{cp_1} = 4.73484 \cdot 10^4 \]
\[ \omega_{cz_1} = 1.51515 \cdot 10^3 \]
\[ \omega_{cp_0} = 100 \]

Table 5.9: Coefficients used inside the Controller in Boost converter model 2.
Figure 5.2.17: Combined step response for Boost converter model 2
Boost converter model 3

The model of the Plant is given in the figure. The coefficients are given in the table.

![Boost converter model diagram]

**Table 5.10: Coefficients used inside the Plant in Boost converter model 3.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$2.2 \times 10^{-5} \text{H}$</td>
</tr>
<tr>
<td>$T_{\text{Switch}(\text{ON})}$</td>
<td>$0.001 \text{ns}$</td>
</tr>
<tr>
<td>$R_i$</td>
<td>$0.05 \Omega$</td>
</tr>
<tr>
<td>$C$</td>
<td>$4.8 \times 10^{-4} \text{F}$</td>
</tr>
<tr>
<td>$L_{\text{ESR}}$</td>
<td>$0.1 \Omega$</td>
</tr>
<tr>
<td>$R_{\text{switch}(\text{ON})}$</td>
<td>$0.1 \Omega$</td>
</tr>
<tr>
<td>$R_{\text{switch}(\text{OFF})}$</td>
<td>$10^8 \Omega$</td>
</tr>
<tr>
<td>$C_{\text{ESR}}$</td>
<td>$8 \text{m\Omega}$</td>
</tr>
</tbody>
</table>
The model of the Controller is given in the figure. The coefficients are given in the table.

\[ B_0 = 0.0070120660 \]
\[ B_1 = 5.292125317 \times 10^{-5} \]
\[ B_2 = -0.0069591447 \]
\[ A_1 = 1.7883149872 \]
\[ A_2 = -0.7883149872 \]
\[ Z_{\text{delay}} = 0.000005 \]

Table 5.11: Coefficients used inside the Controller in Boost converter model 3.
Figure 5.2.21: Combined step response for Boost converter model 3
Boost converter model 4

The model of the Plant is given in the figure. The coefficients are given in the table on next page.

Figure 5.2.22: Model of the Plant, using Simulink components
The model of the Controller is given in the figure. The coefficients for both the Plant and Controller are given in the table. The blocks Soft Start, Voltage Divider, ERROR AMP, Choose Bits and SET-RESET are described in detail in APPENDIX (VHDL).

![Controller model diagram]

Figure 5.2.23: Model of the Controller using Altera DSP builder components (VHDL)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$2.2 \times 10^{-5} \text{H}$</td>
</tr>
<tr>
<td>$T_{\text{Switch (ON)}}$</td>
<td>$1 \text{ns}$</td>
</tr>
<tr>
<td>$R_1$</td>
<td>$0.05 \Omega$</td>
</tr>
<tr>
<td>$R_{\text{Switch (ON)}}$</td>
<td>$0.1 \Omega$</td>
</tr>
<tr>
<td>$R_{\text{Switch (OFF)}}$</td>
<td>$10^8 \Omega$</td>
</tr>
<tr>
<td>$C$</td>
<td>$4.8 \times 10^{-4} \text{F}$</td>
</tr>
<tr>
<td>$C_{\text{ESR}}$</td>
<td>$8 \text{m} \Omega$</td>
</tr>
<tr>
<td>$L_{\text{ESR}}$</td>
<td>$0.1 \Omega$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$1000 \Omega$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$1332 \Omega$</td>
</tr>
<tr>
<td>$R_s$</td>
<td>$2000 \Omega$</td>
</tr>
</tbody>
</table>

Table 5.12: Coefficients used inside the Controller and Plant in Boost converter model 4.

- ERROR AMP is based on same coefficients as for Boost converter model 3
- Minimum bits needed inside controller: [Integer].[Fractional] = [5].[20]
- Minimum FPGA Clock frequency: 4 MHz
Figure 5.2.24: Combined step response for Boost converter model 4
5.3 Used Subsystems

The following pages will describe the used subsystems seen inside the models. A short description is also given (if needed).

Switch

The used switches (acting as NMOS transistors) are containing following components.

- **OBS**: Transport Delay is changed by changing the coefficient $T_{Switch(ON)}$.

Single Piulse

- Unit delay = 1 FPGA clock cycle
- Output pulse will have a duration of 1 FPGA clock cycle.
**ADC**

The ADC introduces a quantization error (quantization block). It also introduces a delay before the data is reached to the controller. It takes [ADC_BITS] clock cycles for the data to reach the controller, when the voltage is sampled.

![Diagram of ADC components](image)

**Figure 5.3.3**: Components inside the ADC blocks.

**Notes:**

- *DFF ALTERA* block consists of a Data flip flop from DSP builder package.
- All voltages above Vref_ADC will be presented as Vref_ADC.
- All negative voltages will be presented as 0.
The DAC introduces a quantization error (quantization block). It also introduces a delay before the data is reached to the comparator. It takes \([\text{DAC\_BITS}]\) clock cycles for the data to reach the Comparator, when the digital value is sampled.

Notes

- **DFF ALTERA** block consists of a Data flip flop from DSP builder package
- Since a DAC normally takes input from 0 up to \(2^{\text{DAC\_BITS}}\), a voltage divider is used at the input to create the same effect.
- Since the ADC quantizer has an output \(D = \frac{A}{V_{\text{ref\_DAC}}} \times 2^{\text{ADC\_Bits}}\), a voltage divider is used at the output to cancel out the ADC effect.
**Voltage Divider (Inside DAC)**

Performs a multiplication needed to create a DAC from ADC quantizer block.

![Voltage Divider Diagram](image)

Figure 5.3.5: Components inside the Voltage Divider blocks.

**BUCK/BOOST SWITCH**

Since the same Plant is used, Both for Buck and Boost mode, the voltage over Ri will become negative during Buck mode. This is a problem since the DAC only can represent positive values. Thus a switch that flips the polarity of the voltage over the resistor Ri is needed.

