

## **ABSTRACT**

BO GAO. Scalable Medium Voltage and High Voltage Super Cascode Power Modules. (Under the direction of Dr. Douglas Hopkins).

This research investigates the fundamental understanding of normally-on JFET super cascode power modules (SCPMs). It then proposes an SCPM structure designed to enhance switching performance, thermal performance and manufacturability. A novel approach for drastically improving avalanche energy handling capability is proposed and verified. A mathematical model of balancing network loss is summarized and based on the model, a methodology for optimizing switching loss is proposed. The optimization methodology is implemented as several computer programs for assisting the designing of different of SCPM topologies proposed in this work. Several 6.5kV demonstration modules are designed, fabricated and tested with a curve tracer, HV probing and double pulse testing. Future possibilities of the SCPM technology are proposed for increasing switching megavolt levels and enabling solid state power controllers and pulse power applications.

© Copyright 2018 by Bo Gao

All Rights Reserved

Scalable Medium Voltage and High Voltage Super Cascode Power Modules

by  
Bo Gao

A dissertation submitted to the Graduate Faculty of  
North Carolina State University  
in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina

2018

APPROVED BY:

---

Douglas Hopkins  
Committee Chair

---

Subhashish Bhattacharya

---

Srdjan Lukic

---

Gregory Buckner

## **DEDICATION**

This work is dedicated to my parents who gave my life and supported me through my education, research and works.

## **BIOGRAPHY**

Bo Gao, born May 20<sup>th</sup> 1992, obtained BSEE from Lanzhou University, China in 2014, MSEE from NC State University, US in 2015. His research includes power semiconductor modules and semiconductor-based electrical power converters, power management integrated circuits, embedded systems engineering and precision analog and digital signal processing.

## **ACKNOWLEDGMENTS**

This work was sponsored by Power America Institute and United Silicon Carbide Incorporated.

The author thanks the help from NCSU PREES Laboratory, United Silicon Carbide

Incorporated, Hesse Mechatronics, Wacker Chemie and NCSU FREEDM Systems Center. The

author also wishes to thank my colleagues, Yang Xu, Adam Morgan, Xin Zhao, Thomas Ballard

and Dr. Douglas Hopkins, for their tremendous amount of help provided for this work.

## TABLE OF CONTENTS

LIST OF TABLES .....	vii
LIST OF FIGURES .....	viii
<b>Chapter 1: Introduction and Background</b> .....	1
Requirements for medium voltage switches .....	1
Structure of this dissertation .....	3
<b>Chapter 2: Review of SCPM Structure</b> .....	4
Existing SCPM structures and limitations .....	4
Problems to be solved by this research .....	12
<b>Chapter 3: Proposed Single SCPM Topology</b> .....	18
Mathematical modelling of switching losses .....	18
Impact of layout optimization for manufacturability .....	22
<b>Chapter 4: Development of an SCPM Implementation</b> .....	28
Electrical interconnection structure .....	28
Thermal simulation and optimization .....	32
Electrical simulation .....	36
Static and dynamic characterization .....	43
<b>Chapter 5: Increasing Avalanche Energy Handling Capability</b> .....	52
Necessity of avalanche energy handling capability .....	52
Directing avalanche energy from diodes to JFETs .....	55
Simulation and verification .....	56
<b>Chapter 6: Scaling Multiple SCPMs</b> .....	62
Using SCPMs as higher voltage JFETs .....	62

Optimizing series connected SCPM design.....	66
Using SCPMs as higher voltage MOSFETs .....	71
Triggering of series connected SCPMs through optical fiber.....	76
<b>Chapter 7: Conclusion and Future Work.....</b>	<b>86</b>
Conclusion of research.....	86
Application on device testing of future semiconductor devices .....	88
Application on solid state circuit breakers and power controllers .....	89

## LIST OF TABLES

Table 2.1	Power switch operation condition comparison of common converters .....	13
Table 4.1	Thermal simulation parameters .....	32
Table 4.2	Capacitor configuration of proposed SCPM .....	36
Table 4.3	Proposed SCPM simulation result.....	37
Table 5.1	Average power dissipation comparison .....	57
Table 6.1	EB value for all possible configurations at N=20 .....	67
Table 6.2	Total cost factor for all possible configurations at N=20.....	68
Table 6.3	Figure of merit for all possible configurations at N=20.....	68
Table 6.4	Simulation result of 2*2*5 2-layer SCPM-J compared with 20-stage SCPM .....	71
Table 6.5	Calculated EB value of SCPM topologies compared in this work.....	85

## LIST OF FIGURES

Figure 2.1	Friedrichs' SCPM.....	4
Figure 2.2	Biela's SCPM and Biela's SCPM with better tolerance to mismatching.....	6
Figure 2.3	Biela's SCPM (turn-on process, only Q5 stage is shown) .....	7
Figure 2.4	Biela's SCPM (turn-off process, only Q5 stage is shown).....	8
Figure 2.5	Li's SCPM.....	10
Figure 2.6	Song's SCOM.....	12
Figure 2.7	Current spike during hard switching in a non-commutating half bridge .....	14
Figure 2.8	Avalanche operation of half bridge switches .....	15
Figure 2.9	Size comparison between a power JFET die and a packaged avalanche diode .....	16
Figure 2.10	Current path of Biela's SCOM during an avalanche breakdown .....	17
Figure 3.1	Zero state non-ZVS capacitor charging model.....	19
Figure 3.2	Physical representation of interested sources of loss .....	20
Figure 3.3	Proposed single SCPM topology.....	23
Figure 3.4	Simulation bench for extracting balancing resistor energy dissipation.....	26
Figure 4.1	Rendering of proposed demonstration SCPM.....	29
Figure 4.2	DBC layout of demonstration SCPM.....	29
Figure 4.3	Current distribution and inductance extraction with flat bar model.....	30
Figure 4.4	Current distribution and inductance extraction with bonding wire model .....	31
Figure 4.5	Drain pad margin vs thermal resistance curve .....	33
Figure 4.6	Thermal simulation model and result .....	34
Figure 4.7	Overall schematic of demonstration SCPM .....	35
Figure 4.8	Demonstration SCOM simulation bench .....	38

Figure 4.9 Simulated switching waveform of demonstration SCPM .....	39
Figure 4.10 Pin-fin baseplate used in water cooled SCPM design.....	40
Figure 4.11 Sloped coolant cavity design for compensating power dissipation difference .....	41
Figure 4.12 Simulated water temperature across baseplate with new design.....	41
Figure 4.13 Simulated water temperature across baseplate with conventional design.....	42
Figure 4.14 SCPM S0 leakage current vs voltage .....	43
Figure 4.15 SCPM S1 leakage current vs voltage .....	44
Figure 4.16 SCPM S1 on-state curve tracer test result .....	45
Figure 4.17 DPT electrical diagram.....	47
Figure 4.18 DPT test setup .....	48
Figure 4.19 DPT at 4kV, 110A.....	49
Figure 4.20 Turn on detail at 3kV, 50A.....	50
Figure 4.21 Turn off detail at 3kV, 50A.....	50
Figure 5.1 Series connecting SCPMs for higher blocking voltage.....	54
Figure 5.2 Topology with avalanche breakdown immunity showing diode resistors .....	55
Figure 5.3 Simulation bench for avalanche current distribution .....	58
Figure 5.4 Gate current of J6 before and during avalanche breakdown .....	59
Figure 5.5 Topology with avalanche breakdown immunity showing gate resistor .....	60
Figure 5.6 Gate current of J6 before and after avalanche breakdown with gate resistor.....	61
Figure 6.1 Unit SCPM stack structure of SCPM-J .....	62
Figure 6.2 SCPM-J using individual Unit SCPMs as JFETs.....	64
Figure 6.3 Simulation bench of 2*2*5 SCPM-J .....	69
Figure 6.4 SCPM-M using individual Unit SCPMs as MOSFETs.....	72

Figure 6.5	SCPM-M with bootstrapping gate drivers.....	75
Figure 6.6	SCPM-M with optical fiber coupled gate drivers .....	77
Figure 6.7	Proposed optical fiber coupled gate driver.....	78
Figure 6.8	Transmitter circuit of proposed optical fiber coupled gate driver.....	78
Figure 6.9	Hardware logic for generation of laser driving signals .....	79
Figure 6.10	TEC controller, LMT01 decoder and system management function blocks.....	80
Figure 6.11	Prototype optical fiber coupled gate driver transmitter without laser diode .....	81
Figure 6.12	Prototype optical fiber coupled gate driver transmitter with laser diode .....	81
Figure 6.13	Receiver circuit of proposed optical fiber coupled gate driver .....	82
Figure 6.14	Hardware logic for recovering gate signal from modulated laser .....	83
Figure 6.15	Prototype of optical fiber coupled gate driver receiver PCB .....	84
Figure 7.1	H-bridge energy recirculation converter with boost converter pre-regulator.....	88

## Chapter 1: Introduction and Background

### 1.1: Requirements for medium voltage switches

Modern world runs on electricity, and with the depletion of fossil fuel, more renewable energy sources are being developed, such as PV, wind and geothermic. Most renewable energy sources are far from densely populated areas. Thus, the generated energy needs to be transmitted hundreds or even thousands of miles to power consumers. Stepping up voltage increases transmission efficiency by reducing resistive loss, but as voltage increases, capacitive loss starts to show its negative effects.

For instance, for transmitting 1MW of power at 10kV, the required current is 100A. At 50kV, the current can be reduced to 20A (both assuming 1MW transmitted from source, not received by consumer). For the same cable resistance, power dissipation caused by  $I^2R$  loss is reduced by 96%. On the other hand, as a side effect, capacitive coupling power to ground and between wires increases by 24 times. This limits how high voltage can be used. However, it needs to be pointed out that despite the increased capacitive coupling, this may not be unintended due to most load on the grid is inductive anyway, so this stray capacitance can be used to improve power factor. Nevertheless, if the AC is replaced with high voltage DC, the power factor issue and the capacitive coupling issue can be solved altogether.

Therefore, HVDC is being favored to build ultra-high capacity grids. Another loss in power transmission is the distribution network. Running mains voltage is always lossy compared to running higher voltage as far as possible, preferably right before consumers' service entrance panel. A point of load solid state transformer (POL SST) then drops distribution voltage to mains voltage near the clients' service entrance panel. Regardless HVDC or POL SST, they all require

power semiconductor switches to do power conversion, and the voltage requirement is usually ranging from a few kV to more than 1MV<sup>[1]</sup>. Additionally, industrial motor drives and electrified locomotives are also high-power applications, sometimes more than 20MW (19.8MW motor power for CRRC-Siemens Velaro CRH380CL, plus power conversion loss and lighting/AC power consumption), that can benefit from medium voltage switches.

For a number of reasons, medium voltage (6.5kV and above) SiC power semiconductors are yet not widely available<sup>[2]</sup>. This calls for the use of novel structures to series connect low voltage (1.2kV~3.3kV) power semiconductor devices to achieve the higher operating voltage.

Furthermore, if simultaneous switching can be achieved, series connected power switches can be further series connected to switch higher voltage for grid connected high voltage applications.

Although there are 6.5kV and above Si-IGBT devices widely available, their poor switching performance and inherent instability compared with SiC unipolar junction devices make them less favorable for technical reasons in modern high frequency power electronics<sup>[3]</sup>. For instance, at 1.1kHz, 6.5kV IGBTs will have more switching loss than its maximum allowable power dissipation<sup>[2]</sup>. As a result, the operating frequency is limited to a few hundreds of Hz, which is not much higher than the mains frequency. This leads to slower control, higher harmonics, and the need of massive LC filtering components in power electronic circuits.

A cascade/cascode topology was used for decades for amplifying linear signals to higher voltages beyond what a single semiconductor device rating. Super cascode topology was recently introduced to allow the use of readily available 1.2kV SiC power semiconductor devices to create higher voltage switching blocks.

## **1.2: Structure of this dissertation**

Though a super cascode topology can be used with any depletion mode devices, such as D-MOSFETs, JFETs, vacuum tubes and even MEMS switches, in this research we specifically address the super cascode using normally-on SiC JFET devices. A review of existing SiC super cascode power module (SCPM) topologies and their principle of operation, as well as shortcomings are presented. Then, a mathematical model for designing and analyzing the balancing network of a SCPM is summarized. Based on the analysis, a new SCPM topology optimized for manufacturability and performance is proposed. A few demonstration modules are designed, fabricated and tested to prove the design concept.

Based on the newly proposed SCPM topology, two methods for further increasing switching voltage are proposed, namely SCPM-J (Mega Cascode-J) and SCPM-M (Mega Cascode-M), each with pros and cons compared with each other. Based on loss models, a computer program is developed to automatically find the most optimum design based on component parameters and target switching voltage.

Finally, future possibilities enabled by this research are discussed.

## Chapter 2: Review of SCPM Structure

### 2.1: Existing SCPM structures and limitations

The cascode connection of semiconductor devices originates back as a method to amplify linear signal. Super Cascode as a method for constructing power electronics switch blocks was initially proposed by Peter Friedrichs, et al in 2003 <sup>[4]</sup>, and the concept of having a dynamic balancing network was further proposed by J. Biela, et al in 2009 <sup>[5]</sup>. Xueqing Li, et al have proposed a new SCPM topology using different SiC JFET devices <sup>[6]</sup>. Because all SCPM topologies discussed here share the same power path which consists of several JFETs plus a single MOSFET connected in series, “SCPM topologies” in this paper refers to “SCPM balancing network” topologies.

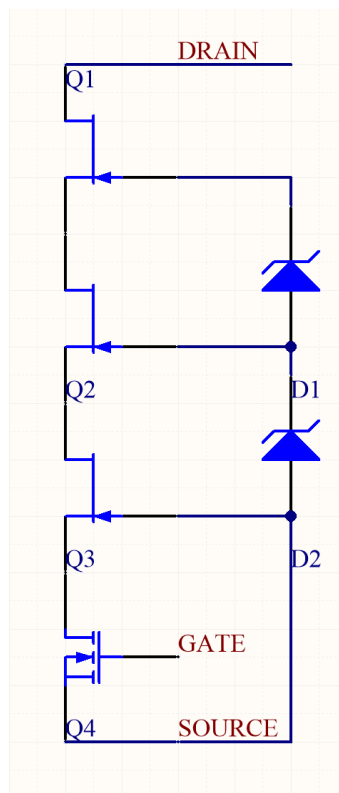


Figure 2.1: Friedrichs' SCPM <sup>[4]</sup>

All SCPMs share the same fundamental principle, sequential switching. In Friedrichs' structure, when MOSFET Q4 turns on, source potential of Q3 is pulled down to module source, which is gate potential. This allows the depletion mode switch Q3 to turn on. As Q3 turns on, it pulls source potential of Q2 to module source level. Due to the existence of D2, gate potential of Q2 is never below negative  $V_F$ , hence when source potential is pulled to module source, Q2 turns on. The same process then turns Q1 on, allowing the SCPM to conduct current.

When Q4 turns off, on-state current through the SCPM charges body charge of Q4 and pulls up source potential of Q3 to a point where Q3 stops conducting due to channel pinching. As Q3 turns off, source potential of Q2 rises, increasing negative gate to source voltage, thus pinching off Q2. Similarly, Q1 turns off by the same mechanism.

When the SCPM reaches turn-off equilibrium, drain to gate leakage current of Q1 flows through D1 and D2 to ground, while drain-to-gate leakage current of Q2 flows through D2 to module source. Since drain-to-source voltage of each stage is equal to drain-to gate voltage minus absolute value of pinch-off voltage, and pinch-off voltage is approximately the same for all JFET devices in an SCPM, here it can be assumed that drain-to-source voltage of each device is only dictated by voltage across the gate-to-gate diode. In this case,  $V_{DS2} \sim V_{D1}$ ,  $V_{DS3} \sim V_{D2}$ . The Q1 drain-to-gate voltage is not clamped by any diodes and hence is equal to module drain-to-source voltage minus all diode voltages,  $V_{D1} + V_{D2}$ . The operating point of an SCPM during its steady state turn-off is referred as static balancing throughout the paper.

The Friedrichs' structure demonstrated 100ns turn off (voltage rise) and 180ns turn on (voltage fall) <sup>[4]</sup>. The current rise and fall time are both less than 40ns. The slow voltage turn-off and turn-

on time severely limits the switching frequency to 10kHz. Despite the 10kHz limitation, the proposed structure is still much faster than Si-IGBT (3860ns and 410ns respectively [3]).

