Delft University of Technology

Faculty of Electrical Engineering, Mathematics and Computer Science

B.A.P. Universal & Modular Gate Driver

Bachelor Graduation Project Electrical Engineering
EE3L11

Authors:
Luc van der Beek - 4262514
Dries Beerkens - 4701860
Jop Kamphuis - 5171016
Federico Penner Heinsohn - 5137020

December 13, 2024

Abstract

The focus of this report is the design and implementation of a universal gate driver for half-bridge circuits. This design includes a programmable dead-time controller and adjustable properties to ensure efficient and reliable switching on a variety of high-voltage applications.

Furthermore, this work includes a comparative understanding of all design choices, such as the options considered for the dead-time controller, the gate driver component and the circuit of the output phase. The designed system is expected to receive as input a PWM signal and generate two output signals to drive a half-bridge configuration. The output pulses must be generated with a programmable dead-time and at the input switching frequency.

The end product of this design will be used in the TU Delft Power Electronics Lab. The specific requirements are determined in order to meet the lab's needs and this report further describes the process from acquiring the technical requirements, the circuit design, and finally, the prototype evaluation.

Preface

The design and development of high-performance power electronics systems often hinge on the choice and control of the switching devices, like IGBTs, Si and SiC MOSFETs. As MOSFET technology continues to evolve, the need for a versatile and adaptive gate driver becomes critical. Gate drivers serve as the interface between low-power control signals and high-power switching devices.

This project focuses on designing a universal gate driver that can adapt to various power transistor characteristics, including programmable dead-time, which ensures safe operation in half-bridge and full-bridge configurations by preventing simultaneous switching of the transistors, thus avoiding short circuits.
Additionally, the gate driver will be designed in such a way that certain components are adjustable, making it adaptable for integration into different systems.

The scope of this project covers the complete design process, from theoretical modeling to practical implementation and testing. This includes selecting the appropriate components for the gate driver, incorporating programmable dead-time control, and evaluating the design's functionality in a prototype.

This project is supervised by H. Vahedi who guided us to the end result. Since this project is part of the curriculum of the TU Delft we also had a Bachelor Thesis professor, Ioan E. Lager. We would also like to credit Bart Roodenburg, Joris Koeners, Harrie Olsthoorn, Mladen Gagic and Hitesh Dialani from the Electrical Sustainable Energy Department for supplying us with different instruments and assistance, as well as providing us with a room to work on our B.A.P. Furthermore we would like to mention Gonzalo Crespo Roche for providing us with information to get the project started.

Contents

1
Introduction

In power electronics, half- and full-bridge circuits play a crucial role in a variety of applications such as electric motor drivers, DC-AC converters, and DC-DC converters. These circuits make use of transistors or switches which, unlike transistors found in typical electronic appliances (smartphones, microprocessors, ...), are designed and rated for high-power applications such as driving motors in electric vehicles and power sources for industrial applications. In the aforementioned examples, smartphones and computing devices typically operate at a logic level of $3.3-5V$ while electric vehicle batteries are typically rated for $400V$, this illustrates the difference in operating ranges for power transistors[1].

The circuit layouts for typical applications of both half- and full-bridge circuits are shown in a simplified manner in figure 1.1. The switching devices are shown by $S1-S4$ while V represents the supply voltage and the $Load$ element represents the component at the receiving end of the circuits [2].

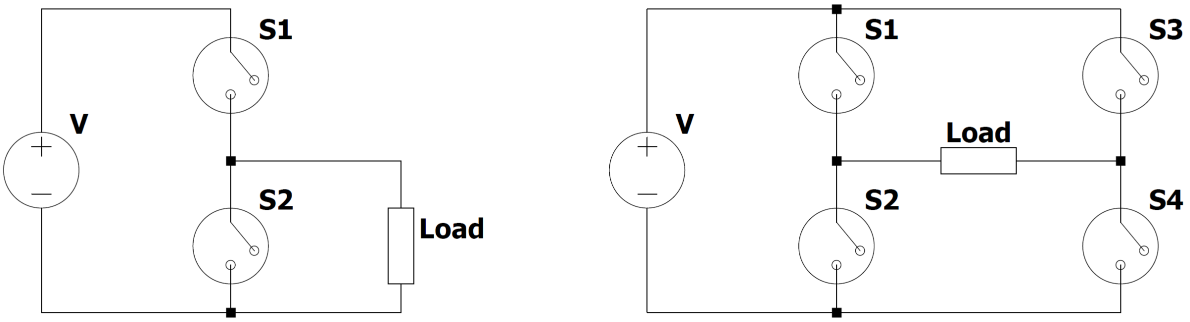


Figure 1.1: Typical examples of a half-bridge (left) and full-bridge circuit (right).

The half-bridge is operated by controlling the gate signals for both transistors, $S1$ and $S2$ in the figure shown above. These signals determine whether the gate at $S1$ should be on, thereby enabling the switch to conduct and providing a connection between source (V) and load; or whether the gate should be off, thereby blocking the connection to the source voltage. Similarly, they indicate whether $S2$ should connect to ground or not.

The full-bridge configuration can be understood as consisting of two parallel half-bridge circuits with the load connected between them. The control signals for the parallel half-bridges are inverted such that the current through the load is bidirectional, i.e. both $S3$ and $S2$ are closed or alternatively, both $S1$ and $S4$ are closed. The set of control signals for the full-bridge circuit can then also be understood as two parallel sets of control signals, each for a half-bridge but in inverted states with respect to each other.

As previously mentioned, voltage ranges in high-power operations are higher than typical appliances. Power switches must be specifically designed to handle and block these high-voltage sources, as well as the inevitable transient currents and voltage spikes which could damage the circuits. The physical capabilities for switches in these circuits requires careful consideration for each application in order to operate these circuits with efficiency and safety. An example to illustrate this dependency is the charge rate required to properly turn a switch on or off. If a switch is turned off faster than it's rated speed, i.e. the discharge current is too high, the device could be potentially damaged which could lead to dangerous system failures.

Moreover, the timing of the control signals for all switches is another important characteristic for properly operating these circuits. Designers take careful consideration of the required switching frequency, as well as the required dead-time in between. The switching frequency is the rate at which a switch is open and closed, it determines how power is transferred from the source to the load in converters. Dead-time refers to the brief moment in time in which both switches in a half-bridge are open. Proper dead-time control is critical to prevent shoot-through, a condition where both the high-side ($S1$) and low-side ($S2$) switches conduct simultaneously, leading to potential device failure. These timing characteristics of the control or gate signals are of great importance to meet the power transfer specifications used in converters and motor drivers. Additionally, proper gate control is an important factor in ensuring the safety of the operators and the devices in the circuits.

The operating conditions previously mentioned in these half- and full-bridge circuits give rise to the need for dedicated control systems to ensure these power switches properly function. These control systems are typically made up of a computing device and a gate driver. The computing device, typically a microcontroller, generates the signals with the specified switching frequency and dead-time. The gate driver device provides the necessary signal strength to either turn the gate on or off. A range of features related to optimization and protection against the high-power handled is often built into the gate driver or control system, these include protections against shoot-throughs as well as gate driving optimization. This project attempts to design and realize one such dedicated control system in the context of research in power converters.

Examples of gate driver systems include:

- UCC21331 by TI [3], which is an integrated half-bridge driver that uses a resistor to set the dead-time.
- 2ASC-17A1HP by Microchip [4], a half-bridge drive which provides software to program in the dead-time as well as other features.
- 2SC0108T2D0x-12 by Power Integrations [5], which is designed for applications limited in space.

In the research field of power converters, different types of power switches are used and tested in half-bridge designs. Each of these types has their own switching frequency capability, voltage and current ratings, turn-on and turn-off requirements, and other characteristics which are exclusive to each device. Testing and evaluating these characteristics is the main focus of researchers. The difference between different types of switches extends to different gate driving requirements. This means researchers often spend time and resources on both the half-/full-bridge design as well as the necessary gate driver and control system for the switches. Using industry-available products like the ones mentioned above require careful consideration when integrating into half-bridge power boards.

To facilitate research purposes in the TU Delft Power Electronics Lab, this project seeks to design a universal gate driver with programmable dead-time and adjustable output phase. This output phase consists of components for safety and optimization including a series resistance to limit current transfer to the gate. These components must be designed such that researchers can fine-tune the gate driver to match the specific needs of various switches. Furthermore, this universal gate driver design must be capable of performing efficiently within a specified frequency range and programmable dead-time to accommodate different devices while ensuring efficient and reliable switching. It should provide some tools for evaluating the half-/full-bridge performance as well it's own, such as LED's placed to respond to certain signals. It should also be designed such that researchers can easily integrate onto their power board designs.

In order to explain the process through which the design was created, the following structure is used. Firstly, the requirements that were set for this project will be outlined in chapter 2. Next, in chapter 3, the reasoning behind the choices for the various components will be discussed. Following this is chapter 4 where the design of the circuit, as well as the implementation of the VHDL code for the CPLD and circuit simulations made in LTspice, will be explained. Next the prototype that was built and tested and the results of these tests will be shown in chapter 5. Finally, in chapter 6, the overall conclusion and further recommendations for improvements will be discussed to improve the current design will be discussed.

2
Requirements

As discussed earlier, the universal gate driver is meant to be used in the context of research. Specifically, the TU Delft Power Electronics Lab proposed this project as a solution to facilitate their research into power converters. In qualitative terms, the gate driver design must be 'attached to any switches used in the lab and help PhD students design and test their converters faster', as can be read in the Project Description, appendix C.

This chapter will address the specific requirements that were provided by the lab through meetings with the project coordinator Hani Vahedi. A general description and overview is provided in section 2.1. The specific technical criteria is then discussed in section 2.2.

2.1. General Description

The designed system is expected to receive as input a PWM signal and generate two output signals to drive a half-bridge configuration. The output pulses must be generated with a programmable dead-time and at the input switching frequency. The expected input to output relation is illustrated in figure 2.1, where the output signals $high - side$ and $low - side$ respectively refer to the signals generated to drive $S1$ and $S2$, in terms of figure 1.1. The required dead-time generated between output signals is also high-lighted in the figure.

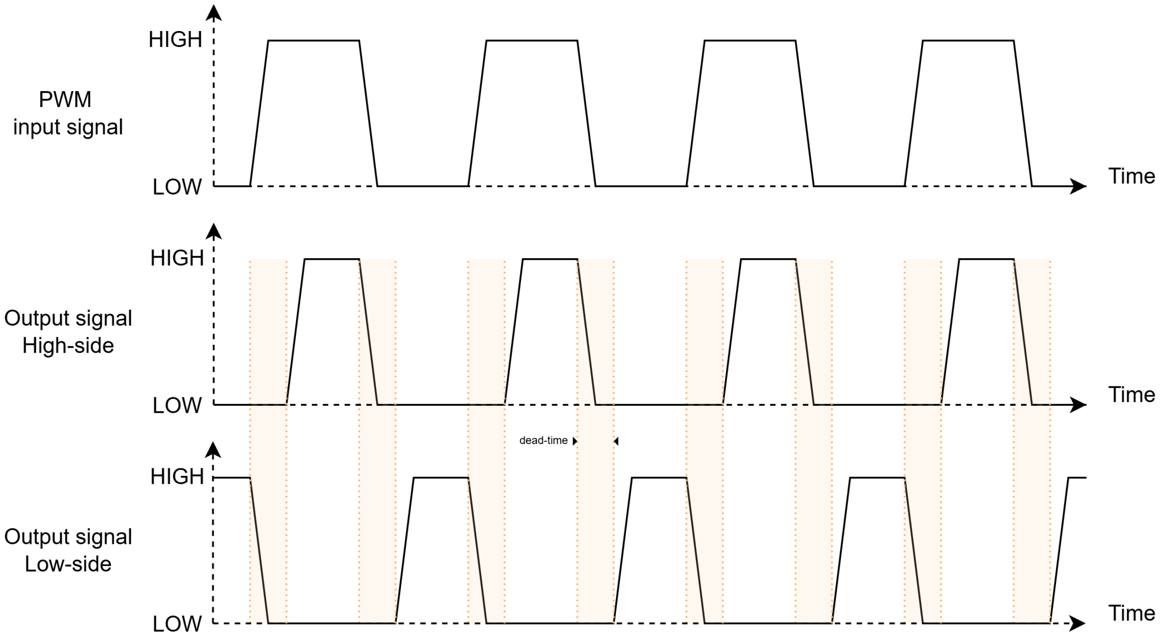


Figure 2.1: Illustration of required input to output relation.

Furthermore, the required output signals are unipolar, not bipolar.

The provided requirements further specify the method and structure by which the output signals are to be generated. Namely, through the use of Integrated Circuits (IC's). The designed system structure must include two IC components to achieve the main functions required by the overall system. Moreover, in order to further accommodate the system's capabilities to different types of switches, the output of the system must include a specific configuration of components to further optimize the output signals. The required IC units and the output components are outlined below.

1. **Programmable Unit**

   A programmable device is required to generate the initial output signals. The component options provided to achieve this functionality fall within the category of Programmable Logic Devices (PLD's) and Micro-Controllers. A list of specific criteria was provided by the lab in order to choose a specific IC device from within this category. The criteria list is provided in table 2.1.

   Regardless of the choice, this device is expected to read the input signal in the expected form and in turn generate the initial half-bridge signals in digital format. This device must be programmed to generate these initial signals with the dead-time included. Within the programming of this device, the key feature required is the ability to adjust the dead-time parameter, thereby making the dead-time programmable.

2. **Gate Driver Unit**

   The initial signals generated by the programmable device will each control a gate driver device. This device must include a digital input, and deliver the required signal strength to the switch gates. The signal strength refers to the previously mentioned physical capability of the signal to properly turn a switch on or off. These capabilities include, among others, the charge/discharge current and the required voltage level.

   These types of devices are widely employed in the industry and are manufactured with added features to enhance performance, protection, and control, among others. Examples include the previously mentioned UCC21331[3] which includes a programmable dead-time within the gate driver IC, other examples of added features include sensors to determine whether a short circuit has occurred and thus, close the switch before a dangerous breakdown occurs.

   Similar to the programmable IC, the gate driver IC must be chosen according to it's provided criteria list, shown in table 2.2. As can be seen in the table, this device must include specific added features. In the end, each product must contain 2 of these gate driver IC's for each output signal.

3. **Output phase**

   This phase of the system consists of series and parallel components located at the output of every gate driver IC. In other words, between the gate driver IC and the switch's input. The series component is a resistor that must be placed to limit current exchange. The parallel components are related to protection and signal optimization, these include a resistor, a capacitor, and a diode.

It is worth noting that the objective of this project is to design a product called 'Universal Gate Driver'. This product must include an IC whose category name is also 'gate driver'. While their is a description and discussion surrounding the gate driver IC, throughout this project thesis, the designed system as a whole is also typically referred to 'gate driver'. This should be kept in mind in the following sections and chapters.

The required layout and components are illustrated in figure 2.2. This block diagram includes the required input to output relation, the two previously mentioned IC components, and the Output Phases for both high-side and low-side.

Additional requirements apply to the overall design:

- Given the real necessity for the product's use in the lab, the required components to implement the design must be accessible in quantity through suppliers frequented by the lab.

- The costs for the IC devices must be kept as low as possible.

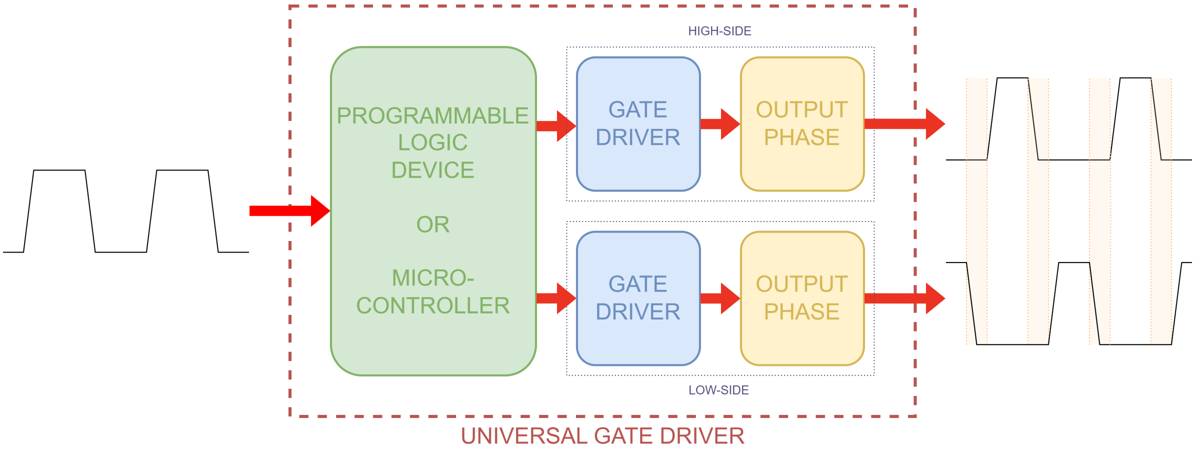


Figure 2.2: Block diagram of required design layout.

- The product is also expected to be designed in the form of a modular card, such that it can be easily attached onto power converter designs. This modularity also refers to it's expected ability of driving only one switch in a half-bridge configuration, and another aspect is the ability to use two of the products to drive a full-bridge configuration.

- The designed circuit is expected to be implemented on a Printed Circuit Board (PCB). The previously discussed IC units must be mounted on the PCB, as well as all other required components, input, and output pins.

Finally, since the product is designed for use in the context of research, certain features should be included in the circuit design to allow researchers to evaluate the Universal Gate Driver system during operation. Through discussions with the project supervisor, it was agreed upon that these features will be limited to Light Emitting Diodes (LED's) attached to certain signals within the system.

2.2. Technical Description

This section addresses the specific criteria for the devices and components previously described. The table format used throughout this section consists of a column for the required feature or property, a description column that provides an explanation for the required property, and lastly, a value column to quantitatively describe the respective property.

As previously mentioned, general requirements are that the components be accessible through commonly used suppliers and to keep the devices costs' are kept low. This implies that quantitatively defining some of these requirements will depend on conducting initial research into the options available from these suppliers. That being said, the scope of this initial research must include the features and parameters outlined in the respective tables.

2.2.1. PLD or Micro-controller

There are three different kind of device options for this part. The FPGA(field-programmable gate array) an MCU(Micro Control Unit) and a CPLD(Complex programmable Logic Device). These devices all must be able to split and invert the PWM signal in a cost efficient way. The requirements are listed below 2.1, and a more practical condition, it should be available in stock of the supplier.

Table 2.1: Specific requirements for programming unit.

Requirement	Description	Value
Dead-time	Dead-time in nanoseconds that should be programmable	300 ns
Programming	The IC can be programmed with JTAG	-
Package	Shape and place of the pins on the chip	xxFP
Propagation delay	Time required for a signal to be received after it has been sent	low
Input Voltage	Voltage it needs to operate	5V
Output voltage	Voltage level at the output pins	5V

To make a well-considered choice between three devices, their purpose and functionality needs to be understood. Here it is explained how each programmable unit works.

FPGA

An FPGA is a semiconductor device that can be programmed to perform a wide range of digital functions. It consists of an array of programmable logic blocks and interconnects, which can be configured to implement custom digital circuits.[6]
The hardware performance of the FPGA depends on Logic blocks, interconnects, I/O blocks and configuration Memory. Since the code used in this project does not heavily depend on the hardware but more on the software, these parts are not taken into consideration.

Microcontroller

An MCU is an integrated circuit intended for specific tasks. The hardware is set, but it can be programmed within the given hardware limitations. The CPU(Central Processing Unit) memory can be divided into volatile and non-volatile. Volatile memory stores variables of the executed code and the non-volatile memory stores the instruction that the CPU executes. The micro-controller also has peripherals which are extra components for different functionalities.

CPLD

A CPLD has the same architectural features but is less complex than an FPGA[7]. The FPGA's architecture is mostly based on LUTS (Look Up Tables) where as a CPLD's logic function is created with mostly gates. This makes a CPLD less complex and flexible. The trade-off between an FPGA and CPLD is not an obvious choice, since the MCU works different in a lot of ways a comparison for consideration besides costs is hard to make.

The main difference between these two are structural. The FPGA mostly relies on interconnects, where a CPLD is connected with lots of gates. These connections are also setup in a different way. The CPLD has more of a symmetrical structure which creates an equal delay between the logic blocks. The FPGA interconnects are not equally distributed which creates an uneven delay.

Another feature of the CPLD compared to the FPGA is that the memory is mostly non-volatile which means it can function immediately on turn-on.

2.2.2. Gate Driver IC

The detailed specifications for the IC are outlined in the table below. An additional key requirement is that the required gate driver must be designed to handle a switching device with maximum ratings of 1.2 kV and 50 A.

Table 2.2: Specific requirements for gate driver unit.

Requirement	**Description**	**Value**
Input voltage	Expected digital signal for a ’high’.	5 V
Output voltage	Required output voltage for a ’high’.	15 V
Isolation	Barrier between the power switches (power-side) and control circuits (digital-side). The parameters related to this requirement are given below.	-
Frequency	The maximum operating frequency for which the gate driver device is rated.	1 MHz
Source/sink current	The rated current the device can generate (source) or receive (sink).	-
DESAT	Protection feature in gate driver IC’s. It monitors and detects when the voltage across the switch’s channel rises abnormally high and indicates a fault has occurred, such as a short circuit.	-
CLAMP	Another protection feature. In a half-bridge configuration, it prevents turned-off switches from dynamically turning on.	-
Propagation delay	The total time required for an input signal to propagate to the device’s output.	-
Cost	The cost on the lab for purchasing the device.	-

Notes on the requirements with undefined values:

- The isolation requirement is further characterized by three parameters:

  i. $CMTI$: Common-Mode Transient Immunity. The maximum tolerable slew rate (dv/dt) of the common mode voltage applied between two isolated circuits [8]. High-slew-rate (high-frequency) transients can cause interference across the isolation barrier and lead to breakdown. Measured in V/ms or $kV/\mu s$.

  ii. V_{ISO}: The maximum voltage the isolation barrier can withstand for a short period without breakdown. Measured in V_{RMS}.

  iii. V_{IORM}: The maximum continuous or repetitive peak voltage that can be applied across the isolation barrier during normal operation. Measured in V_{pk}.

  The values for these parameters must at least match the given maximum ratings for the switching devices (1.2 kV, 50 A).

  Furthermore, these parameters are tested and certified under standardized conditions such as IEC 60747-17 (magnetic coupling) and UL 1577 (optical coupling) [9] [10].

- Gate driver devices usually have different ratings for source and sink current. This property is usually a feature given that many switches have higher charging limitations than for discharging; switches can sometimes have their gates discharged (turn off) faster than they can be charged.

- The DESAT protection feature typically uses a high-voltage diode to sense the voltage across the channel (collector-emitter for IGBT’s; drain-source for MOSFET’s). Additionally, a ’blanking’ capacitor is connected in parallel to filter the signal. A comparator detects when the signal rises over a defined threshold, then the controller indicates a fault has occured and the device’s output is pulled down [11].

- For a turned off switch, the CLAMP feature uses a comparator to sense if the device’s output voltage has reached a defined threshold. If so, the IC’s digital controller enables a transistor to sink the output and discharge the gate, thereby preventing an accidental turn-on [11].

- As previously mentioned, selecting the device will require researching the options through the suppliers. The undefined requirements, including the propagation delay and costs, will be defined during this research process. Overall, the device will be selected by comparing the best parameter values, the accessible quantity, and the lowest price.

2.2.3. Output phase

The specific requirements for this part of the design describe the components and their structure rather than the specific values for the required components. This follows from the key requirement that users be able to adjust the Universal Gate Driver to their specific needs.

Moreover, after discussions with the project supervisor, it was agreed upon that the ability to adjust the output phase will be limited to larger physical sizes of the components. The requirement on larger sizes should facilitate manually soldering components onto the PCB.

The required components and structure are illustrated in figure 2.3. The inputs are connected to the output of the gate driver IC, so that the $outputphase$ is referenced to an isolated virtual ground. The output $OUT+$ connects to the gate at the switch and $OUT-$ to the source (MOSFET) or the emitter (IGBT). The function each required component serves is described in table 2.3.

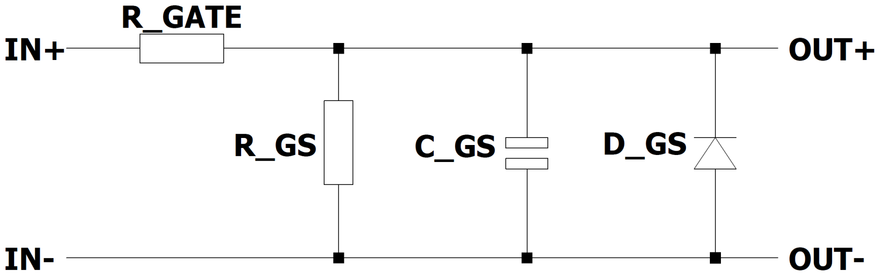


Figure 2.3: Components and structure of the output phase.

Table 2.3: Component definition for output phase.

Requirement	Description
R_{GATE}	Series resistor between the gate driver IC and the switching device limits inrush current, controls switching speed, and mitigates noise or oscillations.
R_{GS}	Pull-down resistor between the gate and source ensures the switch stays off by preventing gate voltage buildup due to noise or parasitic capacitance.
C_{GS}	Bypass capacitor to dampen high-frequency oscillations, especially in high-speed switching applications.
D_{GS}	Diode to clamp the gate voltage and protect the MOSFET/IGBT gate from overvoltage.

The function required by each component in this circuit is further described in section 4.4.

3

Device & Component Selection

3.1. Method

After defining the general and technical requirements, the next step in the design process is the selection of the main devices previously described, i.e. the gate driver IC and the programmable device IC. Using the given requirements to narrow down the search, available devices from different manufacturers and suppliers were considered. This search involved looking through many data sheets and all viable options were compiled onto spreadsheets.

The spreadsheets contain all the required information to weight different features, properties, and costs. In the end, the devices were selected through meetings with the project supervisor. The following sections address the tables and requirements presented in the previous chapter for each chosen device. Additional information on features and the choices made are also described for each device.

Furthermore, the choices on the components used in the Output phase circuit are also described. The choices here were determined by the components relation to the adaptability of the output circuit. This adaptability follows from the requirement that the design be adaptable to different types of gates. All other components and devices required will be further explained in the next chapter, but the full list of components can be found in appendix D.

3.2. CPLD

The first major component that is looked at is the programmable logic device. Three main IC components can be chosen for this: field programmable gate arrays (FPGA), a micro-controller and a complex programmable logic device (CPLD). An overview of the components that were looked at can be seen in appendix B.

Micro-controllers were the first set of devices that were looked at. The device as shown in figure B.3 is one example of many similar micro-controllers. The positive aspect of this option lies in the clock speed, about 500 MHz, which is very high and more than enough to set the required 300 ns dead-time. However, with relatively high cost and propagation delay there are better options to choose from.

FPGAs are considered next. These devices also have high clock speed, but also suffer from relatively high costs. Another problem with FPGAs is the package, it should give easy access to the pins of the IC and should have sufficient space between the pins. All FPGAs that were considered can be seen in figure B.1. In this figure it can be seen that either the costs are high or the package is insufficient. Some FPGAs had a relatively low cost and could be considered, if it weren't for the fact that the CPLDs are cheaper.

CPLDs were the last components to be considered. As stated in section 2.2.1 the structure of the FPGA and CPLD are mostly similar so the time delays and costs are all quite similar. Where the CPLD excels compared to the FPGA is in the package and the fact that for lower clock speeds the CPLD becomes much cheaper than the FPGA. For these reasons a CPLD was chosen as the main IC to focus on. Figure B.2 shows a list of viable CPLDs. All of these CPLDs are able to set a dead-time of 300 ns, have a package that is easy to use when testing, similarly low propagation delays and all work on 5 V. This makes cost the deciding factor in choosing the CPLD. The obvious choice would be CY37032P44-200JXC, however, this CPLD was sold through the "marketplace" on DigiKey which added additional costs. The second choice, ATF1502AS-10AU44, did not have this problem and was therefore chosen as the CPLD that would be used. Only the propagation delay is a bit higher than the other CPLDs, 7.5 ns, this is still low enough as to not cause problems.

Table 3.1 shows the technical specification of the CPLD.

Table 3.1: 1ED3323MC12NXUMA1 Technical Specifications [11]

Requirement	Value
Dead-time	300 ns is achievable with 100 MHz
Programming	Can be done through JTAG in-system programming
Package	44-TQFP
Propagation delay	15 ns
Input Voltage	5 V
Output voltage	5 V

3.3. Gate Driver

The gate driver device was selected out of the spreadsheet shown in B.2. The main family of products considered was the 1ED332xMC12N by Infineon. Initially, the selected device was 1ED3321 but due to availability the 1ED3323 was chosen in the end. With respect to table 2.2, the relevant information for 1ED3323 is shown in table 3.2.

Table 3.2: Specifications of the gate driver 1ED3323MC12N [11]

1ED3323MC12N

Property	Description	Value(s)
Input voltage	Maximum input voltage.	7 V
Output voltage	Maximum output voltage.	40 V
Isolation	CMTI (Common-Mode Transient Immunity) V_{ISO}. Safety certification UL 1577 tested for 1 sec. / 60 sec. V_{IORM}. Reinforced insulation certified by IEC 60747-17.	300 kV/μs 6840 / 5700 V (rms) 1767 V (peak)
Frequency	Maximum rated operating frequency.	1 MHz
Source/sink current	Maximum output current.	-9 / 9 A
DESAT	Required featured included. Here characterized by the maximum fault-to-low delay.	430 ns
CLAMP	Feature included. Here characterized by the threshold voltage and maximum sink current.	2 V / 2 A
Propagation delay	Maximum delay between input and output for turn-on/-off.	84 / 92 ns
Cost	Price per unit of device.	2,69 €

Additional relevant features of the selected gate driver device are:

- **VCC1 and VCC2**

  This gate driver device makes use of two different operating voltages, namely V_{CC1} and V_{CC2}. The input or digital side operates at V_{CC1} while the output or power-side uses V_{CC2}. (Digital and power side are separated by isolation barrier).