![BUCK/BOOST SWITCH Diagram](image)

Figure 5.3.6: Components inside the BUCK/BOOST SWITCH block.
6 DISCUSSION

6.1 Results

Buck converter

By taking a look at the bode plot of the buck converter, it is clearly seen that the gain margin, phase margin and crossover frequencies are in a “good range”. The system is stable enough since $G_m > 20 \, \text{dB}$ and $P_m > 45$ degrees. It is also seen that this stability is achieved at a crossover frequency of 15000 Hz. This is quite good since it makes the system fast. The speed of the system is also demonstrated by looking at the combined step-responses. One can observe the following:

- When the need of current is doubled, there is no voltage drop to observe.
- The time it takes for the currents to rise/fall to its final value is comparable with an ideal case (see combined step-response in method chapter)
- The buck converter is, however, suffering from an overshoot, when the reference voltage is changed (voltage step). It also suffers from sub-harmonic oscillation. This can be seen by observing the “noise” when the voltage is 4.5 V ($D > 50\%$).
- It is also seen that the sub-harmonic oscillation is increased when looking at model 4. This is due to the quantization noise (from the ADC and DAC). The quantization noise can also be observed by looking at the output voltage when $V_o = 1 \, \text{V}$ ($D < 50\%$).
**Buck converter Softstart**

The most interesting thing to observe in model 4 is the softstart. Softstart is needed here since the ADC (and DAC) is not capable of representing the overshoots. But softstart does not only decrease the overshoots, it is also quite fast. This speed can be increased even further, since it is a digital softstart. The cost of this increase in speed is a small (or larger) overshoot, depending on how fast one wish the system to be.

**Boost Converter**

By taking a look at the bode-plot of the Boost converter, it is clearly seen that the stability is good. Both gain margin and phase margin lay in the appropriate intervals ($g_m > 10 \text{ dB} \& P_m > 45 \text{ deg}$). However, the crossover frequency is not that impressive as in the case of the buck converter. This because the crossover frequency only is 400 Hz. The system is not as fast as in the case of the buck converter. One of the reasons for this is the combined buck/boost converter setup. Since it was decided to have the same plant for both buck and boost mode, the analog component values do not match the design guidelines of both converter types. This converter is thus a compromise between 2 ways of designing a system. The major problem is that $R_i$ which slows down the boost converter. However, decreasing $R_i$ by further would make the buck converter unstable. The speed of the system can be observed by looking at the combined step-response here as well. One can observe the following:

- The system is not that fast and thus the voltage drop is present, when dubbing the current needed.
- The time it takes to reach the desired current is following the same slow pattern as the voltage drops.
- Even the boost converter suffers from sub-harmonic oscillation. The bold lines in the combined step responses are the consequence. They are a combination of:
  1. Voltage drops due to the high voltage/current that has to be maintained.
  2. Sub-harmonic oscillation

**Boost converter Softstart**

The softstart feature clearly decreases the overshoots, when comparing the combined stepresponses of model 4 with the others. It is however still big enough to be considered as a problem, especially when sensitive loads are to be connected. This is due to the slow speed of the system. A solution for this is either a slower slope of the softstart or a faster system (changing $R_i$).
6.2 Bits needed

Buck converter.

Since the maximum voltage of the buck converter is 5, only 4 signed bits are needed to represent the values. But since softstart is used, the error given to the error amplifier will never reach to an amount of $2^{4-1} = 2^3$. Thus the system works with only 3 bits, representing the integer part. The fractional bits is dependent on the coefficients used inside the controller (B0, B1, B2, A1, A2). The value of these coefficients decides how good accuracy is needed. In this case, only 5 fractional bits are needed. Thus 8 bits is enough to represent the total error at the input of the controller (error amplifier).

Boost converter

The maximum output voltage of the boost converter is chosen to be 12. Thus, 5 signed integer bits are chosen. In the case of the boost converter, the softstart does not damp all overshoots as in the case of the buck converter. Thus, the number of bits cannot be decreased. The fractional bits are dependent on the coefficients here as well (B0, B1, B2, A1, A2). But in the case of Boost converter, these coefficients are much smaller than the coefficients needed for the buck converter. Thus 20 fractional bits are used. By looking at the problem further, 15 fractional bits have been tested where functionality is maintained. But the results are bad enough to consider a converter with more bits as approved. Thus 20 bits are chosen to be a compromise between good results and lower bound of bits needed.

6.3 Method

Iterative process

Since the thesis started at a theoretical level, many models had to be calculated, implemented, measured and compared. An iterative process was thus considered to be the best way to go. A possible “extra cost” that can be associated with this approach is the amount of work needed. The whole project will act as a combination of smaller sub-projects. During each sub-project, implementation and measurements are needed. Adding up all these subtasks can end in a relatively long “to-do list”. Since there were only simulation designs to consider during this work, this extra cost did not appear as a problem. The benefits, however, were both more and more important:

- **Step by step design**: This is preferable when bigger/many systems are to be designed. It does not only make the system/systems easier to design, but also to understand at later steps.
• **Comparison of results**: The results of each iteration can be compared with each other. With the help of this comparison, the designer/designers can also determine whether something is wrong or not.

• **Predictable**: By starting from ideal mathematical models, one can see what is possible/not possible to achieve at later steps. This can save time since goals that are not achievable will be discarded.

• **Documentation**: The documentation during the first step can act as a template (if possible) for later steps.

• **Replicability**: The project will be described in detail, giving a higher replicability.

**Frequency response**

By looking at the methods/results, the reader can wonder why frequency response is considered only during the mathematical models. The main reason is the cost, when looking for the frequency response. Finding the frequency response in a software/simulation environment is easy, but requires expensive tools in real hardware. As discussed in the background chapter, this thesis is intended to act as a reference for future work. Thus, model 4 is only considered when performing **combined step response**. The results of model 2 and 1 should be compared with model 4, since they were Simulink models. That is the reason for using frequency response for model 1 and combined step response for the others.

**Combined step response**

Since this converter should be fast, both when $V_{ref}$ and $R_C$ change their values, a combination of both voltage steps and step loads was chosen to be measured. It is also reactively easy and cheap to perform this type of measurement in real hardware. All that is needed is an oscilloscope and a circuit that can switch between 2 different resistances.