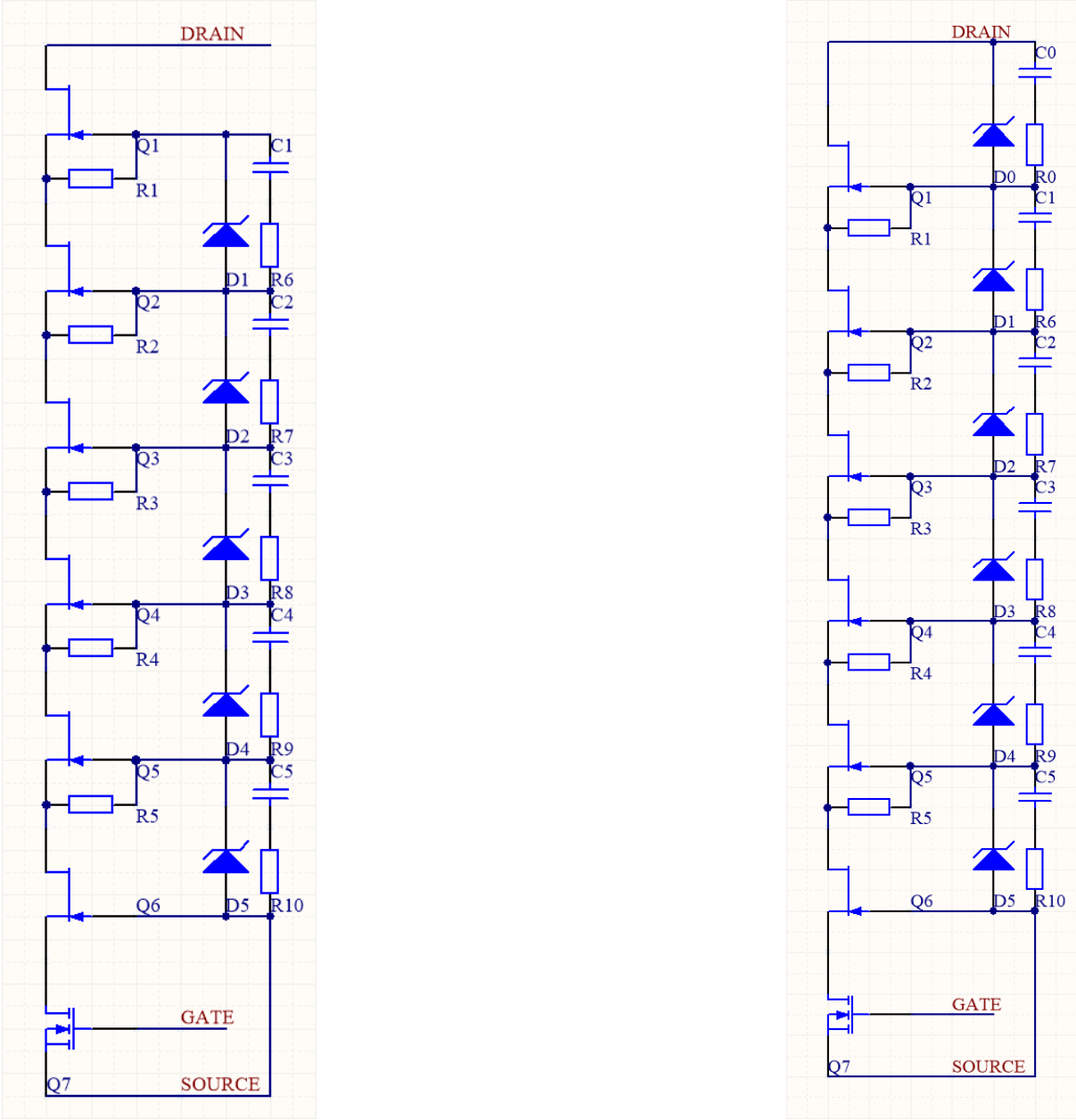


Figure 2.2: Biela's SCPM (left) and Biela's SCPM with better tolerance to mismatching (right) [5]

To improve switching speed, the Kolar team (Biela, et al) from ETH added a capacitive balancing network to Friedrichs' structure, Fig. 2 left. In this structure, in addition to the static

balancing discussed before, there is a dynamic balancing circuit providing charge balance through capacitors C1~C5. The avalanche diodes D1-D5 serve the same purpose as in Friedrichs' structure while C1-C5 and R6-R10 serve as a dynamic balancing network.

When Q7 turns on, source voltage of Q6 is pulled to module source, allowing Q6 to turn on. As Q6 turns on, source voltage of Q5 is pulled module ground. In the meantime, C5 charges the gate capacitance of Q5, accelerating turn on speed of Q5. As Q5 is turning on, source voltage of Q5 collapses, allowing C4 to charge the gate capacitance of Q4. This chain reaction propagates to Q1, allowing Q1 to be turned on with acceleration from C1. Ideally, C1 supports all the charge of Q1 gate, C2 supports all the charge of Q2 gate, plus charge from C1 flowing down, C3 supports all the charge of Q3 gate, plus all the charge from C1 and C2 flowing down, etc. Therefore,  $\Delta Q_{C1} = Q_{G1} - Q_{D1}$ ,  $\Delta Q_{C2} = \Delta Q_{C1} + Q_{G2} - Q_{D2}$ , etc. In a perfectly balanced system where the voltage across all JFETs are the same, and all JFETs and diodes are matched, then  $\Delta Q_{Cn} = n * (Q_G - Q_D)$ , hence  $C_n = (Q_G - Q_D) / V_{DS} * N$ , where  $V_{DS}$  is drain to source voltage of each individual JFET.

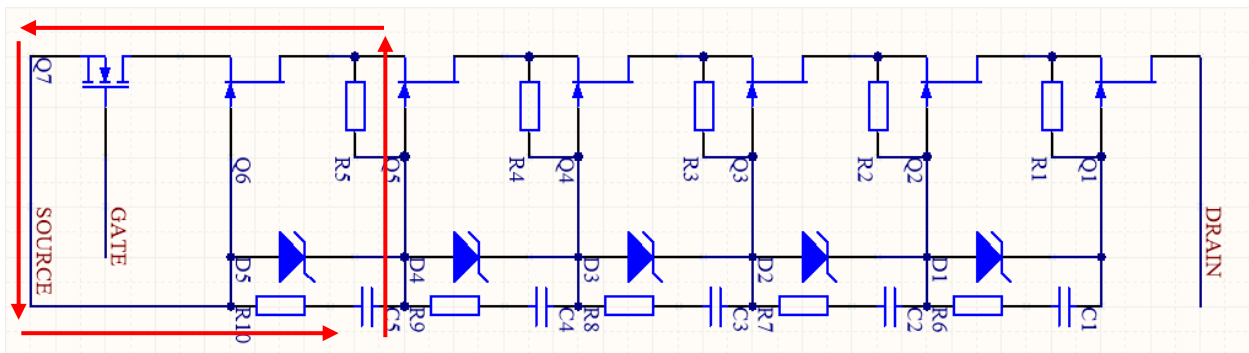


Figure 2.3: Biela's SCPM (turn on process, only Q5 stage is shown)

When Q7 turns off, on-state current through the SCPM increases the Q6 source potential, pinching off Q6. As Q6 turns off, source potential of Q5 rises, C5 absorbs charge from elevated gate potential. Similarly, the turn off process propagates through all stages to Q1.

After dynamic balancing has been established, leakage currents through the devices force static equilibrium, which is the same as Friedrich's structure, with the exception that the added currents through R1~R5 cause further matchings of leakage current in each stage, so that each diode has approximately the same current, reducing the static voltage balancing mismatch.

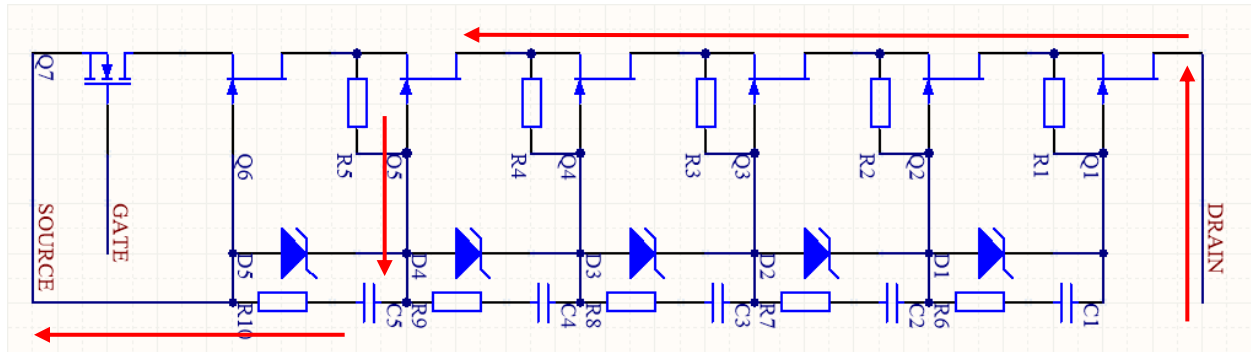


Figure 2.4: Biela's SCPM (turn off process, only Q5 stage is shown)

By changing capacitance values, per-stage voltage during dynamic balancing can be fine-tuned to mitigate voltage mismatch or to artificially introduce voltage balance in order to optimize power loss. Series resistors R6~R10 are used to control slew rate to fine tune switching speed of each stage as well as to control EMI. While dynamic balance voltage changes, static voltage needs to be changed accordingly, by adjusting R1~R5 and/or D1~D5.

A variant of Biela's structure is given in Fig. 2 right and shows an extra balancing state added across SCPM Q1 drain to gate. This provides a more stable biasing current source in parallel to Q1 drain-to-gate leakage, to improve static voltage balancing and to improve switching speed. However, because the balancing voltage needs to be lower than JFET voltage rating, the Zener voltage of the diodes must be lower than rated voltage of JFETs. In case of an avalanche event, avalanche current can flow straight through D0-D5 to ground, catastrophically damaging them. Therefore, this design is not avalanche proof. It is also worth mentioning that even Figure 2 left is

not fully avalanche proof. In case an avalanche voltage greater than  $5 \cdot V_D + V_{DG\_MAX}$ , where  $V_{G\_SMAX}$  is drain to gate breakdown voltage of a JFET, avalanche current will still flow through the D-G junction of J1 and D1-D5 to module source, bypassing Q2-Q7, destroying D1-D5. This failure mode also applies to Friedrichs' SCPM as well as other existing SCPM structures.

The Biela's SCPM has a much more improved switching performance compared with Friedrichs' SCPM. The reported turn off (voltage rise) is 50ns and the corresponding turn on (voltage fall) is 35ns. Current fall and rise times are similar to the voltage rise and fall times <sup>[5]</sup>.

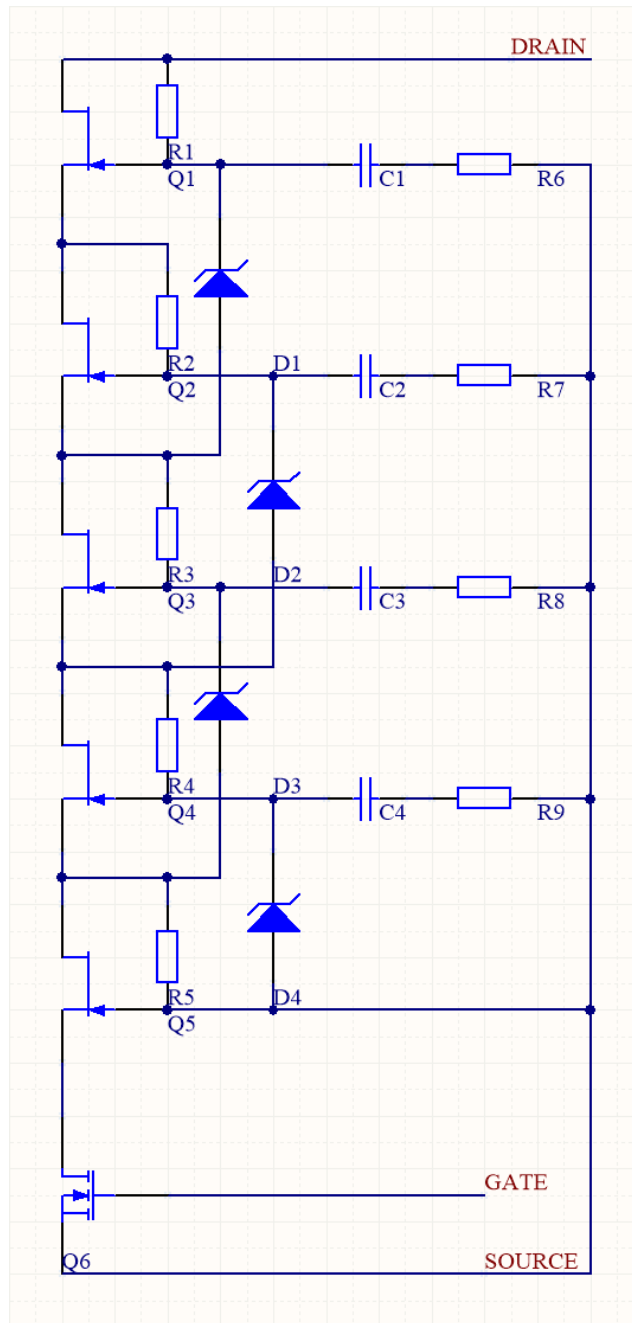


Figure 2.5: Li's SCPM <sup>[6]</sup>

Xueqing Li, et al at United Silicon Carbide Incorporated proposed another SCPM topology. This topology connects capacitors across gates and module source, instead of across each gate. This design allows for individual voltage balancing of the JFET stages without significantly affecting

other stages. The connection of avalanche diodes is also different than in the Biela's structure. Li's structure connects diodes across the gate of an upper stage to the source of the lower next stage. This reduces voltage imbalance caused by leakage current mismatch of avalanche diodes at the cost of less effective gate voltage balancing due to reduced intra-stage gain. The Li's structure also moves leakage current compensation resistors from gate-to-source to gate-to-drain to create a negative feedback on voltage sharing at the cost of more static current contributed by the balancing network. The Li SCPM has achieved 50ns voltage rise during turn on, and 50ns voltage fall during turn off [6].

Xiaoqing Song, et al at NC State University have proposed another SCPM topology which achieves 10kV and 15kV blocking voltage and 75ns voltage rise and 85ns voltage fall at 8kV, 40A, Fig. 6 [7]. However, their SCPM structure is not discussed here because their SCPM topology drives a high positive voltage (comparable with MOSFET gate voltage) to the gates of JFETs through diodes, while the JFETs used in this research cannot handle positive gate voltage more than 2V.

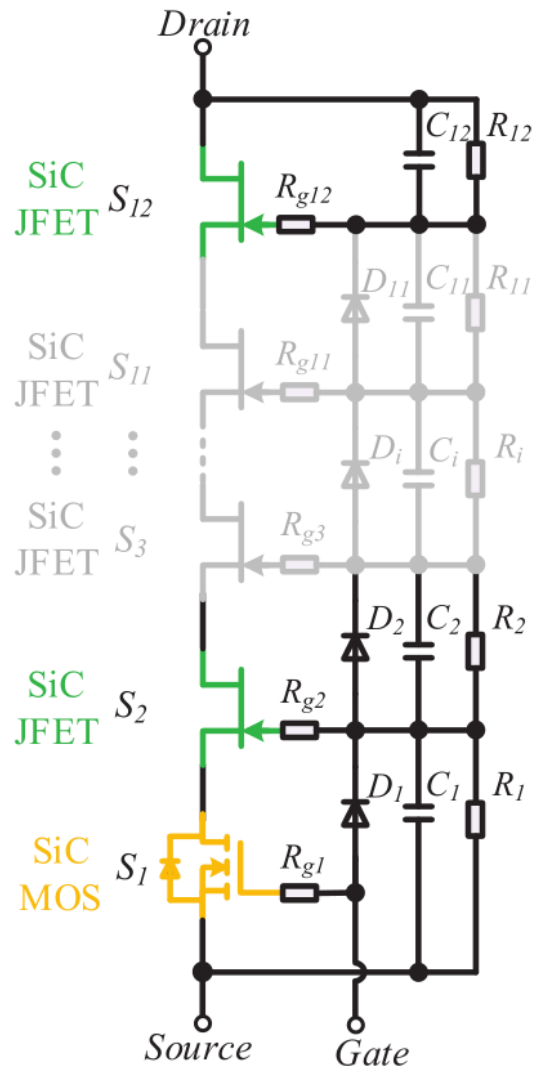


Figure 2.6: Song's SCPM, note that their JFETs see  $\sim 15V \cdot n \cdot V_f$  positive gate driving voltage <sup>[7]</sup>

## 2.2: Problems to be solved by this research

All the SCPMs mentioned above suffer from a common issue: voltage scalability. With a given JFET device and a given maximum drain to source blocking voltage, the overall voltage of a module is dictated by stage count of JFETs in series. However, as stage count increases, balancing network capacitive loss increases rapidly. For instance, in Biela's structure, the

capacitor of a lower stage has to hold/release charge from current stage JFET as well as upper stages. Therefore, the lower the stage, the higher the capacitance is required.

At same voltage per stage, assuming the top capacitor stores  $1x$  mJ energy, then the second from top stage capacitor stores  $2x$  mJ energy, the third stage from top stage capacitor stores  $3x$  mJ energy, etc. An  $n$ -stage SCPM will have totally  $1x+2x+\dots+(N-1)x$  mJ energy stored in balancing network during turn-off process, and this energy gets dissipated as heat during turn-on process. Using Gaussian formula, total balancing network switching loss is  $x*(N^2-N)$  mJ in non-ZVS mode operation (details explained in chapter 3). The second order term limits the maximum stage count allowed in an SCPM to switch efficiently.