- **RDY status signal & UVLO**

  The RDY status is an additional output signal meant to communicate whether the undervoltage lockout (UVLO) is operational. The UVLO independently measures the power supply voltages for input and output sides. If the supply goes below the threshold, the RDY output signal and the device's output are pulled down

- **/FLT signal**

The /FLT signal indicates if DESAT is triggered and a fault has occurred. This is an active-low signal so that if DESAT triggers, the signal is respectively pulled down.

- **/RST signal**

  Active-low input signal. The device's output is low if /RST = Low. If DESAT is triggered and the output is disabled, /RST must be pulled low for a specified time period in order to reset the device (re-enable).

In choosing the device, it was established that the additional output signals RDY and /FLT must be used to control LED's (green and red, respectively). This addition should facilitate evaluating the performance and state of the board.

As previously described, the choice for this specific device was made by weighing the technical requirements as well as the general requirements (stock availability, costs, etc..). In the end, this device was chosen as sufficient for the intended application.

3.4. Output phase

The final step, after having discussed the CPLD and gate driver, are the connections to the transistors in the half-bridge, the gate resistance.

The gate resistance is an important part of the circuit. The switches used in half-bridge configuration are usually transistors. These transistors have many internal parasitic capacitance's and inductance's, which result into undesired behaviour such as [12]:

- Ringing, where oscillations happen due to fast switching behaviour of the transistor.
- Additional noise in the load.
- High current and voltage transient behaviour, which may result damage to the transistor.
- Reverse recovery of the Body-diode, which reduces the switching frequency of the transistors.

Choosing the optimal gate resistance will help reduce these problems. The optimal value of the gate resistance is dependent on the transistors that are connected to the gate driver.

Since the objective is to create a universal gate driver, the gate resistance needs to be adaptable. Two options were considered to make the gate resistance adaptable. First, through-hole pins were considered. These pins serve as places where resistors could be inserted, but this was thought to take too much space in the desired PCB size. Also, the through-hole connection can lead to problems when dealing with high frequencies, making it undesirable. Thus, this option was discarded.

Second, resistors implemented as empty surface-mounted device. The PCB would have spaces where SMD resistor could be mounted on when needed. The package size of the resistor was chosen as the 0805 package, which is big enough to solder. An additional benefit to this implementation is that the resistors could be placed much closer together.

Thus the gate resistance was implemented as a SMD 0805 set of resistors. Section 4.4 goes into further detail regarding the gate resistance.

4

Circuit Design

This chapter outlines the design of the project's primary electronic circuit and PCB. It includes the overall design approach, details of the CPLD used as a dead-time generator, the gate driver circuit, the output phase, and the design choices made throughout the process. An overview of the complete system is also provided to present a comprehensive understanding of the circuit's operation and structure.

4.1. Approach

With the selection of required components addressed in previous chapters, this chapter focuses on the subsequent design process. This includes configuring the devices, integrating components into a functional system, and designing the circuit and PCB.

The circuit and PCB schematics were developed using Altium Designer. Initial schematics were prepared by the bachelor student group, with final adjustments and PCB layout completed by the project supervisor.

The design process involved programming the CPLD to provide sufficient functionality as a dead-time generator, configuring the gate driver with the necessary external components, input-output connections, and peripheral devices, and designing the output phase to ensure reliable signal delivery and power handling to the load.

The following sections provide a detailed explanation of each step, describing how the components were integrated and configured to meet the project's operational requirements.

4.2. CPLD

The hardware description of the logic gates is designed using VHDL (VHSIC Hardware Description Language) and the standard used to transmit the description to the IC unit is JTAG (Joint Test Action Group). Even though the JTAG standard and the description of the programmable logic gates relate to the unit's hardware, this part of the CPLD design is described in section 4.2.1, Software. The external circuitry required to operate the IC device within the overall system is described in section 4.2.2, Hardware. This includes a description of the input and output connections and other features.

The purpose of the VHDL code is to split the incoming PWM signal into two separate signals, to invert one of the signals, and lastly to add a specific dead-time to both the high- and low signal paths.

4.2.1. Software

In order to accomplish these goals a finite-state machine (FSM) is implemented in the VHDL code. The code then operates in the following manner: Initially, after a reset signal has been issued, the FSM is in the 'Reset' state. Once both gate drivers send a RDY signal, the FSM transitions to the 'Count High' state upon receiving the first PWM input signal. From here on out it will keep moving between states in a cyclical manner until either the PWM signal stops or when another reset is issued. These states are: Count High, Hold High, Count Low, and Hold Low. An overview of the FSM of this process can be seen in figure 4.1. below.

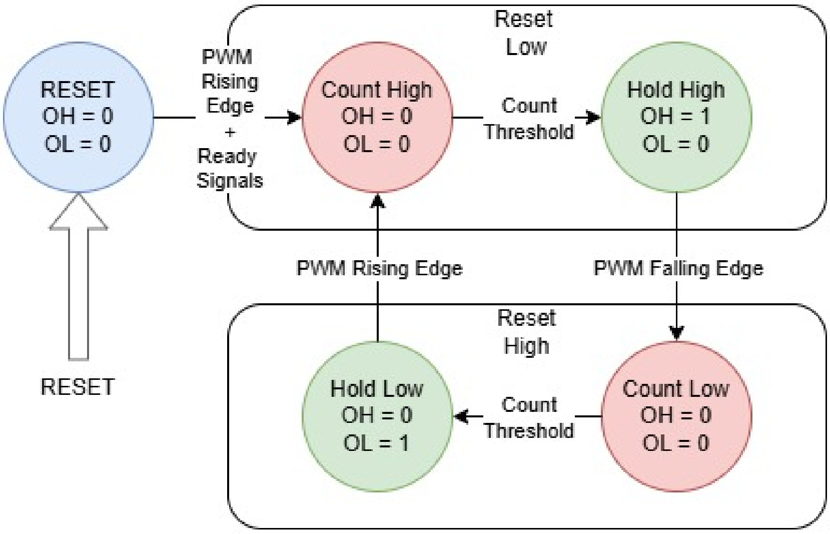


Figure 4.1: States of the FSM.

The Reset state and its conditions for moving into the loop can be seen in the top left of the figure above. Furthermore, the states surrounded by the "Reset Low" box indicate the states during which the low-side output, "OL", is kept low. The states surrounded by the "Reset High" box indicate a similar function for the high-side output, "OH". The transition between these two areas where either the high-side or the low-side is kept low is always dependent on the PWM signal. A rising edge of the PWM signal will move the FSM from the "Reset High" to the "Reset Low" area and vice versa for a falling edge. Within these two areas the count state, indicated in red, maintains a low level output for both outputs.

The count state uses the incoming clock signal to increase the value of count until it reaches a predefined threshold. Once this count threshold is reached the state will move to its respective hold state, shown in green, where either the high- or low-side output will be high.

The FSM will keep moving through the states until either the PWM signal stops or until the reset is enabled. Through this process there is always a minimum amount of time where both outputs are kept low, thus ensuring safe operation at the output of the device.
A simulated output cycle of the FSM can be seen in figure 4.2.

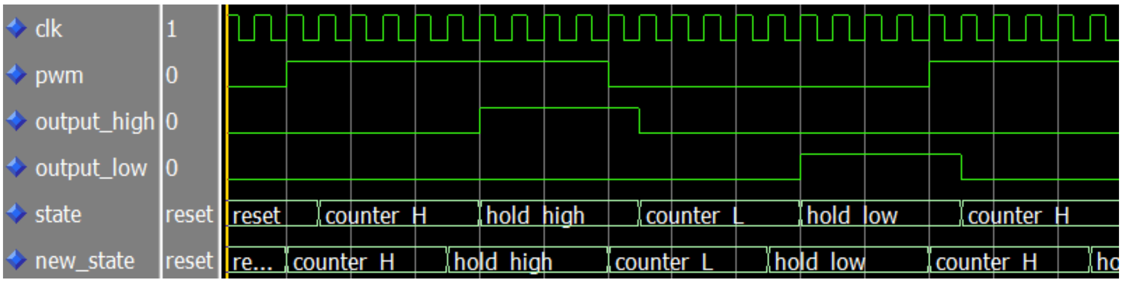


Figure 4.2: Simulated FSM output.

The degree of customization of the dead-time is dependent on the clock speed of the CPLD. For the ATF1502-10AU44 this speed is 20 MHz which means that the code has a step size of 50 ns. As a result, the dead-time can be adjusted to within 25 ns of the intended setting. Should the clock speed of the device be changed through use of a different clock, the step size would then also change accordingly.

A minimum dead-time setting in the code ensures that no invalid values can be entered. This is done to ensure that it isn't possible to accidentally enter a setting which might result in damage to the half-bridge or gate drivers. Another option offered by the code is to turn off the output from the CPLD to one or both of

the gate drivers. Furthermore, several LED's that indicate either a ready or fault situation in either of the gate drivers will be switched on if the code received an input from the gate driver.
The full code with comments can be found in Appendix A.

JTAG

Section 4.2.1 discusses the code that is used to program the dead-time. This code needs to be uploaded onto the CPLD. For this purpose, the CPLD has in-system programming (ISP) based on the IEEE standard 1149.1 interface [13]. This standard gives specifics about testing and programming of a logic devices, such as the chosen CPLD, known as JTAG which is named after the Joint Test Action Group.

The JTAG interface consists of four reserved pins on the CPLD: Test Mode Select (TMS), Test Clock (TCK), Test Data In (TDI) and Test Data Out (TDO) and enables the user to access a few operating modes, most importantly the ISP mode that is used to upload the code to the CPLD.

The written VHDL code cannot simply be uploaded as a VHDL file, the CPLD also needs to know which internals should be connected to which physical pin. Therefore, the VHDL file needs to be transformed into a JEDEC file, named after the Joint Electron Device Engineering Council. The manufacturer of the CPLD, Atmel, provides software to facilitate transforming the VHDL file into the JEDEC. Namely, Pof2Jed, the software that changes a .pof file into a JEDEC file. This .pof can be obtained by using third party software such as quartus II which also requires the pinout of the CPLD to be made. Furthermore, the program required to upload the JEDEC file to the CPLD is called ATMISP. ATMISP works in conjunction with Atmel's programming board ATDH1150USB, which connects a PC's USB port to the JTAG interface of the CPLD and can upload then the JEDEC file[14].

4.2.2. Hardware

CPLD to Gate Driver Connection

The signals used between the CPLD and the gate driver are digital signals. These digital signals use '1' and '0' to indicate signal behaviour, however, this is a simplification and not what happens in reality. In reality the '1' of a digital signal refers to a 'high' voltage level and the '0' refers to a 'low' voltage level, both high and low defined relative to the operating voltage. The physical implementation of the input/output pins (I/O-pin) from a device determine where the voltage threshold lies. The CPLD uses transistor-transistor logic (TTL) for its I/O-pins and the gate driver uses CMOS logic for the I/O-pin. Both of these logic devices have a separate value at which a '1' and '0' can be seen, figure 4.3 shows these different voltage levels for TTL and CMOS logic.

From figure 4.3 it can be it can be seen that the TTL input voltage at a pin needs to be at least a 2 V for a '1', and the voltage input for CMOS needs to be at least 3.5 V. The CMOS voltage thresholds for the gate driver can be defined by equation 4.2 for a '1' and equation 4.1 for a '0' [11].

$$V_{low} = 0.3 * V_{cc} \tag{4.1}$$

$$V_{high} = 0.7 * V_{cc} \tag{4.2}$$

TTL and CMOS voltages have a certain range in which neither a '1' or '0' can be determined. These range are indicated in figure 4.3 by the light gray area. When the voltage level is in this region the signal is ignored and does not get put through.

The voltage level threshold for detecting a '1' by the gate driver created some problems. Namely, the output voltage range of a CPLD I/O pin can be seen in figure 4.3, however, the datasheet for the CPLD gave a typical output voltage of 2.4 V when the current is 4 mA [16]. This means that there is a chance that a high at the output of the CPLD does not equal a high at the input of the gate driver. This is an issue. The output voltage of a TTL device is determined by the current that is going through it, a low current means a high output voltage, which is also implied in the CPLD data sheet [16]. Since the gate driver uses CMOS logic for inputs and it is implied in its datasheet that the resistance of the gate driver is high [11], the current from the CPLD to the gate driver should be sufficiently low, so that the voltage between the two devices is always above 3.5 V when a '1' should be transmitted. However, if we are going by the typical voltage as indicated

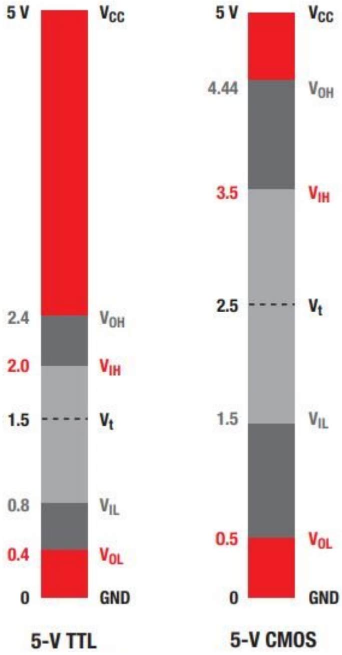


Figure 4.3: TTL voltage levels and CMOS voltage levels [15]

in the CPLD data sheet, the output voltage is 2.4 V which would be problematic since this would mean that ’1’ of the CPLD should not instill a ’1’ in the gate driver. Making the assumption that these two devices will always match up could therefore not be made. Moreover, making a wrong assumption about the voltages between the CPLD and gate driver, and not having the components to test the voltages, could render the design unusable. Therefore, a possible solution had to be found in case the assumption, that the voltages would match up correctly, was false and the voltages would not match.

Connecting Options
Three options were considered to ensure the voltages for high would match.

First, the power supply voltage of the gate driver could be lowered to 3.3 V. The new high threshold, using equation 4.2, would then be 2.31 V, this would guarantee that a high at the output of the CPLD would instill a high at the input of the gate driver. However, a step down would be needed between the power supply of the PCB and the power supply to the gate driver. This also means that there would be 3.3 V, 5 V and 15 V all on the same PCB and close together which can create a lot of problems. Therefore, this option was discarded.

Second, a new CPLD or gate driver could be found. The gate driver was already the best option that could be found based on price and functionality, replacing it would only lead to worse or a more expensive option, neither of which are desirable. Replacing the CPLD with a different one was also not an option due to the output voltage always having a range between 2.4 V and the supply voltage. This range was also the standard in different option we had for CPLD’s [17], [18]. So, a replacement of the component with a different one lead to the same problems.

This leads to the third option, placing a TTL-compatible buffer between the chosen CPLD and gate driver. This buffer can measures a ’high’ at the input for 2 V or higher and gives a voltage output closer to the supply voltage (5 V), thus bridging the gap between 2.4 and 3.5 V. It does not significantly impact the input signal except for a relatively small delay of 6 ns, which is acceptable [19]. Thus, a buffer was included in the design to address the difference between the logic families used by the devices.

Buffer Choice
The chosen buffer must hold to the requirements as stated stated above. The specifications for the chosen

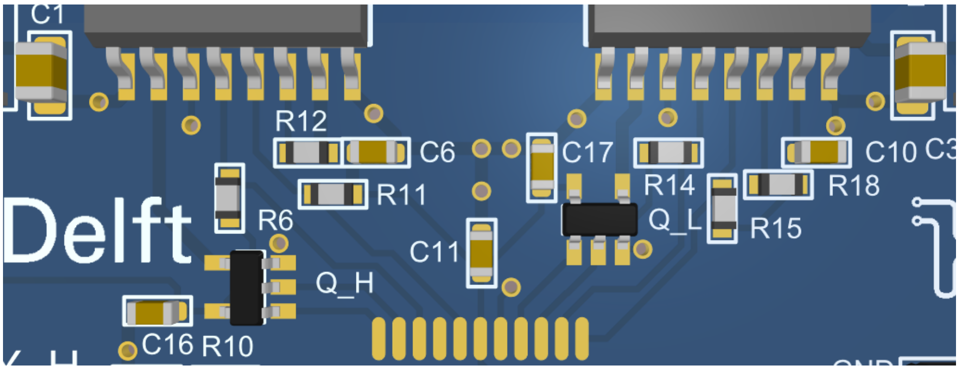


Figure 4.4: PCB design implementation of buffer.

buffer, CAHCT1G125QDBVRQ1, can be seen in table 4.1.

Table 4.1: Specifications of CAHCT1G125QDBVRQ1 [19]

CAHCT1G125QDBVRQ1

Requirement	**Value**
Voltage Supply	5 V
Input Voltage High	2 V
Output Voltage	5 V
Propagation delay	6 ns
Maximum Frequency	1 MHz
Maximum Output Current	8 mA

Table 4.1 shows that the input voltage to register a 'high' is 2 V. With the CPLD typical output voltage being 2.4 V, the buffer should properly register a 'high' input signal. The buffer raises the output voltage closer to the operating voltage (5 V), which is sufficient to register a 'high' in the gate driver. The buffer's voltage compatibility, 1 MHz maximum operating frequency, and low propagation delay make it suitable for connecting the CPLD to the gate driver.

The chosen buffer further includes an enable signal, namely /OE or Output Enable. This signal will be controlled by the CPLD's VHDL code.

The PCB's implementation of the buffer is shown in the figure below 4.4.

User Components

The final components added to the design were to increase the ease of use of the design. This consists of adding LED's which light up when certain things happen in the circuit, these LED's light up when:

- The gate driver detects a DESAT problem, and sets the fault pin to low. The colour of this LED is red.
- When both 5 and 15 V are connected to the gate driver it gives a ready signal, this ready signal is used to turn on corresponding green LED's.
- Each input for the PWM also has an LED that is turned on through the code when the PWM is set to any one of the input pins.

4.2.3. CPLD Design Overview

Figure 4.6 shows the full schematic of the CPLD, it shows the implemented LED's for fault and ready at the top, the input LED's at the bottom. The input and its connection to the CPLD can be seen on the left side of the schematic. The JTAG connection can be seen at the top.

Figure 4.5 shows the PCB implementation of the CPLD. Where the bottom pins are the input pins and the pins on the right are the JTAG pins. Furthermore, the positions of the LED's can be seen as well as the colour of each LED.

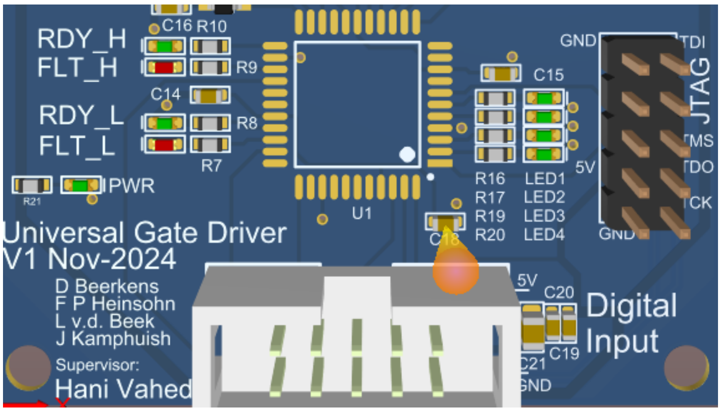


Figure 4.5: The Complete PCB Implementation of the CPLD Component

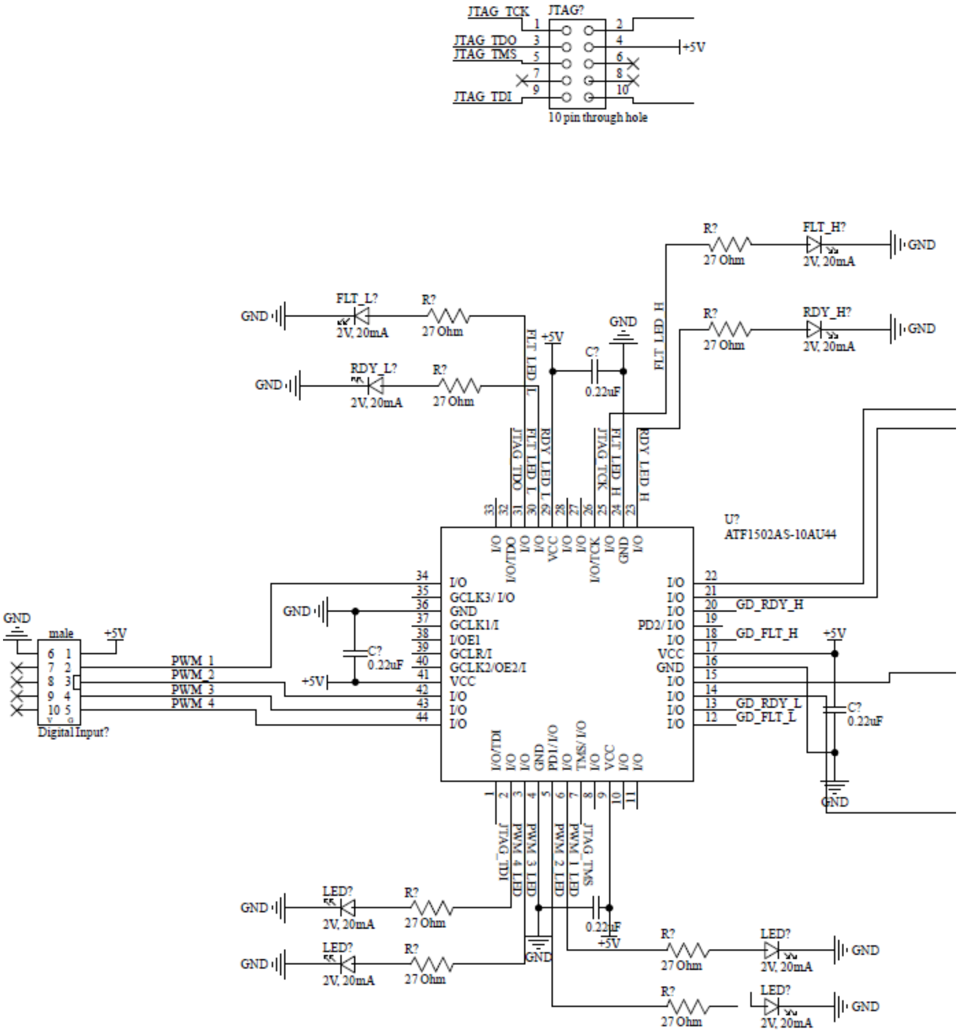


Figure 4.6: Design Schematic of the CPLD

4.3. Gate Driver

This section covers the design and configuration of the gate driver circuit, which is responsible for controlling the power switching devices. This section details the choice of gate driver components, the necessary external circuitry, and the interface with the CPLD and Output phase.

4.3.1. Recommended Application

The gate driver IC unit was designed into the circuit according to the manufacturer's recommended layout for unipolar applications, shown in figure 7, page 13 of the datasheet[11]. This includes pull-up resistors for the RDY and /FLT output signals, and bypass capacitors for the supply at both input and output sides.

Furthermore, the manufacturer specifies a recommended circuit layout to make use of both the DESAT and CLAMP features.

The DESAT pin of the IC should (a) connect to the virtual ground through a blanking capacitor in parallel to a Zener diode, and (b) connect to the drain/collector through a high-voltage diode in series to a resistor. The diode chosen for this project's application is thus rated for 1.2 kV and 1 A, which follows from the given maximum ratings for power switches (1.2 kV, 50 A).

The CLAMP pin should connect to the node leading to switch's gate, after the series resistance R_{GATE} in the Output phase (OUT+ node in figure 2.3).

The Altium Design implementation of this recommended circuit layout is shown in figure 4.7. The device's input and output are shown by the pins IN+ and OUT, respectively. The connection to the input signal is labeled as PWM_L and it originates from the low-side buffer. The output is connected to GD_OUT_L, or the low-side Output phase.

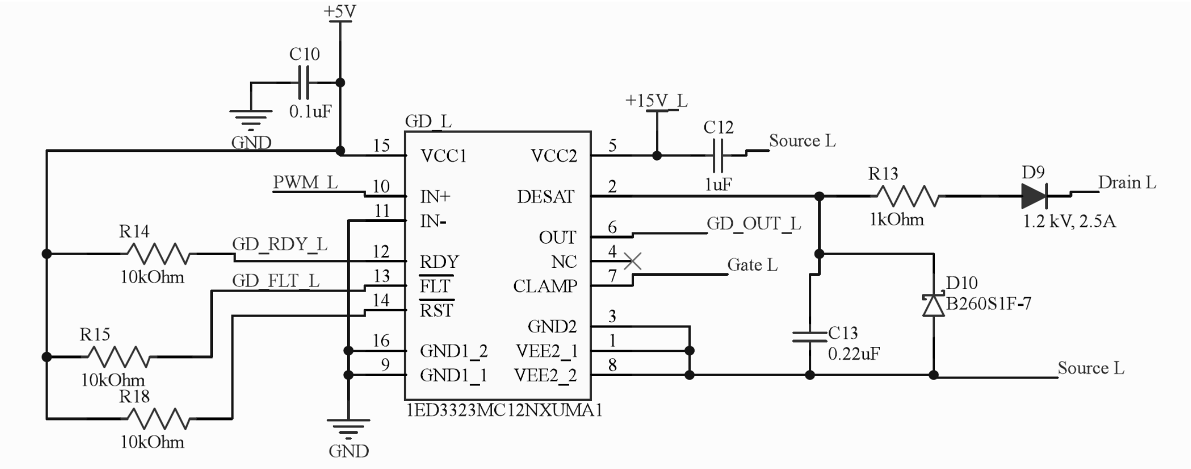


Figure 4.7: Altium Design schematic of gate driver circuit (low-side).

The input pin IN- is for inverting input signals. This device offers the possibility of using an input signal with an inverted logic, such that the transistor's gate is high for IN- equals low. The intended use of the device does not require the use of this feature, thus the IN- pin is connected to digital-side's common ground.

The pull-up resistors and their resistance values can be seen connected to the output pins RDY and $\overline{\text{FLT}}$. These two outputs pins are also connected to the CPLD to control the LED's, as previously described.

The $\overline{\text{RST}}$ pin is also connected to a pull-up resistor. During the design process it was established by the project supervisor that it would not be necessary to allow the user to control this signal. Control over this signal is required in case the DESAT protection features is triggered; by setting this signal to 'low' for the required time, the device is reset and can be operated again. In order to do so, the user would have to be provided a button or a similar manual input mechanism, and this was decided to not be necessary by the project supervisor. With $\overline{\text{RST}}$ permanently pulled up, if a DESAT event occurs the PCB's power supply needs to be disconnected to reset the gate driver's output.

The DESAT and CLAMP pins can be seen connected to the recommended configuration of external components, and ultimately connected to the switch's pins, in this case $Drain\ L$, $Source\ L$, and $Gate\ L$ in terms of a power MOSFET.

4.3.2. DC-DC Converter

As previously discussed, this device has separate power supply inputs for the digital- and power-side. These power supply inputs are shown by VCC1 and VCC2 in figure 4.7. Moreover, given as a requirement for this device in 2.2 is that the input- or digital-side must operate at 5 V and that the output-side operate at 15 V.

Therefore, it was decided to use an isolated DC-DC converter. This converter can then be connected to the digital-side's common power supply of 5 V and in turn generate the required 15 V for the power-side. Given that this device also bridges the digital- and output-side, an isolation barrier is also required between the device's individual inputs and outputs.

The chosen converter is the TBA1-0513E by Traco Power. The manufacturer's datasheet also provides the recommended bypass capacitors to attach to either side of the isolation barrier[20]. The converter and the bypass capacitors can be observed in the Altium schematic shown in figure 4.8.

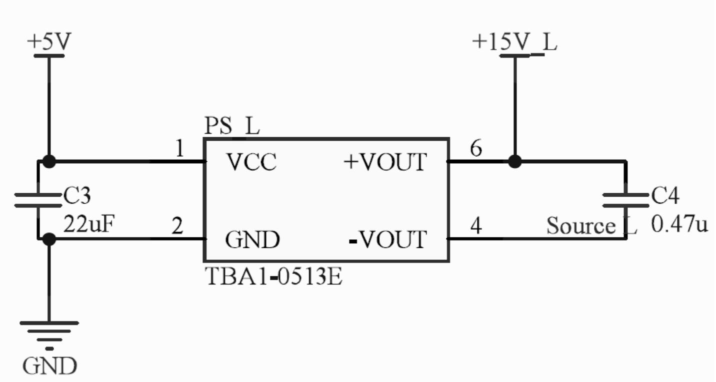


Figure 4.8: Altium Design schematic of DC-DC converter (low-side).

Table 4.2: Specifications of TBA1-0513E [20]

TBA1-0513E

Requirement	Value
Input Voltage	5 V
Output Voltage	15 V
Output Current	66 mA
Isolation (I/O tested for 60 s.)	1.5 kV

The input pin of the converter (VCC) is connected to the input bypass capacitor and the board's power rail, the same can be seen for the VCC1 input pin connected to it's respective bypass capacitor and the same $+5V$. Similarly, the output pin of the converter (+VOUT) is connected to it's bypass capacitor, and the line $+15V\_L$ provides the required operating voltage for the output-side of the gate driver IC, or VCC2. The output is also referenced to the isolated virtual ground, shown in figure 4.8 by OUT- connected to $Source\ L$.

4.3.3. Gate Driver Design Overview

The circuit design is shown in the schematic of figure 4.7. The PCB implementation of this circuit is shown below in figure, where both high-side and low-side gate drivers can be seen. These are shown by GD_H and

GD_L. Furthermore the DC-DC converters are shown by the black rectangular units shown at the sides of the picture.