![Example design of simple circuit](image)

Figure 6.3.1: Example design of simple circuit
**Speed of system clock**

The minimum clock frequency of the switch was 200 kHz, as stated in the background chapter. To obtain functionality, a minimum system clock of 4 MHz was needed. The reason for this is the Nyquist criterion. 200 kHz switching frequency needs a sample rate of 400 kHz as a minimum. A bit accuracy of 10 bits for the ADC and DAC were chosen. Totally this results in $200 \times 10^3 \times 2 \times 10 = 4$ MHz.

**Reliability**

Since all measurements were performed in a simulation environment, this thesis has high reliability. No extra noise will be added, from an unexpected source. This problem is possible, if the measurements were performed in real hardware. If the same simulation environment is used, all that is needed is a copy of the models. For those who wish to perform simulations in other environments, the accuracy of the simulation steps (in time) shall be chosen with care. Also the algorithms used during simulations shall preferably be the same.

**Validity**

One disadvantage of simulations is the validity. This is because simulations tend to be close to an ideal environment. Even if there have been efforts made to avoid ideal models, one can always expect noise from unexpected sources, during hardware implementation.
7 CONCLUSIONS

7.1 Performance and complexity

In the theory chapter it is explained that slope compensation is done by modifying the control voltage to a ramp, instead of a constant limit. A basic solution would be to let the DAC convert a minimum of 10 voltages every switching cycle (see figure below).

![Digital Slope compensation](image)

Figure 7.1.1: Digital Slope compensation

In this case, it will correspond to a DAC bit rate of 40 Mbps. To obtain a good slope compensation, 100 conversions would be a better choice. This will correspond to a DAC bit rate of 400 Mbps. Suppose the DAC can sample its input at either the clock edge. This would correspond to a clock frequency of 800 MHz. These values are based on the switching frequency. For those who wish a higher switching frequency, the requirements on the DAC will grow even more. This reminds of the price and complexity needed to obtain a stable system, for all duty cycles using current mode control. Comparing this to a voltage mode converter, it is easily seen that voltage mode is a preferable choice, when price and complexity are to be reduced. This is even more clear by looking at the boost converter. 25 bits were needed to maintain functionality. Using multipliers with a 25 bit wide bus will either take too much chip space, or take several clock cycles to calculate. But is there any benefits using current mode control? As mentioned in the introduction chapter, current mode control offers better stability (see figure 7 in [14]). Current mode control implemented into FPGA:s is thus a good choice when performance and flexibility weights more than price and complexity.
7.2 Future work

There are totally 5 types of current mode control methods [15]. “Constant-frequency with turn-on at clock time” is the one used in this thesis. Another interesting set-up is hysteric control method (also current mode control). This type of converter does not have a fixed switching frequency. Instead the switching frequency depends on the rising time and falling time of the inductor current. One advantage of this method is that there is no need of slope compensation. As described in the theory chapter, the coefficients (A0, A1, B0, B1, B2) are related to the switching frequency. Using a system with the switching frequency varying, these coefficients have to vary as well. This drawback can be turned into an advantage, if a fast digital system is used. Using FPGA:s, a coefficient bank could be implemented, creating room for a more general solution. With this approach, different voltage levels and duty cycles could be used with the same converter. Problem statements to be answered is given below:

- What is the relation between the amount of coefficients needed, for a certain input voltage range?

- Is it possible for the system to determine the inductor and capacitor values, without any user input?


APPENDIX

The following pages contain the code that describes model 4 for both buck and boost mode, inside the controller. For each block, the VHDL code is given for Boost converter followed by buck converter. This is the same code that can be found inside the simulation models. Since model 1, 2 and 3 does only contain simulink blocks, no code is given for these models.
-- This block takes a digital value and gives a digital value with less amount of bits. This is because
the amount – of bits taken from DIGITAL_CONTROLLER block is more than the bits ADC can take
in. Thus this block – is needed to choose out the bits that are needed to maintain functionality.

-- * INT_BITS IS the amount of bits that represent the integer bits.
-- * FRACTIONAL IS the amount of bits that represent the fractional bits.
-- * DAC_BITS is the amount of bits the DAC takes in.
-- * MSB_BIT_TO_DAC and LSB_TO_DAC is the MSB and LSB bits choosen out from
-- * DIGITAL_CONTROLLER block output. To know which bits to take out from the
-- * DIGITAL_CONTROLLER block output, eather a simulation is needed or the
-- script in matlab can be used. The script in matlab is not 100 % proven to work for any situation.

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.ALL;

entity CHOOSE_BITS_BOOST is
  generic
  (    MSB_BIT_TO_DAC:      natural := 39;
       LSB_TO_DAC:              natural := 30;
       INT_BITS:                      natural := 5;
       FRACTIONAL:               natural := 20;
       DAC_BITS   :                 natural := 10
  );
  port
    (    i : in   SIGNED(((2*INT_BITS + 3) + 2*FRACTIONAL - 1) downto 0);
        o : out  SIGNED((DAC_BITS) - 1 downto 0)
  );
end CHOOSE_BITS_BOOST;

architecture BEHAV of CHOOSE_BITS_BOOST is
begin
  o <= i(MSB_BIT_TO_DAC downto LSB_TO_DAC);
end BEHAV;
-- This block takes a digital value and gives a digital value with less amount
-- of bits. This is because the amount of bits taken from DIGITAL_CONTROLLER block
-- is more than the bits ADC can take in. Thus, this block is needed to choose
-- out the bits that are needed to maintain functionality.

-- INT_BITS is the amount of bits that represent the integer bits.
--
-- FRACTIONAL is the amount of bits that represent the fractional bits.

-- DAC_BITS is the amount of bits the DAC takes in.
--
-- MSB_BIT_TO_DAC and LSB_TO_DAC is the MSB and LSB bits chosen out from
-- DIGITAL_CONTROLLER block output. To know which bits to take out from the
-- DIGITAL_CONTROLLER block output, either a simulation is needed or the script
-- in matlab can be used. The script in matlab is not 100% proven to work for
-- any situation.