Table 1 summarizes the operating conditions of active power switches for common power converter topologies. Synchronized rectification is not considered since this research is focused on medium voltage and high voltage applications, reducing diode forward drop voltage does not significantly improve efficiency, but additional switching loss can be detrimental. All converters are assumed to operate in CCM due to interest of operating at high power level:

Table 2.1: Power switch operation condition comparison of common converters

	Buck	Boost	Forward	Flyback	Half Bridge	LLC
ZVS (on)	No	No	No	No	Yes/No	Yes
ZCS (off)	No	No	No	No	No	Yes
Ringig	Low	Low	High	High	Low	Low

Most converter topologies used in high power systems are based on half bridge switching elements, such as buck, boost, (phase shift) half bridge, full bridge, LLC and more. In particular, this research focuses to optimize SCPM designs for 2 general operation modes: ZVS and non-

ZVS. The ZVS mode is commonly seen in topologies with cycle-to-cycle switching-node current commutation, such as ZVS half bridge converters, ZVS full bridge converters and LLC converters. Non-ZVS mode is commonly seen in topologies with no cycle-to-cycle switching-node current commutation, such as voltage source inverters, buck converters and boost converters.

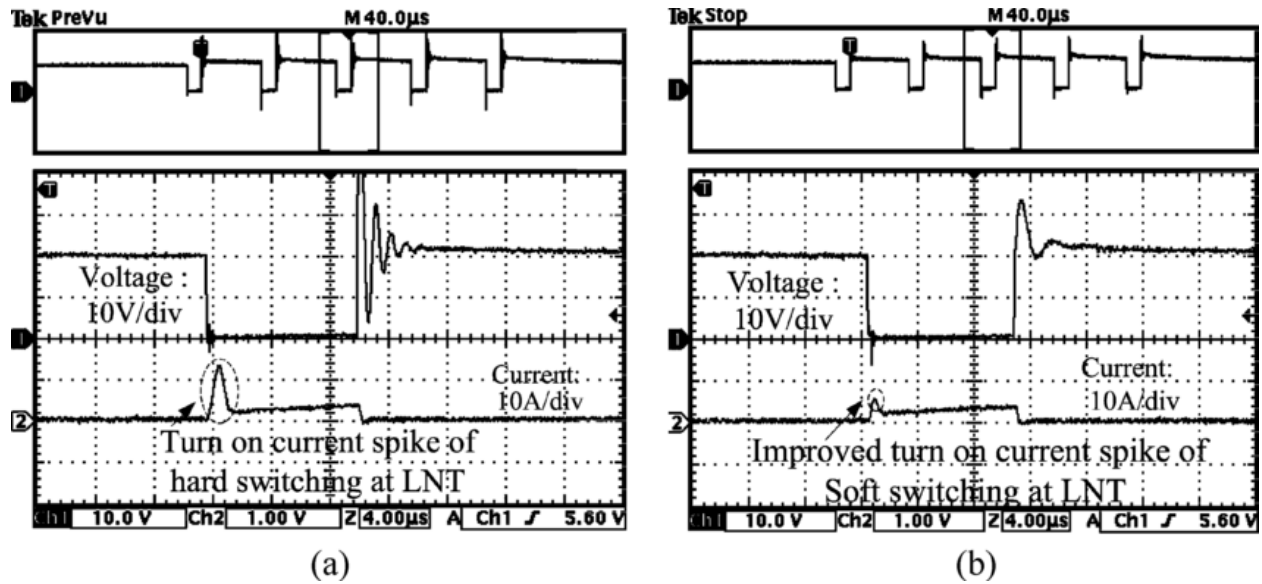


Figure 2.7: Current spike during hard switching in a non-commutating half bridge [8]

During non-ZVS SCPM turn-on process, stored charge in JFET junctions and balancing network capacitors are shorted by lower stages, being converted to heat. Therefore, reducing balancing network power loss is the goal for reducing switching loss in topologies such as voltage source inverters and buck/boost converters.

For a ZVS converter where the inductor/transformer current commutates during each switching cycle, if the timing is correct, the commutation current of magnetics can soft charge JFET junctions and balancing network capacitors, the capacitive loss can be greatly reduced, but not completely eliminated due to a series balancing resistor. In addition, the energy loss during the

SCPM transits between on and off states (partially conducting) still contributes to power loss. ZCS topologies such as LLC are free of this issue, but non-ZCS topologies suffer from this issue, even if they are ZVS.

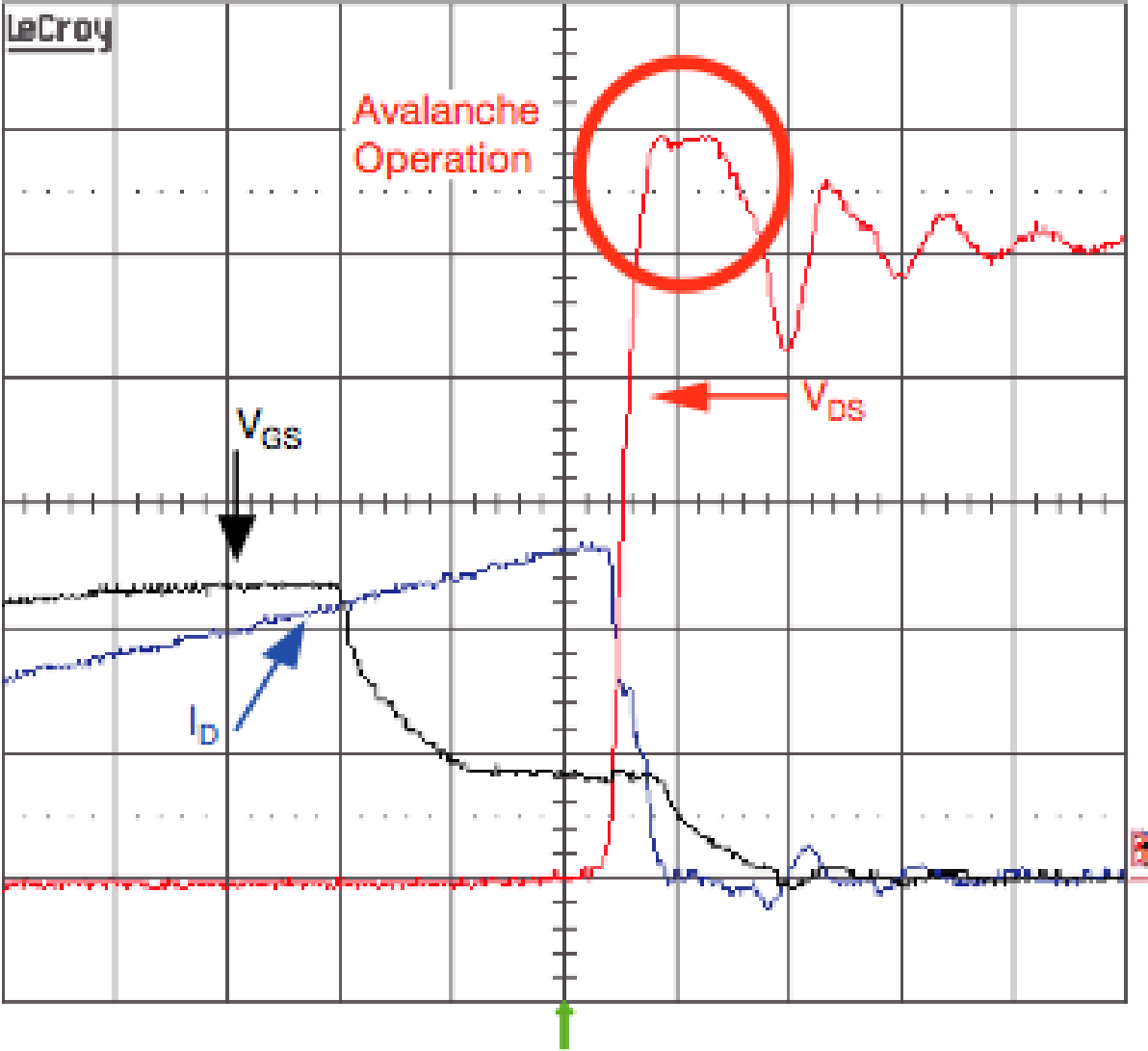


Figure 2.8: Avalanche operation of half bridge switches

A monolithic power semiconductor has a relatively large die size to dissipate the heat caused by avalanche breakdown. However, in an SCPM, avalanche current travels through avalanche

diodes in the balancing network, which have much smaller die size compared with power JFETs. Any avalanche current through the SCPM may be easy to manage for a large power JFET, but easily fail a small avalanche diode.



Figure 2.9: Size comparison between a power JFET die (gold on the left) and a packaged avalanche diode (black on the right), estimated die size of the avalanche diode is 1/6 of its package size

For instance, 100nH of stray inductance in the DC bus decoupling loop at 100A stores 0.5mJ of energy. At 10kHz operation, the leakage avalanche power loss caused by this inductance is 50W, assuming the worst case, which is when there is no junction capacitance in SCPM. This is not much for a stack of high-power semiconductor module to dissipate, but for the avalanche diodes in the balancing network, 50W is more than sufficient to destroy them, according to information obtained from AU1PM datasheet.

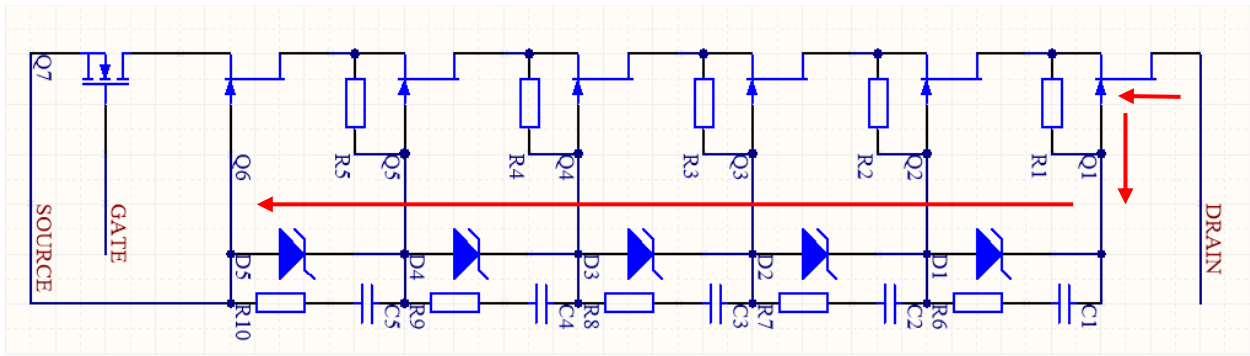


Figure 2.10: Current path of Biela's SCPM during an avalanche breakdown

As a result, avalanche energy handling capability is very important for SCPMs, even if they are used in a half bridge application.

In this research, the proposed SCPM aims to solve these issues with existing SCPM structures:

1. Leakage current mismatch sensitivity
2. Limited avalanche handling capability
3. Limited voltage scalability

## Chapter 3: Proposed Single SCPM Structure

### 3.1: Mathematical modelling of switching losses

A new super cascode power module topology is proposed in this research to address the medium voltage ( $\leq 10\text{kV}$ ) applications mentioned above. To facilitate optimization of design, a simplified non-ZVS switching power loss model was developed as a worst-case scenario. The model contains:

1. Balancing network non-ZVS switching loss
2. DBC drain pad capacitive coupling switching loss
3. Overall layout stray inductance switching loss

Power losses related to power JFETs, such as junction capacitive loss, partial conductive loss,  $R_{\text{DS(on)}}$  loss, as well as leakage current loss, are not modelled since they are similar in all SCPM topologies with balancing networks that are both statically and dynamically balanced. Hence, they are not considered optimizable terms. Static leakage loss on balancing networks is also not discussed, since this is considered negligible compared with other power losses.

The abovementioned simplified loss model can be expressed as the following expression:

$$P=f*(2*E_B+2*E_C+E_L) \quad (1)$$

Here,  $f$  is switching frequency,  $E_B$  is stored energy of balancing network capacitors,  $E_C$  is stored energy of DBC drain pads and  $E_L$  is stored energy of stray inductance in power module.

Capacitive losses are modelled as frequency times twice stored energy because during zero state

non-ZVS charging, equal amount of capacitor energy is dissipated on series resistance, as derived here:

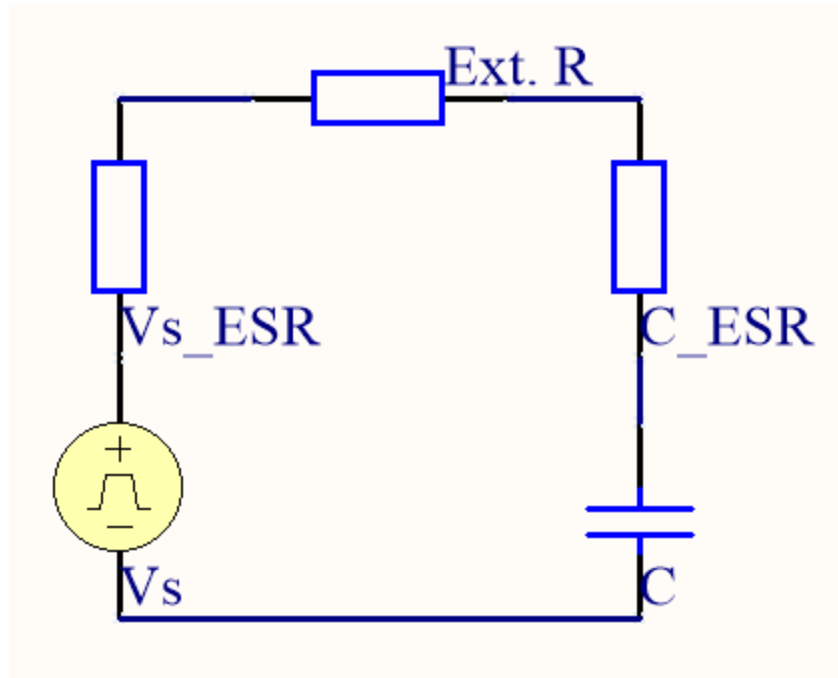


Figure 3.1: Zero state non-ZVS capacitor charging model

Here, R is the overall resistance of ESR or source, external resistor and ESR of capacitor C. Stray capacitors are not modelled for simplicity.

Using KCL,  $I_R = I_C$ . Using KVL,  $V_S = V_R + V_C$ . Therefore:

$$i_C(t) = (V_S - v_C(t)) / R$$

$$E_R = \int (i_R(t)^2 * R) dt = \int \left\{ \left[ \frac{(V_S - v_C(t))}{R} \right]^2 * R \right\} dt = \frac{1}{R} \int (V_S - v_C(t))^2 dt$$

Replacing  $v_C(t)$  with zero state non-ZVS charging voltage formula  $v_C(t) = V_S - V_S * e^{-t/\tau}$ , the abovementioned formula can be simplified as:

$$E_R = 1/R * \int [V_S - (V_S - V_S * e^{-t/\tau})]^2 dt = V_S/R * \int (e^{-t/\tau})^2 dt = V_S/R * \int e^{-2t/\tau} dt = V_S/R * \int e^{-2t/RC} dt$$

Solving the integral equation and integrate from 0 to infinite:

$$E_R = 1/2 * V_S^2 * C = E_C$$

Therefore, for a zero state non-ZVS circuit, energy used to charge a capacitor is always twice the energy being charged into the capacitor, regardless value of series resistor R.

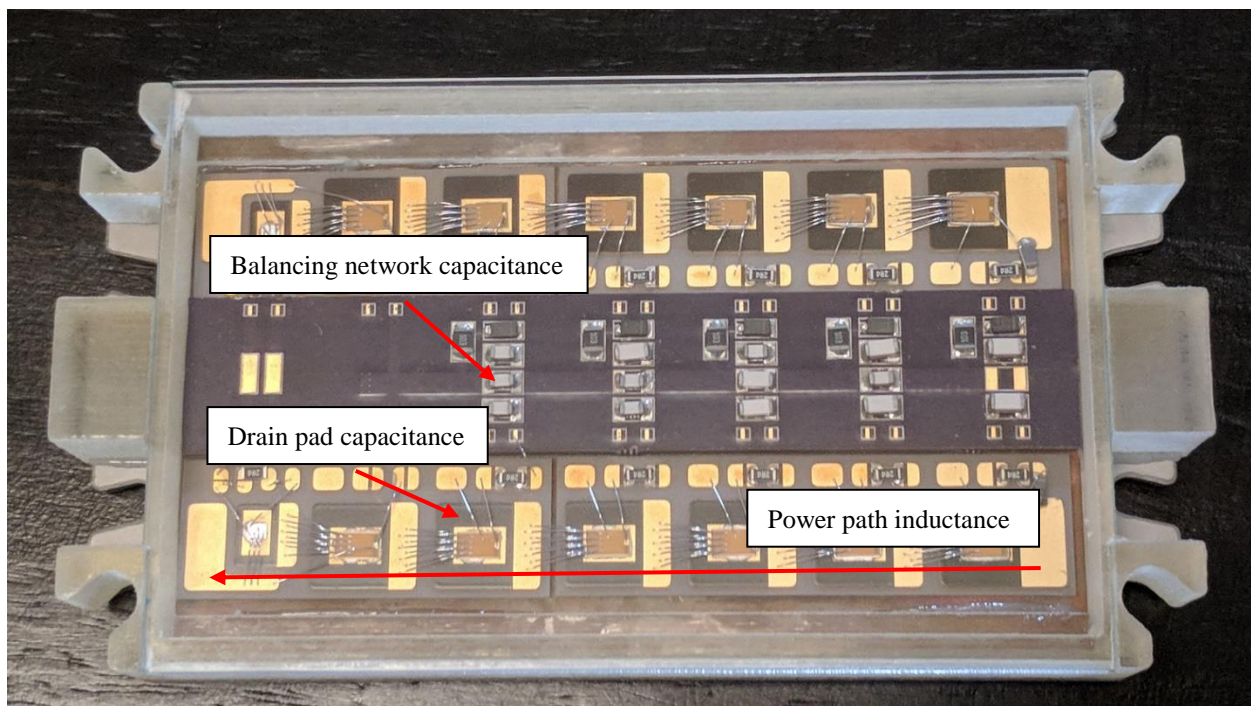


Figure 3.2: Physical representation of interested sources of loss

Since the 3 sources of switching loss come from completely different mechanisms (balancing network capacitance, drain pad capacitance and stray inductance, seen in Fig. 12), a special generalization of losses can be deduced to provide a relatively accurate approximation of loss value for optimization. For instance, regardless of the balancing network, the same configuration (parallel amount and serial amount) of JFET semiconductor area occupies the same total drain

pad area, including interconnections pad size. Similarly, for the same configuration of JFETs, all the stray inductance can be considered the same regardless of balancing network design for a 2D planer layout design.