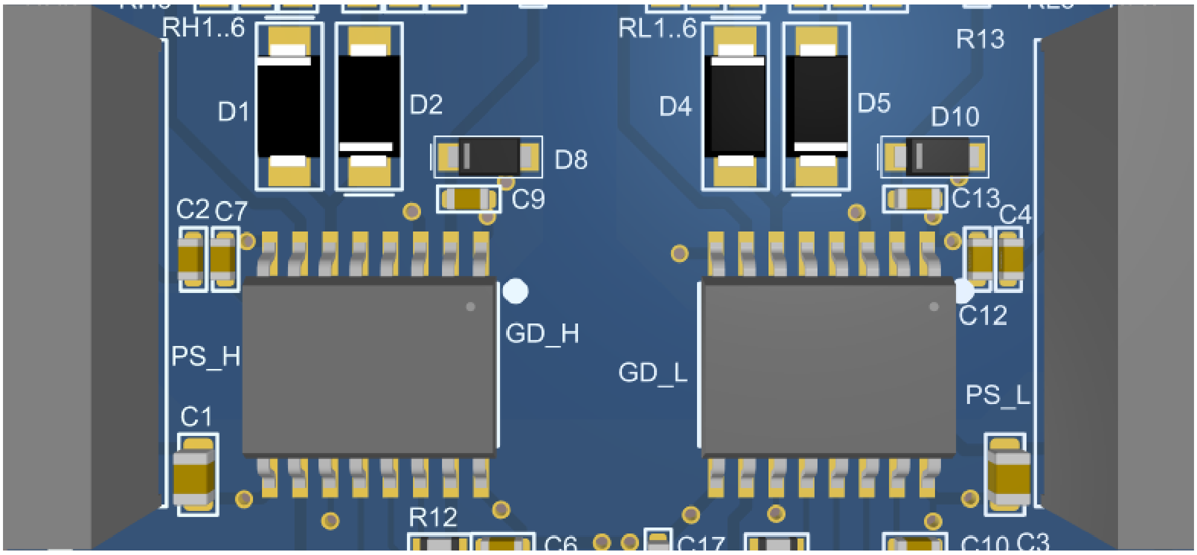


Figure 4.9: PCB Implementation of the gate driver IC units.

4.4. Output Phase

This section covers the circuit following the gate driver output, including the gate resistance and its implementation, as well as the components used for gate-to-source protection. These components include the pull-down resistor, bypass capacitor, and clamp diode.

4.4.1. Gate Resistance

The gate resistance is an important part of the circuit as it regulates the amount of current that flows from the gate driver to the transistor. Current flows to the transistor and turns on the transistor when the gate charge is sufficiently high. When the transistor turns off, the charge on the gate flows back through the gate resistor towards the gate driver. The turn-on and turn-off times for transistors are different and can be influenced by the gate resistance.

Therefore, two different paths are included in the design, using an on-resistor (R_{on}) for the turn-on of the transistor and using an off-resistor (R_{off}) for the turn-off of the transistor. These two paths can be seen in figure 4.10. The benefit of having two paths lies in the ability to regulate the turn-off and turn-on times more precisely and thereby optimizing the switching behaviour of the transistors regardless of the type of transistor that is used, making this implementation better for a universal gate driver. These paths are formed by placing diodes connected in series with R_{on} and R_{off}, oriented in opposite directions.

The final decision to be made has to be done regarding the size of the resistors. The design will be implemented on a PCB, thus the components will be relatively small (millimeter range). The size that was chosen for these R_{on} and R_{off} resistors was the 0805 package (1.25 mm by 2 mm) [21], this is large enough to easily replace the gate resistors. Also, the decision was made to put three resistor in parallel instead of just one resistor for each path, this creates a wider range of resistance values without having the a lot of specific resistor values on hand.

4.4.2. Protection and Optimization

After discussing the CPLD, the gate driver, their connection, and the gate resistance, it is also important to look at what it all connects to, the half-bridge. As stated before, the half-bridge consists of two transistors that act as switches are turned on by the PWM input signal. The simplification that is often made when talking about switches in design is that the switch turns on and off instantaneously. In reality, switching takes a small amount of time. This time stems from internal transistor capacitance [12]. Especially the gate capacitance requires some attention. This capacitance results in a couple problems that need to be dealt with.

Pull-down Resistor

The gate capacitance needs to be charged to turn on and allow current to flow. Conversely, immediately when the gate has to turn off, this capacitance holds the charge that needs to be released. To help the capacitor discharge faster a resistor can be connected between the gate and source terminals of the transistor. This resistor draws current from the gate and allows it to float to (virtual) ground, decreasing the turn-off speed and increasing the switching frequency [22].

Clamp Diode

The fact that the gate capacitance holds charge means that it can also get trapped temporarily. This build-up of charge could remain inside the transistor, and, if the internal charge difference is big enough, the increase in voltage inside the transistor could destroy the gate. This phenomenon is called electrostatic discharge (ESD). To protect the transistor from ESD, a diode is included between the gate and source terminals of the transistor and parallel to the aforementioned resistor. The breakdown voltage of the diode is set to such a point that if the voltage rises, the diode lets current go through and clamps to voltage below the point where ESD damages the transistor[23].

Bypass Capacitor

Finally, the time it takes for the gate capacitance to fully charge could be slower than a transient response at the gate or source terminals. This sudden change of voltage could lead to the transistor turning on even when that is unwanted. To prevent the consequences of this transient response from happening a small capacitor is added between the gate and source terminals of the transistor and parallel to the previously mentioned resistor and diode. For fast changes in voltage this capacitor will act as a small resistance between the gate and the source, which keeps the gate voltage and source voltage close enough together for the transistor to not turn on. As the transient response dies down, the resistor takes over and the transistor can operate as intended [24], [25].

These three components are meant for protections and optimizations, their values depend on the transistors that are connected therefore they should be big enough to be changed easily. For that reason, the size of resistor and capacitor are the 0805 size package and diode has size DO-214AC (SMA) package (2.4 mm by 3.99 mm) [26].

4.4.3. Output Phase Design Overview

Figure 4.13 shows the schematic view of the output phase. It includes the R_{on} and R_{off} paths and the switch resistor, capacitor and diode as well as an added output LED.

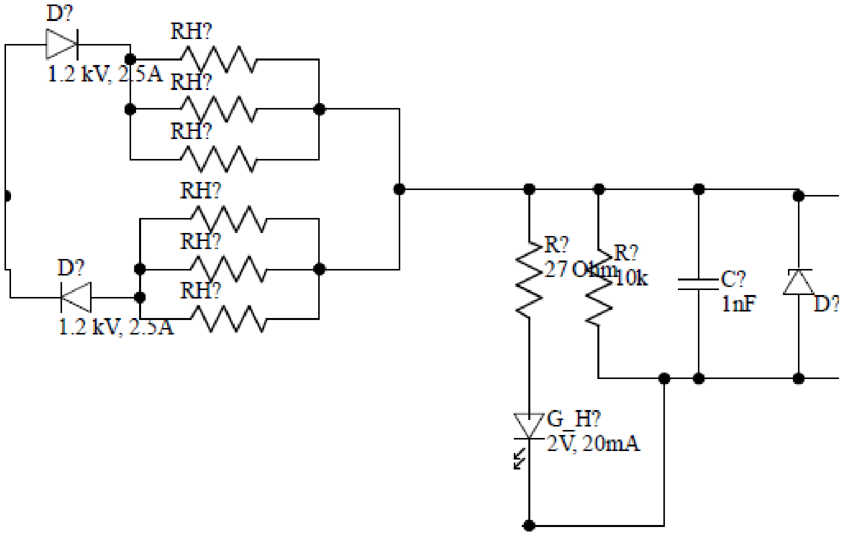


Figure 4.10: Output Phase of the Low-side

Figure 4.11 shows the the PCB implementation of the output phase. Where the 4 diodes at the bottom each connect to 3 parallel resistors which are the R_{on} and R_{off} implementations, which is the same as shown in figure 4.13.

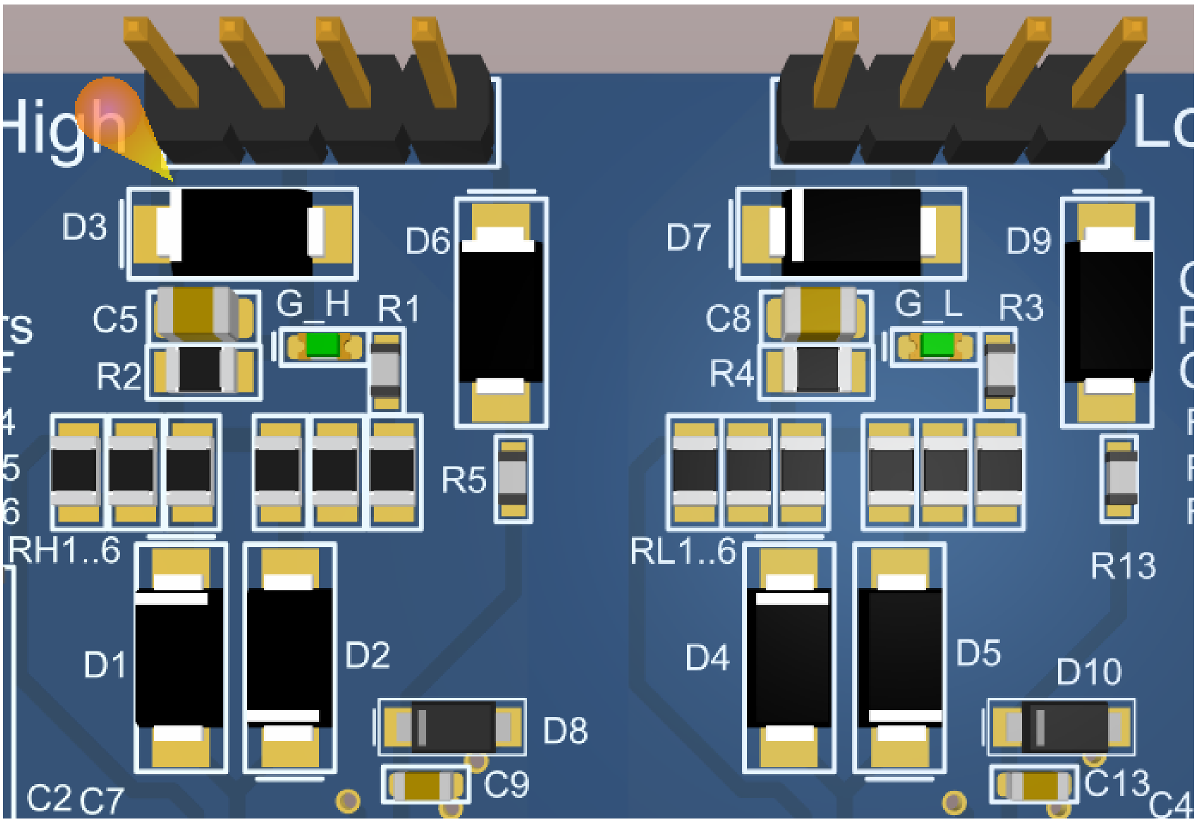


Figure 4.11: Implementation of the Output Phase on the PCB

4.5. Design Overview

4.5.1. Circuit Schematic & PCB

Figure 4.12 shows the PCB layout of the design. With the outputs at the top, the gate driver and the DC-DC converter in the middle and the CPLD at the bottom. As can be seen in the figure, the outputs are put on two sets of 4x1 pins, where the third pin from the left is left empty to create space between the source and drain connections.

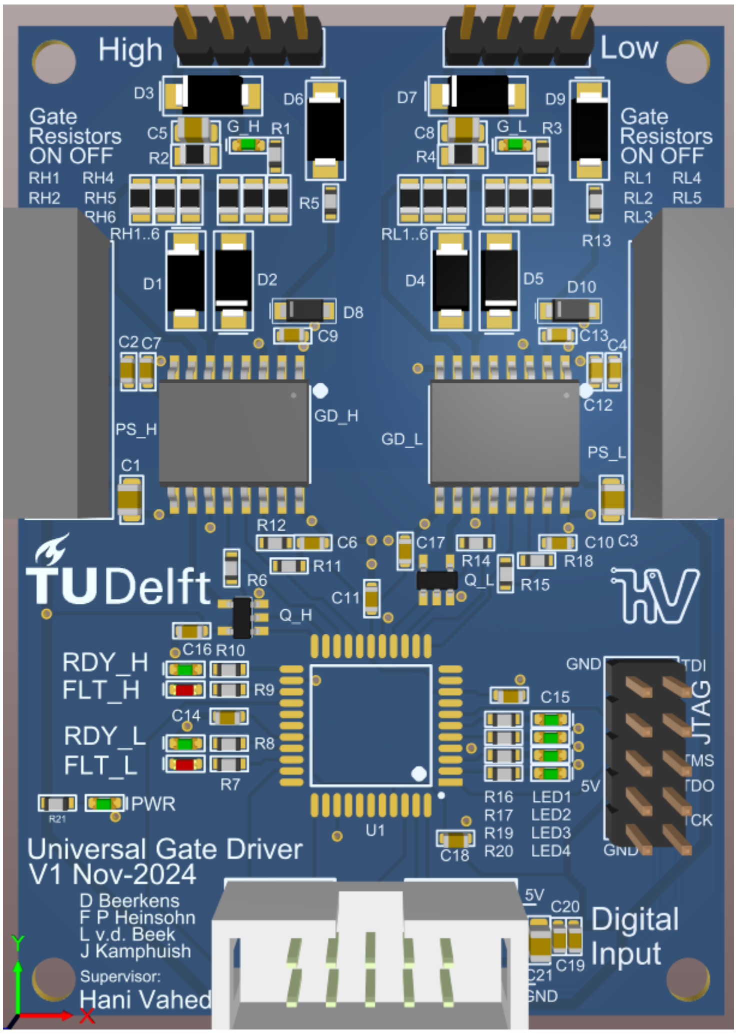


Figure 4.12: Full Design PCB

Figure 4.13 shows the full schematic of the design and a simplified version of the connections.

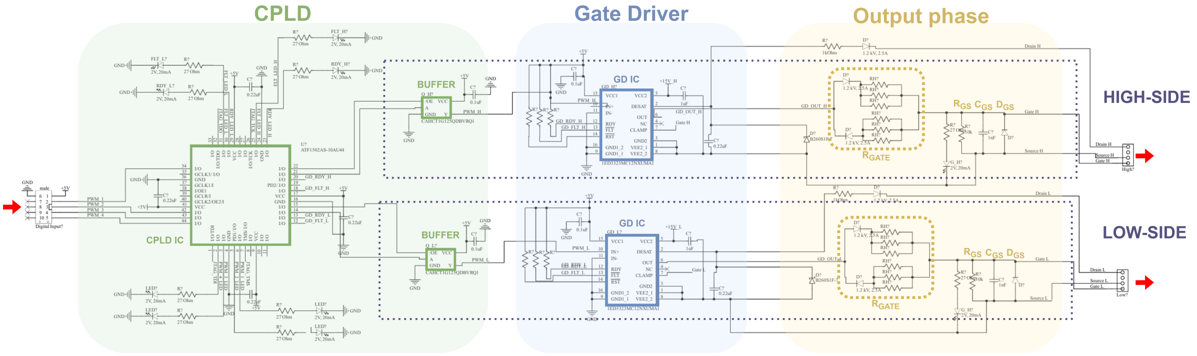


Figure 4.13: Full Design Schematic

4.5.2. Circuit Simulations

In order to get an overview of the effects of each component that was added to the output stage of the overall device, the design is simulated using LTspice. The design can be seen in the image below.

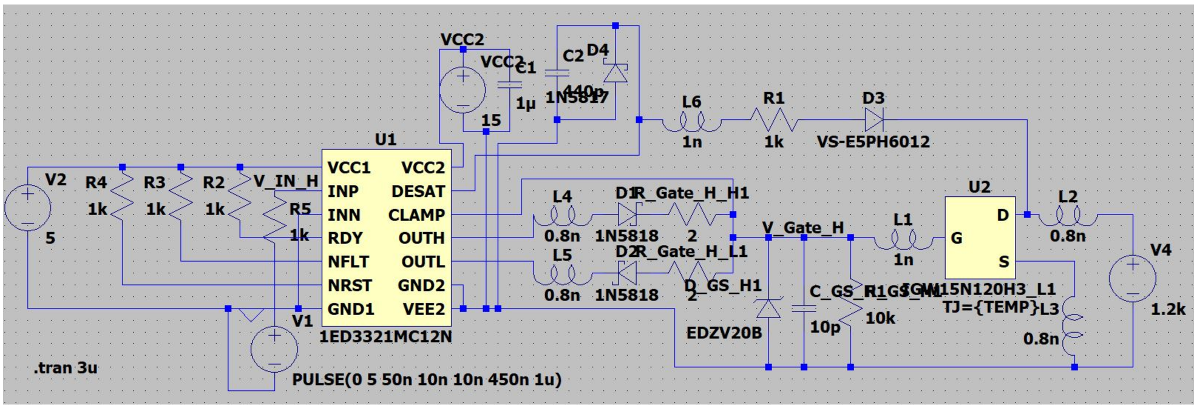


Figure 4.14: Design of single branch

This design only encompasses a single branch, in this case the high branch, of the two branches. An identical design, albeit with a different input for the PWM signal, is used for the low branch of the design. The model that was used for the gate driver is slightly different from the one that will be used for the physical design. This is due to the fact that there was no model of the exact component available through the Infineon website. After contacting Infineon regarding the availability of such a model they suggested the use of this model instead since it belongs to the same family of gate drivers and behaves in a similar manner to the gate driver that has been chosen for the physical design.

Using this design the effect of several components on the device's output are examined. Furthermore the expected behavior of some components of the gate drivers such as the low VCC cut-off function and the propagation delay have been tested as well. The results of these tests can be seen in the images below.

In the first image, image F.15, the relation between the rise time of the voltage at the gate of the MOSFET and the value of the gate resistors can be seen. For the higher branch the value of the gate resistors is set to 0.5 Ω , whereas for the lower branch the gate resistors are set to 20 Ω. The resulting graph clearly shows that the rise time differs significantly between both signals.

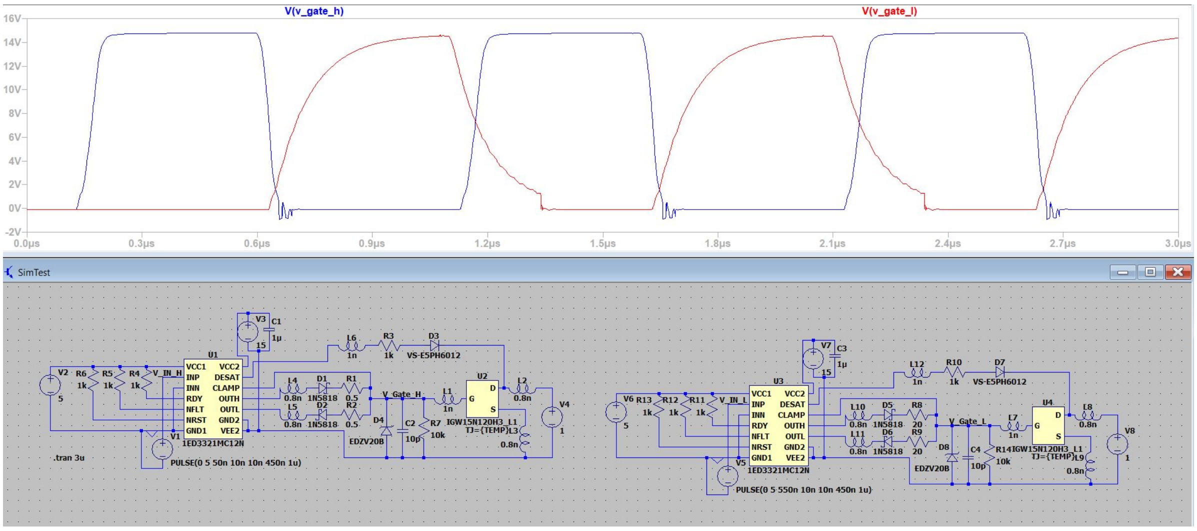


Figure 4.15: Output Gate Voltage With Different Gate Resistance Settings

The image below, image F.16, shows the difference in current running through the gate resistors when they are set to 20Ω and 0.5Ω respectively.

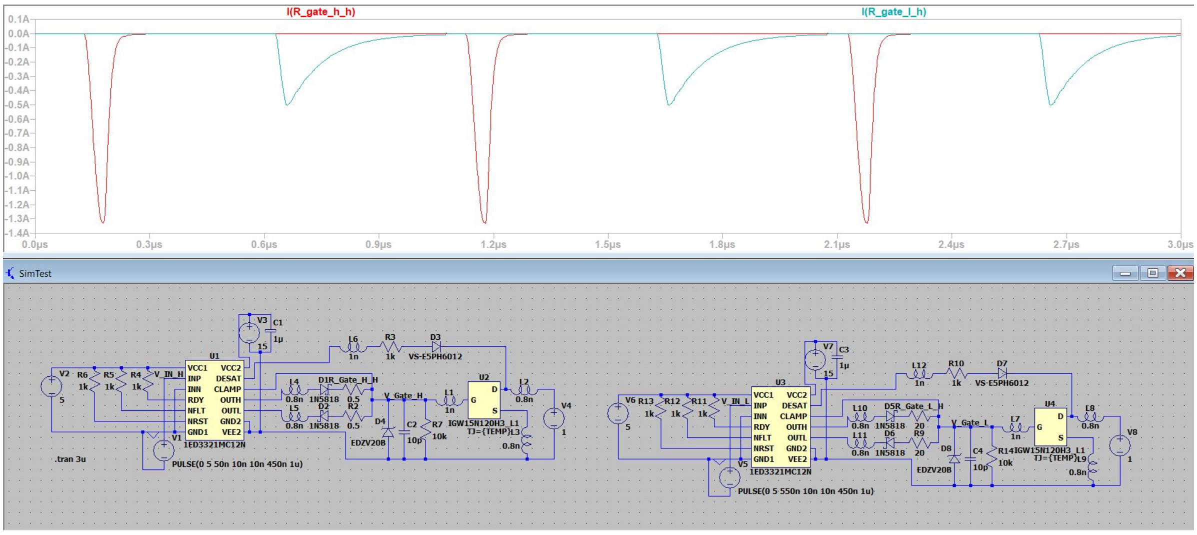


Figure 4.16: Current through gate resistors with different resistance settings

The next image shows the low VCC cut-off behaviour as well as the cut-off behaviour of the gate to source diode, which ensures that the gate voltage operates within the desired voltage range.

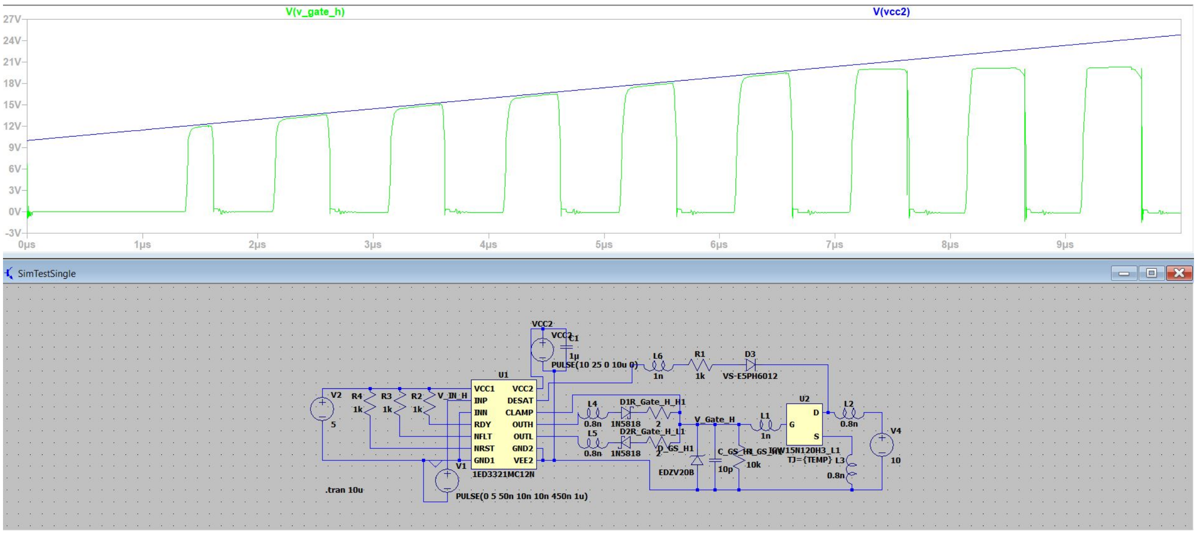


Figure 4.17: Low VCC cut-off + cut-off diode behaviour.

These simulation tests were made to study the behavior of the system and to decide upon the range of values for the design's components. Ultimately the goal is to achieve a similar output as the one that can be seen in figure 4.18 below. Here the outgoing signal is split in two, V-gate-h and V-gate-l, and there is a set amount of dead-time between the two outputs.

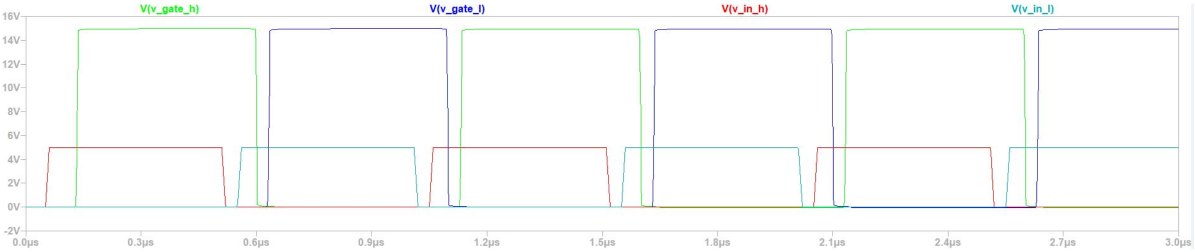


Figure 4.18: LTspice simulation of input signals at gate drivers and their corresponding output signals with added dead-time.

5

Prototype Evaluation & Result Analysis

This chapter discusses the results that were obtained whilst working with the universal gate driver prototype. First, the general throughput of the CPLD and gate driver was tested and is discussed. Second, the need for the buffer is discussed. Finally, the programmed dead-time is shown and the results are discussed based on whether the requirements are met.

5.1. General Tests

This section discusses the general tests that were done to check the functionality of the CPLD and gate driver as well as checking if the buffers between the CPLD and gate drivers are necessary.

5.1.1. Signal Throughput

Figure 5.1 shows the PCB with the LED's on.

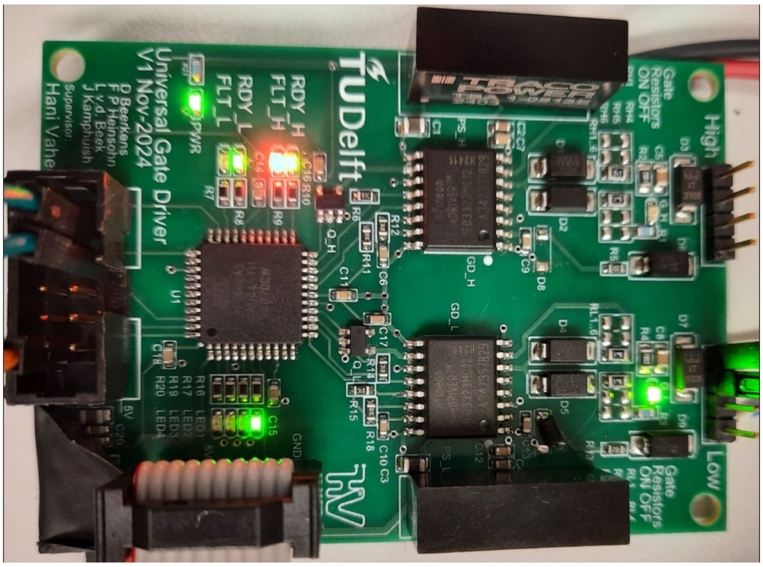


Figure 5.1: PCB with working LED, showing functionality of the circuit.

The top-left LED is the power LED, this indicates whether the power supply is properly working on the board.
The LED's on the right indicate some states of the FSM that was discussed in section 4.2.1, meaning that the code was uploaded successfully.

The LED's on the left indicate two things, first, both green LED's show that the gate driver gives a 'ready' signal, this means that both the 5 V and 15 V sides have sufficient power and that the gate driver can be used.

Which follows from the previously mentioned built-in feature of the gate driver, UVLO.

The red LED indicates a 'fault' from the gate driver, this indicates that a DESAT event has triggered. In this case the FLT LED for the high-side is on since there is not a connection between the source and gate outputs while 15 V is being held at the output of the gate driver. The fault LED for the low-side is off since there is a connection between the source and gate outputs and there is no DESAT problem.

Furthermore, the output LED is on, this shows that the CPLD gives a signal to the gate driver and the gate driver can correctly output the signal. Part of this signal can be seen in figure 5.2.

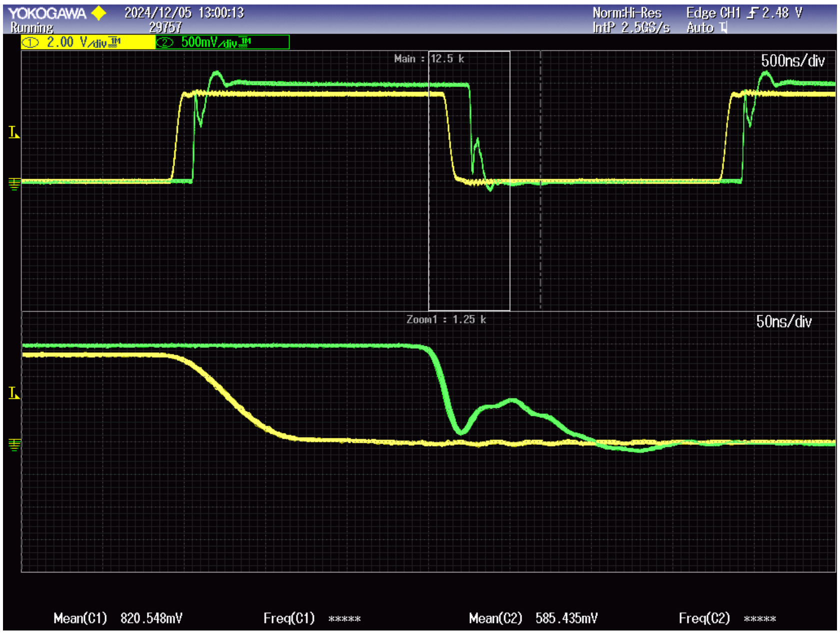


Figure 5.2: 10 kHz input signal (yellow) and Output (green) of a PWM signal (measured with 10x probe). The bottom shows a zoomed in part of the signal. R_{off} = 5.1 Ω

Figure 5.2 shows the input signal (yellow) and the output signal (green). It can be seen that the green signal has a delay of about 150 ns on the PWM. This delay stems from the internal delays of the CPLD, buffer, gate driver and from the long wires that are used during testing. Furthermore, during turn off, the output signal has a small bump, this bump is undesirable since it could give problems like cross-capacitance between switches, transient issues and, if the voltage rises high enough, it could lead to the transistor turning itself on again. What is expected at turn-on and turn-off is a small overshoot and oscillation called 'ringing', which was also briefly mentioned in 3.4. This means that the size of R_{off} plays a role in controlling the size of the bump.