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.ALL;

entity CHOOSE_BITS_BUCK is
generic
(
  MSB_BIT_TO_DAC    : natural := 39;
  LSB_TO_DAC        : natural := 30;
  INT_BITS          : natural := 5;
  FRACTIONAL        : natural := 20;
  DAC_BITS          : natural := 10
);
port
(
  i: in        SIGNED(((2*INT_BITS + 3) + 2*FRACTIONAL - 1) downto 0);
  o: out       SIGNED((DAC_BITS) - 1 downto 0)
);
end CHOOSE_BITS_BUCK;

architecture BEHAV of CHOOSE_BITS_BUCK is
begin
  o <= i(MSB_BIT_TO_DAC downto LSB_TO_DAC);
end BEHAV;
-- This block takes a digital value that represents an error and outputs
-- a Vcontrol that is supposed to represent the upper limit of the current in a
-- peak current mode DC-DC buck/BOOST converter.

-- INT_ BITS IS the amount of bits that represent the integer bits.
-- FRACTIONAL IS the amount of bits that represent the fractional bits.
--
-- THE analog component values B0_VALUE, B1_VALUE, B2_VALUE, A1_VALUE, A2_VALUE
-- are the coefficients that is generated by the biriticha-calculator. Since
-- fractional values are used, each coefficient needs to be multiplied
-- with 2^FRACTIONAL and rounded. The formulas is given below:
--
-- B0_VALUE = round(B0*2^FRACTIONAL);
-- B1_VALUE = round(B1*2^FRACTIONAL);
-- B2_VALUE = round(B2*2^FRACTIONAL);
-- A1_VALUE = round(A1*2^FRACTIONAL);
-- A2_VALUE = round(A2*2^FRACTIONAL);
--

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.ALL;

entity DIGITAL_CONTROLLER_BOOST is

generic
(
    INT_BITS        : natural := 5;
    FRACTIONAL      : natural := 20;
    B0_VALUE        : integer :=  7352;
    B1_VALUE        : integer :=   55;
    B2_VALUE        : integer := -7298;
    A1_VALUE        : integer := 1875184;
    A2_VALUE        : integer := -826609
);

port
(
    SYSTEM_CLK   : in    std_logic;
    RESET_BLOCK  : in    std_logic;
    SAMPLE_CLK   : in    std_logic;
    E            : in    SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
    Y            : out   SIGNED(((2*(INT_BITS + FRACTIONAL) + 2)) downto 0)
);
end DIGITAL_CONTROLLER_BOOST;
architecture BEHAV of DIGITAL_CONTROLLER_BOOST is

begin

PRECALC1 <= ((resize((Yn2*A2),PRECALC1'length) + Yn1*A1) + (resize((En1*B1),PRECALC1'length) + En2*B2));
Yn0 <= (resize((En0*B0),Yn0'length) + PRECALC2);
Y <= Yn0;

process(SYSTEM_CLK)
begin
if(rising_edge(SYSTEM_CLK)) then
if(RESET_BLOCK = '0') then
En0 <= (others => '0');
En1 <= (others => '0');
En2 <= (others => '0');
SAMPLE_CLK_OLD <= '0';
PRECALC2 <= (others => '0');
Yn1 <= (others => '0');
Yn2 <= (others => '0');
else
if((SAMPLE_CLK = '1') and (SAMPLE_CLK_OLD = '0')) then
SAMPLE_CLK_OLD <= '1';
PRECALC2 <= PRECALC1;
En0 <= E;
elsif((SAMPLE_CLK = '0') and (SAMPLE_CLK_OLD = '1')) then
Yn2 <= Yn1;
Yn1(Yn1'length - 1) <= Yn0(Yn0'length - 1);
Yn1((Yn1'length - 2) downto 0) <= Yn0(((Yn1'length - 2) + FRACTIONAL) downto FRACTIONAL);
En2 <= En1;
En1 <= En0;
SAMPLE_CLK_OLD <= '0';
end if;
end if;
end if;
end process;
end BEHAV;
This block takes a digital value that represents an error and outputs a Vcontrol that is supposed to represent the upper limit of the current in a peak current mode DC-DC buck/BOOST converter.

INT_BITS IS the amount of bits that represent the integer bits.
FRACTIONAL IS the amount of bits that represent the fractional bits.

THE analog component values B0_VALUE, B1_VALUE, B2_VALUE, A1_VALUE, A2_VALUE are the coefficients that is generated by the biritcha-calculator. Since fractional values are used, each coefficient needs to be multiplied with 2^FRACTIONAL and rounded. The formulas is given below:

\[
\begin{align*}
    B0\_VALUE &= \text{round}(B0 \times 2^\text{FRACTIONAL}); \\
    B1\_VALUE &= \text{round}(B1 \times 2^\text{FRACTIONAL}); \\
    B2\_VALUE &= \text{round}(B2 \times 2^\text{FRACTIONAL}); \\
    A1\_VALUE &= \text{round}(A1 \times 2^\text{FRACTIONAL}); \\
    A2\_VALUE &= \text{round}(A2 \times 2^\text{FRACTIONAL}); \\
\end{align*}
\]

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.ALL;

entity DIGITAL_CONTROLLER_BUCK is
end DIGITAL_CONTROLLER_BUCK;