Assuming each stage in an SCPM blocks the same voltage  $V$ , and the base plate of SCPM is tied to module source, then the capacitive energy stored in an SCPM for  $N$  number of JFET stages is:

$$E_C = M * (E_1 + E_2 + E_3 + \dots + E_N)$$

Where  $M$  is number of JFET strings in parallel,  $E_1 \sim E_N$  are energy stored per drain pad. The abovementioned formula can also be written as:

$$E_C = 1/2 * M * C * V^2 * (1^2 + 2^2 + 3^2 + \dots + N^2)$$

The abovementioned sequence can be further simplified as:

$$E_C = 1/2 * M * C * V^2 * (2N^3 + 3N^2 + N) \tag{2}$$

$E_L$  can be modelled with the following assumptions:

1. Junction capacitances are omitted
2. Balancing network capacitances are omitted
3. Interconnect capacitances are omitted

Those assumptions define the worst-case scenario where all inductive energy stored in the power decoupling loop being avalanche energy.

With those assumptions,  $E_L$  at turn off current  $I$  can be modelled as:

$$E_L = 1/2 * I^2 * L \tag{3}$$

Where  $L$  is overall module stray inductance. Because this research focuses on the power modules, not application circuits, only parasitic parameters inside the modules are being researched on. Despite external inductance also contributes to  $E_L$ , only sources of parasitic inductance within the module are being discussed. Similarly, external capacitance contributing to  $E_C$  is also ignored.

With these, the  $E_C$  is optimized during DBC layout design and  $E_L$  is optimized during busbar design.  $E_B$  is dependent on JFET gate charge and number of stages. Later in this work, optimizing of  $E_B$  is heavily discussed in chapter 6. However, in this chapter,  $E_B$  is assumed to be fixed for a given JFET model and module serial/parallel configuration. This balancing network also focuses to improve the following:

1. Reducing leakage current mismatch impact on static balancing operation point
2. Divert avalanche current from avalanche diodes to JFETs
3. Reduce manufacturing cost and improve manufacturability

### **3.2: Impact of layout optimization for manufacturability**

This SCPM design uses balancing network capacitors connected across gates to achieve three goals:

1. To provide dynamic balancing and high switching performance
2. To eliminate high voltage capacitors in the upper stages to lower risk of voltage breakdown and to reduce manufacturing cost
3. To eliminate upper stage gate to lower stage source connections to avoid crossing bonding wires

In this new design, gate-source resistors are not used since the avalanche diode chosen has a sharp knee point so strictly even current distribution among stages is no longer needed. In addition, a bias resistor from top JFET drain to gate is introduced to supply a parallel minimum bias current independent of JFET leakage current. This guarantees worst-case startup behavior of the proposed SCPM. Finally, avalanche diodes and capacitors are connected in parallel directly, which are then connected across gates between stages via a series resistor which sets switching speed. This allows the resistors which dissipate the majority of balancing network switching energy to be placed on the DBC right next to power JFETs and allows the placement of the rest of the balancing network on a low cost, low mechanical stress FR4 located next to the DBC. This reduces manufacturing cost while increasing thermal structural reliability, all due to the reduction of DBC size.

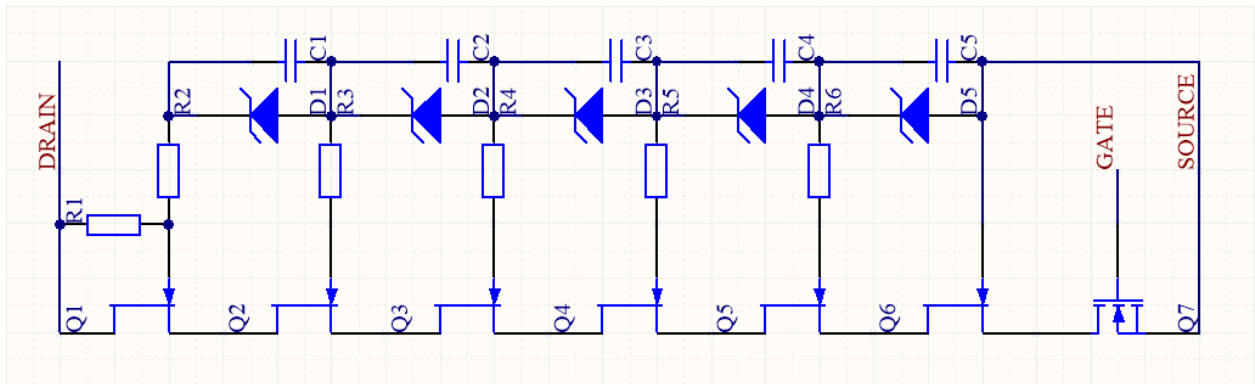


Figure 3.3: Proposed single SCPM topology

Figure 13 shows a 6-stage implementation of the proposed SCPM structure. In the proposed SCPM implementation, R1 provides additional bias current for D1-D5, which set the static balancing voltage. The R2-R6 set rise and fall times, while C1-C5 have values based on gate charge of Q1-Q5 and body charge of D1-D5. In a perfectly balanced system where voltage across all JFETs are the same and all JFETs and diodes are matched,  $\Delta Q_{cn} = n \cdot (Q_G - Q_D)$ , hence:

$$C_n=(Q_G-Q_D)/V_{DS}*N \quad (4)$$

Here,  $V_{DS}$  is drain to source voltage of each individual JFET. Using capacitor energy formula

$$E=1/2*V^2*C:$$

$$E_{C1}=1/2*(Q_G-Q_D)*V_{DS}, E_{C2}=2*E_{C1}, \text{ etc.}, E_{Cn}=n*E_{C1}$$

For an N-stage SCPM, there are N-1 stages of balancing capacitors. To obtain balancing network capacitive energy loss per switch,  $E_B$ , applying Gaussian formula,  $E_B$  can be expressed as:

$$E_B=\sum(E_{Cn})=(E_{C1}+E_{CN-1})*(N-1)/2=1/4*(Q_G-Q_D)*V_{DS}*(N^2-N) \quad (5)$$

Unique to this SCPM approach is the sole use of nonlinear V-I behavior of avalanche diodes to define static voltages. This reduces the off-state bias current requirement, which reduces the module leakage current. This does require binned avalanche diodes with matching I-V characteristics. Note the cost saved on DBC, wire bonding and HV MLCCs due to the use of conventional PCB and simplified balancing circuit design can easily offset the additional avalanche diode binning cost.

Capacitance of balancing capacitors are determined with previously mentioned formula 4,  $C_n=(Q_G-Q_D)/V_{DS}$ . With UJN1202Z operating at 1000V balancing voltage, from simulation as well as trials and errors with DPT testing, it was determined that  $Q_G=300\text{nC}$ . Datasheet value shows  $Q_G=235\text{nC}$ ,  $Q_{GD}=165\text{nC}$  at  $V_{DS}=600\text{V}$ . Applying root square law on  $Q_{GD}$ , calculated ideal  $Q_G$  at  $V_{DS}=1000\text{V}$  is  $285\text{nC}$ , close to the measured value  $300\text{nC}$ . The diode charge,  $Q_D$ , is ignored due to the miniscule impact to overall gate charge balance.

In this configuration, dynamic balancing network energy loss is:

$$E_B = 1/4 * 300\text{nC} * 1000\text{V} * (6^2 - 6) = 4.5\text{mJ}$$

Since both JFETs and series gate resistors are in the charging/discharging path of balancing capacitors, the total dynamic balancing power is dissipated by both JFETs and resistors. Due to the gain of the JFETs, JFETs dissipate much more balancing network dynamic energy than resistors, and the exact ratio is determined by simulation.

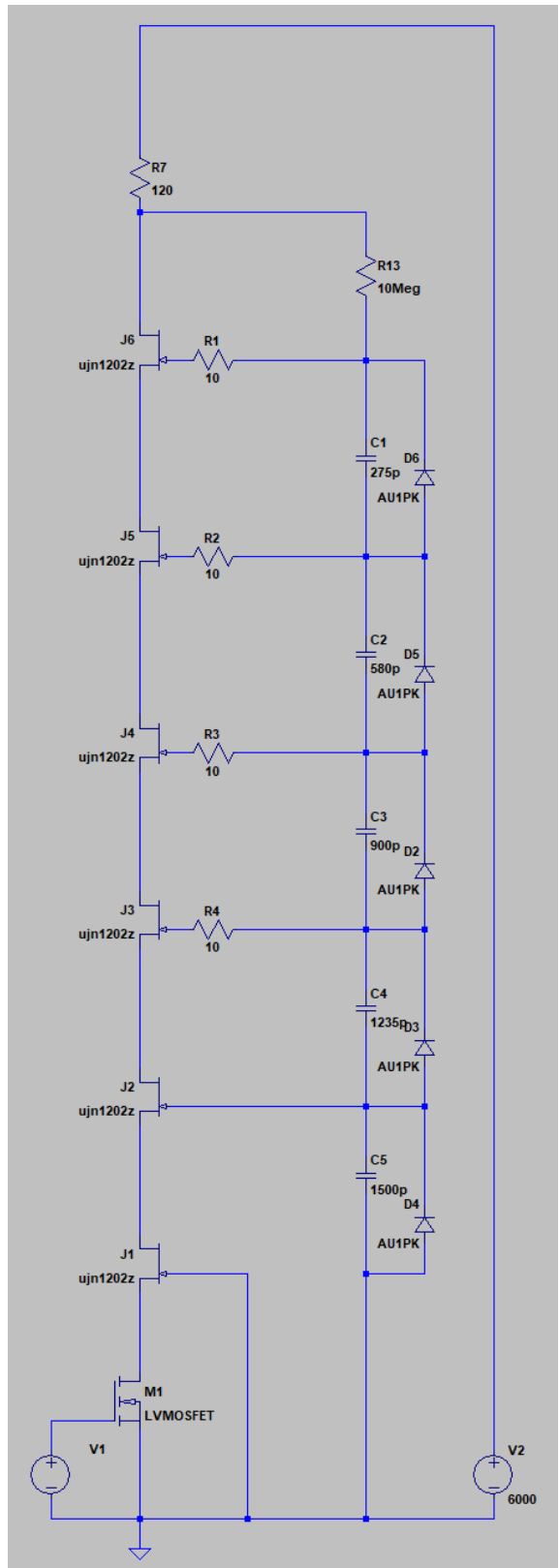


Figure 3.4: Simulation bench for extracting balancing resistor energy dissipation

According to simulation, each of the resistors dissipates an average 39.8mW/kHz (56.9uJ for R1, 38.5uJ for R2, 32.8uJ for R3 and 31.0uJ for R4). By putting these resistors on highly thermally conductive DBCs instead of the PCB, a much smaller resistor package can be used, and PCB aging due to high temperature components becomes much slower, thus increasing long term reliability. With 1206 package and 25 mil AlN DBC, each high-power resistor (Vishay RCP) can dissipate 11W according to datasheet, allowing the balancing network to operate at more than 193kHz (limited by R1).

## Chapter 4: Development of an SCPM Implementation

### 4.1: Electrical interconnection structure

To verify the proposed SCPM structure, a 6.5kV, 100A SCPM was fabricated and tested. The module was housed in an Infineon E3-like housing at the request at the research sponsor, and to have it as a direct replacement of Si IGBTs. In this design, 2 sets of proposed structure (Fig. 13) are used in parallel to double the current handling capability, forming a 100A SCPM from 50A rated power devices. More SCPM strings can be connected in parallel to form higher current modules if necessary. To facilitate balancing between the parallel connected SCPM strings, source and gate of each stage in one string are connected the source and gate of their respective stage in the other string. The JFETs, MOSFETs and resistors are placed on 4 separate pieces of DBC substrates, as shown in Fig. 15 and Fig. 16. The separation of DBC pieces mitigates stress, improving long term reliability and manufacturability. This also helps increasing production yield because having 4 dies attached and wire bonded properly is easier than having 14 dies attached properly and bonded properly.

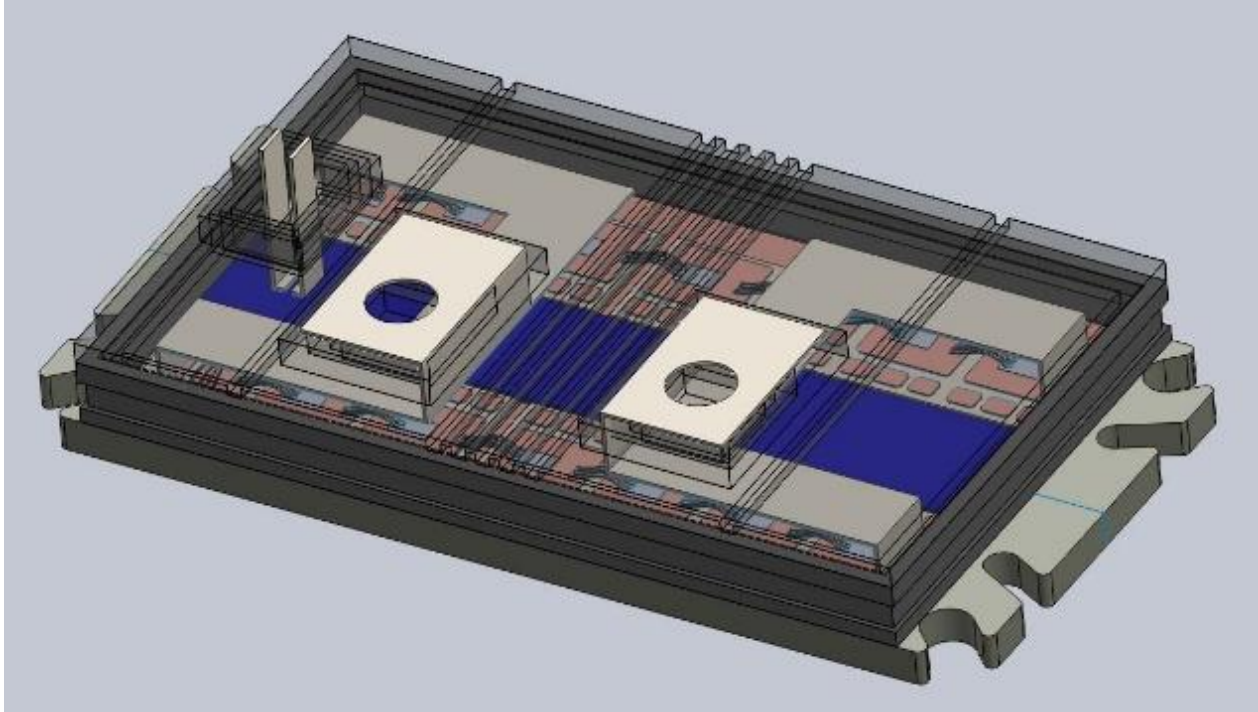


Figure 4.1: Rendering of proposed demonstration SCPM

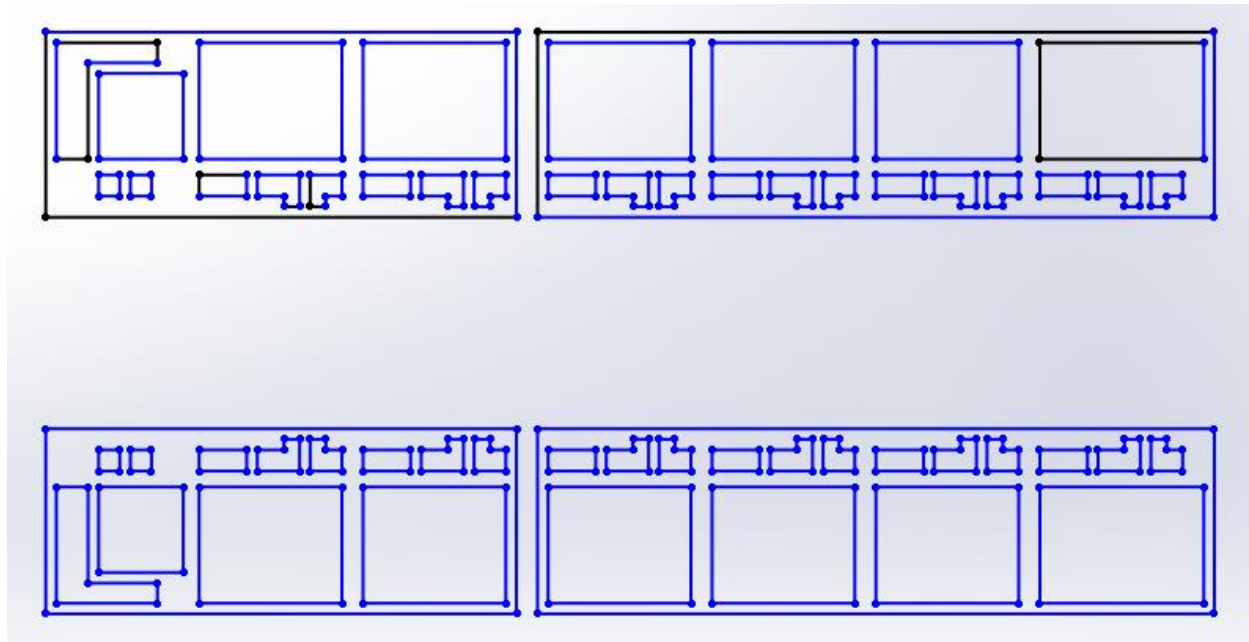


Figure 4.2: DBC layout of demonstration SCPM

Using Ansys Q3D, inductance analysis is performed on the demonstration SCPM implementation in connect with team member Dr. Yang Xu of the PREES Lab at NCSU. During simulation, the busbar connections are adjusted to form a closed loop to better resemble actual module busbar connection to system busbar.