The size of R_{off} that was used initially was 5 Ω, see figure 5.2.

Figure 5.3 shows the same input but with a R_{off} of 2.5 Ω, this was done by adding a second 5 Ω resistor to the turn-off path of the gate resistance, see figure 4.11. It can be seen that there is still a bump present but now there are small oscillation, meaning that there is ringing at turn-off, and thus a faster turn-off time.

Figure 5.4 shows the same set up as the previous figures but now R_{off} was set to 1.67 Ω. It again shows the small oscillations due to ringing, but now it stabilizes quicker. However, the bump still has not become smaller even though the gate resistance has become 3 times as small. This would mean that there is either noise in the system or the bypass diode has a wrong value for the used transistor. So, the final change made to the PCB was changing the bypass capacitor to 100pF, a higher value would also have been used but it was not possible to obtain a larger valued capacitor in the correct package, within the limited time left.

Figure 5.5, shows this final output simulation. And again, more oscillation, yet, the bump is still in at the

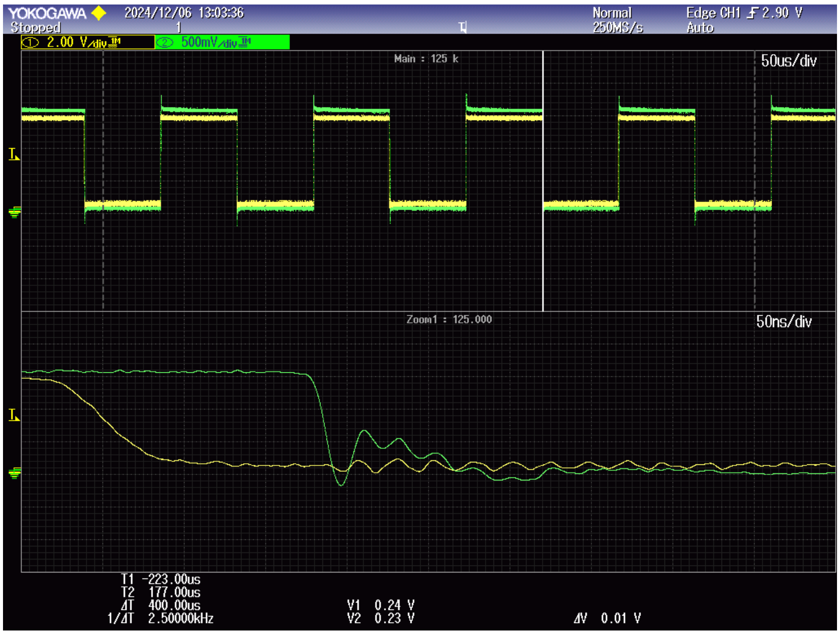


Figure 5.3: 10 kHz input signal (yellow), output signal (green) measured with 10x probe. R_{off} = 2.5 Ω

output. This leads us to suspect that noise is the main culprit in creating it.

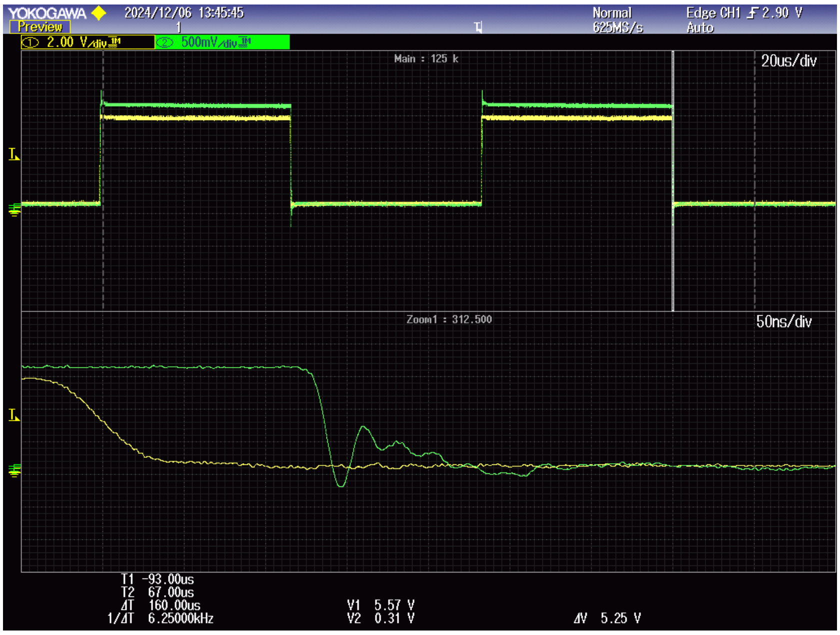


Figure 5.4: 10 kHz input signal (yellow), output signal (green) measured with 10x probe. R_{off} = 1.67 Ω

5.1.2. Buffer Test

In section 4.2.2 the doubt surrounding the compatibility of the CPLD and gate driver is discussed. To test if the buffers were necessary two PCB's were made, one with the buffer and one with the location of the buffer shorted, see figure 5.6.

Using two probes and an oscilloscope, the voltage levels at the input and output of the buffer were tested, the result can be seen in figure 5.7.

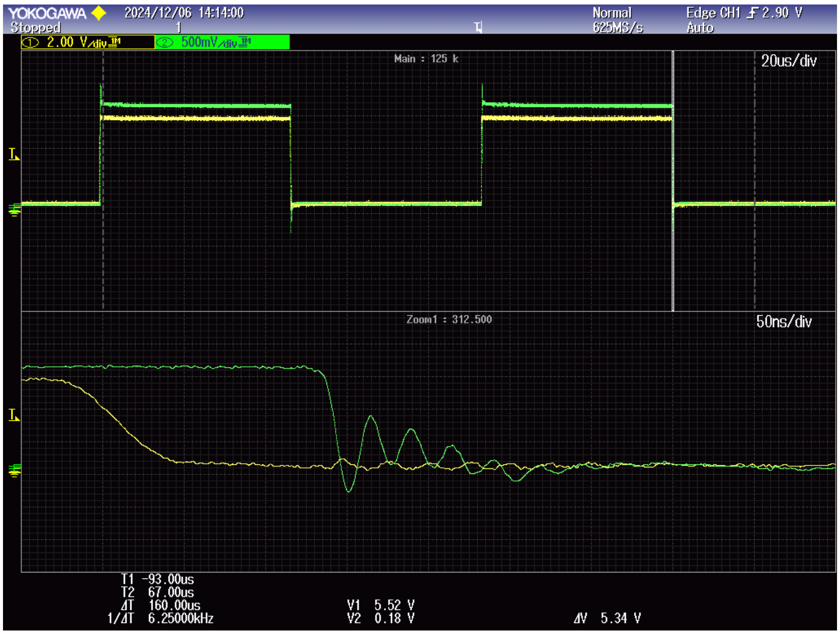


Figure 5.5: 10 kHz input signal (yellow), output signal (green) measured with 10x probe. R_{off} = 1.67 Ω and C_{bypass} = 100 pF

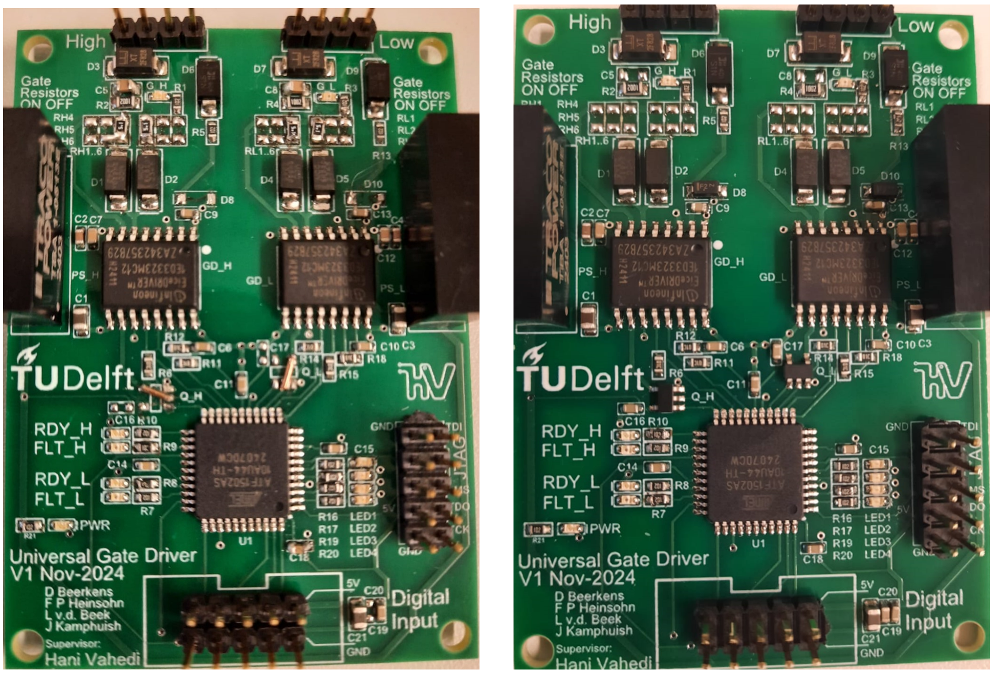


Figure 5.6: Left: PCB without a buffer between CPLD and gate driver. Right: PCB with a buffer between CPLD and gate driver.

From the left image of figure 5.7 it can be seen that the voltage level at the input is around 4.2 V and the output voltage is close to 5 V. The right image shows the voltage to be around 4 V in the situation where there is a direct connection between the CPLD and gate driver. This shows that the buffers between the CPLD and gate driver are not necessary. When looking only at the voltage levels.

Figure 5.8, shows the same input and output signal equal to figure 5.3 but now on the PCB that does not have a buffer.

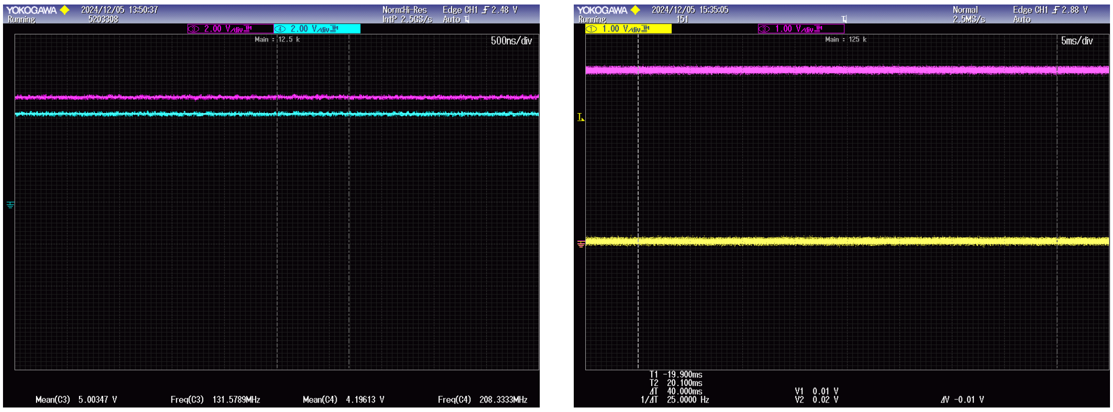


Figure 5.7: Left: Voltage at input of buffer (blue) and output of buffer (purple). Right: Voltage level without buffer (purple) and a 0 V reference (yellow).

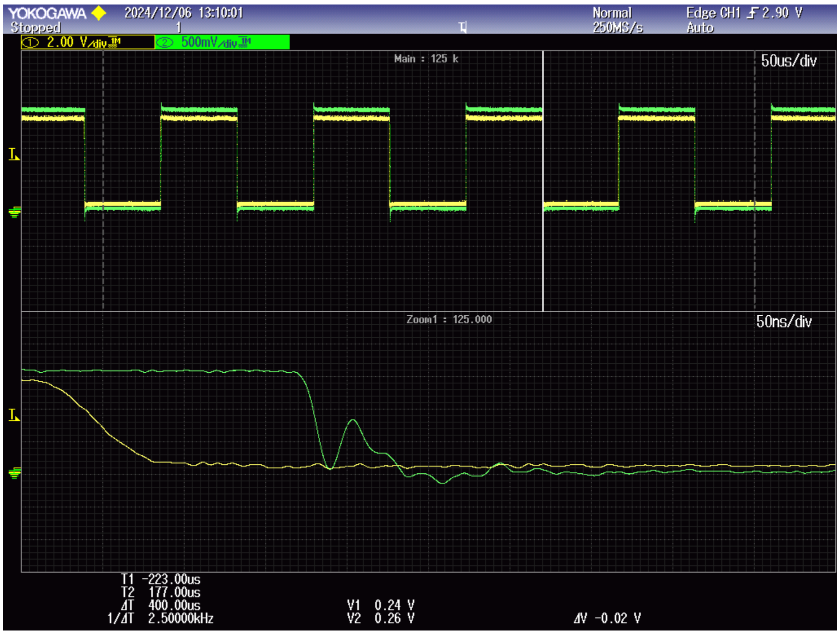


Figure 5.8: 10 kHz input signal (yellow), output signal (green) measured with 10x probe. $R_{off} = 2.5\,\Omega$. Without buffer.

There is only a small difference in the oscillation and there is a bigger bump. This change between a circuit with and without a buffer is small enough that the conclusion can be drawn that the buffers are not needed on the PCB and can be removed to reduce the price as per the requirements.

5.2. Functionality Based on Requirements

In this section the tests that were done to show the functionality based on the initial requirements will be discussed.

The first objective of the device is to be able to add an adjustable dead-time to the incoming PWM signal. In figure 5.9 the incoming PWM signal can be seen in yellow, and the two outgoing signals can be seen in green and purple. There is a clear, in this case exaggerated, amount of dead-time where both output signals are kept low, thus indicating a successful application of dead-time to the incoming signal.

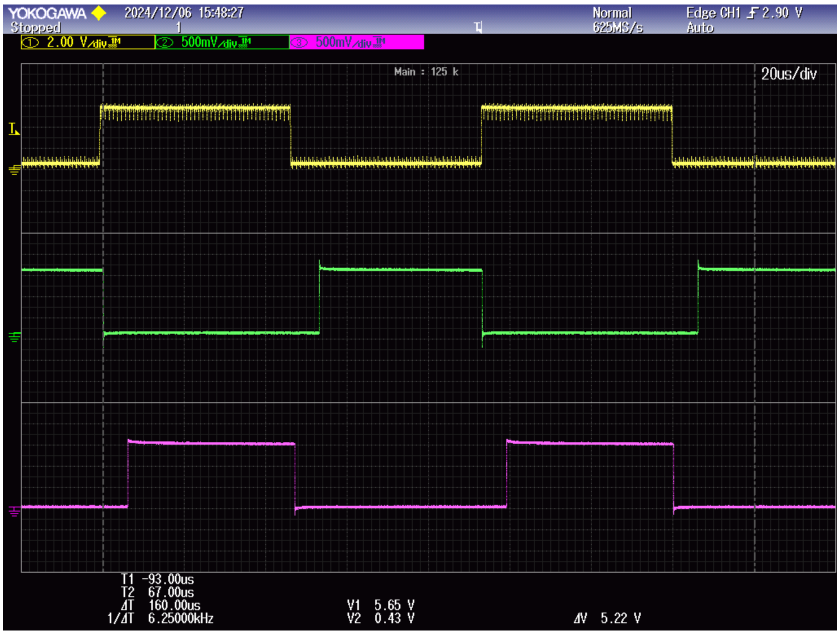


Figure 5.9: From top to bottom: Incoming PWM signal and the two output signals, low- and high-side respectively.

The device also requires for the dead-time to be adjustable. In the following images the dead-time has been set to 100 ns in image 5.10, whereas the dead-time has been set to 300 ns in figure 5.10. Both have a slight extra offset of about 150 ns, this is due to the propagation delay of the entire device. This is done by simply changing the dead-time integer value in the VHDL code and thus either increasing or decreasing the value in steps of 50 ns since the CPLD runs on a 20 MHz clock signal.

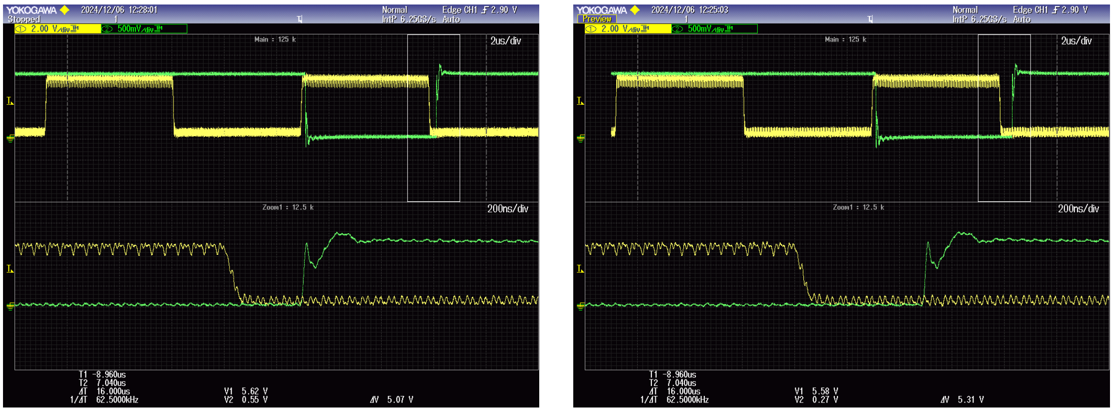


Figure 5.10: Left: 100 ns Dead-Time Right: 300 ns Dead-Time.

In figure 5.11, we can also see that both the incoming PWM signal and the signal that is between the buffer and gate driver are between 0 V and 5 V. The green output signal, however, jumps between 0 and 15 V instead. This clearly demonstrates the difference in voltage levels between both sides of the device.

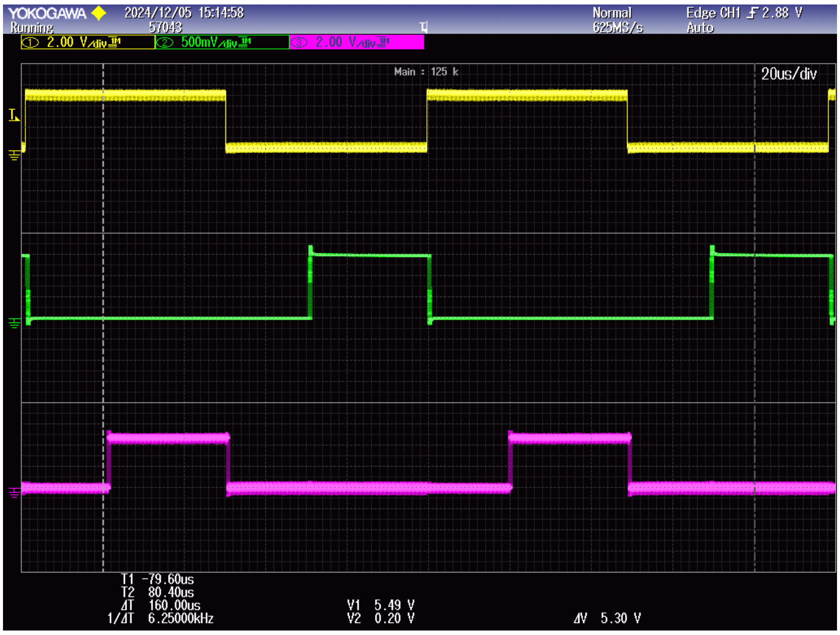


Figure 5.11: From top to bottom: Incoming PWM signal at 5V, low-side output signal at 15V, and high-side gate driver input at 5V.

As can be seen in some of the previous images, as well as in image 5.12 below, there is quite some noise active in the signal. This is mostly due to the fact that we are using an external oscillator to function as our clock signal. This seems to add quite an amount of jitter to the signal and this can be seen in the large oscillations within the PWM signals.

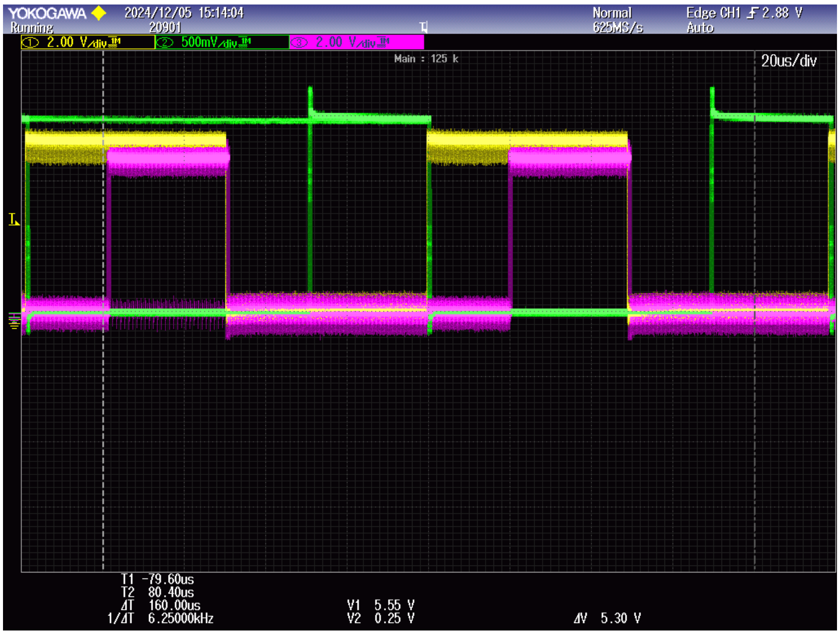


Figure 5.12: Input and output signals with significant amount of noise.

Removing the clock and only running the device using the PWM signal greatly reduces any jitter in the signal. An example of this can be seen in image 5.2.

It is therefore quite clear that a clock signal should be added to the PCB in the form of a crystal oscillator. This should improve the output quite significantly, and result in a better functioning device. For this it is recommend to add the following component: CSX-750FCC20000000T [27]. This crystal oscillator is capable of running on the 5 V supply voltage and is compatible with the TTL input of the CPLD. Adding this component will allow it to generate a stable 20 MHz clock signal for the CPLD .

6
Discussion and Conclusion

The goal of this thesis is to create a gate driver that can be adapted to various MOSFET characteristics. In order to accomplish this goal several conditions had to be met:

- Be able to split an incoming PWM signal into two separate PWM signals, with one of those signals being inverted.
- Create an adjustable dead-time.
- Create a step-up in voltage from 5 V to 15 V to drive the transistor gates.
- Create an interchangeable and adjustable set of components at the output phase.

The programmable unit met all specified parameters except for the operating frequency, which was set to 20 MHz, at which all tests were conducted. It is also capable of generating different dead-times. The gate driver fulfills all its specified criteria. The split PWM signals are the inverse of one another. The design has an adjustable dead-time that works as expected and the voltage provided to the gates moves between 0 V and 15 V. Furthermore, it is possible to remove, replace, and change the values of several components such as the gate resistors at the output phase.

However, the presence of noise and distortion in the expected output indicates the need for further optimization to enhance signal quality. Because of this distortions and noise it is difficult to ascertain whether certain parameters of the hardware behave as expected and desired. However, these results provide a basis for improvements in the next iterations of the design, which are discussed in the following section.

Furthermore, a critical issue encountered is that we were unable to run all the tests that were originally envisioned in the test plan. Therefore more tests still remain to be done to better characterize the design's effectiveness and functionality.

That being said, it can be concluded that the project reached its main goal. The output signals sufficiently show the desired behaviour which at least demonstrates the design concept has been proven to work. This means that it is possible to design a cost-efficient gate driver that can be adapted to operate in a wide range of half-bridge setups with a programmable dead-time.

Recommendations and Future Work

In order to improve the current design we give the following recommendations:

- Add an external 20 MHz crystal oscillator to the global clock input.
- Add a global reset button that can reset the CPLD and possibly the Gate driver IC.
- Change the breakdown voltage of the DESAT blanking diode to allow the DESAT to be triggered.
- Keep the buffer in the design.
- Change the LED resistor to one of a higher resistance value.

The main problem we aim to solve with these adjustments is the large amount of noise that is generated in the current design. This would be achieved by adding the crystal oscillator component to the global clock. Adding a global reset button will allow for easier initialization of the VHDL code. Changing the DESAT blanking diodes will ensure the DESAT component functions as intended. The addition of a buffer, although not strictly necessary for the operation of the device, ensures that the voltage at the input of the gate drivers is kept at 5 V, instead of roughly 4 V, when they are not included. Furthermore, they add some functionality to the code where one or both of the gate drivers can be turned on or off to test separate components. Lastly, the change in LED resistors is simply a quality of life improvement.

With the expected reduction of noise in the improved design several new options for future work become available. Firstly, the reduced noise level would allow for several tests that were already carried out to be done more accurately. Next some tests that turned out to be inconclusive could be repeated and the level at which they were affected by the noise could be ascertained. This could then allow for a more precise choice of components for the output phase of the design. Once this is done a range of recommended settings for specific, often used, transistors could be made, thus improving the speed at which tests can be carried out.

Bibliography

[1] E. Osmanbasic. “High voltage vehicles: Why 800-volt evs are on the rise.” (2023), [Online]. Available: https://www.engineering.com/high-voltage-vehicles-why-800-volt-evs-are-on-the-rise/ (visited on 06/12/2024).

[2] M. H. Andreas Volke, IGBT Modules: Technologies, driver and application. 2017.

[3] Ucc21331cdr: Isolated dual-channel gate drive. [Online]. Available: https://www.ti.com/lit/ds/symlink/ucc21331.pdf?ts=1733707804179&ref_url=https%253A%252F%252Fnl.mouser.com%252F.

[4] 2asc-17a1hp: Dual-channel augmented high performance. [Online]. Available: https://nl.mouser.com/datasheet/2/268/2ASC_17A1HP_spec_V03-3444522.pdf.

[5] 2sc0108t2d0-12: Data sheet. [Online]. Available: https://nl.mouser.com/datasheet/2/328/2SC0108T2D0-12-1387062.pdf.

[6] A. Ayodele. “Fpga vs. microcontroller: Understanding the key differences.” (2023), [Online]. Available: https://www.wevolver.com/article/fpga-vs-microcontroller (visited on 04/11/2024).

[7] AVAQ. “What is complex programmable logic device(cpld).” (2023), [Online]. Available: https://www.avaq.com/technology/what-is-complex-programmable-logic-device-cpld (visited on 04/11/2024).

[8] A. Devices, “Common-mode transient immunity,” 2020.

[9] S. C. H. (47), “Semiconductor devices - part 17: Magnetic and capacitive coupler for basic and reinforced insulation,” 2020.

[10] A. Devices, “Understanding the safety certification of digital isolators,” 2015.

[11] 1ed3323mc12nxuma1: Gate driver. [Online]. Available: https://nl.mouser.com/datasheet/2/196/Infineon_1ED332xMC12N_DataSheet_v01_02_EN-3360451.pdf.

[12] T. Instruments, External gate resistor design guide for gate drivers. [Online]. Available: https://www.ti.com/lit/ab/slla385a/slla385a.pdf?ts=1732042358879&ref_url=https%253A%252F%252Fwww.google.com%252F.

[13] IEEE, “Ieee standard for test access port and boundary-scan architecture,” IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001), pp. 1–444, 2013. doi: 10.1109/IEEESTD.2013.6515989.

[14] Atdh1150usb: Jtag programmer. [Online]. Available: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/UserGuides/Atmel-8909-CPLD-ATDH1150USB-ATF15-JTAG-ISP-Download-Cable-UserGuide.pdf&ved=2ahUKEwjq7YmB9tiJAxVu1gIHHbS3JMUQFnoECBcQAQ&usg=AOvVaw1x7xtTY_hlYoEjExrWAVYr.

[15] T. Instruments, Logic guide. [Online]. Available: https://www.ti.com/lit/sg/sdyu001ab/sdyu001ab.pdf.

[16] Atf1502as-10au44, cpld. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-0995-CPLD-ATF1502AS(L)-Datasheet.pdf.

[17] Cy37032p44-200jxc: Cpld. [Online]. Available: https://www.digikey.nl/nl/products/detail/rochester-electronics-llc/CY37032P44-200JXC/12108607?s=N4IgTCBcDaIMIEODMB2ADEsAFALDgtGGmgFIA

[18] Max 7000: Cpld. [Online]. Available: https://rocelec.widen.net/view/pdf/efjbibfz0z/ALTES00397-1.pdf?t.download=true&u=5oefqw.

[19] Cahct1g125qdbvrq1: Buffer. [Online]. Available: https://www.ti.com/lit/ds/symlink/sn74ahct1g125-q1.pdf?ts=1733475172454&ref_url=https%253A%252F%252Fnl.mouser.com%252F.

[20] Tba1-0513e: Dc-dc converter. [Online]. Available: https://www.tracopower.com/products/tba1e.pdf.

[21] Resistor sizes and packages. [Online]. Available: https://eepower.com/resistor-guide/resistor-standards-and-codes/resistor-sizes-and-packages/#.

[22] T. Instruments, External gate resistor design guide for gate drivers. [Online]. Available: https://www.ti.com/lit/ab/slla385a/slla385a.pdf?ts=1733470356721&ref_url=https%253A%252F%252Fwww.google.com%252F.

[23] T. Instruments, Fundamentals of mosfet and igbt gate driver circuits. [Online]. Available: https://www.tij.co.jp/jp/lit/ml/slua618a/slua618a.pdf?ts=1733462583316.

[24] Toshiba, How do esd protection diodes operate? [Online]. Available: https://toshiba.semicon-storage.com/eu/semiconductor/knowledge/faq/diode_tvs-diodes/how-do-esd-protection-diodes-operate.html.

[25] Toshiba, Mos is sensitive to static electricity. how do you protect mosfets from static electricity? [Online]. Available: https://toshiba.semicon-storage.com/ap-en/semiconductor/knowledge/faq/mosfet/mos-is-sensitive-to-static-electricityhow-do-you-protect-mosfets.html.