--

generic
(
    INT_BITS : natural := 5;
    FRACTIONAL : natural := 20;
    B0_VALUE : integer := 948000;
    B1_VALUE : integer := 85326;
    B2_VALUE : integer := -862675;
    A1_VALUE : integer := 1270199;
    A2_VALUE : integer := -221624
);

port
(
    SYSTEM_CLK : in std_logic;
    RESET_BLOCK : in std_logic;
    SAMPLE_CLK : in std_logic;
    E : in SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
    Y : out SIGNED(((2*(INT_BITS+FRACTIONAL) + 2)) downto 0)
);

end DIGITAL_CONTROLLER_BUCK;
architecture BEHAV of DIGITAL_CONTROLLER_BUCK is

signal SAMPLE_CLK_OLD : std_logic;
signal En0      : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := (others => '0');
signal En1      : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := (others => '0');
signal En2      : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := (others => '0');
signal Yn0      : SIGNED(((2*(INT_BITS + FRACTIONAL) + 2)) downto 0) := (others => '0');
signal Yn1      : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := (others => '0');
signal Yn2      : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := (others => '0');
signal PRECALC1 : SIGNED((2*(INT_BITS + FRACTIONAL) + 1) downto 0) := (others => '0');
signal PRECALC2 : SIGNED((2*(INT_BITS + FRACTIONAL) + 1) downto 0) := (others => '0');

constant B0 : SIGNED((INT_BITS+FRACTIONAL-1) downto 0):=to_signed(B0_VALUE,(INT_BITS + FRACTIONAL));
constant B1 : SIGNED((INT_BITS+FRACTIONAL-1) downto 0):=to_signed(B1_VALUE,(INT_BITS + FRACTIONAL));
constant B2 : SIGNED((INT_BITS+FRACTIONAL-1) downto 0):=to_signed(B2_VALUE,(INT_BITS + FRACTIONAL));
constant A1 : SIGNED((INT_BITS+FRACTIONAL-1) downto 0):=to_signed(A1_VALUE,(INT_BITS + FRACTIONAL));
constant A2 : SIGNED((INT_BITS+FRACTIONAL-1) downto 0):=to_signed(A2_VALUE,(INT_BITS + FRACTIONAL));

begin

PRECALC1 <= ((resize((Yn2*A2),PRECALC1'length) + Yn1*A1) +
(resize((En1*B1),PRECALC1'length) + En2*B2));

Yn0      <= (resize((En0*B0),Yn0'length) + PRECALC2);

Y        <= Yn0;

process(SYSTEM_CLK)
begin
if(rising_edge(SYSTEM_CLK)) then
if(RESET_BLOCK = '0') then
En0 <= (others => '0');
En1 <= (others => '0');
En2 <= (others => '0');
SAMPLE_CLK_OLD <= '0';
PRECALC2 <= (others => '0');
Yn1 <= (others => '0');
Yn2 <= (others => '0');
else
if((SAMPLE_CLK = '1') and (SAMPLE_CLK_OLD = '0')) then
SAMPLE_CLK_OLD <= '1';
PRECALC2 <= PRECALC1;
elseif((SAMPLE_CLK = '0') and (SAMPLE_CLK_OLD = '1')) then
Yn2 <= Yn1;
Yn1 <= Yn0(Yn0'length - 1);
Yn1((Yn1'length - 2) downto 0) <= Yn0(((Yn1'length - 2) +
FRACTIONAL) downto FRACTIONAL); 
En2 <= En1;
En1 <= En0;
SAMPLE_CLK_OLD <= '0';
end if;
end if;
end process;
end BEHAV;
-- This block is a set-reset block with a leading_edge blanking protection
-- against glitches in a current-mode DC-DC BUCK/BOOST converter.

-- BLANKING_WINDOW represent amount of system clock cycles to wait until RESET
-- input is enabled. This will protect against glitches that occour on the RESET
-- pin. The higher value BLANKING_WINDOW is set to, the longer it takes until
-- RESET pin is enabled.
-
-- Q and Q_PRIM are outputs connected to the switches in the analog part.
-- SAFE reset is an output activiating a switch, preventing the voltage to grow
-- during a reset of the whole system.

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.ALL;

entity SET_RESET_BOOST is
  generic
  (  
    BLANKING_WINDOW         : natural := 5  
  );
  port  
  (  
    SYSTEM_CLK              : in        std_logic;
    RESET_BLOCK             : in        std_logic;
    S                       : in        std_logic;
    R                       : in        std_logic;
    Q                       : out std_logic;
    Q_PRIM                  : out std_logic;
    SAFE_RESET              : out std_logic
  );
end SET_RESET_BOOST;

architecture BEHAV of SET_RESET_BOOST is
begin
  process(SYSTEM_CLK)
  variable BLANKING_COUNTER : integer range 0 to BLANKING_WINDOW;
  begin
    if (rising_edge(SYSTEM_CLK)) then
      if(RESET_BLOCK = '0') then
        BLANKING_COUNTER := 0;
        Q          <= '0';
        Q_PRIM     <= '0';
        SAFE_RESET <= '1';
      else
        SAFE_RESET <= '0';
        if(BLANKING_COUNTER < BLANKING_WINDOW) then
          BLANKING_COUNTER := BLANKING_COUNTER + 1;
        end if;
        if ((R = '1')and(BLANKING_COUNTER=BLANKING_WINDOW)) then
          Q          <= '0';
          Q_PRIM     <= '1';
        elseif (S = '1') then
          BLANKING_COUNTER := 0;
          Q          <= '1';
          Q_PRIM     <= '0';
        end if;
      end if;
    end if;
  end process;
end BEHAV;
-- This block is a set-reset block with a leading_edge blanking protection
-- against glitches in a current-mode DC-DC BUCK/BOOST converter.

-- BLANKING_WINDOW represent amount of system clock cycles to wait until RESET
-- pin is enabled. This will protect against glitches that occur on the RESET
-- pin. The higher value BLANKING_WINDOW is set to, the longer it takes until
-- RESET pin is enabled.

-- Q and Q_PRIM are outputs connected to the switches in the analog part.
-- SAFE reset is an output activating a switch, preventing the voltage to grow
-- during a reset of the whole system.