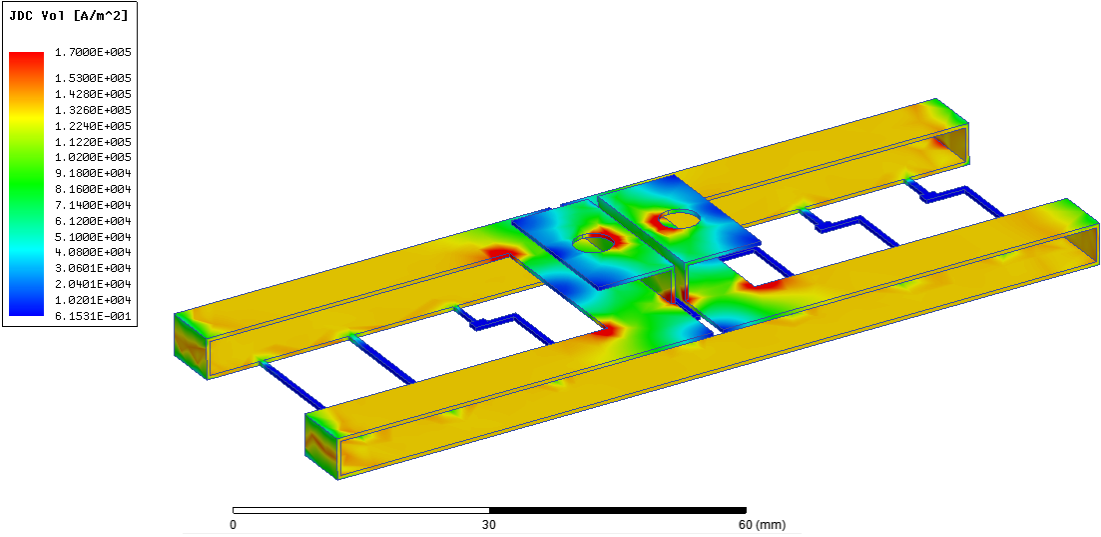


Figure 4.3: Current distribution and inductance extraction with flat bar model

Current path was modelled as a flat bar instead of individual bonding wires as shown in Fig. 17. To determine the error of this approach, a separate simulation on a small segment with bonding wires was performed and the results show negligible difference, Fig. 18.

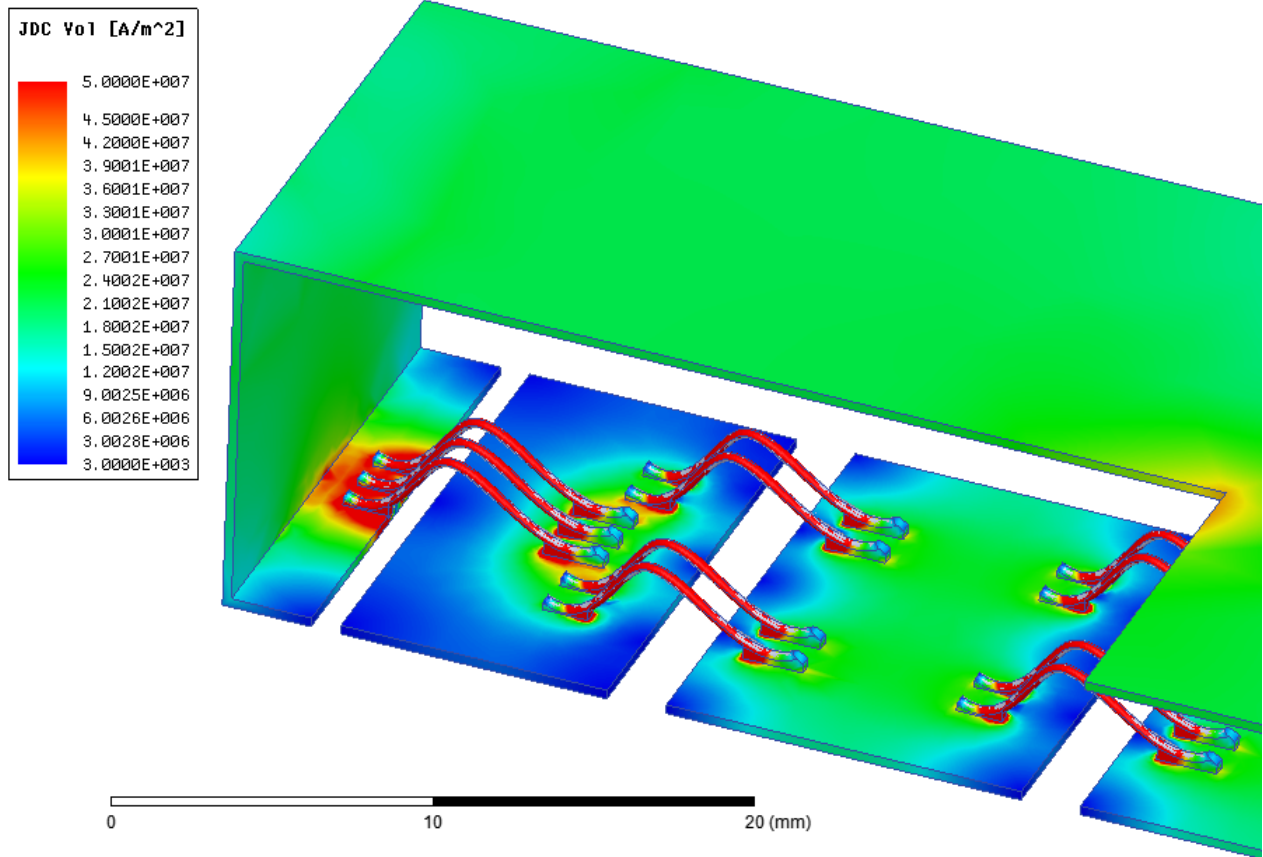


Figure 4.4: Current distribution and inductance extraction with bonding wire model

Simulation results show 23nH of power loop inductance, while industrial standard modules in the same package size (E3) have a typical power loop inductance of 50nH. At 100A turn-off current, this only contributes  $E_L=0.115\text{mJ}$  switching energy loss according to formula 3,  $E_L=1/2 \cdot I^2 \cdot L$ , hence is ignored in future analysis.

## 4.2: Thermal simulation and optimization

Thermal performance of the proposed physical structure was studied with the help of finite element simulation software by the other team member. Solder used in simulation is a general Sn63Pb37 because this simulation was carried out before process development, hence solder type was not determined yet. Based on 0.635mm (12mil) and 1.016mm (40mil) DBC thickness with AlN ceramic and AlSiC baseplate, the pad size was chosen to provide minimum area (hence minimum capacitive coupling) for both 3.5mm\*5.5mm dies and 8.0mm\*8.0mm dies while the thermal performance is not degraded. Parameters used in simulation are listed below:

Table 4.1: Thermal simulation parameters from COMSOL library and online material databases

Material	Thickness	Thermal Conductivity
Si MOSFET	0.15mm	130W/mk
SiC JFET	0.15mm	450W/mk
Solder (Die to DBC, Sn63Pb37)	0.10mm	50W/mk
DBC Top Cu	0.25mm	400W/mk
DBC AlN	0.63mm	180W/mk
DBC Bottom Cu	0.25mm	400W/mk
Solder (DBC to Baseplate, Sn63Pb37)	0.10mm	50W/mk
AlSiC Baseplate	3.00mm	180W/mk

Simulation results are plotted for both 8mm\*8mm JFET and 3.5mm\*5.5mm JFET devices on both 25mil AlN DBC and 40mil AlN DBC, as shown in Fig. 19 to determine the optimum drain pad size on DBC:

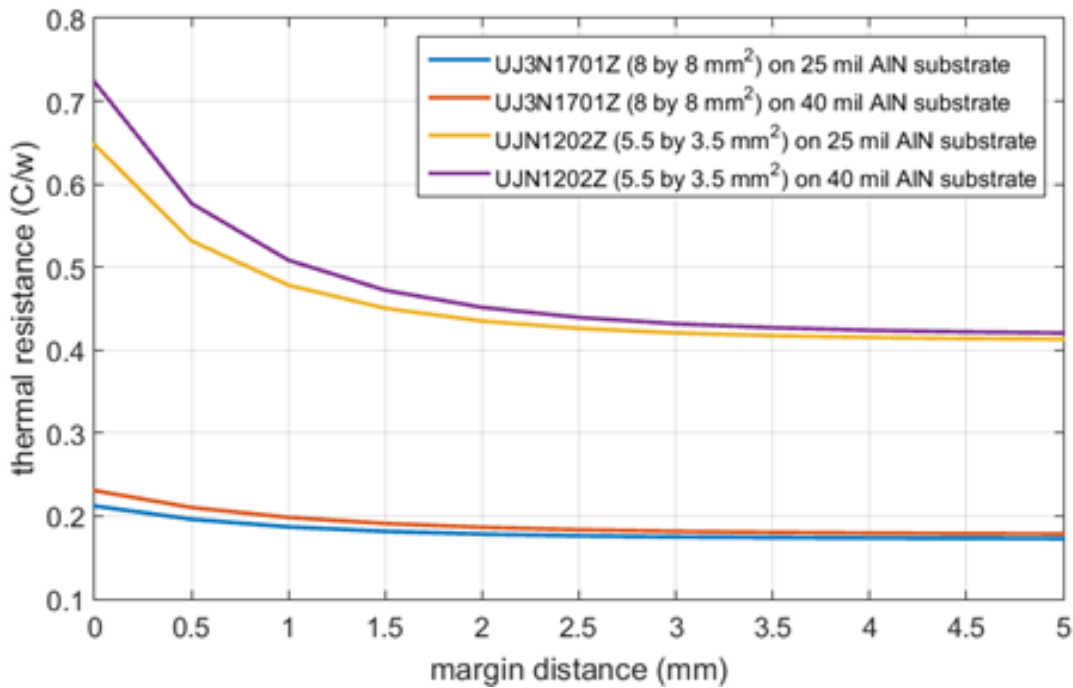


Figure 4.5: Drain pad margin vs thermal resistance curve

From simulation, the thermal resistance decreases <10% above 1.5mm margin space between dies and edge pf drain pad. 1.5mm margin space is chosen in this demonstration, but other margin values can be used as well according to manufacturing requirements. Since the DBC is designed for 200A SCPM made of two 100A SCPM strings with 8.0mm\*8.0mm JFETs, the pad size was determined to be 11mm\*13mm to accommodate the largest possible JFETs and wire bonding space within the confines of the sponsor’s E3 package outline.

Using previously mentioned formula 2:

$$E_C = 1/2 * M * C * V^2 * (2N^3 + 3N^2 + N)$$

The substrate capacitive coupling switching energy loss  $E_C$  is:

$$E_C = 1/2 * 2 * C * 1000V^2 * (2*6^3 + 3*6^2 + 6) = 1.09mJ/pF$$

With 11mm\*13mm, 0.635mm DBC,  $C=33.8\text{pF}$ ,  $E_c=36.84\text{mJ}$ . With 1.016mm DBC,  $C=21.2\text{pF}$ ,  $E_c=23.10\text{mJ}$ .

A detailed simulation was carried out in connect with Dr. Yang Xu of the PREES Lab at NCSU on one DBC section to verify thermal performance of UJN1202Z JFETs and USU141A MOSFET on 0.635mm AlN DBC. This simulation shows  $0.45^\circ\text{C/W}$  between JFETs and baseplate, and  $0.61^\circ\text{C/W}$  between cascode MOSFET and baseplate.

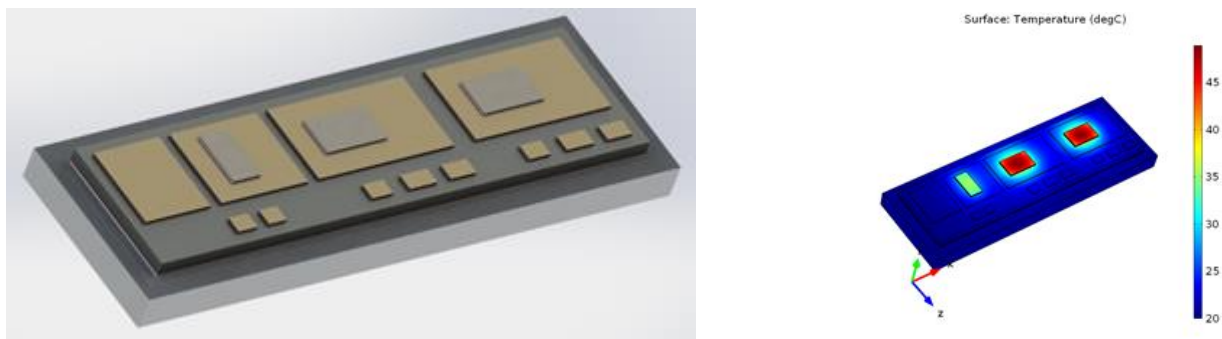


Figure 4.6: Thermal simulation model and result

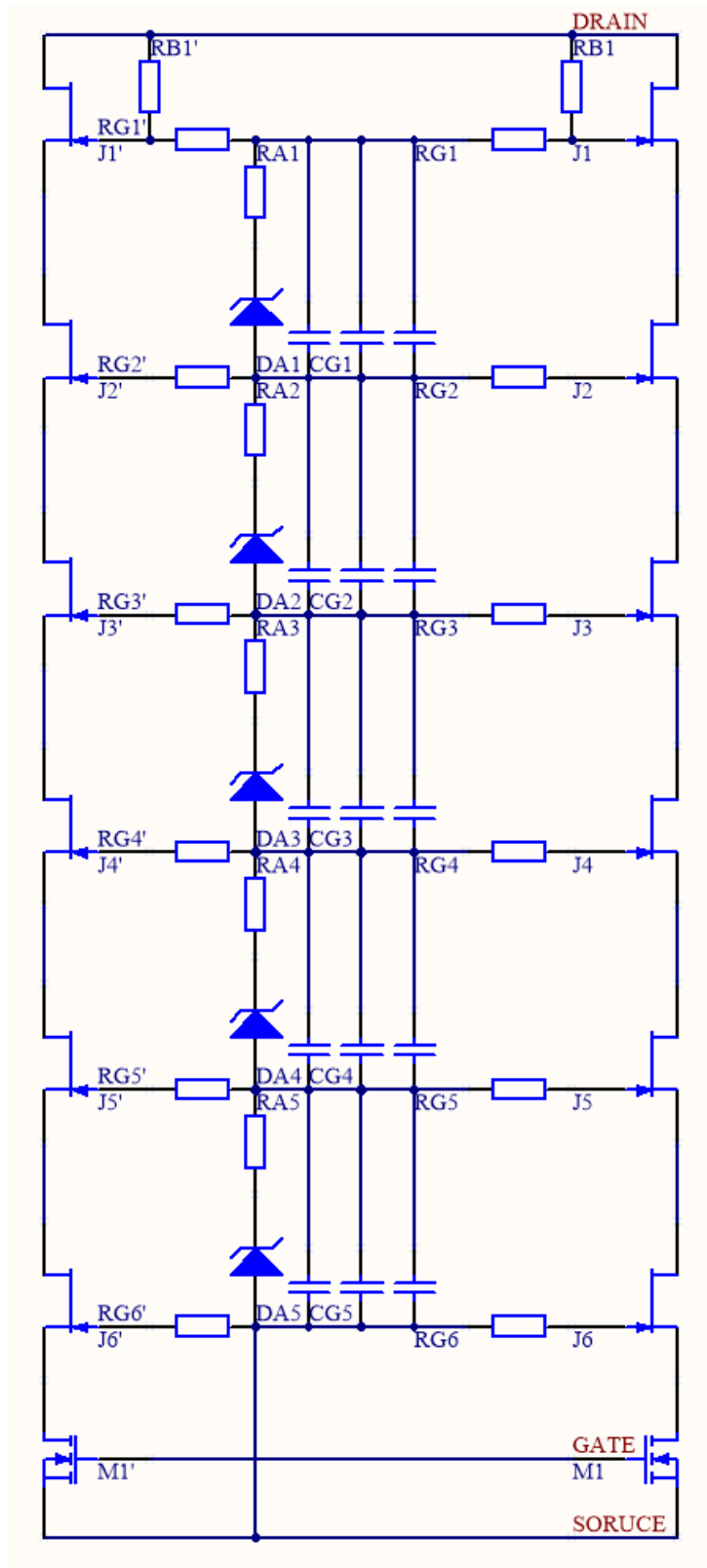


Figure 4.7: Overall schematic of demonstration SCPM

### 4.3: Electrical simulation

Due to the uneven power dissipation caused by the nature of sequentially switching power semiconductor stacks described earlier in chapter 2 and chapter 3, JFETs in lowest stages (i.e. J6 and J6' in Fig. 22) dissipate more power than JFETs in upper stages (i.e. J1 and J1' in Fig. 22) during switching. To mitigate this uneven power dissipation, the balancing network is tuned so that JFETs in lower stages share slightly lower voltage than JFETs in upper stages, as this offsets the power differences. Based on simulation in LTSpice with UJN1202Z JFET model from USCi and standard E12 capacitance values, the capacitor values in Table 3 are chosen (3 capacitors per stage, shared by both parallel strings, shown in Fig. 22):