[26] Diode.com. "Do-214ac package size." (2023), [Online]. Available: https://www.diodes.com/assets/Package-Files/DO-214AC.pdf.

[27] Csx-750f: Crystal oscillator: Cpld. [Online]. Available: https://cfd.citizen.co.jp/cms/cfd/pdf/english/CSX-750F_E.pdf.

[28] C2012x5r1v226m125ac: 22uf capacitor. [Online]. Available: https://product.tdk.com/system/files/dam/doc/product/capacitor/ceramic/mlcc/catalog/mlcc_commercial_general_en.pdf.

[29] C1608x7r1h474k080ac: 0.47uf capacitor. [Online]. Available: https://product.tdk.com/system/files/dam/doc/product/capacitor/ceramic/mlcc/catalog/mlcc_commercial_general_en.pdf.

[30] C0805c102k5ractu: 1nf capacitor. [Online]. Available: https://nl.mouser.com/datasheet/2/447/KEM_C1002_X7R_SMD-3316098.pdf.

[31] Cl10b104kb8nnnc: 0.1uf capacitor. [Online]. Available: https://nl.mouser.com/datasheet/2/585/MLCC-1837944.pdf.

[32] 06035c105kat2a: 1uf capacitor. [Online]. Available: https://nl.mouser.com/datasheet/2/40/AVX_X7RDielectric_777024-1853642.pdf.

[33] C0603c224k9rac7867(the same as c0603c224k9ractu): 0.22uf capacitor. [Online]. Available: https://nl.mouser.com/datasheet/2/447/KEM_C1002_X7R_SMD-3316098.pdf.

[34] S1n-13-f: Schottky diode. [Online]. Available: https://nl.mouser.com/datasheet/2/115/DIOD_S_A0004888657_1-2542530.pdf.

[35] Sk510aq-l: Schottky diode. [Online]. Available: https://www.littelfuse.com/~/media/electronics/datasheets/tvs_diodes/littelfuse_tvs_diode_smaj_datasheet.pdf.pdf.

[36] Smaj18ca: Zener diode. [Online]. Available: https://nl.mouser.com/datasheet/2/54/SMAJ-778033.pdf.

[37] 150060rs55040: Red led. [Online]. Available: https://www.we-online.com/components/products/datasheet/150060RS55040.pdf.

[38] 150060vs75000: Green led. [Online]. Available: https://www.we-online.com/components/products/datasheet/150060VS75000.pdf.

[39] 61300411121: 1x4 pin connector. [Online]. Available: https://www.we-online.com/components/products/datasheet/61300411121.pdf.

[40] 61301021121: 2x5 pin connector. [Online]. Available: https://www.we-online.com/components/products/datasheet/61301021121.pdf.

[41] M3 hole: Separator holes. [Online]. Available: https://nl.mouser.com/datasheet/2/990/Essentra_3_12_2024_1485423-3421833.pdf.

[42] Tba1-0513e: Dc-dc converter. [Online]. Available: https://nl.mouser.com/datasheet/2/687/tba1e_datasheet-3049838.pdf.

[43] Rncp0805ftd10k0: 10 kohm resistor (0805). [Online]. Available: https://nl.mouser.com/datasheet/2/385/SEI_rncp-3077653.pdf.

[44] Rncp0603ftd1k00: 1 kohm resistor. [Online]. Available: https://nl.mouser.com/ProductDetail/SEI-Stackpole/RNCP0603FTD1K00?qs=FESYatJ8odIL5qVsa0n8TA%3D%3D.

[45] Rncp0603ftd10k0: 10 kohm resistor (0603). [Online]. Available: https://nl.mouser.com/datasheet/2/385/SEI_rncp-3077653.pdf.

[46] Mmsz5237bt1g, zener diode. [Online]. Available: https://nl.mouser.com/datasheet/2/308/1/MMSZ5221BT1_D-2316205.pdf.

A

VHDL Code for Programming Dead-Time

This appendix contains the VHDL code that is used to program the CPLD. The main functionality is that it can take an incoming PWM signal, split it into two signals, invert one of the signals and then add a set amount of Dead-Time to both signals.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;


entity S4_FSM is

        port (
                -- inputs
                clk : in std_logic;               --typically 20 MHz
                pwm : in std_logic;               --typically between 1kHz and 1 MHz.
                ready_H, ready_L : in std_logic;  --ready signal indicating GD's can be used.
                fault_H, fault_L : in std_logic;  --DESAT fault signal
                reset_input : in std_logic;       --Global reset signal

                -----------------------------------------------------------

                -- outputs
            --Signals that send the modified PWM signal to High- and Low side Gate Drivers.
                output_high, output_low                                  : out std_logic;
            --Signals that enable the buffer on the High- and/or Low side.
                enable_high, enable_low                                  : out std_logic;
            --Signals that turn on or off the ready LED for the High- and/or Low side.
                ready_led_H, ready_led_L                                 : out std_logic;
            --Signals that turn on or off the fault LED for the High- and/or Low side.
                fault_led_H, fault_led_L                                 : out std_logic;
            --Signals that turn on/off the four PWM LED's to show which PWM signal is being used.
                pwm_led_1, pwm_led_2, pwm_led_3, pwm_led_4               : out std_logic
        );
end entity;

architecture behavioural of S4_FSM is
```

```
-- ONLY CHANGE THE CONSTANT VALUES AFTER ":="

--Change this value to set a different Dead-Time. Steps are of size 50ns (i.e. 6 = 300ns)
--due to the fact that the current clock speed is 20 MHz.
constant deadtime                       : integer range 1 to 16                := 6;

--Set to '1' to turn off the high side buffer. Set to '0' to turn on high side buffer.
constant output_high_enable             : std_logic                             := '0';

--Set to '1' to turn off the low side buffer. Set to '0' to turn on low side buffer.
constant output_low_enable              : std_logic                             := '0';

--Set from 1-4 to indicate which PWM input is being used.
constant led_on                         : integer range 1 to 4                  := 1;

---------------------------------------------------

-- These are the states used for the FSM.
type states is
(
        --Reset state where both outputs to the buffer/Gate Driver are low.
        --Using the Reset input puts the system in this state.
        --Incoming PWM signal moves the system to the counter_H state.
        reset,

        --High side count state. Both outputs to the buffer/Gate Driver are low.
        --Once the count threshold has been reached the system moves to the hold_high state.
        counter_H,

        --Low side count state. Both outputs to the buffer/Gate Driver are low.
        --Once the count threshold has been reached the system moves to the hold_low state.
        counter_L,

        --Hold high state. In this state the high side output to the buffer/Gate Driver is high,
        --while the low side output is low. A falling edge of the PWM signal moves the system to
        --the counter_L state.
        hold_high,

        --Hold low state. In this state the low side output to the buffer/Gate Driver is high,
        --while the high side output is low. A rising edge of the PWM signal moves the system to
        --the counter_H state.
        hold_low
);


---------------------------------------------------
-- signal declarations
signal state, new_state : states := reset;
signal ID, ire, ile : std_logic;
signal state_check: std_logic;

---------------------------------------------------
begin

process(clk, reset_input, state, pwm, ready_H, ready_L, ire, ile, ID,fault_H, fault_L)
```

```
--Integer count is set to 1 to account for the state update running one clock signal behind.
variable count : integer range 0 to 32 := 1;


begin

        --Clock signal that updates the current state.
        if (rising_edge(clk)) then

                Id <= pwm;

            --Detection of a rising edge in the PWM signal.
                ire <= not ID and pwm;

            --Detection of a falling edge in the PWM signal.
                ile <= ID and not pwm;

            --If the Reset input is high, the state is kept at Reset state.
                if (reset_input = '1') then
                        state <= reset;

            --If Reset is low and both Gate Drivers are ready, the states are updated to
            --their new states.
                elsif (ready_H = '1' and ready_L = '1') then
                        state <= new_state;

            --If the Gate Drivers are not ready, the system is kept in the Reset state even
            --if the Reset input isn't high.
                else
                        state <= reset;
                end if;

                -- clock cycle counter
            --When in either the counter_H or counter_L state the count value is updated,
            --if it is in any other state the value is set back to 1.
                if (state = counter_H) then
                        count := count + 1;
                elsif (state = counter_L) then
                        count := count + 1;
                else
                        count := 1;
                end if;
        else
        end if;

-------------------------------------------------------

--Code to turn on/off one or both of the ready LEDs indicating the Gate Drivers are ready.

        if (ready_H = '1' and ready_L = '1') then
                ready_led_H <= '1';
                ready_led_L <= '1';
        elsif (ready_H = '0' and ready_L = '1') then
                ready_led_H <= '0';
                ready_led_L <= '1';
        elsif (ready_H = '1' and ready_L = '0') then
```

```
                ready_led_H <= '1';
                ready_led_L <= '0';
        else
                ready_led_H <= '0';
                ready_led_L <= '0';
        end if;

-------------------------------------------------------

--Code to turn on/off one or both of the fault LEDs indicating the Gate Drivers are
--experiencing a fault.

        if (fault_H = '0') then
                fault_led_H <= '1';
        else
                fault_led_H <= '0';
        end if;

        if (fault_L = '0') then
                fault_led_L <= '1';
        else
                fault_led_L <= '0';
        end if;

-------------------------------------------------------

--Code to turn on/off a LED to show which PWM signal is being used.

        if (led_on = 1) then
                pwm_led_1 <= '1';
                pwm_led_2 <= '0';
                pwm_led_3 <= '0';
                pwm_led_4 <= '0';
        elsif (led_on = 2) then
                pwm_led_1 <= '0';
                pwm_led_2 <= '1';
                pwm_led_3 <= '0';
                pwm_led_4 <= '0';
        elsif (led_on = 3) then
                pwm_led_1 <= '0';
                pwm_led_2 <= '0';
                pwm_led_3 <= '1';
                pwm_led_4 <= '0';
        else
                pwm_led_1 <= '0';
                pwm_led_2 <= '0';
                pwm_led_3 <= '0';
                pwm_led_4 <= '1';
        end if;

-------------------------------------------------------

-- FSM states

        case state is
```

```
    --During Reset State the outputs are kept low, whilst the buffer enables
    --are kept high to turn off the buffers.
        when reset =>

                output_high                 <= '0';
                output_low                  <= '0';
                enable_high                 <= '1';
                enable_low                  <= '1';


                if (ire = '1') then
                        new_state <= hold_high;
                else
                        new_state <= reset;
                end if;

-------------------------------------------------------

        when counter_H =>

                output_high                    <= '0';
                output_low                     <= '0';
                enable_high                    <= output_high_enable;
                enable_low                     <= output_low_enable;

                if (count >= deadtime) then
                        new_state <= hold_high;
                else
                        new_state <= counter_H;
                end if;

-------------------------------------------------------

        when counter_L =>

                output_high                    <= '0';
                output_low                     <= '0';
                enable_high                    <= output_high_enable;
                enable_low                     <= output_low_enable;

                if (count >= deadtime) then
                        new_state <= hold_low;
                else
                        new_state <= counter_L;
                end if;

-------------------------------------------------------

        when hold_high =>

                output_high                      <= '1';
                output_low                       <= '0';
                enable_high                      <= output_high_enable;
                enable_low                       <= output_low_enable;
```

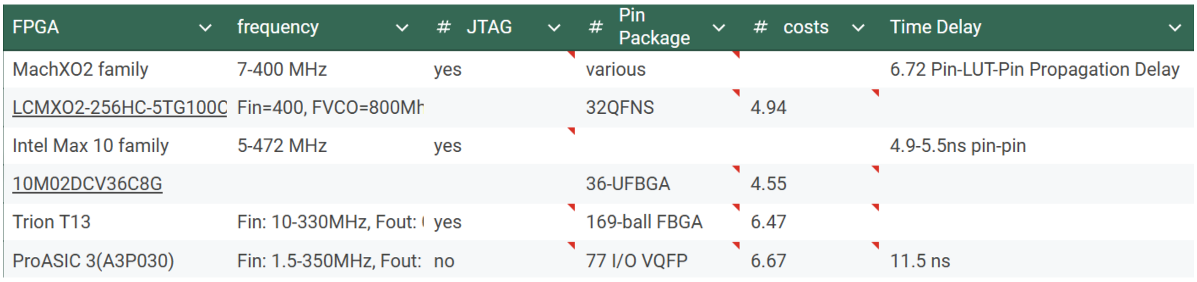
```
                        if (ile = '1') then
                                new_state <= Counter_L;
                        else
                                new_state <= hold_high;
                        end if;

        -------------------------------------------------------

                when hold_low =>

                        output_high                <= '0';
                        output_low                 <= '1';
                        enable_high                <= output_high_enable;
                        enable_low                 <= output_low_enable;


                        if (ire = '1') then
                                new_state <= Counter_H;
                        else
                                new_state <= hold_low;
                        end if;

        -------------------------------------------------------
                --In the others state both output PWM signals are kept low, and the
                --buffers are turned off.
                        when others =>

                                output_high          <= '0';
                                output_low           <= '0';
                                enable_high          <= '1';
                                enable_low           <= '1';

        end case;

end process;

end behavioural;
```

B

Device Sheets

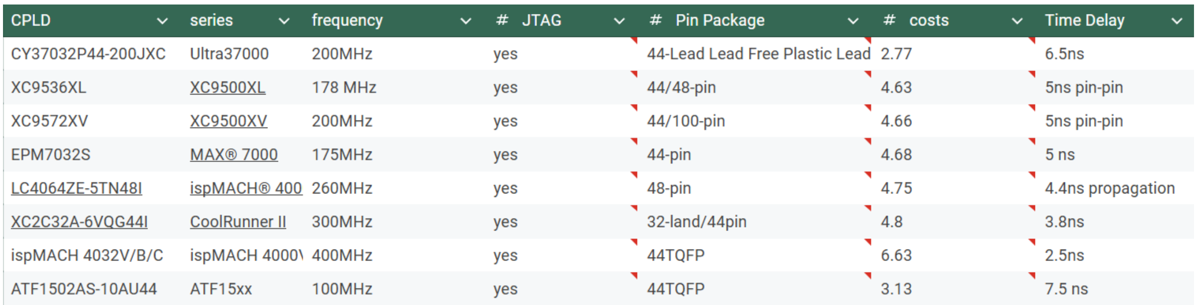
This appendix shows the spreadsheets that were used to decide on the main integrated circuit devices.

B.1. Programmable Logic Devices



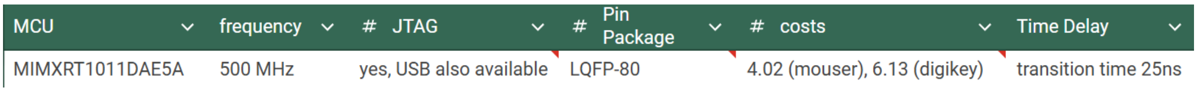
FPGA	frequency	# JTAG	# Pin Package	# costs	Time Delay
MachXO2 family	7-400 MHz	yes	various		6.72 Pin-LUT-Pin Propagation Delay
LCMXO2-256HC-5TG100C	Fin=400, FVCO=800Mh		32QFNS	4.94	
Intel Max 10 family	5-472 MHz	yes			4.9-5.5ns pin-pin
10M02DCV36C8G			36-UFBGA	4.55	
Trion T13	Fin: 10-330MHz, Fout: (	yes	169-ball FBGA	6.47	
ProASIC 3(A3P030)	Fin: 1.5-350MHz, Fout:	no	77 I/O VQFP	6.67	11.5 ns

Figure B.1: List of all FPGAs that were considered.



CPLD	series	frequency	# JTAG	# Pin Package	# costs	Time Delay
CY37032P44-200JXC	Ultra37000	200MHz	yes	44-Lead Lead Free Plastic Lead	2.77	6.5ns
XC9536XL	XC9500XL	178 MHz	yes	44/48-pin	4.63	5ns pin-pin
XC9572XV	XC9500XV	200MHz	yes	44/100-pin	4.66	5ns pin-pin
EPM7032S	MAX® 7000	175MHz	yes	44-pin	4.68	5 ns
LC4064ZE-5TN48I	ispMACH® 400	260MHz	yes	48-pin	4.75	4.4ns propagation
XC2C32A-6VQG44I	CoolRunner II	300MHz	yes	32-land/44pin	4.8	3.8ns
ispMACH 4032V/B/C	ispMACH 4000V	400MHz	yes	44TQFP	6.63	2.5ns
ATF1502AS-10AU44	ATF15xx	100MHz	yes	44TQFP	3.13	7.5 ns

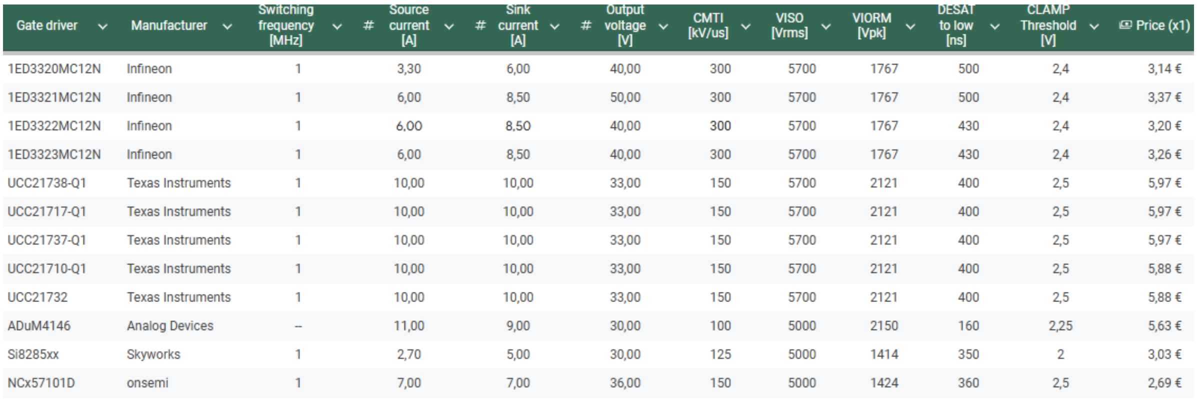
Figure B.2: List of all CPLDs that were considered.



MCU	frequency	# JTAG	# Pin Package	# costs	Time Delay
MIMXRT1011DAE5A	500 MHz	yes, USB also available	LQFP-80	4.02 (mouser), 6.13 (digikey)	transition time 25ns

Figure B.3: The Microcontroller that was considered.

B.2. Gate driver sheets

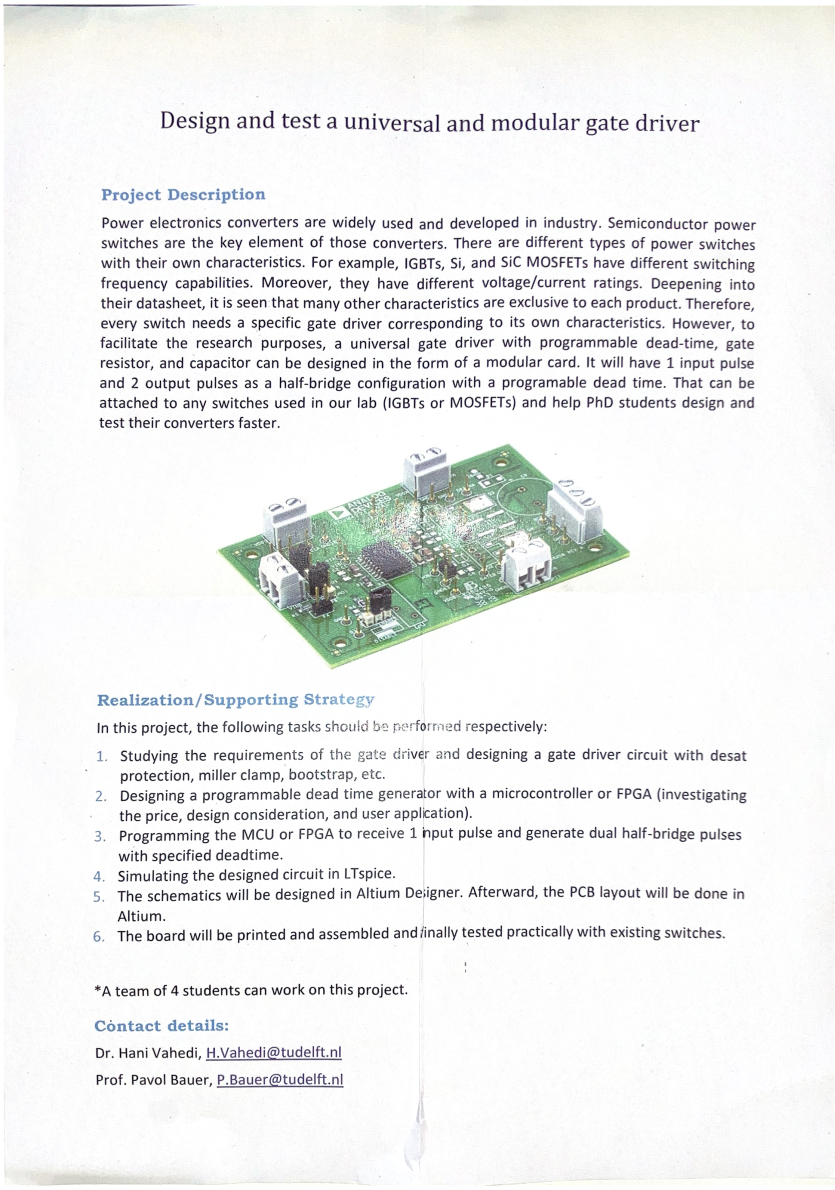


Gate driver	Manufacturer	Switching frequency [MHz]	Source current [A]	Sink current [A]	Output voltage [V]	CMTI [kV/us]	VISO [Vrms]	VIORM [Vpk]	DESAT to low [ns]	CLAMP Threshold [V]	Price (x1)
1ED3320MC12N	Infineon	1	3,30	6,00	40,00	300	5700	1767	500	2,4	3,14 €
1ED3321MC12N	Infineon	1	6,00	8,50	50,00	300	5700	1767	500	2,4	3,37 €
1ED3322MC12N	Infineon	1	6,00	8,50	40,00	300	5700	1767	430	2,4	3,20 €
1ED3323MC12N	Infineon	1	6,00	8,50	40,00	300	5700	1767	430	2,4	3,26 €
UCC21738-Q1	Texas Instruments	1	10,00	10,00	33,00	150	5700	2121	400	2,5	5,97 €
UCC21717-Q1	Texas Instruments	1	10,00	10,00	33,00	150	5700	2121	400	2,5	5,97 €
UCC21737-Q1	Texas Instruments	1	10,00	10,00	33,00	150	5700	2121	400	2,5	5,97 €
UCC21710-Q1	Texas Instruments	1	10,00	10,00	33,00	150	5700	2121	400	2,5	5,88 €
UCC21732	Texas Instruments	1	10,00	10,00	33,00	150	5700	2121	400	2,5	5,88 €
ADuM4146	Analog Devices	–	11,00	9,00	30,00	100	5000	2150	160	2,25	5,63 €
Si8285xx	Skyworks	1	2,70	5,00	30,00	125	5000	1414	350	2	3,03 €
NCx57101D	onsemi	1	7,00	7,00	36,00	150	5000	1424	360	2,5	2,69 €

Figure B.4: List of all Gate Drivers that were considered.

C

Project Description



Design and test a universal and modular gate driver

Project Description

Power electronics converters are widely used and developed in industry. Semiconductor power switches are the key element of those converters. There are different types of power switches with their own characteristics. For example, IGBTs, Si, and SiC MOSFETs have different switching frequency capabilities. Moreover, they have different voltage/current ratings. Deepening into their datasheet, it is seen that many other characteristics are exclusive to each product. Therefore, every switch needs a specific gate driver corresponding to its own characteristics. However, to facilitate the research purposes, a universal gate driver with programmable dead-time, gate resistor, and capacitor can be designed in the form of a modular card. It will have 1 input pulse and 2 output pulses as a half-bridge configuration with a programable dead time. That can be attached to any switches used in our lab (IGBTs or MOSFETs) and help PhD students design and test their converters faster.

Realization/Supporting Strategy

In this project, the following tasks should be performed respectively:

1. Studying the requirements of the gate driver and designing a gate driver circuit with desat protection, miller clamp, bootstrap, etc.
2. Designing a programmable dead time generator with a microcontroller or FPGA (investigating the price, design consideration, and user application).
3. Programming the MCU or FPGA to receive 1 input pulse and generate dual half-bridge pulses with specified deadtime.
4. Simulating the designed circuit in LTspice.
5. The schematics will be designed in Altium Designer. Afterward, the PCB layout will be done in Altium.
6. The board will be printed and assembled and finally tested practically with existing switches.

*A team of 4 students can work on this project.

Contact details:

Dr. Hani Vahedi, H.Vahedi@tudelft.nl

Prof. Pavol Bauer, P.Bauer@tudelft.nl

D
List of Components Used in Design

Table D.1: All components that are used on one PCB

Value/description	Component Name	Quantity	Datasheet
22uF	C2012X5R1V226M125AC	3	[28]
0.47uF	C1608X7R1H474K080AC	2	[29]
1nF	C0805C102K5RACTU	2	[30]
0.1uF	CL10B104KB8NNNC	5	[31]
1uF	06035C105KAT2A	2	[32]
0.22uF	C0603C224K9RAC7867(the same as C0603C224K9RACTU)	7	[33]
schottky 1A	S1N-13-F	2	[34]
schottky 5A	SK510AQ-L	4	[35]
Zener 18.2V, 22A ipp TVS	SMAJ18CA	2	[36]
LED RED 2V, 20mA	150060RS55040	2	[37]
LED GREEN 2V, 20mA	150060VS75000	9	[38]
1ED3323MC12NXUMA1	1ED3323MC12NXUMA1	2	[11]
conn 4pin through hole	61300411121	2	[39]
conn 10 pin through hole	61301021121	2	[40]
M3 hole	M3 hole	4	[41]
TBA1-0513E	TBA1-0513E	2	[42]
CAHCT1G125QDBVRQ1	CAHCT1G125QDBVRQ1	2	[19]
10kOhm	RNCP0805FTD10K0	2	[43]
1kOhm	RNCP0603FTD1K00	13	[44]
10kOhm	RNCP0603FTD10K0	6	[45]
ATF1502AS-10AU44	ATF1502AS-10AU44	1	[16]
Zener 8.2V, 500mW	MMSZ5237BT1G	2	[46]

E

Cost of Design

In this appendix the cost of one or multiple PCB boards with components can be found. It is assumed that regardless of the amount of boards at least one ATDH1150USB JTAG programmer is purchased. The cost of this device is: €95,60.

Table E.1: Costs Per Board

Number of Boards	1	10	100
Cost	€33.655	€261.13	€2114.5

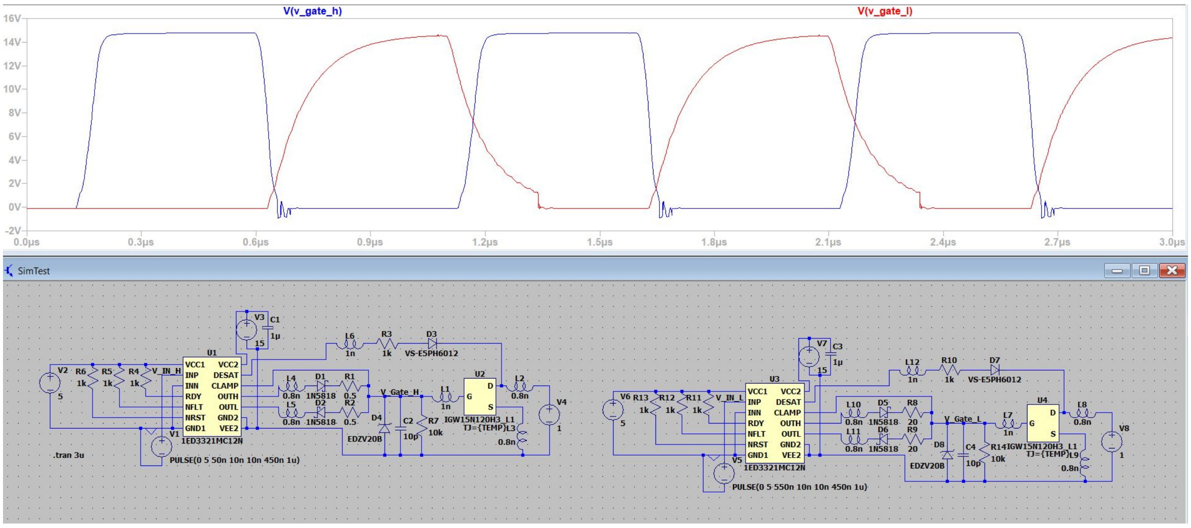
Table E.2: Costs of Making the PCB's (EuroCircuits)

Number of Boards	1	10	100
Cost	€46.01	€108.30	€364.00

F

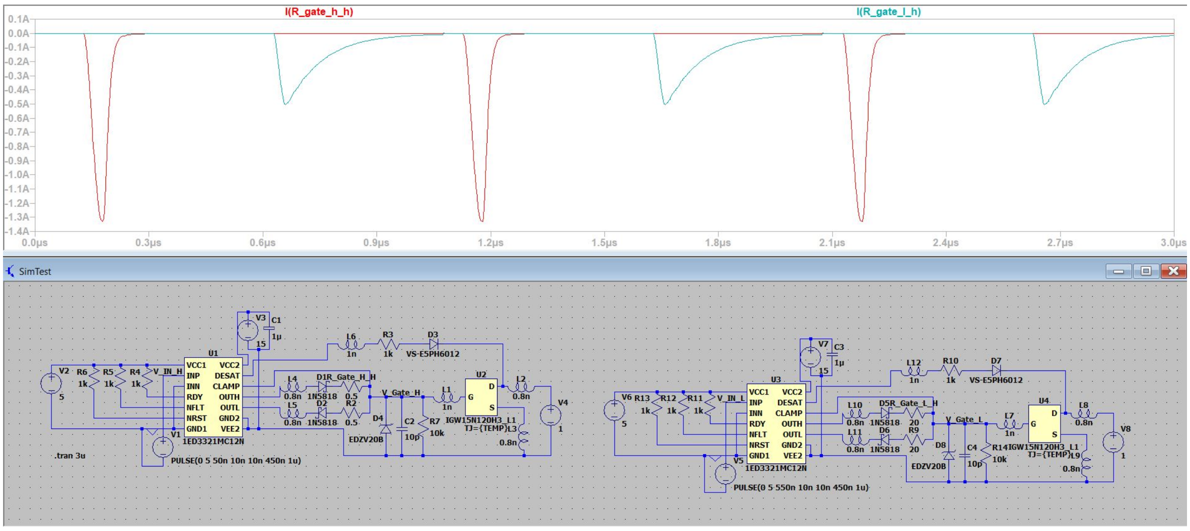
Simulations

This appendix contains a number of results from simulations that were run using LTspice. Information regarding what was tested can be found in the comments. Values can be seen in the schematics in the figures.