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.ALL;

entity SET_RESET_BUCK is
  generic
    (   BLANKING_WINDOW : natural := 5
  );
  port
    (   SYSTEM_CLK              : in        std_logic;
    RESET_BLOCK             : in        std_logic;
    S                       : in        std_logic;
    R                       : in        std_logic;
    Q                       : out       std_logic;
    Q_PRIM                  : out       std_logic;
    SAFE_RESET              : out       std_logic
  );
end SET_RESET_BUCK;

architecture BEHAV of SET_RESET_BUCK is
begin
  process(SYSTEM_CLK)
  variable BLANKING_COUNTER : integer range 0 to BLANKING_WINDOW;
  begin
    if (rising_edge(SYSTEM_CLK)) then
      if(RESET_BLOCK = '0') then
        BLANKING_COUNTER := 0;
        Q                   <= '0';
        Q_PRIM              <= '0';
        SAFE_RESET          <= '1';
      else
        SAFE_RESET          <= '0';
        if(BLANKING_COUNTER < BLANKING_WINDOW) then
          BLANKING_COUNTER := BLANKING_COUNTER + 1;
        end if;
        if (R = '1') and (BLANKING_COUNTER = BLANKING_WINDOW) then
          Q             <= '0';
          Q_PRIM        <= '1';
        elsif (S = '1') then
          BLANKING_COUNTER := 0;
          Q             <= '1';
          Q_PRIM        <= '0';
        end if;
      end if;
    end if;
  end process;
end BEHAV;
This block takes a digital value that represents $V_{ref}$ and outputs a digital value that grows towards $V_{ref}$ by 1 step each sampling cycle.

**INT_ BITS** is the amount of bits that represent the integer bits.

**FRACTIONAL** is the amount of bits that represent the fractional bits.

**STARTING_VALUE** is the value the output begins at. This is needed for BOOST converter and should be the value of the input voltage to the system.

**STEP_BITS** is representing amount of fractional bits needed for each step. The more bits, the smaller step. For example:

- **2 bits** => a step is $2^{-2}$
- **4 bits** => a step is $2^{-4}$
- **8 bits** => a step is $2^{-8}$
- **n bits** => a step is $2^{-n}$

----- OBS: ADC_BITS < STEP_BITS < FRACTIONAL -----
STEP_BITS should not be too high or too low. In both cases, it can lead to instability.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.ALL;

entity SOFT_START_BOOST is
    generic
    (    
      -- This is needed for the BOOST CONVERTER.
      -- The start value should be more than
      -- Vin on the converter input.
      START_VALUE        : natural := 5;
      STEP_BITS_UP       : natural := 7;
      STEP_BITS_DOWN     : natural := 10;
      INT_BITS           : natural := 5;
      FRACTIONAL         : natural := 20
    );