Table 4.2: Capacitor configuration of proposed SCPM

Stage from Top (J1)	Total Capacitance	Capacitor 1	Capacitor 2	Capacitor 3
1	550pF	220pF	N/A	330pF
2	1160pF	470pF	220pF	470pF
3	1800pF	330pF	100pF	470pF
4	2470pF	1000pF	4700pF	1000pF
5	3000pF	1000pF	1000pF	1000pF

With the capacitors listed in Table 3, the simulation result shows both good power sharing (Table 4) and voltage sharing (Fig. 23). There is a tradeoff between power sharing and voltage sharing. To achieve highest possible operating voltage, voltage must be distributed evenly, which means there will be unevenly high-power dissipation on the lowest JFETs (J6 and J6' in Fig. 22), hence thermally limiting switching voltage, current and frequency of the entire SCPM. To reduce power dissipation on lowest stage JFETs (J6 and J6' in Fig. 22), 900V balancing voltage on J6 and J6' is chosen to limit their power dissipation. Table 4 lists switching energy  $E_{sw}$  and balancing voltage  $V_{OFF}$  of Q1-Q7 at 6kV DC bus, 100A resistive load current with 23nH

parasitic inductance and 50% duty cycle. Simulation was done with only one SCPM string (J1~J6, M1, RA1~RA5, DA1~DA5, CG1~CG5, capacitances of CG1~CG5 are halved), Fig. 23. Full module switching energy loss values can be calculated from doubling the simulation result.

Table 4.3: Proposed SCPM simulation result

	J1	J2	J3	J4	J5	J6	M1
$E_{SW}$	0.96mJ	1.32mJ	1.55mJ	1.70mJ	1.62mJ	2.02mJ	0.01mJ
$V_{OFF}$	1253V	1076V	976V	923V	863V	890V	15V

Ignoring gate resistor loss, the proposed SCPM has 18.32W/kHz switching power loss, excluding drain pad capacitive switching loss. In addition, this module has 36.84W/kHz drain pad capacitive power loss and 480W  $R_{DS(on)}$  power loss. Based on thermal and electrical simulation results, the demonstration module is able to switch at 115kHz (assumption based on  $E_C$  distributes the same way as  $E_B$ ) when baseplate is grounded to module source and baseplate temperature is held at 0°C. If a smaller drain pad specifically designed for 3.5mm\*5.5mm JFETs were used in a smaller module then the switching frequency can be further increased to 200kHz. With a more realistic 40°C baseplate temperature, maximum switching frequencies are 90kHz and 150kHz respectively, assuming all other conditions remain the same. The proposed SCPM has 22ns turn on and 49ns turn off at 6kV DC bus, 100A resistive load current with 23nH parasitic inductance and 50% duty cycle operation, Fig. 24.

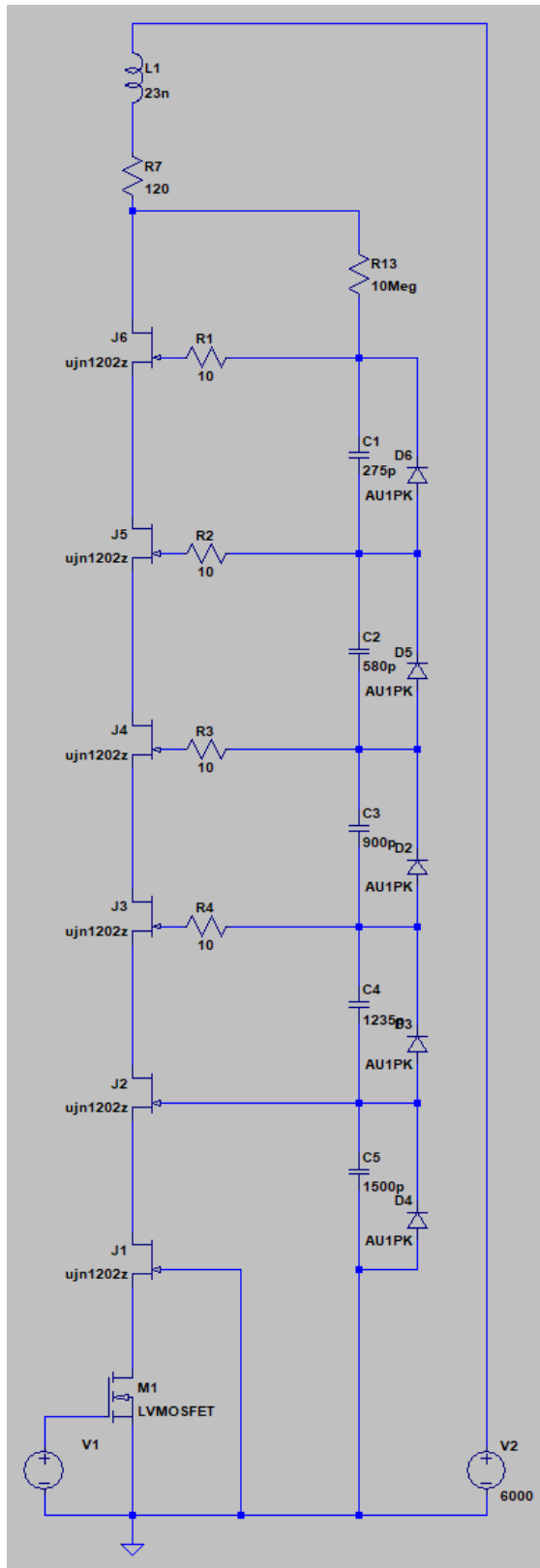


Figure 4.8: Demonstration SCPM simulation bench

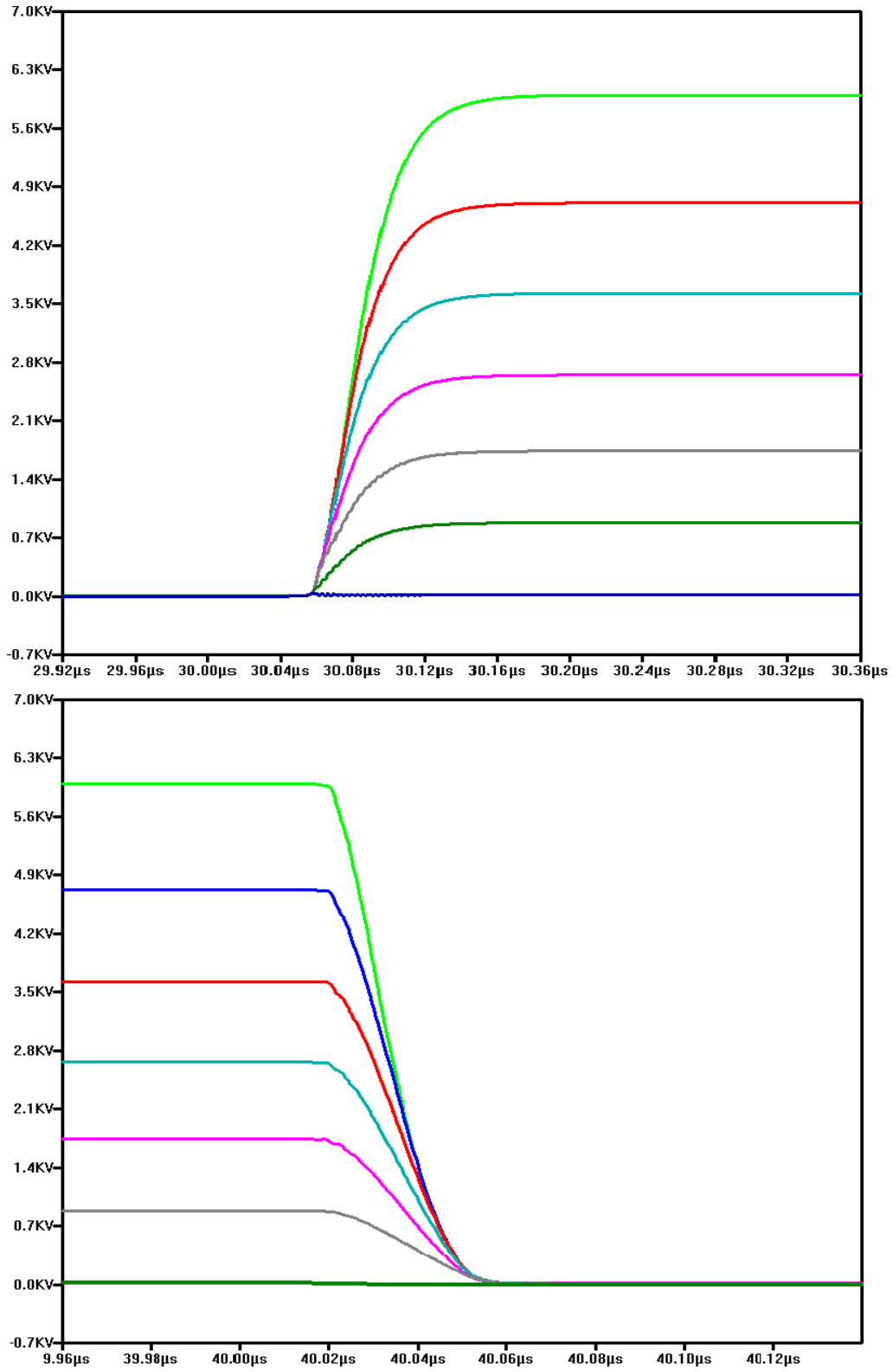


Figure 4.9: Simulated switching waveform of demonstration SCPM

Despite voltage adjusting, JFETs in lower stages (i.e. J6 and J6' in Fig. 22) still dissipate more power than JFETs in upper stages (i.e. J1 and J1' in Fig. 22). To provide further thermal balancing, an asymmetric cooling solution is proposed. The newly proposed thermal solution utilizes a pin-fin baseplate (Fig. 25) and a sloped coolant cavity (Fig. 26), to allow more heat to be extracted from JFETs in lower stages (i.e. J6 and J6' in Fig. 22), thus allowing better thermal distribution across the entire SCPM.

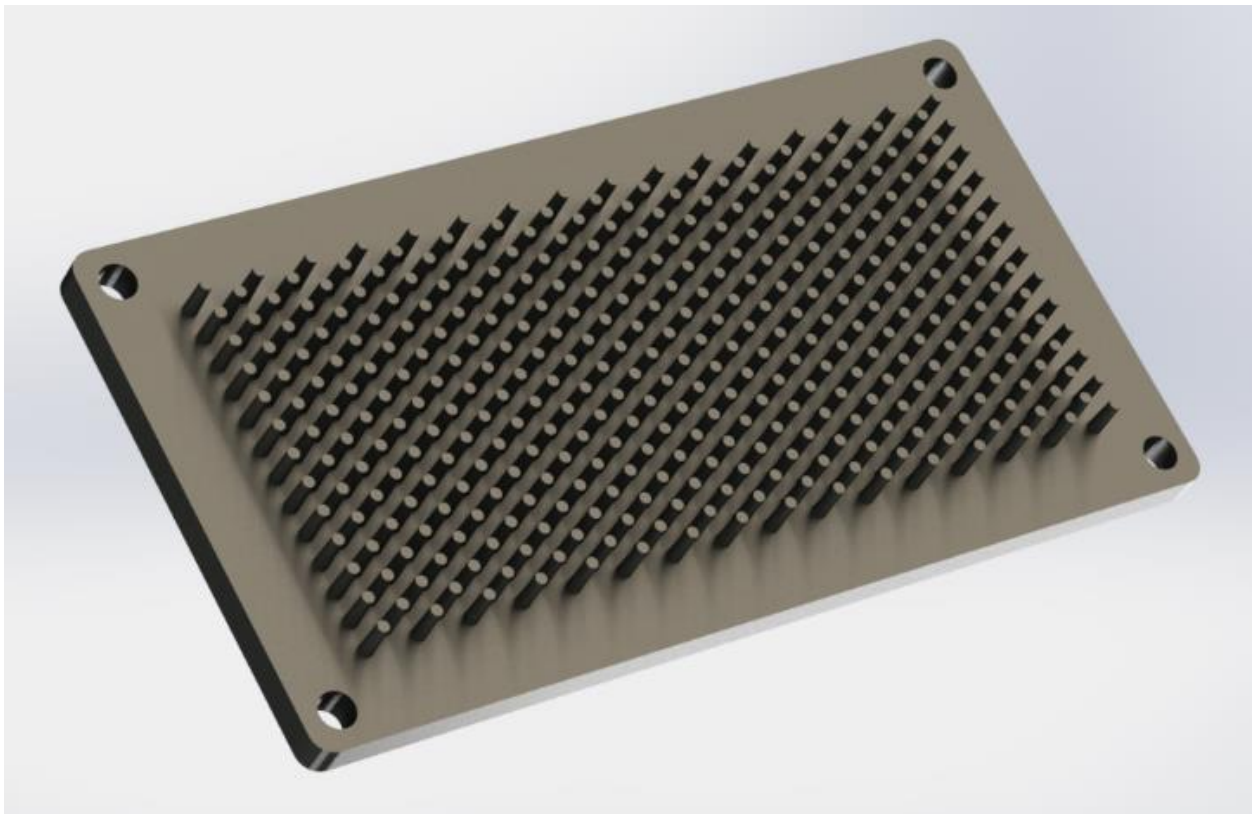


Figure 4.10: Pin-fin baseplate used in water cooled SCPM design

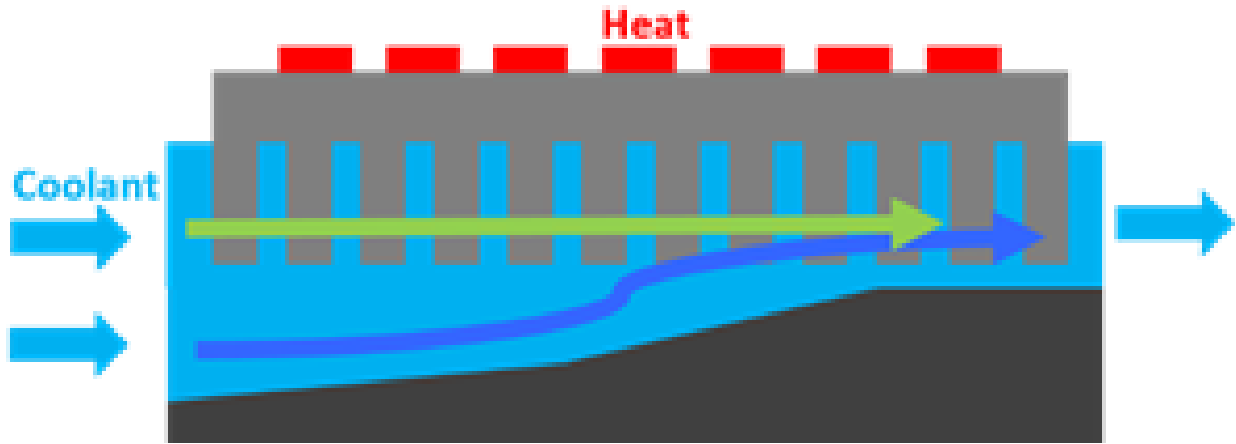


Figure 4.11: Sloped coolant cavity design for compensating power dissipation difference