```
.tran 3u
PULSE(0 5 50n 10n 10n 450n 1u)
PULSE(0 5 550n 10n 10n 450n 1u)
```

Figure F.1: Output Pulses With Different Resistance Settings

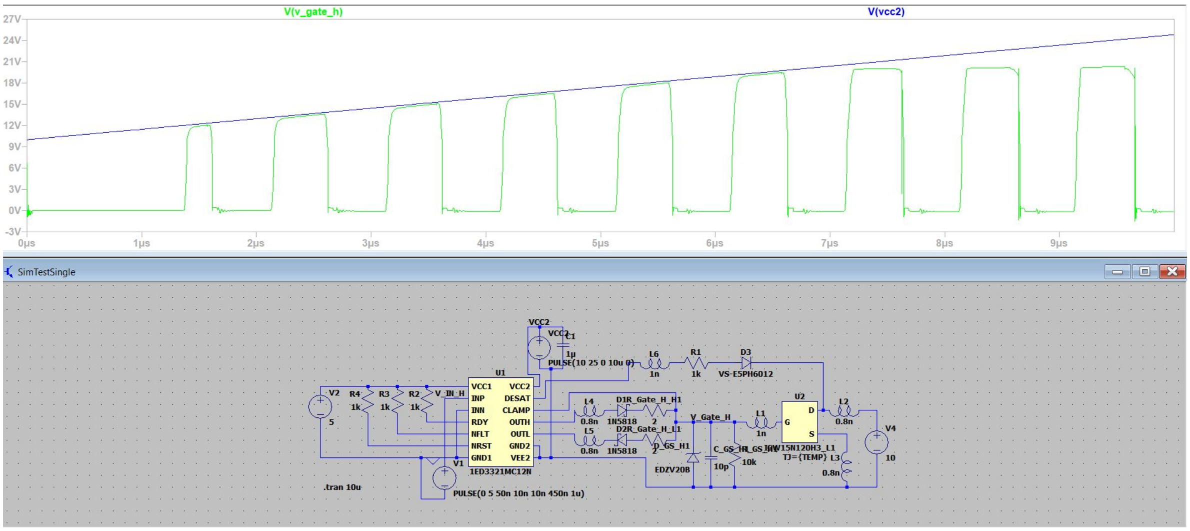


.tran 3u

PULSE(0 5 50n 10n 10n 450n 1u)

PULSE(0 5 550n 10n 10n 450n 1u)

Figure F.2: Output Pulses With Different Resistance Settings Measured Current



.tran 10u

PULSE(10 25 0 10u 0)

PULSE(0 5 50n 10n 10n 450n 1u)

Figure F.3: Low VCC Cut-Off + Cut-Off Diode Behaviour

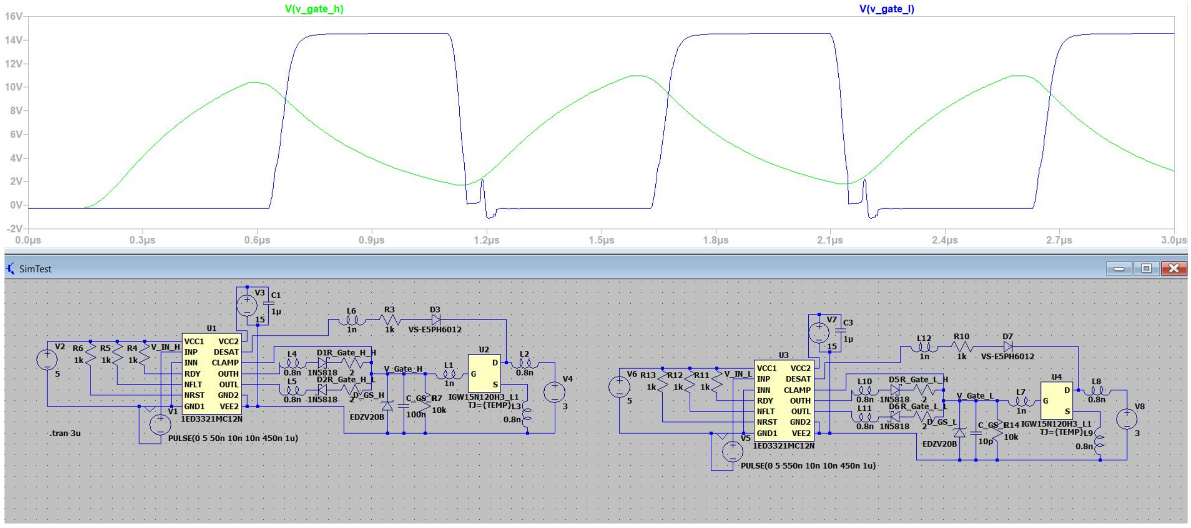


Figure F.4: Capacitor GS Different Values Resulting MOSFET Gate Voltage

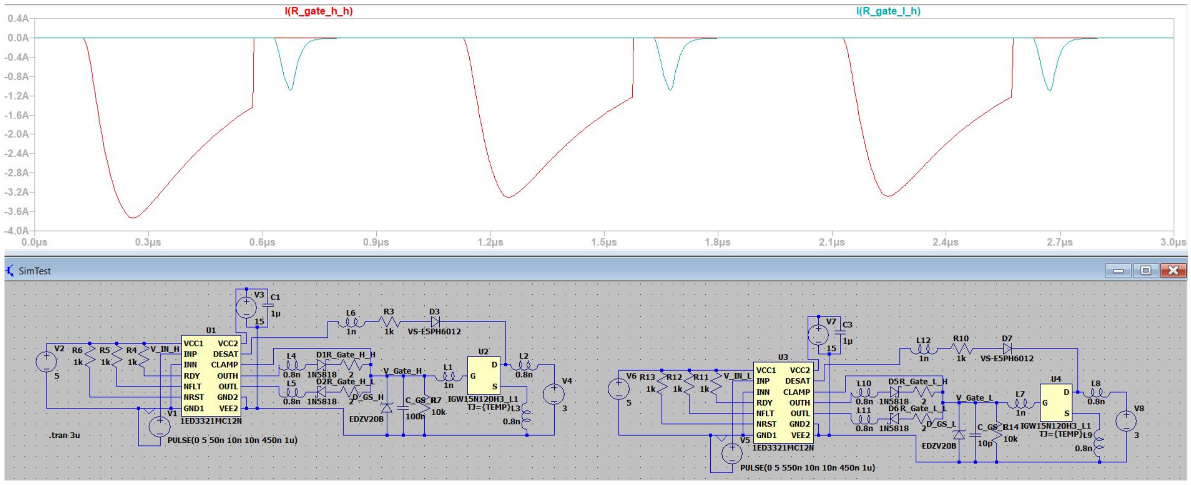


Figure F.5: Capacitor GS Different Values Resulting R Gate Current

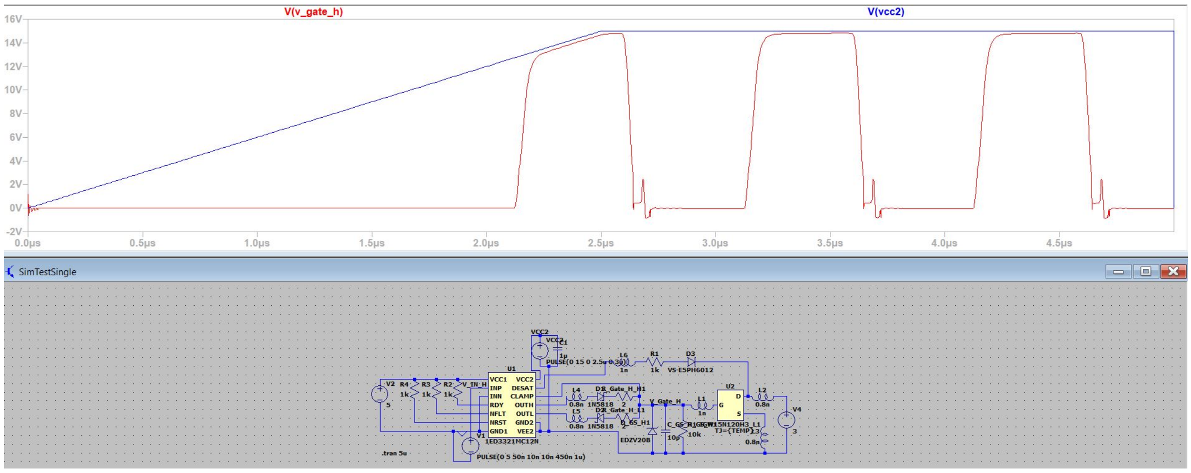
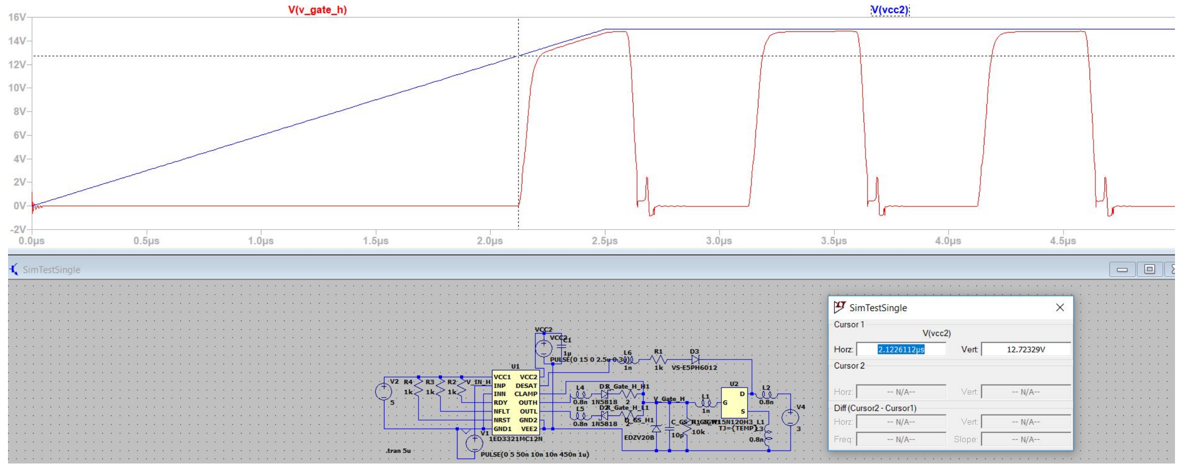


Figure F.6: Turn on Behaviour



SimTestSingle

Cursor 1

V(vcc2)

Horz: 2.1226112µs Vert: 12.72329V

Cursor 2

Horz: -- N/A-- Vert: -- N/A--

Diff (Cursor2 - Cursor1)

Horz: -- N/A-- Vert: -- N/A--

Freq: -- N/A-- Slope: -- N/A--

Figure F.7: Turn on Behaviour with Value

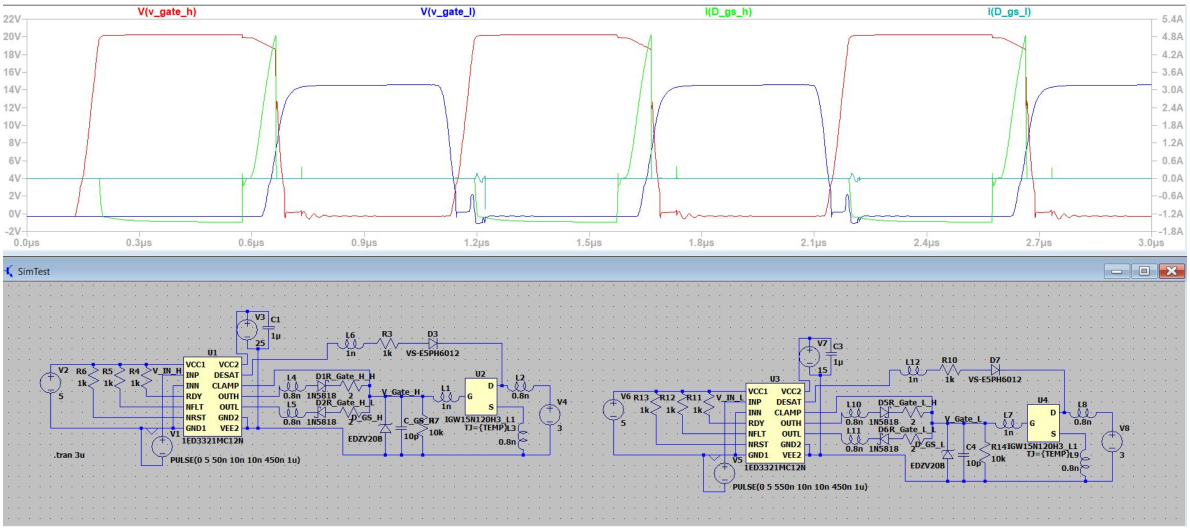
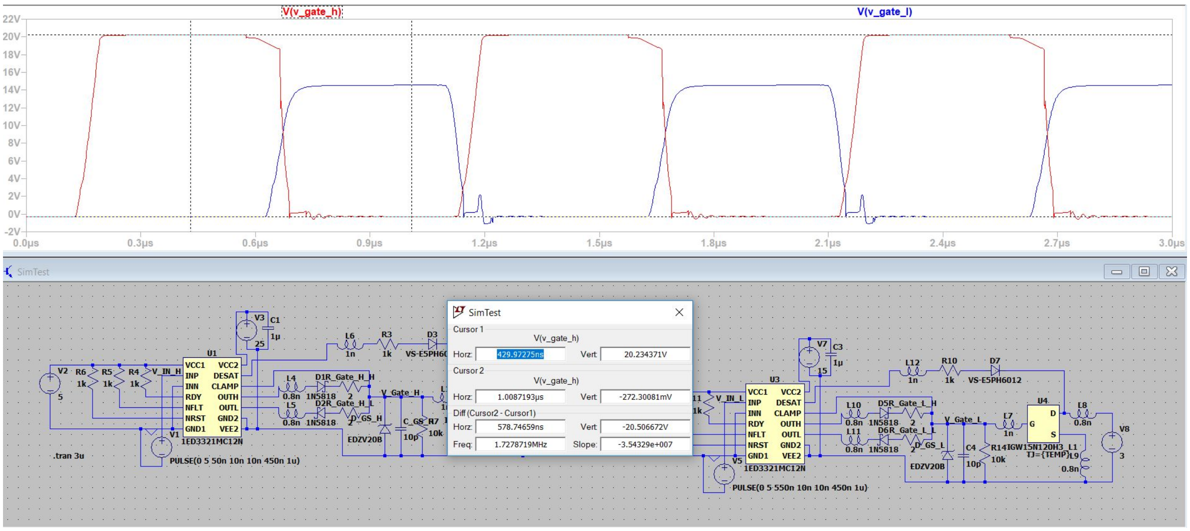


Figure F.8: Diode GS Function Cap at Breakdown Voltage with Current



SimTest

Cursor 1

V(v_gate_h)

Horz: 429.97275ns Vert: 20.234371V

Cursor 2

V(v_gate_h)

Horz: 1.0087193µs Vert: -272.30081mV

Diff (Cursor2 - Cursor1)

Horz: 578.74659ns Vert: -20.506672V

Freq: 1.7278719MHz Slope: -3.54329e+007

Figure F.9: Diode GS Function Cap at Breakdown Voltage

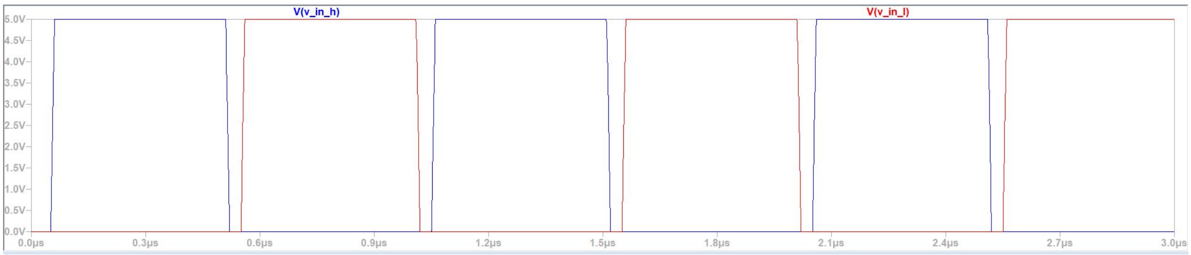


Figure F.10: Gate Driver Input Pulses

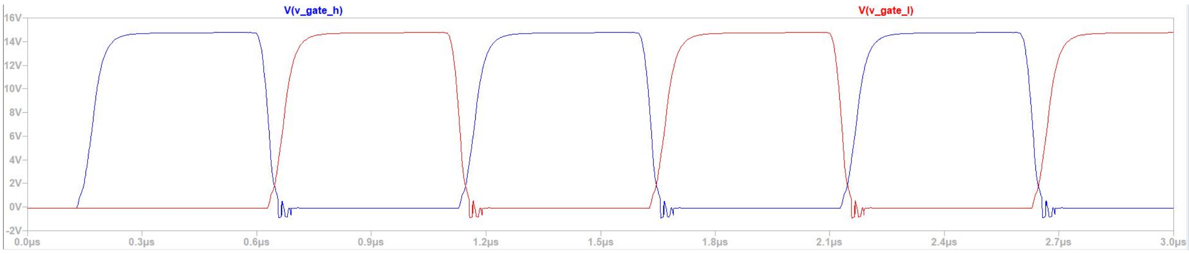
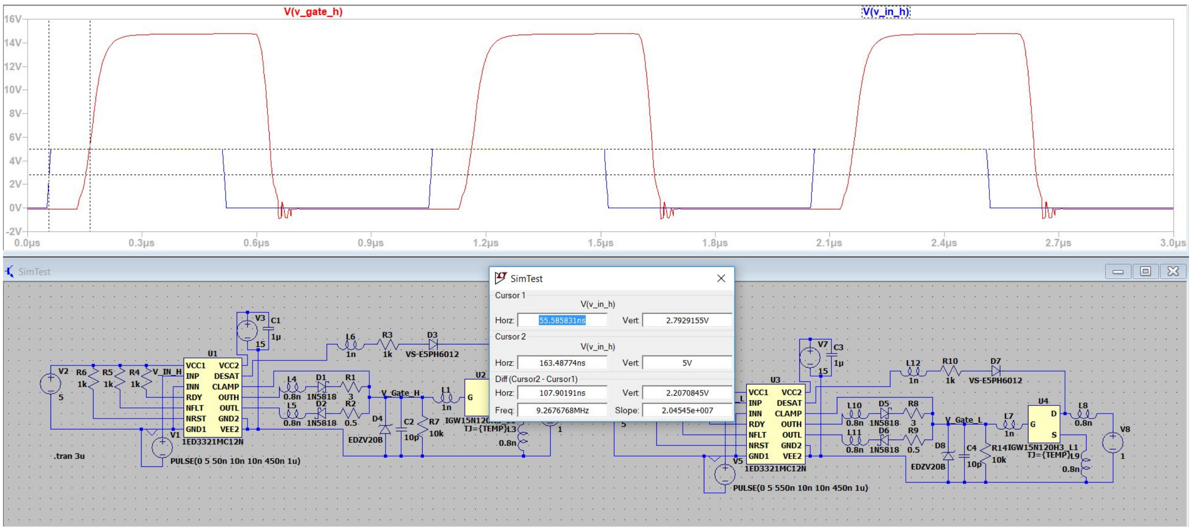


Figure F.11: Gate Driver MOSFET Gate Pulses



SimTest

Cursor 1

V(v_in_h)

Horz: 55.585831ns Vert: 2.7929155V

Cursor 2

V(v_in_h)

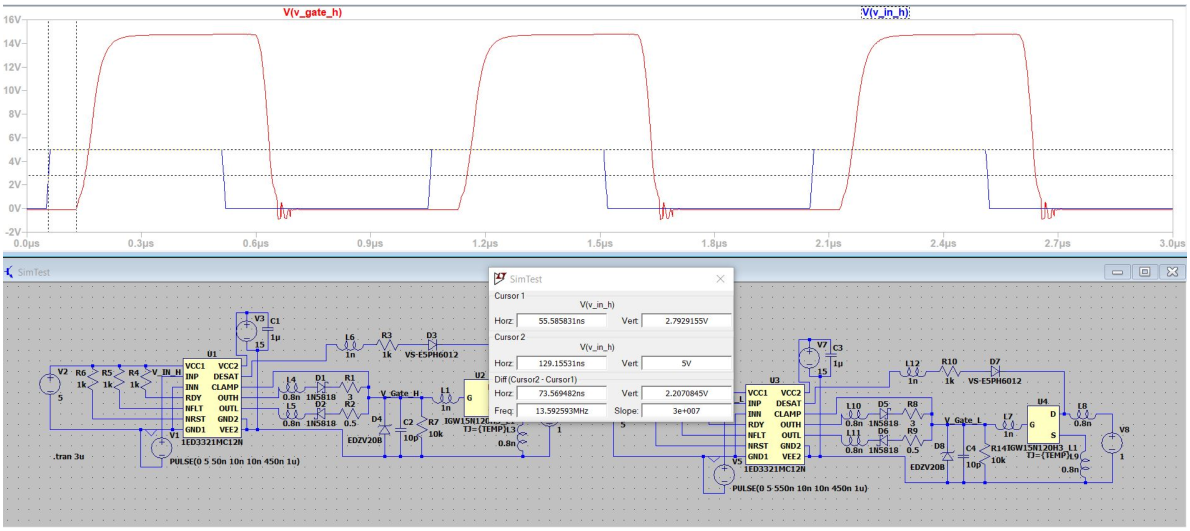
Horz: 163.48774ns Vert: 5V

Diff (Cursor2 - Cursor1)

Horz: 107.90191ns Vert: 2.2070845V

Freq: 9.2676768MHz Slope: 2.04545e+007

Figure F.12: Gate Driver Propagation Delay



SimTest

Cursor 1

V(v_in_h)

Horz: 55.585831ns Vert: 2.7929155V

Cursor 2

V(v_in_h)

Horz: 129.15531ns Vert: 5V

Diff (Cursor2 - Cursor1)

Horz: 73.569482ns Vert: 2.2070845V

Freq: 13.592593MHz Slope: 3e+007

Figure F.13: Gate Driver Propagation Delay II

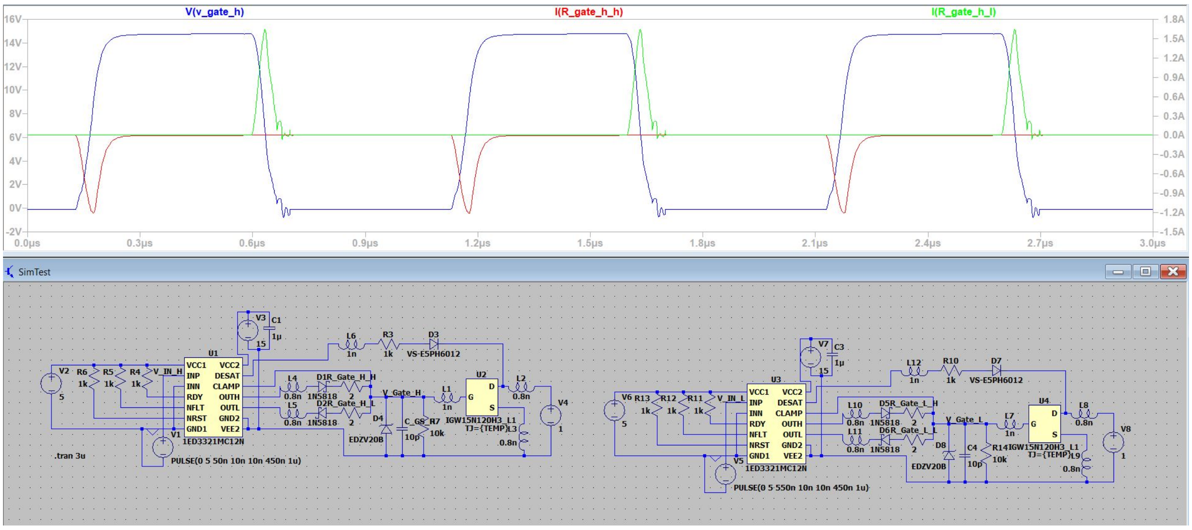


Figure F.14: Output Pulse Gate Voltage and Gate Resistor Current

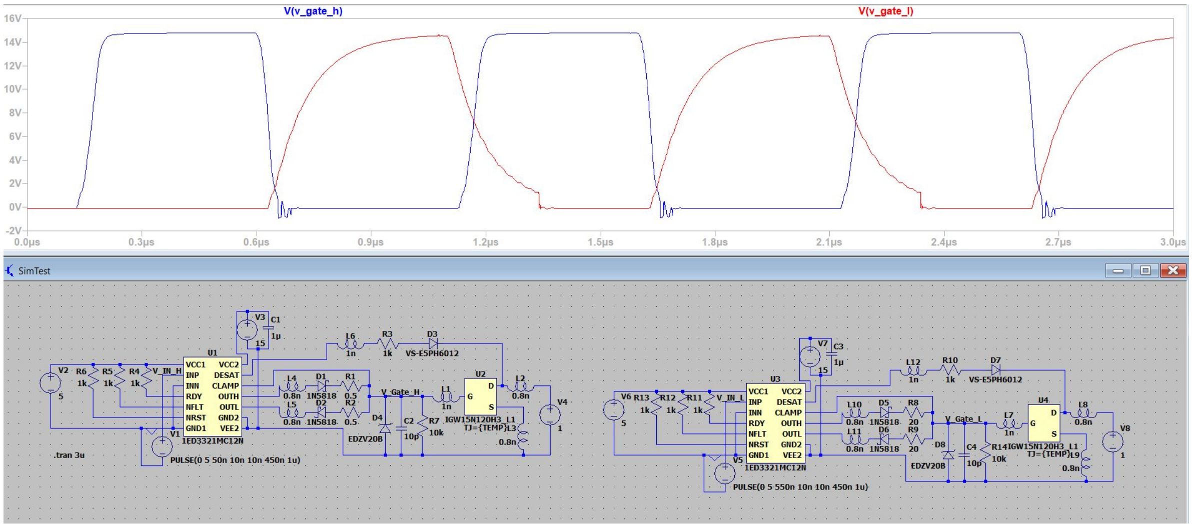


Figure F.15: Output Pulses With Different Resistance Settings

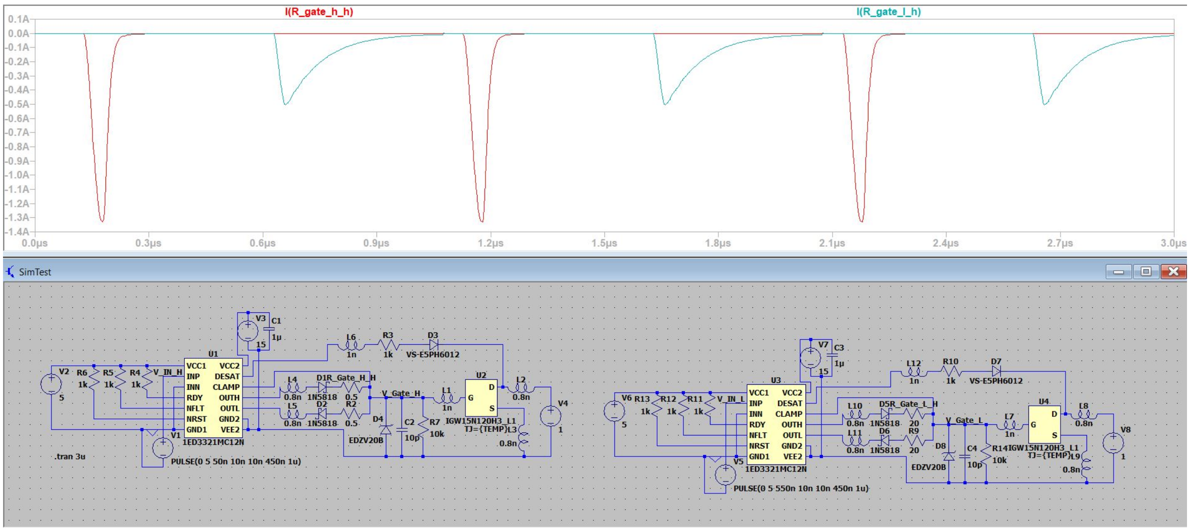


Figure F.16: Output Pulses With Different Resistance Settings Measured Current

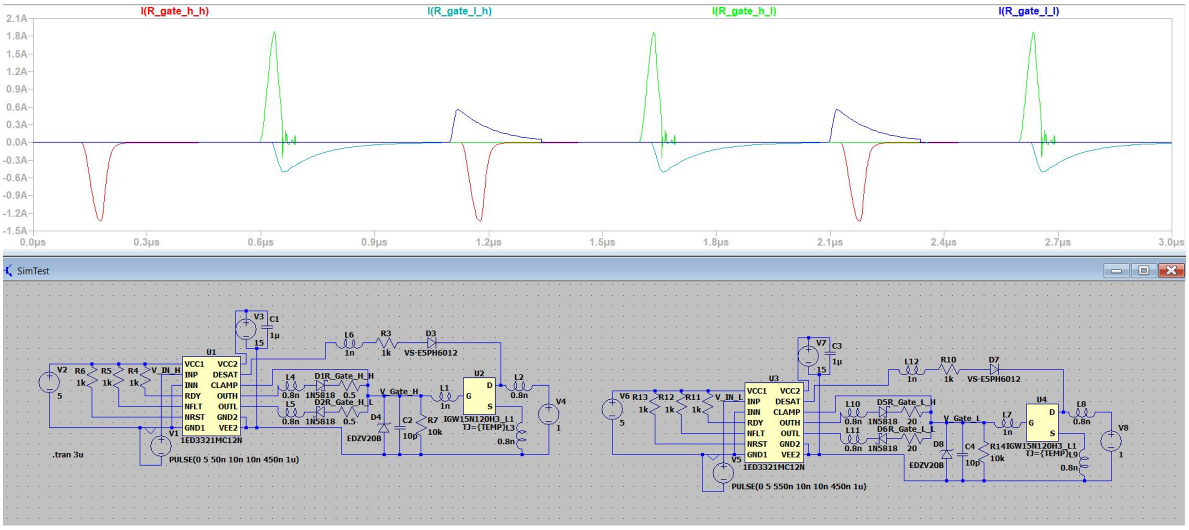


Figure F.17: Output Pulses With Different Resistance Settings Measured Current II

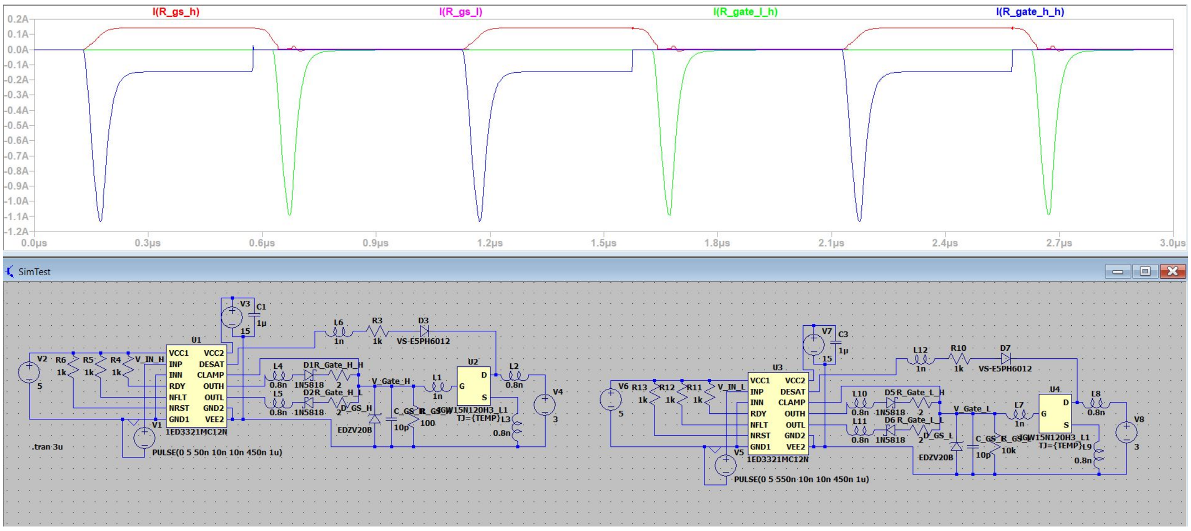


Figure F.18: Resistor GS Different Value resulting Current in Gate Resistors

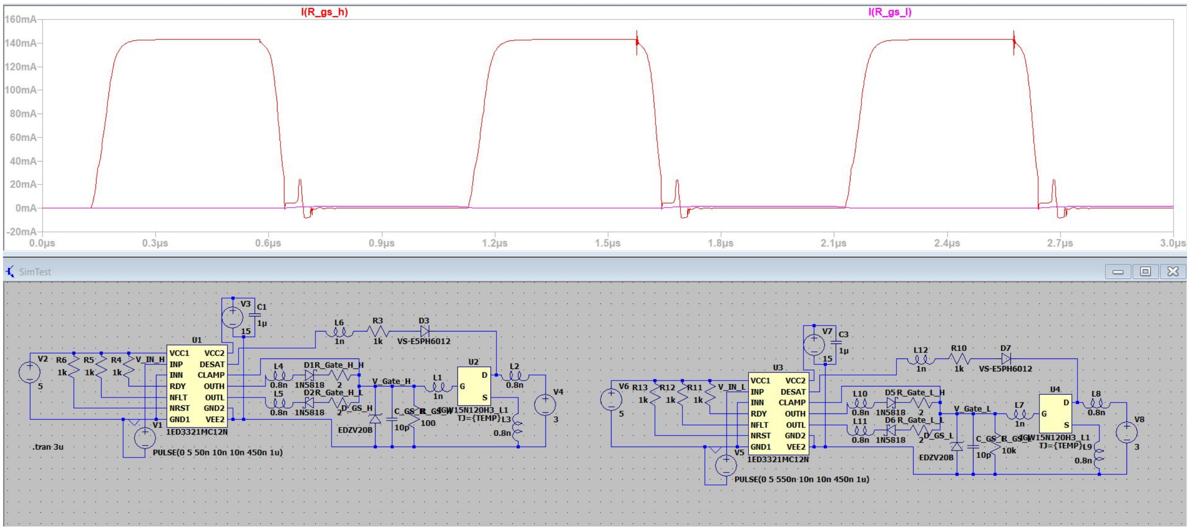


Figure F.19: Resistor GS Different Value resulting Current

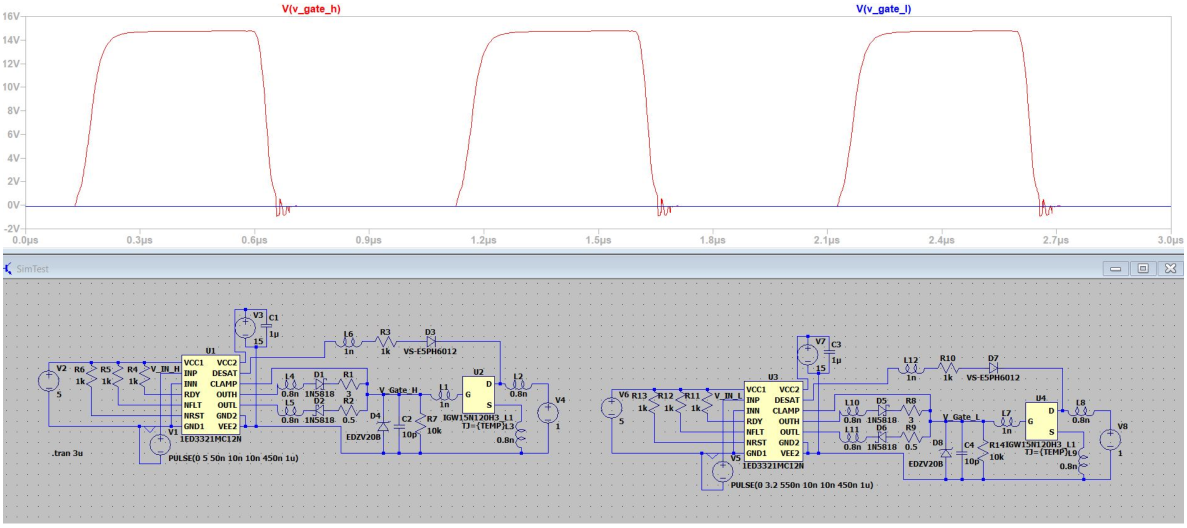


Figure F.20: Supply Voltage Threshold to activate Gate Driver

G

Instruction Manual

This appendix contains a step-by-step guide of how to use the product, as well as all software and hardware required to program the CPLD and an overview of all necessary components.

Universal and Modular Gate Driver Instruction Manual

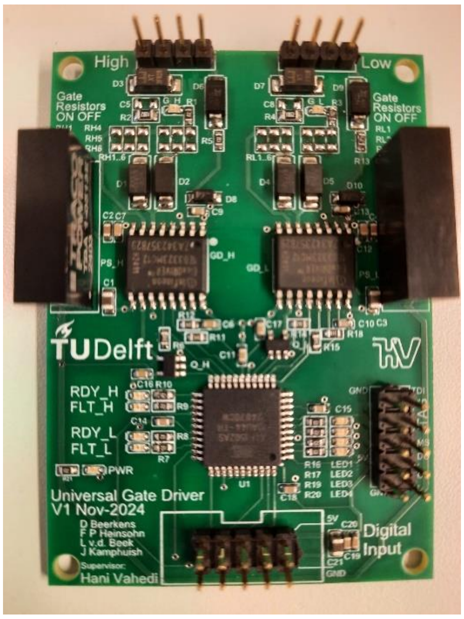
This appendix contains a step-by-step guide of how to use the product, as well as all software and hardware required to program the CPLD and an overview of all necessary components.

Required Hardware

For this project it is necessary to have the following components:

-1x ATDH1150USB JTAG programmer.

-1x PCB (provided) with components seen in the images below.



Value/description	Component Name	Quantity
22uF	C2012X5R1V226M125AC	3
0.47uF	C1608X7R1H474K080AC	2
1nF	C0805C102K5RACTU	2
0.1uF	CL10B104KB8NNNC	5
1uF	06035C105KAT2A	2
0.22uF	C0603C224K9RAC7867(the same as C0603C224K9RACTU)	7
schottky 1A	S1N-13-F	2
schottky 5A	SK510AQ-L	4
Zener 18.2V, 22A ipp TVS	SMAJ18CA	2
LED RED 2V, 20mA	150060RS55040	2
LED GREEN 2V, 20mA	150060VS75000	9
1ED3323MC12NXUMA1	1ED3323MC12NXUMA1	2
conn 4pin through hole	61300411121	2
conn 10 pin through hole	61301021121	2
M3 hole	M3 hole	4
TBA1-0513E	TBA1-0513E	2
CAHCT1G125QDBVRQ1	CAHCT1G125QDBVRQ1	2
10kOhm	RNCP0805FTD10K0	2
1kOhm	RNCP0603FTD1K00	13
10kOhm	RNCP0603FTD10K0	6
ATF1502AS-10AU44	ATF1502AS-10AU44	1
Zener 8.2V, 500mW	MMSZ5237BT1G	2

Note: Resisters used to create the gate resistance have not been included in the list above. These can be chosen as needed for the specific design. Connections are of size 0805.

Required Software

The project requires the following software to be able to adjust the code and program the CPLD:

Quartus II 13.0sp1 (64-bit) Web Edition

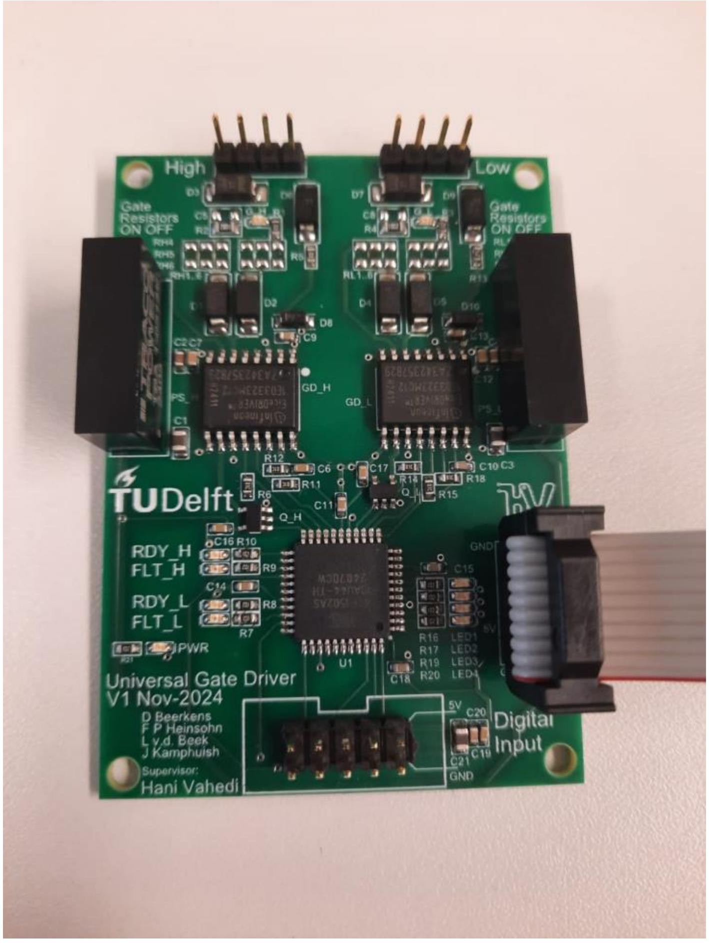
> Download link: https://www.intel.com/content/www/us/en/software-kit/711791/intel-quartus-ii-web-edition-design-software-version-13-0sp1-for-windows.html

POF2JED 4.45.1

> Download link: https://www.microchip.com/en-us/products/fpgas-and-plds/spld-cplds/pld-design-resources

ATMISP v7.3

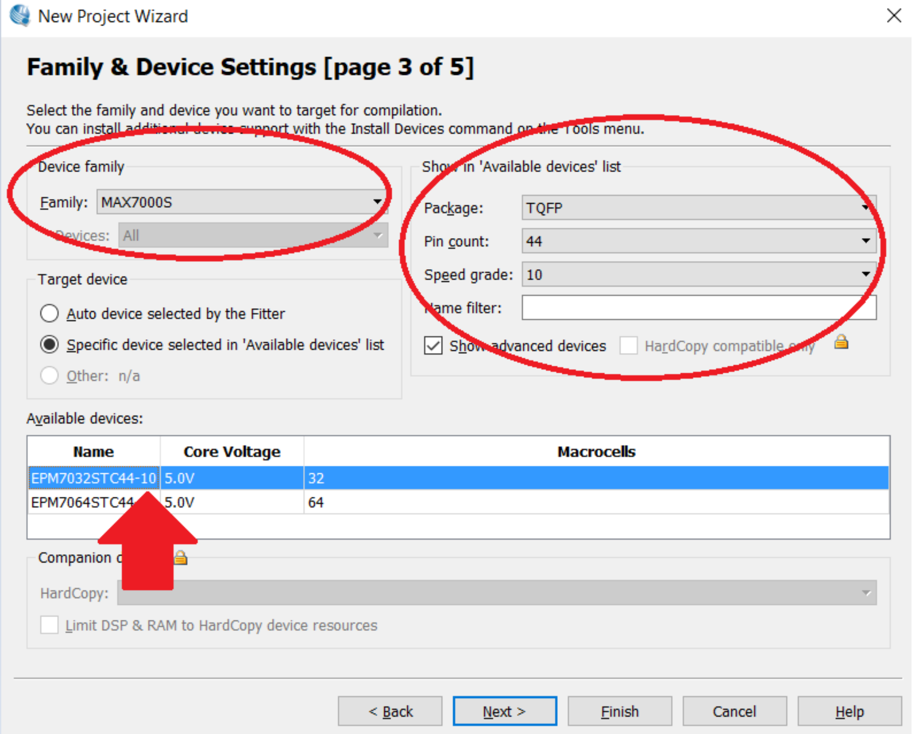
> Download link: https://www.microchip.com/en-us/products/fpgas-and-plds/spld-cplds/pld-design-resources
>
> Make sure the ATDH1150USB JTAG programmer is connected to the CPLD and the computer when running the ATMISP software for the first time, this ensures that all drivers are properly installed. Make sure the cable is connected to the JTAG pins with the red wire on the bottom as seen in the image below.



Uploading the VHDL Code

In order to upload the code the following steps are required:

1. Open Quartus and create new project (You can name this S4_FSM to match the code or change this later). Make sure the settings are set to:



New Project Wizard

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

Device family

Family: MAX7000S

Devices: All

Target device

○ Auto device selected by the Fitter

● Specific device selected in 'Available devices' list

○ Other: n/a

Show in 'Available devices' list

Package: TQFP

Pin count: 44

Speed grade: 10

Name filter:

☑ Show advanced devices ☐ HardCopy compatible only

Available devices:

Name	Core Voltage	Macrocells
EPM7032STC44-10	5.0V	32
EPM7064STC44-10	5.0V	64

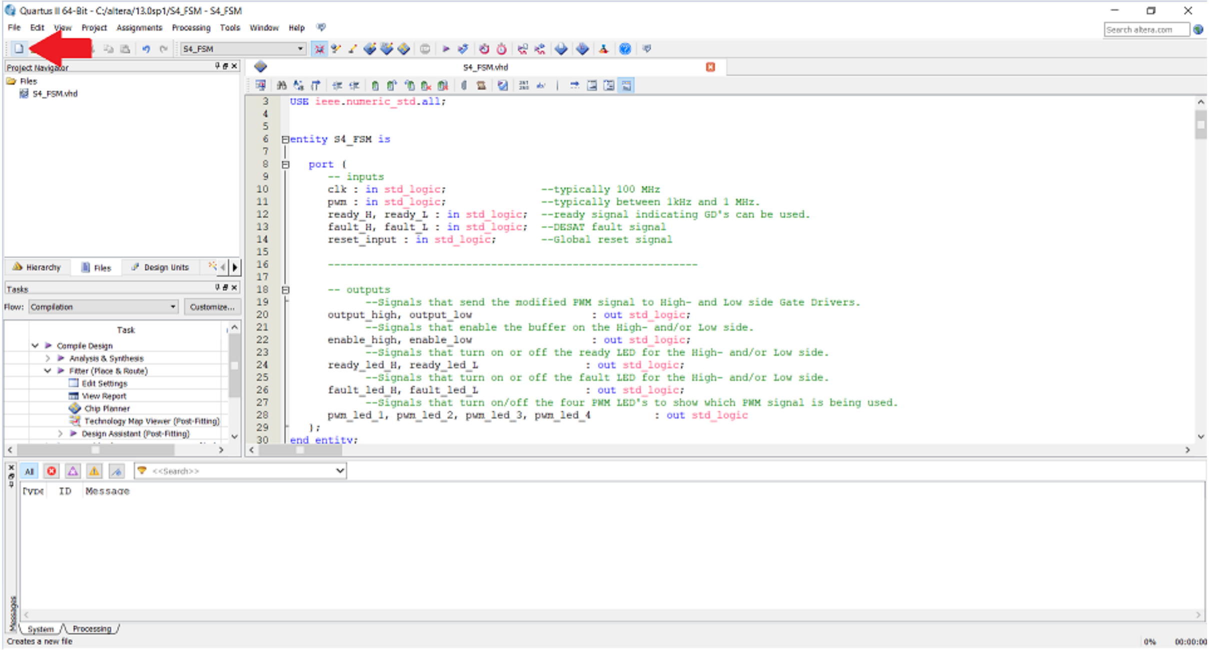
Companion device

HardCopy:

☐ Limit DSP & RAM to HardCopy device resources

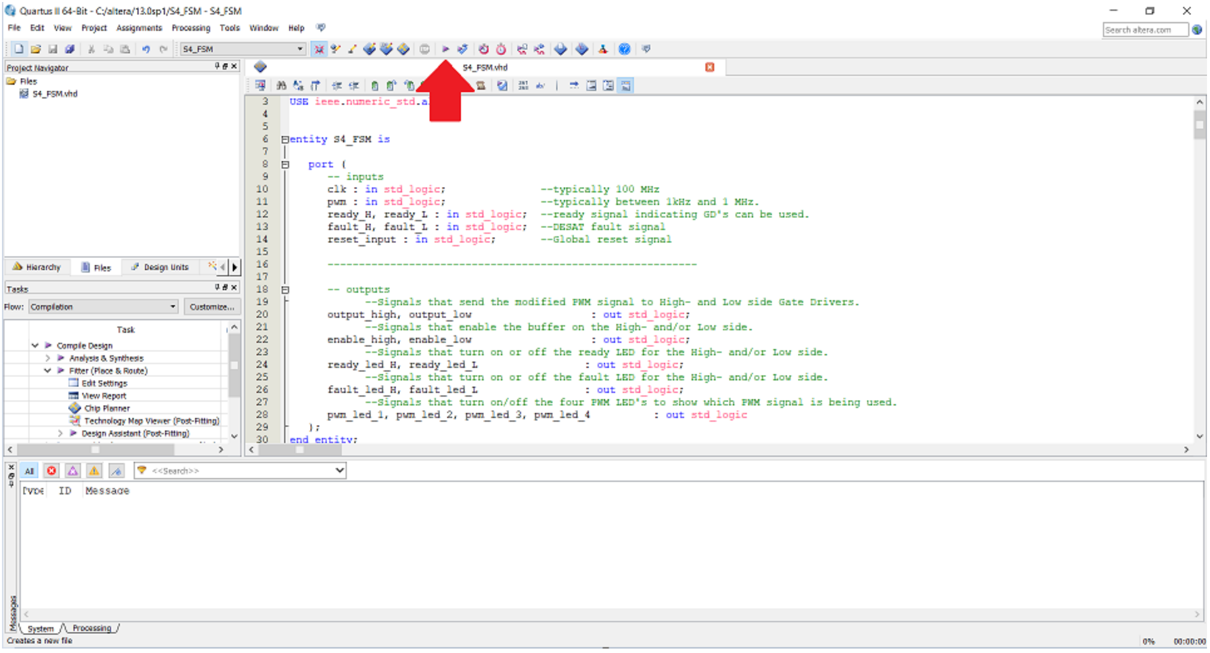
   a. Family: MAX7000S
   b. Package: TQFP
   c. Pin Count: 44
   d. Speed Grade: 10
   e. And select the “EPM7032STC44-10 5.0V 32” option.

2. Click finish and create a VHDL file in the file map on the left by clicking “new” and selecting VHDL file. Put the code that can be found below in the file that has been created. Make sure this file is set to top level and that the name of the file matches the name of the code (in this case “S4_FSM”), this can be done by right clicking on the file.



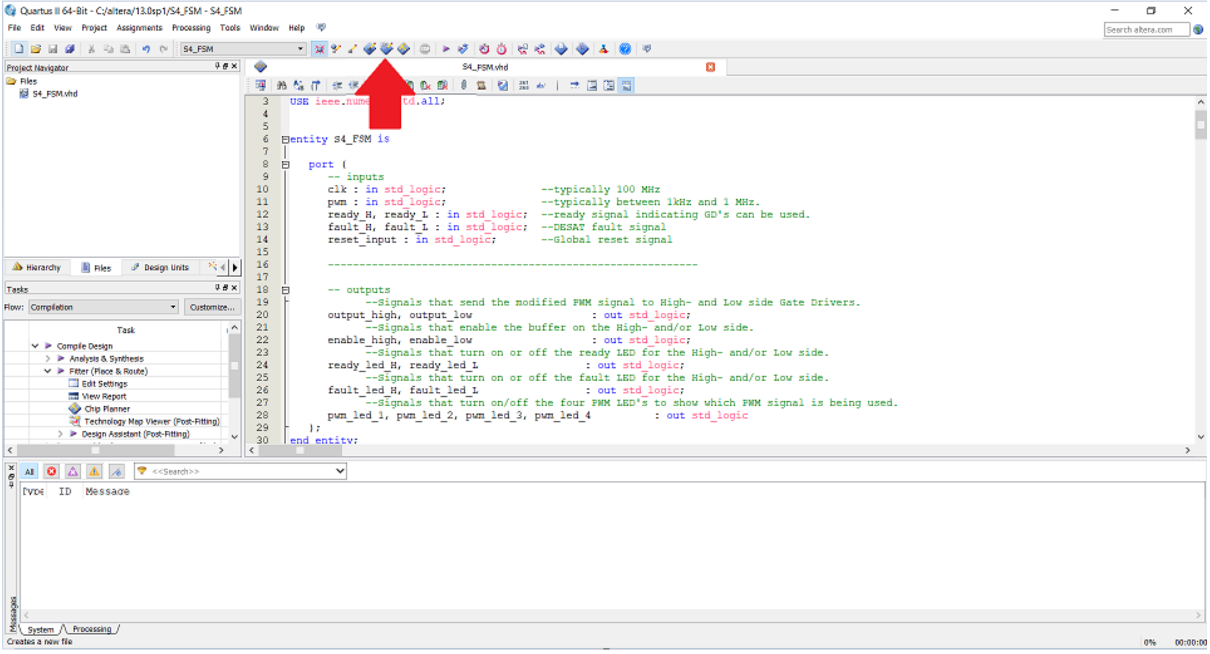
```vhdl
USE ieee.numeric_std.all;