    port
    (    
      SYSTEM_CLK    : in  std_logic;
      RESET_BLOCK   : in  std_logic;
      SAMPLE_CLK    : in  std_logic;
      Vref_in       : in  SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
      Vref_out      : buffer    SIGNED(24 downto 0)
    )
end SOFT_START_BOOST;
```
architecture BEHAV of SOFT_START_BOOST is

type UP_OR_DOWN is (UP, DOWN);

signal SAMPLE_CLK_OLD : std_logic;
signal DIRECTION : UP_OR_DOWN := UP_OR_DOWN'(UP);

signal Old_Vref   : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0)     := (others => '0');
signal Soft_Vref1 : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0)     := (others => '0');
signal Soft_Vref2 : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0)     := (others => '0');

constant STARTING_VALUE : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0)  :=
                          to_signed(START_VALUE*(2**FRACTIONAL),(INT_BITS + FRACTIONAL));

constant STEP_UP        : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0)  :=
                          to_signed(2**(FRACTIONAL - STEP_BITS_UP),(INT_BITS + FRACTIONAL));

constant STEP_DOWN      : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0)  :=
                          to_signed(2**(FRACTIONAL - STEP_BITS_DOWN),(INT_BITS + FRACTIONAL));

begin
  vref_out <= Soft_Vref2 + STARTING_VALUE;
  process(SYSTEM_CLK)
  begin
    if(rising_edge(SYSTEM_CLK)) then
      if(RESET_BLOCK = '0') then
        DIRECTION      <= UP;
        SAMPLE_CLK_OLD <= '0';
        Old_Vref       <= (others => '0');
        Soft_Vref1     <= (others => '0');
        Soft_Vref2     <= (others => '0');
      else
        if((SAMPLE_CLK = '1') and (SAMPLE_CLK_OLD = '0')) then
          -- Depending on direction, change the value of Vref
          -- by adding a step value each sample cycle.
          if(DIRECTION = UP) then
            if(Vref_out < Vref_in) then
              Soft_Vref2 <= (Soft_Vref1 + STEP_UP);
            end if;
            elsif(DIRECTION = DOWN) then
              if(Vref_out > Vref_in) then
                Soft_Vref2 <= (Soft_Vref1 - STEP_DOWN);
              end if;
            end if;
            -- If Vref changed, change direction if needed.
            if(Vref_in > Old_Vref) then
              DIRECTION <= UP;
              Old_Vref <= Vref_in;
            elsif(Vref_in < Old_Vref) then
              DIRECTION <= DOWN;
              Old_Vref <= Vref_in;
            end if;
            elsif((SAMPLE_CLK = '0') and (SAMPLE_CLK_OLD = '1')) then
              SAMPLE_CLK_OLD <= '0';
              Soft_Vref1 <= Soft_Vref2;
            end if;
          end if;
        end if;
      end if;
    else
      if(SAMPLE_CLK_OLD = '1') then
        SAMPLE_CLK_OLD <= '0';
        Soft_Vref1 <= Soft_Vref2;
      end if;
    end if;
  end if;
  end process;
end BEHAV;
This block takes a digital value that represents Vref and outputs a digital value that grows towards Vref by 1 step each sampling cycle.

* INT_ BITS IS the amount of bits that represent the integer bits.

* FRACTIONAL IS the amount of bits that represent the fractional bits.

* STARTING_VALUE is the value the output begins at. This is needed for the BOOST converter and should be the value of the input voltage to the system.

* STEP_BITS is representing amount of fractional bits needed for each step. The more bits, the smaller step. For example:

  ** 2 bits => a step is 2^-2
  ** 4 bits => a step is 2^-4
  ** 8 bits => a step is 2^-8
  ** n bits => a step is 2^-n

---

====== **** OBS: ADC_BITS < STEP_BITS < FRACTIONAL **** ======
---

STEP_BITS should not be too high or too low. In both cases, it can lead to instability.
---

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.ALL;

entity SOFT_START_BUCK is
  generic
  (
    -- This is needed for the BOOST Converter. The start value should be more than Vin on the converter input.
    START_VALUE : natural := 5;
    STEPBITS_UP  : natural := 7;
    STEPBITS_DOWN: natural := 10;
    INT_BITS     : natural := 5;
    FRACTIONAL   : natural := 20
  );

  port
  (
    SYSTEM_CLK : in std_logic;
    RESET_BLOCK: in std_logic;
    SAMPLE_CLK : in std_logic;
    Vref_in    : in SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
    Vref_out   : buffer SIGNED(24 downto 0)
  );
end SOFT_START_BUCK;
architecture BEHAV of SOFT_START_BUCK is

type UP_DOWN is (UP, DOWN);
signal SAMPLE_CLK_OLD : std_logic;
signal DIRECTION : UP_OR_DOWN := UP_OR_DOWN'(UP);
signal Old_Vref : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := (others => '0');
signal Soft_Vref1 : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := (others => '0');
signal Soft_Vref2 : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := (others => '0');

constant STARTING_VALUE : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := to_signed(START_VALUE*(2**FRACTIONAL),(INT_BITS + FRACTIONAL));
constant STEP_UP : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := to_signed(2**(FRACTIONAL - STEP_BITS_UP),(INT_BITS + FRACTIONAL));
constant STEP_DOWN : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) := to_signed(2**(FRACTIONAL - STEP_BITS_DOWN),(INT_BITS + FRACTIONAL));

begin
  vref_out <= Soft_Vref2 + STARTING_VALUE;
  process(SYSTEM_CLK)
  begin
    if(rising_edge(SYSTEM_CLK)) then
      if(RESET_BLOCK = '0') then
        DIRECTION <= UP;
        SAMPLE_CLK_OLD <= '0';
        Old_Vref <= (others => '0');
        Soft_Vref1 <= (others => '0');
        Soft_Vref2 <= (others => '0');
      else
        if((SAMPLE_CLK = '1') and (SAMPLE_CLK_OLD = '0')) then
          SAMPLE_CLK_OLD <= '1';
          -- Depending on direction, change the value of Vref
          -- by adding a step value each sample cycle.
          if(DIRECTION = UP) then
            if(Vref_out < Vref_in) then
              Soft_Vref2 <= (Soft_Vref1 + STEP_UP);
            end if;
          elsif(DIRECTION = DOWN) then
            if(Vref_out > Vref_in) then
              Soft_Vref2 <= (Soft_Vref1 - STEP_DOWN);
            end if;
          end if;
          -- If Vref changed, change direction if needed.
          if(Vref_in > Old_Vref) then
            DIRECTION <= UP;
            Old_Vref <= Vref_in;
          elsif(Vref_in < Old_Vref) then
            DIRECTION <= DOWN;
            Old_Vref <= Vref_in;
          end if;
        else
          if((SAMPLE_CLK = '0') and (SAMPLE_CLK_OLD = '1')) then
            SAMPLE_CLK_OLD <= '1';
            -- Depending on direction, change the value of Vref
            -- by adding a step value each sample cycle.
            if(DIRECTION = UP) then
              if(Vref_out < Vref_in) then
                Soft_Vref2 <= (Soft_Vref1 + STEP_UP);
              end if;
            elsif(DIRECTION = DOWN) then
              if(Vref_out > Vref_in) then
                Soft_Vref2 <= (Soft_Vref1 - STEP_DOWN);
              end if;
            end if;
            -- If Vref changed, change direction if needed.
            if(Vref_in > Old_Vref) then
              DIRECTION <= UP;
              Old_Vref <= Vref_in;
            elsif(Vref_in < Old_Vref) then
              DIRECTION <= DOWN;
              Old_Vref <= Vref_in;
            end if;
          else
            if((SAMPLE_CLK = '0') and (SAMPLE_CLK_OLD = '1')) then
              SAMPLE_CLK_OLD <= '0';
              Soft_Vref1 <= Soft_Vref2;
            end if;
          end if;
        end if;
      end if;
    end if;
  end process;
end BEHAV;
-- * This block takes 2 inputs (a connects to SOFT_START block and
--   b connects to VOLTAGE_DIVIDER block), and performs a subtraction.
-- * INT_ BITS IS the amount of bits that represent the integer bits.
-- * FRACTIONAL IS the amount of bits that represent the fractional bits.

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

dentity SUBTRACTOR_BOOST is
generic
(
  INT_BITS           : natural := 5;
  FRACTIONAL         : natural := 20
);
port
(
  a      : in    SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
  b      : in    SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
  result : out   SIGNED((INT_BITS + FRACTIONAL - 1) downto 0)
);
end entity;

architecture behav of SUBTRACTOR_BOOST is
begin
  process(a,b)
  begin
    result <= a - b;
  end process;
end behav;
-- * This block takes 2 inputs (a connects to SOFT_START block and
--   b connects to VOLTAGE_DIVIDER block), and performs a subtraction.
-- * INT_BITS IS the amount of bits that represent the integer bits.
-- * FRACTIONAL IS the amount of bits that represent the fractional bits.

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity SUBTRACTOR_BUCK is
  generic
  (
    INT_BITS           : natural := 5;
    FRACTIONAL         : natural := 20
  );
  port
  (
    a      : in    SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
    b      : in    SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
    result : out   SIGNED((INT_BITS + FRACTIONAL - 1) downto 0)
  );
end entity;

architecture behav of SUBTRACTOR_BUCK is
begin
  process(a, b)
  begin
    result <= a - b;
  end process;
end behav;
This block takes a digital value from the ADC and outputs a
digital value to the rest of the system.

ADC BITS IS the amount of bits the ADC gives to the FPGA.

INT_ BITS IS the amount of bits that represent the integer bits.

FRACTIONAL IS the amount of bits that represent the fractional bits.

Since the output voltage is divided with a resistive divider outside
the FPGA, one need to multiply back with this value inside the FPGA,
This is the "DIVIDER_VALUE"

since the ADC output represents a fractional value of its Vref,
one need to multiply the digital value with the Vref.

Also, the DIVIDER_VALUE needs to be multiplied WITH 2^FRACTIONAL where
FRACTIONAL is fractional bits used.

The formula is: DIVIDER_VALUE = round(Vref_ADC*DIVIDER_GAIN*2^FRACTIONAL)
where:
- ** DIVIDER_GAIN is the value the resistive divider outside the FPGA
divides the voltage with.
- ** Vref_ADC is the reference voltage to the ADC.
- ** FRACTIONAL is the amount of bits that represent the fractional values

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.ALL;

entity VOLTAGE_DIVIDER_BOOST is
  generic
  (    ADC_BITS          : natural := 10;
      INT_BITS          : natural := 5;
      FRACTIONAL        : natural := 20;

      -- ADC maximal voltage = 3 (V).
      -- Voltage divider gain = 5/3.
      -- (5/3)*3*(2^FRACTIONAL) = 15728640
      DIVIDER_VALUE  : natural := 15728640--5242880
  );

  port
  (
    SYSTEM_CLK  : in      std_logic;
    RESET_BLOCK : in      std_logic;
    ADC         : in      UNSIGNED((ADC_BITS - 1) downto 0);
    Y           : out     SIGNED((INTBITS + FRACTIONAL - 1) downto 0)
  );
end VOLTAGE_DIVIDER_BOOST;
architecture BEHAV of VOLTAGE_DIVIDER_BOOST is

constant VOLTAGE_DIVIDER_GAIN : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) :=
    to_signed(DIVIDER_VALUE,(INT_BITS + FRACTIONAL));

signal ADC_EXTENDED_BITS : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
signal Y1 : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
signal Y2 : SIGNED((2*(INT_BITS + FRACTIONAL) - 1) downto 0);

begin

    --ADC is unsigned. first resize it and then convert it to SIGNED.
    -- This is because the length needs to be "Integer_bits + Fractional bits"
    ADC_EXTENDED_BITS <= SIGNED(resize(ADC,ADC_EXTENDED_BITS'length));

    -- Now the extended bits shall represent a value such that 0 < Value < 1.
    -- This is the same as dividing the ADC value by 2^ADC_BITS
    Y1 <= resize(ADC_EXTENDED_BITS*(2**(FRACTIONAL - ADC_BITS)),Y1'length);

    -- take out the needed bits to output
    Y <= Y2((Y2'length - 1 - INT_BITS) downto ((Y2'length - 1 - INT_BITS) - (Y'length - 1)));

process(SYSTEM_CLK)
begin
    if(rising_edge(SYSTEM_CLK)) then
        if(RESET_BLOCK = '0') then
            Y2 <= (others => '0');
        else
            Y2  <= Y1*VOLTAGE_DIVIDER_GAIN;
        end if;
    end if;
end process;
end BEHAV;
This block takes a digital value from the ADC and outputs a digital value to the rest of the system.

ADC BITS IS the amount of bits the ADC gives to the FPGA.

INT_ BITS IS the amount of bits that represent the integer bits.

FRACTIONAL IS the amount of bits that represent the fractional bits.

Since one divided the output voltage with a resistive divider outside the FPGA, this value needs to be multiplied back inside the FPGA. This is the "DIVIDER_VALUE"

Since the ADC output represents a fractional value of its Vref, the digital value needs to be multiplied with the Vref.

Also, one need to multiply the DIVIDER_VALUE WITH 2\(^\text{FRACTIONAL}\) where FRACTIONAL is the fractional bits.

The formula is: DIVIDER_VALUE = round(Vref_ADC*DIVIDER_GAIN*2\(^\text{FRACTIONAL}\)) where:
- **DIVIDER_GAIN** is the value the resistive divider outside the FPGA divides the voltage with.
- **Vref_ADC** is the reference voltage to the ADC.
- **FRACTIONAL** is the amount of bits that represent the fractional values.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.ALL;

entity VOLTAGE_DIVIDER_BUCK is
  generic
  (ADC_BITS : natural := 10;
   INT_BITS : natural := 5;
   FRACTIONAL : natural := 20;
   -- ADC maximal voltage = 3 (V).
   -- Voltage divider gain = 5/3.
   -- (5/3)*3*(2\(^\text{FRACTIONAL}\)) = 15728640
   DIVIDER_VALUE b : natural := 5242880)
  port
  (SYSTEM_CLK : in std_logic;
   RESET_BLOCK : in std_logic;
   ADC : in UNSIGNED((ADC_BITS - 1) downto 0);
   Y : out SIGNED((INT_BITS + FRACTIONAL - 1) downto 0)
  );

end VOLTAGE_DIVIDER_BUCK;
```
architecture BEHAV of VOLTAGE_DIVIDER_BUCK is