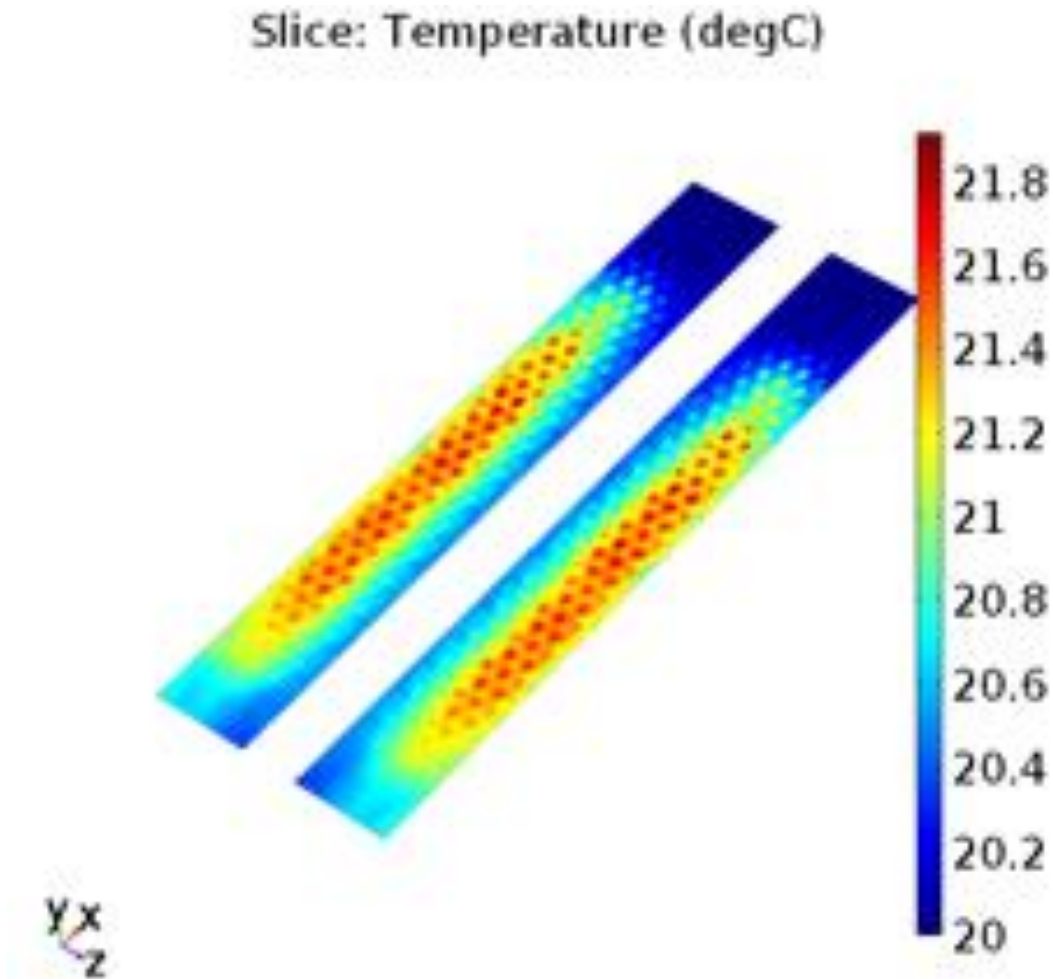


Figure 4.12: Simulated water temperature across baseplate (pin-fin side) with new design

A comparison was done with a conventional symmetric cooling design under the same power dissipation and inlet/outlet coolant pressure difference (Fig. 27, Fig. 28). Result shows the newly proposed design has a more even temperature distribution. Simulation was also done with the JFETs of the lowest stage (J6, J6') dissipating 30% more power than other JFETs, J1~J5 and J1'~J5'.

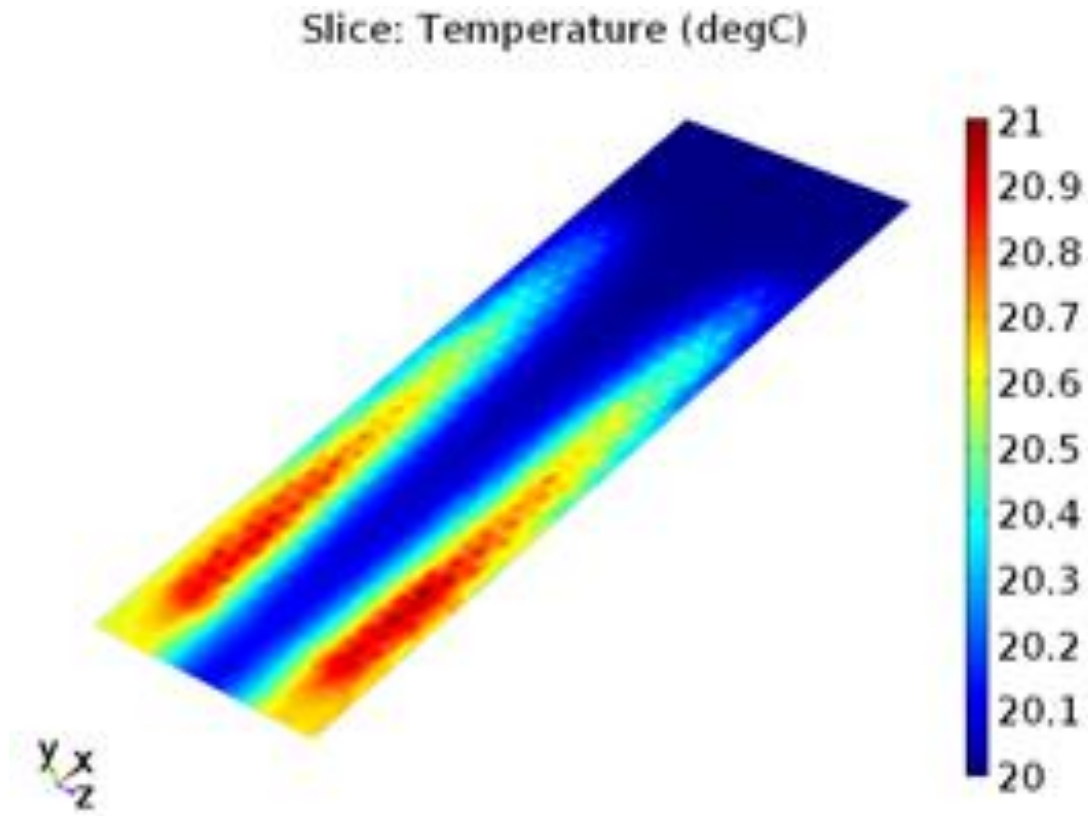


Figure 4.13: Simulated temperature across baseplate (pin-fin side) with conventional design

It is worth noting that the newly proposed design contributes to higher long-term reliability due to the more evenly distributed stress on DBC substrates. Also, the more complicated geometry reduces flow rate for the same pressure difference. Under same flow rate, the newly proposed design should outperform conventional design in terms of highest temperature value.

#### 4.4: Static and dynamic characterization

The proposed SCPM is then fabricated and measured. Totally 4 demonstration SCPMs were fabricated. The first one (S0-6500-050) was populated with one string of the JFETs on one side and used for static blocking and leakage tests. Test voltage was 6100V limited by test equipment.

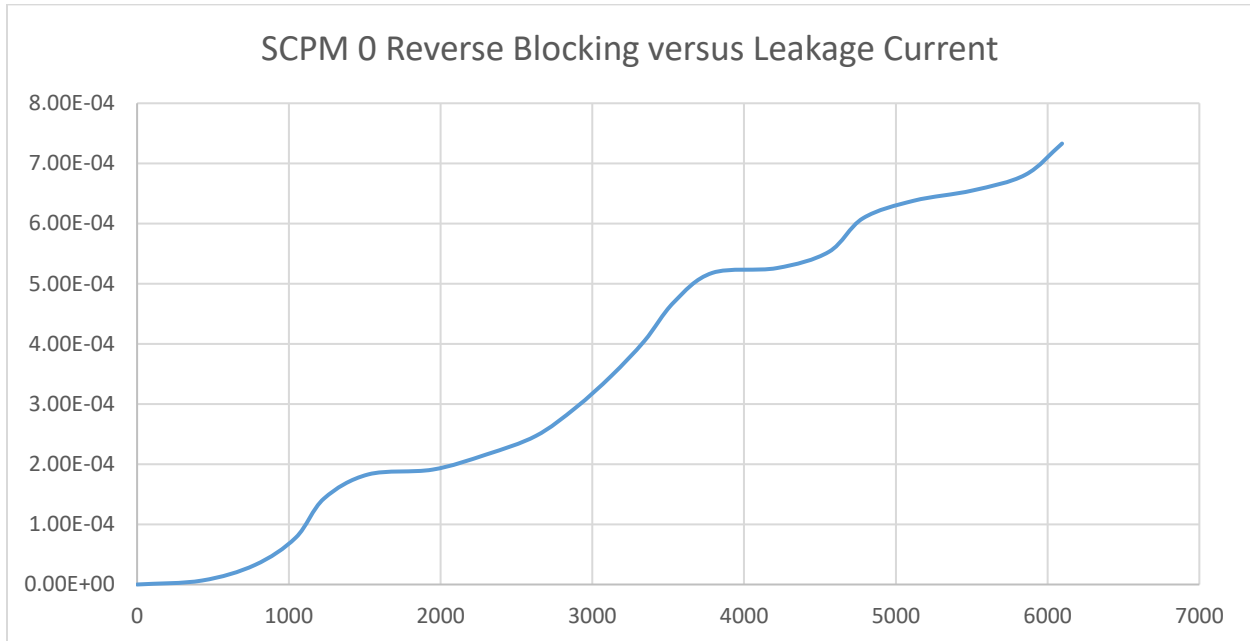


Figure 4.14: SCPM S0 leakage current vs voltage

Leakage current – blocking voltage curve shows a sectioned exponential curve, which is unique to SCPMs. This is caused by the successive breaking down of stages as blocking voltage increases. Also, because the existence of multiple leakage current path and bias resistor across drain and gate of Q1/Q1', the proposed SCPM has higher leakage current than monolithic power switches. This limits the application to those which can tolerate higher leakage current. Safety interrupting devices, such as solid circuit power controllers/circuit breakers, cannot use this SCPM without an additional mechanical interrupting relay. For most other applications, this should not be an issue.

The second module (S1-6500-100) was built with 2 parallel strings of devices and tested for static blocking leakage, on-state resistance and switching performance.

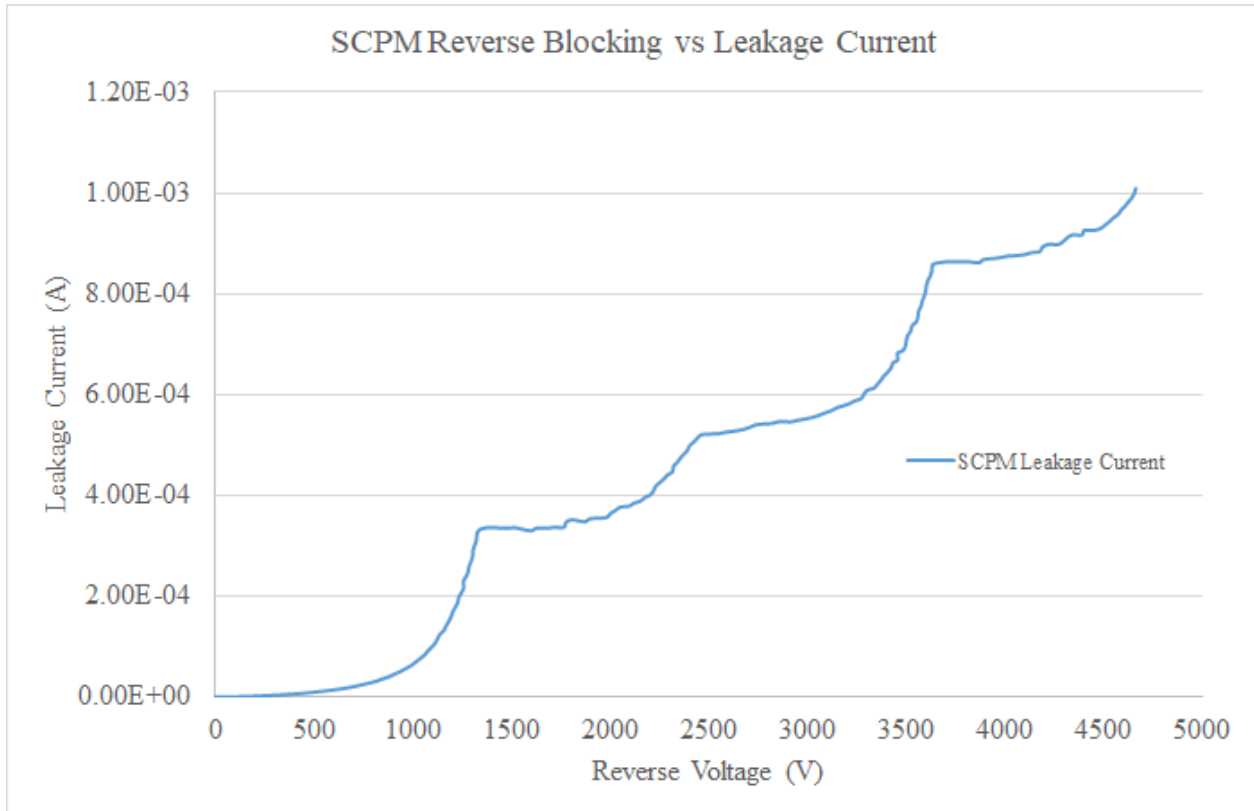


Figure 4.15: SCPM S1 leakage current vs voltage

Off-state leakage test shows 1mA leakage at 4.7kV (equipment limit). At 4.7kV steady state, the balancing network bias resistor should have negligible current flowing through, and leakage current through JFETs should be 0.6mA totally. However, here, 60uA/JFET is used as total drain leakage current, which is the typical value from datasheet. In practice, this value can go to as high as 500uA, therefore the discrepancy between measured value and theoretical value does not disprove the concept.

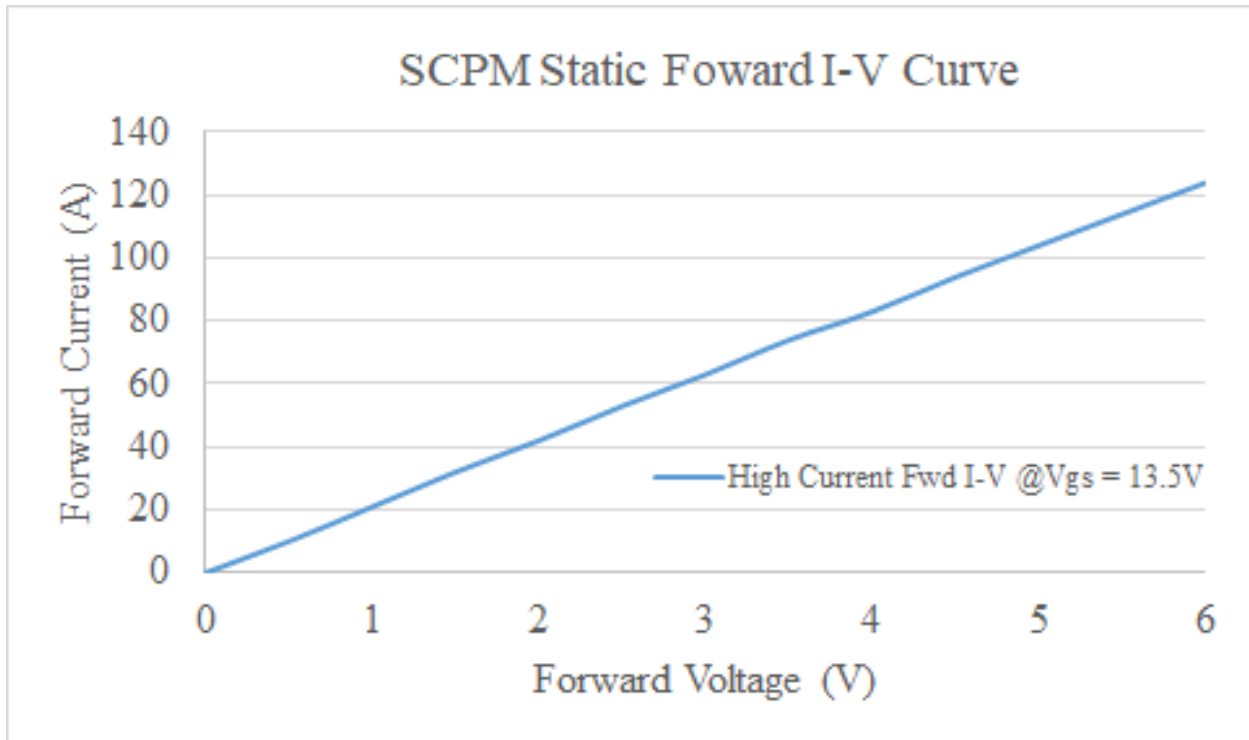


Figure 4.16: SCPM S1 on-state curve tracer test result

Curve tracer test shows 48.5mΩ overall resistance. SCPM S1 has 6 JFETs in series per string, and 2 strings in parallel. Ignoring MOSFET resistance (typ. 0.9mΩ), measured average RDSON of each JFET is  $48.5\text{m}\Omega \cdot 2 / 6 = 16.67\text{m}\Omega$ . This number conforms to datasheet value of UJN1202Z (16mΩ). From 0A to 100A, the demonstration SCPM shows highly linear V<sub>DS</sub>-I<sub>DS</sub> curve, which indicates the SCPM is far from the desaturation limit. However, to prevent damage to the SCPM and/or test equipment, the test was limited to 6V/124A.