entity S4_FSM is

    port (
        -- inputs
        clk : in std_logic;                 --typically 100 MHz
        pwm : in std_logic;                 --typically between 1kHz and 1 MHz.
        ready_H, ready_L : in std_logic;    --ready signal indicating GD's can be used.
        fault_H, fault_L : in std_logic;    --DESAT fault signal
        reset_input : in std_logic;         --Global reset signal

        ------------------------------------------------------------


        -- outputs
            --Signals that send the modified PWM signal to High- and Low side Gate Drivers.
        output_high, output_low                     : out std_logic;
            --Signals that enable the buffer on the High- and/or Low side.
        enable_high, enable_low                     : out std_logic;
            --Signals that turn on or off the ready LED for the High- and/or Low side.
        ready_led_H, ready_led_L                   : out std_logic;
            --Signals that turn on or off the fault LED for the High- and/or Low side.
        fault_led_H, fault_led_L                   : out std_logic;
            --Signals that turn on/off the four PWM LED's to show which PWM signal is being used.
        pwm_led_1, pwm_led_2, pwm_led_3, pwm_led_4          : out std_logic
    );
end entity;
```

3. Next run and compile the code by clicking on the “Run” button.

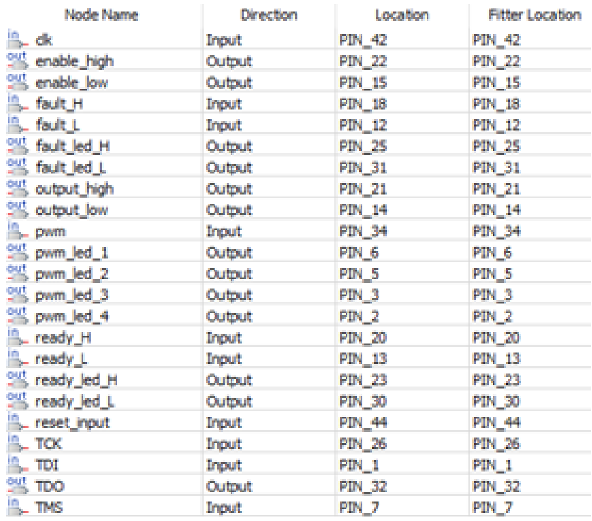


```
USE ieee.numeric_std.a


entity S4_FSM is

   port (
      -- inputs
      clk : in std_logic;                 --typically 100 MHz
      pwm : in std_logic;                 --typically between 1kHz and 1 MHz.
      ready_H, ready_L : in std_logic;    --ready signal indicating GD's can be used.
      fault_H, fault_L : in std_logic;    --DESAT fault signal
      reset_input : in std_logic;         --Global reset signal

      ------------------------------------------------------------

      -- outputs
            --Signals that send the modified PWM signal to High- and Low side Gate Drivers.
      output_high, output_low                        : out std_logic;
            --Signals that enable the buffer on the High- and/or Low side.
      enable_high, enable_low                        : out std_logic;
            --Signals that turn on or off the ready LED for the High- and/or Low side.
      ready_led_H, ready_led_L                      : out std_logic;
            --Signals that turn on or off the fault LED for the High- and/or Low side.
      fault_led_H, fault_led_L                      : out std_logic;
            --Signals that turn on/off the four PWM LED's to show which PWM signal is being used.
      pwm_led_1, pwm_led_2, pwm_led_3, pwm_led_4           : out std_logic
   );
end entity;
```

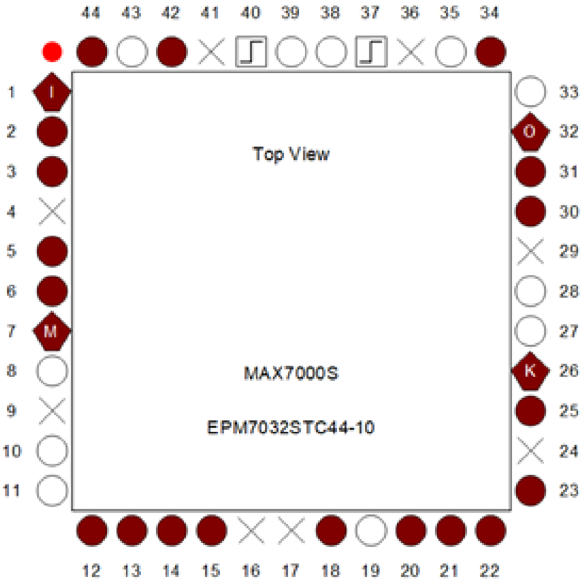
4. Once the code has compiled, set the pin layout as follows:



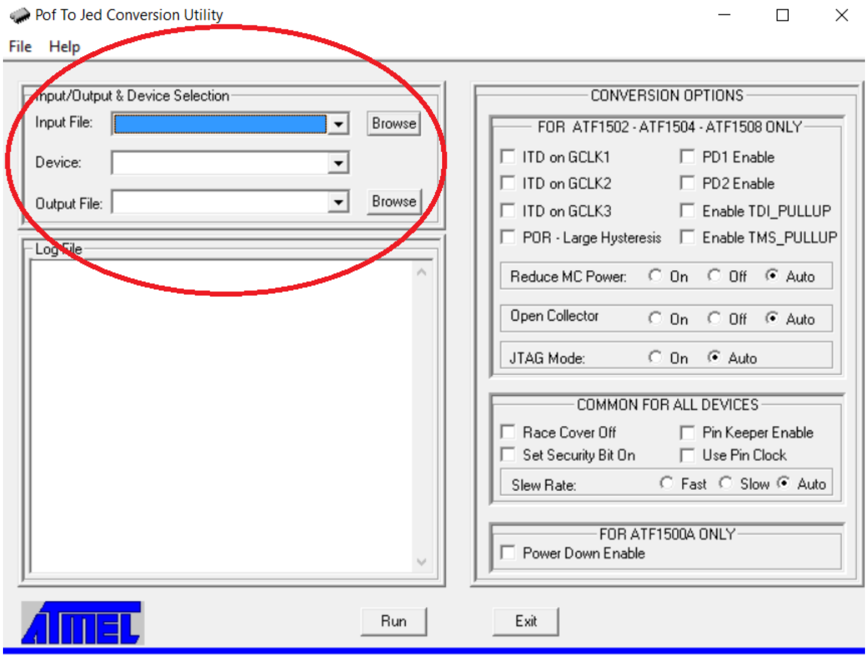
```
USE ieee.nume      td.all;


entity S4_FSM is

   port (
      -- inputs
      clk : in std_logic;                 --typically 100 MHz
      pwm : in std_logic;                 --typically between 1kHz and 1 MHz.
      ready_H, ready_L : in std_logic;    --ready signal indicating GD's can be used.
      fault_H, fault_L : in std_logic;    --DESAT fault signal
      reset_input : in std_logic;         --Global reset signal