constant VOLTAGE_DIVIDER_GAIN : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0) :=
        to_signed(DIVIDER_VALUE,(INT_BITS + FRACTIONAL));

signal ADC_EXTENDED_BITS : SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
signal Y1 :SIGNED((INT_BITS + FRACTIONAL - 1) downto 0);
signal Y2 :SIGNED((2*(INT_BITS + FRACTIONAL) - 1) downto 0);

begin

    --ADC is unsigned. So resize it first and then convert it to SIGNED.
    -- This is because one want the length to be "Integer_bits + Fractional bits"
    ADC_EXTENDED_BITS <= SIGNED(resize(ADC,ADC_EXTENDED_BITS'length));

    -- Now the extended bits needs to represent a value such that 0 < Value < 1.
    -- This is the same as dividing the ADC value by 2^ADC_BITS.
    Y1 <= resize(ADC_EXTENDED_BITS*(2**(FRACTIONAL - ADC_BITS)),Y1'length);

    -- take out the needed bits to output
    Y <= Y2((Y2'length - 1 - INT_BITS) downto ((Y2'length - 1 - INT_BITS) - (Y'length - 1)));

    process(SYSTEM_CLK)
    begin
        if(rising_edge(SYSTEM_CLK)) then
            if(RESET_BLOCK = '0') then
                Y2 <= (others => '0');
            else
                Y2 <= Y1*VOLTAGE_DIVIDER_GAIN;
            end if;
        end if;
    end process;

end BEHAV;