      ----------------------------------------------------------


      -- outputs
            --Signals that send the modified PWM signal to High- and Low side Gate Drivers.
      output_high, output_low                      : out std_logic;
            --Signals that enable the buffer on the High- and/or Low side.
      enable_high, enable_low                      : out std_logic;
            --Signals that turn on or off the ready LED for the High- and/or Low side.
      ready_led_H, ready_led_L                    : out std_logic;
            --Signals that turn on or off the fault LED for the High- and/or Low side.
      fault_led_H, fault_led_L                    : out std_logic;
            --Signals that turn on/off the four PWM LED's to show which PWM signal is being used.
      pwm_led_1, pwm_led_2, pwm_led_3, pwm_led_4          : out std_logic
   );
end entity;
```



Node Name	Direction	Location	Fitter Location
clk	Input	PIN_42	PIN_42
enable_high	Output	PIN_22	PIN_22
enable_low	Output	PIN_15	PIN_15
fault_H	Input	PIN_18	PIN_18
fault_L	Input	PIN_12	PIN_12
fault_led_H	Output	PIN_25	PIN_25
fault_led_L	Output	PIN_31	PIN_31
output_high	Output	PIN_21	PIN_21
output_low	Output	PIN_14	PIN_14
pwm	Input	PIN_34	PIN_34
pwm_led_1	Output	PIN_6	PIN_6
pwm_led_2	Output	PIN_5	PIN_5
pwm_led_3	Output	PIN_3	PIN_3
pwm_led_4	Output	PIN_2	PIN_2
ready_H	Input	PIN_20	PIN_20
ready_L	Input	PIN_13	PIN_13
ready_led_H	Output	PIN_23	PIN_23
ready_led_L	Output	PIN_30	PIN_30
reset_input	Input	PIN_44	PIN_44
TCK	Input	PIN_26	PIN_26
TDI	Input	PIN_1	PIN_1
TDO	Output	PIN_32	PIN_32
TMS	Input	PIN_7	PIN_7



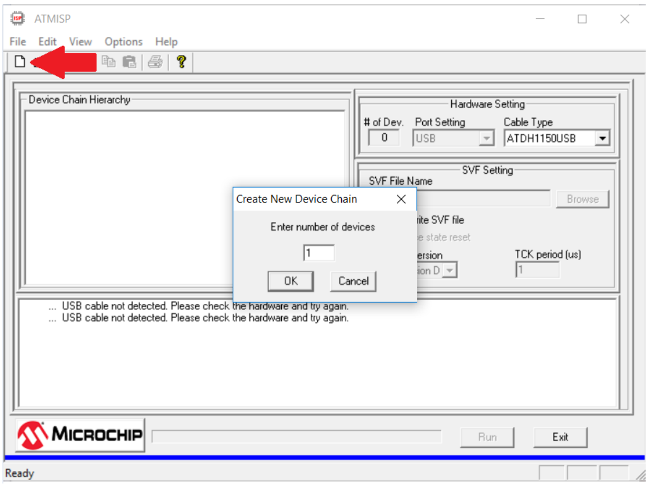
5. When this is done, run the program one more time and then open the POF2JED software.



□ ITD on GCLK1 □ PD1 Enable
□ ITD on GCLK2 □ PD2 Enable
□ ITD on GCLK3 □ Enable TDI_PULLUP
□ POR - Large Hysteresis □ Enable TMS_PULLUP
Reduce MC Power: ○ On ○ Off ● Auto
Open Collector ○ On ○ Off ● Auto
JTAG Mode: ○ On ● Auto
□ Race Cover Off □ Pin Keeper Enable
□ Set Security Bit On □ Use Pin Clock
Slew Rate: ○ Fast ○ Slow ● Auto
□ Power Down Enable

a. Enter the .pof file that was created by Quartus into the input file section.
b. Set a folder for the output to create a JEDEC file.
c. Click Run, the program now creates a .JEDEC file.

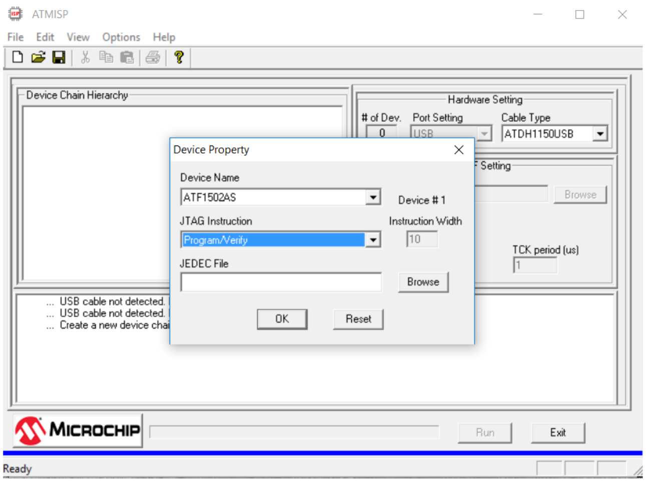
6. Make sure the device is connected to the PCB, and open the ATMISP software



# of Dev. 0 Port Setting USB Cable Type ATDH1150USB

Create New Device Chain
Enter number of devices
1

... USB cable not detected. Please check the hardware and try again.
... USB cable not detected. Please check the hardware and try again.



# of Dev. 0 Port Setting USB Cable Type ATDH1150USB

Device Property
Device Name ATF1502AS
Device # 1
JTAG Instruction Program/Verify
Instruction Width 10
JEDEC File

... USB cable not detected.
... USB cable not detected.
... Create a new device chai

a. Click “new” in the top left corner, click “ok”.
b. Select ATF1502AS for the Device Name.
c. Select Program/Verify.
d. Select JEDEC file that was created in step 5.
e. Click “ok” and “Run”. The light on the device should turn from green to red and back to green once the code has been successfully uploaded.

7. The code should now be uploaded to the device.

Adjusting the VHDL Code

The code can be found at the end of this document.

In the VHDL code the constant values after the line “ONLY CHANGE THE CONSTANT VALUES AFTER ":=" “ can be adjusted.

The options include changing the:

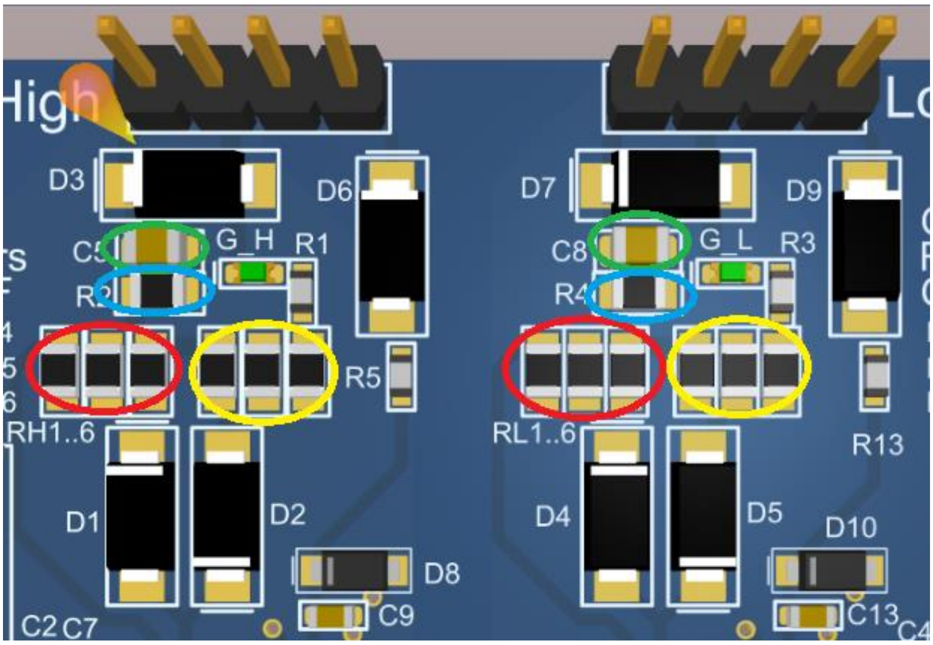
1. Dead-time (variable name: deadtime).
2. Turning on/off one or both of the gate drivers(output_high_enable, output_low_enable).
3. Setting a LED to indicate which incoming PWM signal is used. (led on)

The dead-time currently has a minimum setting of 50ns and takes steps of 50ns (i.e. entering “3” results in a dead-time of 150ns). This is due to the clock speed being set to 20 MHz. If the clock speed is changed this will affect the nanoseconds per step taken and thus the code should be adapted accordingly.

Adjusting the Hardware Components

Several components of the output phase (see image) can be changed to suit the specific needs of the transistor. These are:

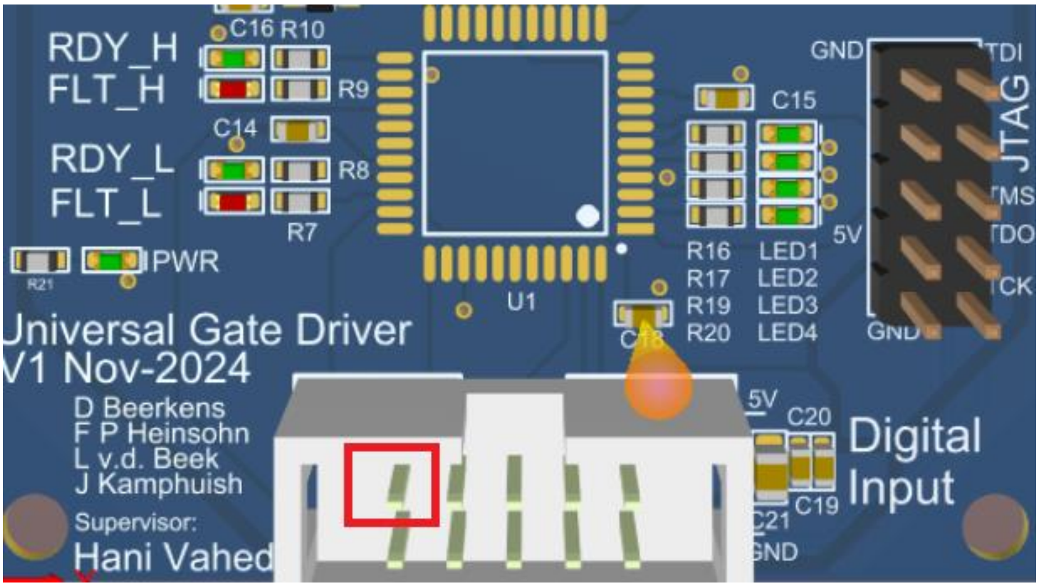
1. Gate Resistor: both turn on (Red) and turn off (Yellow) for both the high- and low side of the device consist of three resistors that can be replaced and set.
2. Gate-to-source Resistor can be changed (Blue).
3. Gate-to-source Capacitor can be changed(Green).



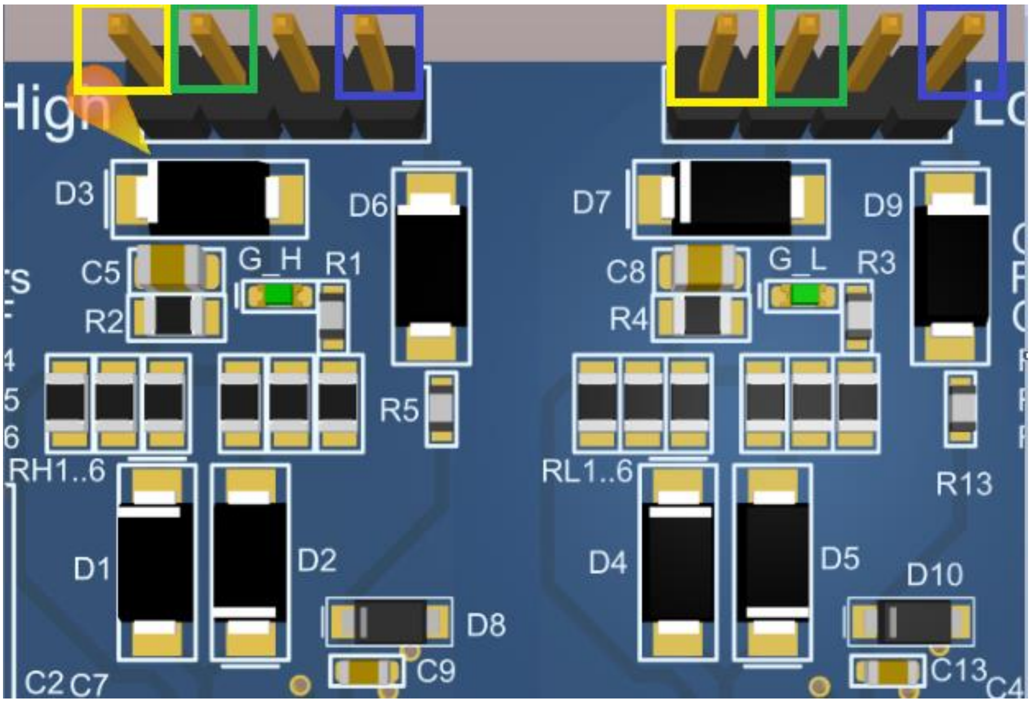
Connecting the Input and Output

Once everything of the device has been set up, the incoming PWM signal and the outgoing signals to the transistors should be attached as follows:

1. Attach the PWM signal to the PWM input connector. PWM signal should be on the top left pin(Red), see image below.



2. Attach the transistor in the manner as seen in the image below: Gate(Yellow)-Source(Green)-Drain(Blue).



VHDL Code:

```
LIBRARY ieee;

USE ieee.std_logic_1164.all;

USE ieee.numeric_std.all;



entity S4_FSM is


        port (

                -- inputs

                clk : in std_logic;              --typically 100 MHz

                pwm : in std_logic;              --typically between 1kHz and 1 MHz.

                ready_H, ready_L : in std_logic;  --ready signal indicating GD's can be used.

                fault_H, fault_L : in std_logic;  --DESAT fault signal

                reset_input : in std_logic;       --Global reset signal


                ----------------------------------------------------------


                -- outputs

       --Signals that send the modified PWM signal to High- and Low side Gate Drivers.

                output_high, output_low                         : out std_logic;

       --Signals that enable the buffer on the High- and/or Low side.

                enable_high, enable_low                         : out std_logic;

       --Signals that turn on or off the ready LED for the High- and/or Low side.

                ready_led_H, ready_led_L                        : out std_logic;

       --Signals that turn on or off the fault LED for the High- and/or Low side.

                fault_led_H, fault_led_L                        : out std_logic;

       --Signals that turn on/off the four PWM LED's to show which PWM signal is being used.

                pwm_led_1, pwm_led_2, pwm_led_3, pwm_led_4          : out std_logic
        );
end entity;

architecture behavioural of S4_FSM is




-- ONLY CHANGE THE CONSTANT VALUES AFTER ":="

--Change this value to set a different Dead-Time. Steps are of size 50ns (i.e. 6 = 300ns)
--due to the fact that the current clock speed is 20 MHz.
constant deadtime              : integer range 1 to 16     := 6;

--Set to '1' to turn off the high side buffer. Set to '0' to turn on high side buffer.
constant output_high_enable    : std_logic                 := '0';

--Set to '1' to turn off the low side buffer. Set to '0' to turn on low side buffer.
constant output_low_enable     : std_logic                 := '0';

--Set from 1-4 to indicate which PWM input is being used.
constant led_on                : integer range 1 to 4      := 1;

---------------------------------------------------


-- These are the states used for the FSM.
type states is
(
    --Reset state where both outputs to the buffer/Gate Driver are low.
    --Using the Reset input puts the system in this state.
    --Incoming PWM signal moves the system to the counter_H state.
        reset,
    --High side count state. Both outputs to the buffer/Gate Driver are low.
    --Once the count threshold has been reached the system moves to the hold_high state.
        counter_H,

    --Low side count state. Both outputs to the buffer/Gate Driver are low.
    --Once the count threshold has been reached the system moves to the hold_low state.
        counter_L,

    --Hold high state. In this state the high side output to the buffer/Gate Driver is high,
    --while the low side output is low. A falling edge of the PWM signal moves the system to
    --the counter_L state.
        hold_high,

    --Hold low state. In this state the low side output to the buffer/Gate Driver is high,
    --while the high side output is low. A rising edge of the PWM signal moves the system to
    --the counter_H state.
        hold_low
);



--------------------------------------------------
-- signal declarations
signal state, new_state : states := reset;
signal ID, ire, ile : std_logic;
signal state_check: std_logic;

--------------------------------------------------
begin

process(clk, reset_input, state, pwm, ready_H, ready_L, ire, ile, ID,fault_H, fault_L)
```

```
--Integer count is set to 1 to account for the state update running one clock signal behind.
variable count : integer range 0 to 32 := 1;


begin

    --Clock signal that updates the current state.
        if (rising_edge(clk)) then

                Id <= pwm;

       --Detection of a rising edge in the PWM signal.
                ire <= not ID and pwm;

       --Detection of a falling edge in the PWM signal.
                ile <= ID and not pwm;

       --If the Reset input is high, the state is kept at Reset state.
                if (reset_input = '1') then
                        state <= reset;

       --If Reset is low and both Gate Drivers are ready, the states are updated to
       --their new states.
                elsif (ready_H = '1' and ready_L = '1') then
                        state <= new_state;

       --If the Gate Drivers are not ready, the system is kept in the Reset state even
       --if the Reset input isn't high.
                else
                        state <= reset;
        end if;


        -- clock cycle counter
    --When in either the counter_H or counter_L state the count value is updated,
    --if it is in any other state the value is set back to 1.
        if (state = counter_H) then
            count := count + 1;
        elsif (state = counter_L) then
            count := count + 1;
        else
            count := 1;
        end if;
    else
    end if;

------------------------------------------------------


--Code to turn on/off one or both of the ready LEDs indicating the Gate Drivers are ready.


    if (ready_H = '1' and ready_L = '1') then
        ready_led_H <= '1';
        ready_led_L <= '1';
    elsif (ready_H = '0' and ready_L = '1') then
        ready_led_H <= '0';
        ready_led_L <= '1';
    elsif (ready_H = '1' and ready_L = '0') then
        ready_led_H <= '1';
        ready_led_L <= '0';
    else
        ready_led_H <= '0';
        ready_led_L <= '0';
    end if;

------------------------------------------------------

--Code to turn on/off one or both of the fault LEDs indicating the Gate Drivers are
--experiencing a fault.

    if (fault_H = '0') then
        fault_led_H <= '1';
    else
        fault_led_H <= '0';
    end if;

    if (fault_L = '0') then
        fault_led_L <= '1';
    else
        fault_led_L <= '0';
    end if;

------------------------------------------------------

--Code to turn on/off a LED to show which PWM signal is being used.

    if (led_on = 1) then
        pwm_led_1 <= '1';
        pwm_led_2 <= '0';
        pwm_led_3 <= '0';
        pwm_led_4 <= '0';
    elsif (led_on = 2) then
        pwm_led_1 <= '0';
        pwm_led_2 <= '1';
```

```
                pwm_led_3 <= '0';

                pwm_led_4 <= '0';

        elsif (led_on = 3) then

                pwm_led_1 <= '0';

                pwm_led_2 <= '0';

                pwm_led_3 <= '1';

                pwm_led_4 <= '0';

        else

                pwm_led_1 <= '0';

                pwm_led_2 <= '0';

                pwm_led_3 <= '0';

                pwm_led_4 <= '1';

        end if;


-----------------------------------------------------


-- FSM states


        case state is


       --During Reset State the outputs are kept low, whilst the buffer enables

       --are kept high to turn off the buffers.

                when reset =>


                        output_high         <= '0';

                        output_low              <= '0';

                        enable_high         <= '1';

                        enable_low              <= '1';



                        if (ire = '1') then
```

```
            new_state <= hold_high;
        else
            new_state <= reset;
        end if;

------------------------------------------------------

    when counter_H =>

        output_high        <= '0';
        output_low             <= '0';
        enable_high        <= output_high_enable;
        enable_low             <= output_low_enable;

        if (count >= deadtime) then
            new_state <= hold_high;
        else
            new_state <= counter_H;
        end if;

------------------------------------------------------

    when counter_L =>

        output_high        <= '0';
        output_low             <= '0';
        enable_high        <= output_high_enable;
        enable_low             <= output_low_enable;

        if (count >= deadtime) then
            new_state <= hold_low;
```

```
                else
                        new_state <= counter_L;
                end if;

------------------------------------------------------

        when hold_high =>

                output_high                     <= '1';
                output_low                       <= '0';
                enable_high                     <= output_high_enable;
                enable_low                       <= output_low_enable;


                if (ile = '1') then
                        new_state <= Counter_L;
                else
                        new_state <= hold_high;
                end if;

------------------------------------------------------

        when hold_low =>

                output_high             <= '0';
                output_low                     <= '1';
                enable_high             <= output_high_enable;
                enable_low                     <= output_low_enable;


                if (ire = '1') then
                new_state <= Counter_H;
            else
                new_state <= hold_low;
            end if;

    ------------------------------------------------------------
     --In the others state both output PWM signals are kept low, and the
     --buffers are turned off.
            when others =>

                output_high    <= '0';
                output_low     <= '0';
                enable_high    <= '1';
                enable_low     <= '1';

    end case;

end process;

end behavioural;
```

H

Time Management and Task Division

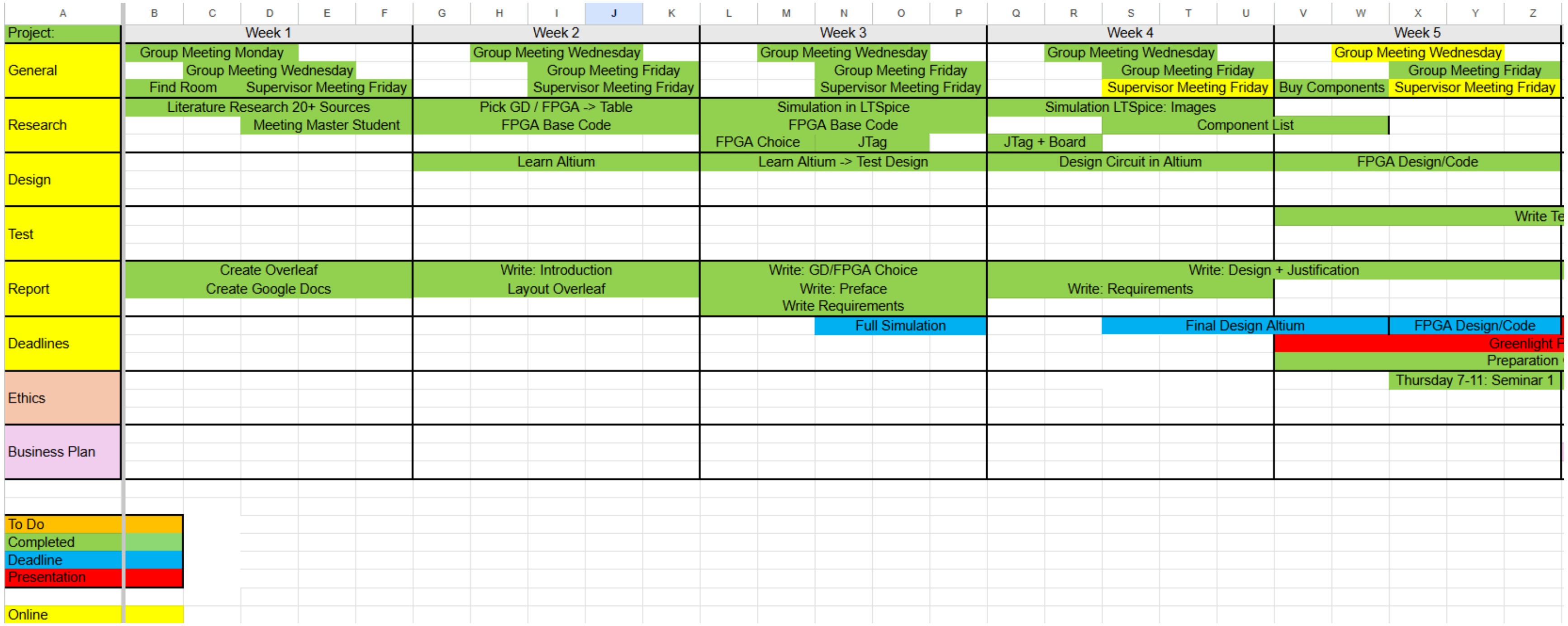
In this appendix the way in which the planned out our project, as well as the division of tasks among the group members, will be shown.

In figures H.1 and H.2 the Gantt Charts that were made to keep track of our project can be seen. Although it proved impossible to completely follow the chart as initially set out, it provided an overall structure with goals to work towards. When needed adaptations were made. During weekly meetings tasks were then further subdivided.

In table H.1, the main tasks performed throughout the project are given on the left column while the group members who mainly contributed towards these tasks are given on the right column.

Table H.1: Contributions increase from left to right.

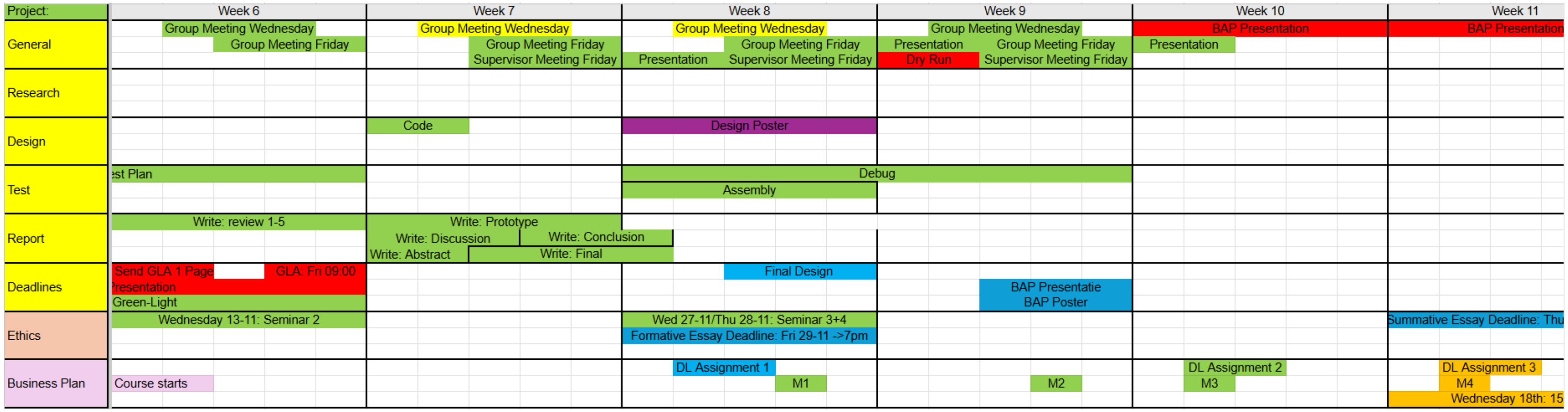
Tasks	**Contributors**
Literature research	Group effort
CPLD Choice	Jop, Dries
Gate Driver Choice	Fede, Jop
Buffer Choice	Fede
VHDL Code	Luc, Dries
JTAG Implementation	Dries , Jop
Altium circuit design	Dries, Luc
LTspice Simulation	Luc, Fede
Prototype building & evaluation	Dries, Luc, Fede
Finding other circuit components	Dries
Project Management	Luc, Jop
Thesis Layout	Fede
Thesis Writing	Group effort
Presentations	Group effort



Project:	Week 1	Week 2	Week 3	Week 4	Week 5
General	Group Meeting Monday	Group Meeting Wednesday	Group Meeting Wednesday	Group Meeting Wednesday	Group Meeting Wednesday
	Group Meeting Wednesday	Group Meeting Friday	Group Meeting Friday	Group Meeting Friday	Group Meeting Friday
	Find Room Supervisor Meeting Friday	Supervisor Meeting Friday	Supervisor Meeting Friday	Supervisor Meeting Friday	Buy Components Supervisor Meeting Friday
Research	Literature Research 20+ Sources	Pick GD / FPGA -> Table	Simulation in LTSpice	Simulation LTSpice: Images	
	Meeting Master Student	FPGA Base Code	FPGA Base Code	Component List	
			FPGA Choice JTag	JTag + Board	
Design		Learn Altium	Learn Altium -> Test Design	Design Circuit in Altium	FPGA Design/Code
Test					Write Te
Report	Create Overleaf	Write: Introduction	Write: GD/FPGA Choice	Write: Design + Justification	
	Create Google Docs	Layout Overleaf	Write: Preface	Write: Requirements	
			Write Requirements		
Deadlines			Full Simulation	Final Design Altium	FPGA Design/Code
					Greenlight F
					Preparation
Ethics					Thursday 7-11: Seminar 1
Business Plan					

To Do	🟧
Completed	🟩
Deadline	🟦
Presentation	🟥
Online	🟨

Figure H.1: Gantt Chart created for weeks 1-5



Project:	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
General	Group Meeting Wednesday Group Meeting Friday	Group Meeting Wednesday Group Meeting Friday Supervisor Meeting Friday	Group Meeting Wednesday Group Meeting Friday Presentation Supervisor Meeting Friday	Group Meeting Wednesday Presentation Group Meeting Friday Dry Run Supervisor Meeting Friday	BAP Presentation Presentation	BAP Presentation
Research						
Design		Code	Design Poster			
Test	est Plan		Debug Assembly			
Report	Write: review 1-5	Write: Prototype Write: Discussion Write: Conclusion Write: Abstract Write: Final				
Deadlines	Send GLA 1 Page GLA: Fri 09:00 resentation Green-Light		Final Design	BAP Presentatie BAP Poster		
Ethics	Wednesday 13-11: Seminar 2		Wed 27-11/Thu 28-11: Seminar 3+4 Formative Essay Deadline: Fri 29-11 ->7pm			Summative Essay Deadline: Thu
Business Plan	Course starts		DL Assignment 1 M1	M2	DL Assignment 2 M3	DL Assignment 3 M4 Wednesday 18th: 15

Figure H.2: Gantt Chart created for weeks 6-11