A DPT setup was built for testing dynamic switching performance of the SCPM (Fig. 32, Fig. 33). The DPT test setup consists of a high voltage power supply, a large decoupling capacitor, two small, low-inductance local decoupling capacitors, a custom inductor, a custom 8.5kV, 200A Schottky diode module (D1-8500-200), the device under test, a voltage sensor, a current sensor, an oscilloscope, an isolated gate driver, and a test waveform generator. The D1-8500-200

module consists of two parallel connected strings of five 1.7kV 50A Cree Schottky diodes connected in series.

Because the SCPMs have two balancing mechanisms, static balancing and dynamic balancing, to make sure the DUT (SCPM) goes into dynamic balancing mode, a series of pre-pulses are applied before the main pulses arrive. This is implemented with a custom programmed microcontroller board based on Cypress PSoC 5LP CY8C5888LTI. The PSoC board is then connected to a computer with an USB isolator. Neither the USB isolator nor the gate driver isolator are necessary since the source terminal of DUT sits on ground potential. However, to mitigate transient noise as well as to protect operator from faulty circuit, which can happen if the high voltage power supply develops a faulty ground connection, both isolators are added. Combined, the two isolators have a total breakdown voltage above 8kV, higher than the maximum 4kV used in actual testing.

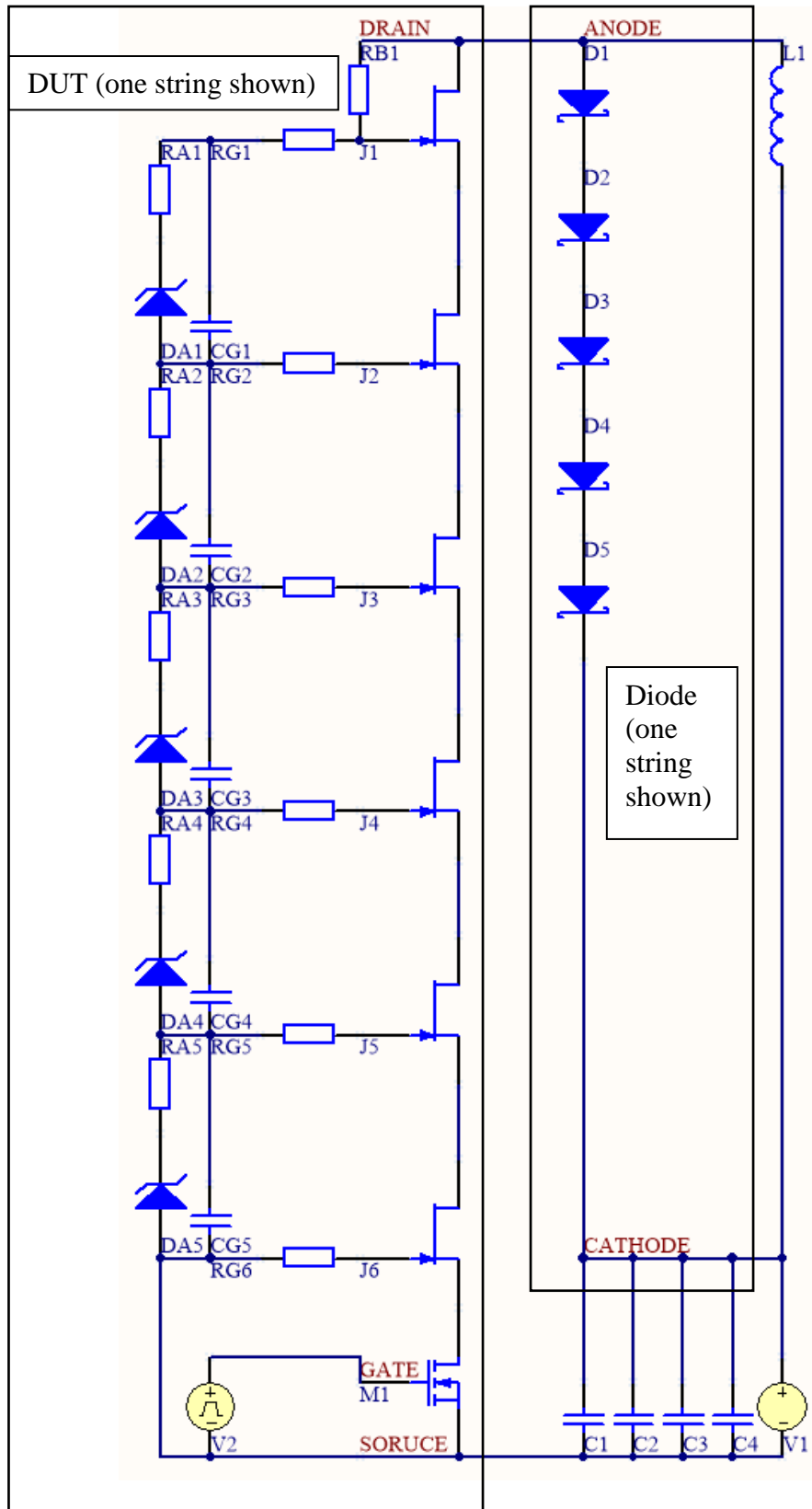


Figure 4.17: DPT electrical diagram

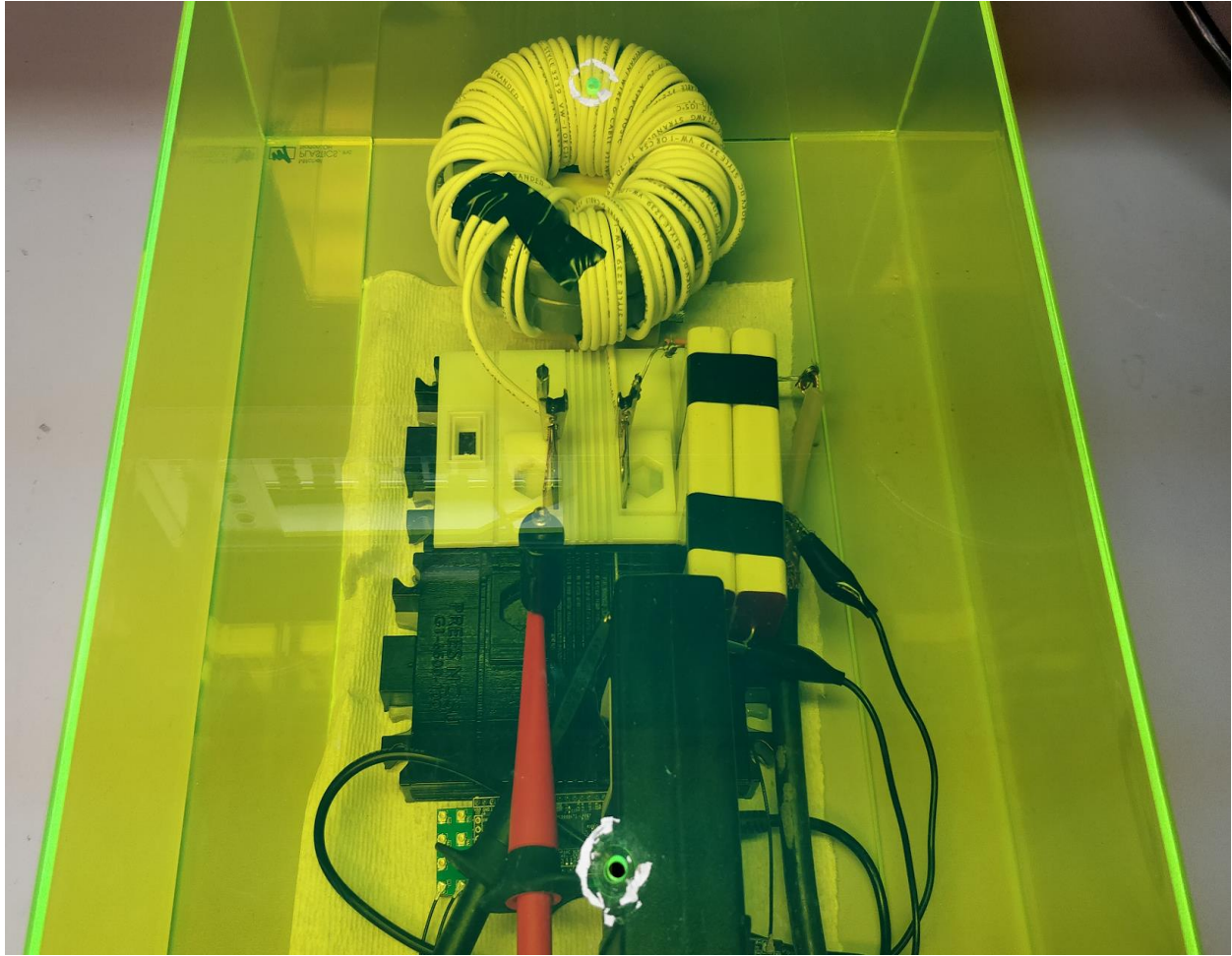


Figure 4.18: DPT test setup

The DPT test was conducted on the SCOM S1 at 4kV DC bus voltage and 110A peak pulse, and the results are shown in Fig. 34 (magenta: drain-to-source voltage, cyan: drain-to-source current, yellow: gate-to-source voltage). The test result shows 45ns current rise and 50ns current fall. The slower turn on time compared with simulation is expected due to loop inductance. During simulation, the series inductance was set to 23nH, which is the inductance of the module itself. However, during testing, the inductance shall include the entire DC bus decoupling path, which is at least 10x higher than 23nH using a parallel rectangular conductor assumption. Fig. 34 shows the voltage fall and voltage rise are 200ns and 45ns respectively, which are also expected due to

the large capacitive load from the Schottky diode module. The noise seen on gate signal (yellow trace on Fig. 34) comes from improper probing, it does not reflect true gate signal.

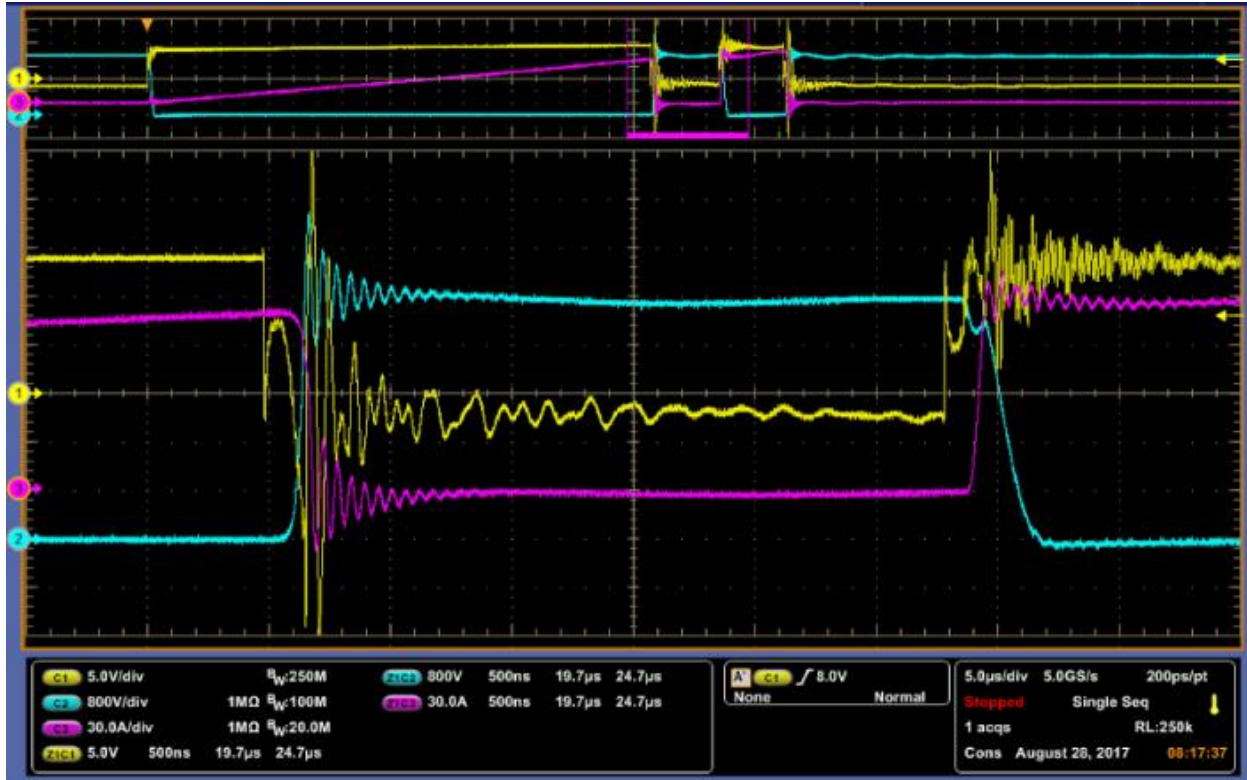


Figure 4.19: DPT at 4kV, 110A

Note that the drain overshoot voltage rises to 5.6kV without clipping indicating the SCPM S1 can block at least 5.6kV.

To measure switching speed without the large capacitive load of D1-8500-200 Schottky diode module, a discrete TO247 device-based SCPM (SD1-6500-050) was fabricated using six USCi UJN1202K JFETs connected in series and packed tightly (schematic identical to Fig. 13). The gate and source terminal of the MOSFET were shorted together to force operation in third quadrant mode. Testing was up to 3kV before failure due to catastrophic avalanche breakdown.

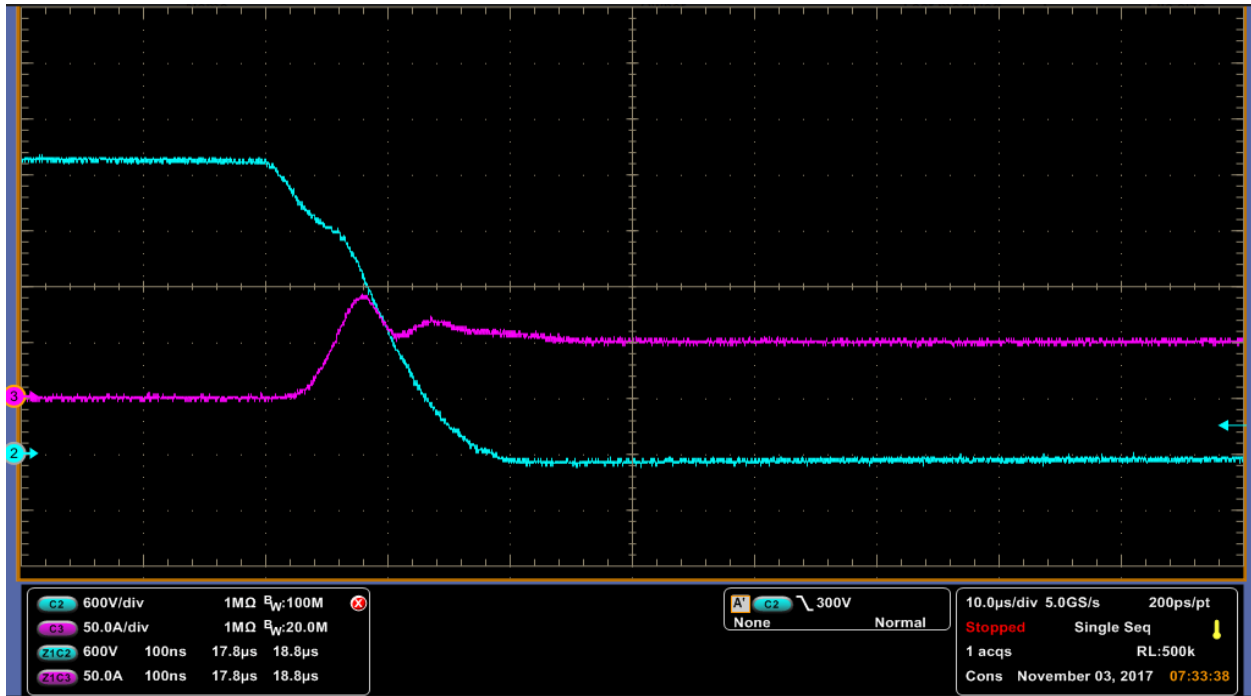


Figure 4.20: Turn on detail at 3kV, 50A

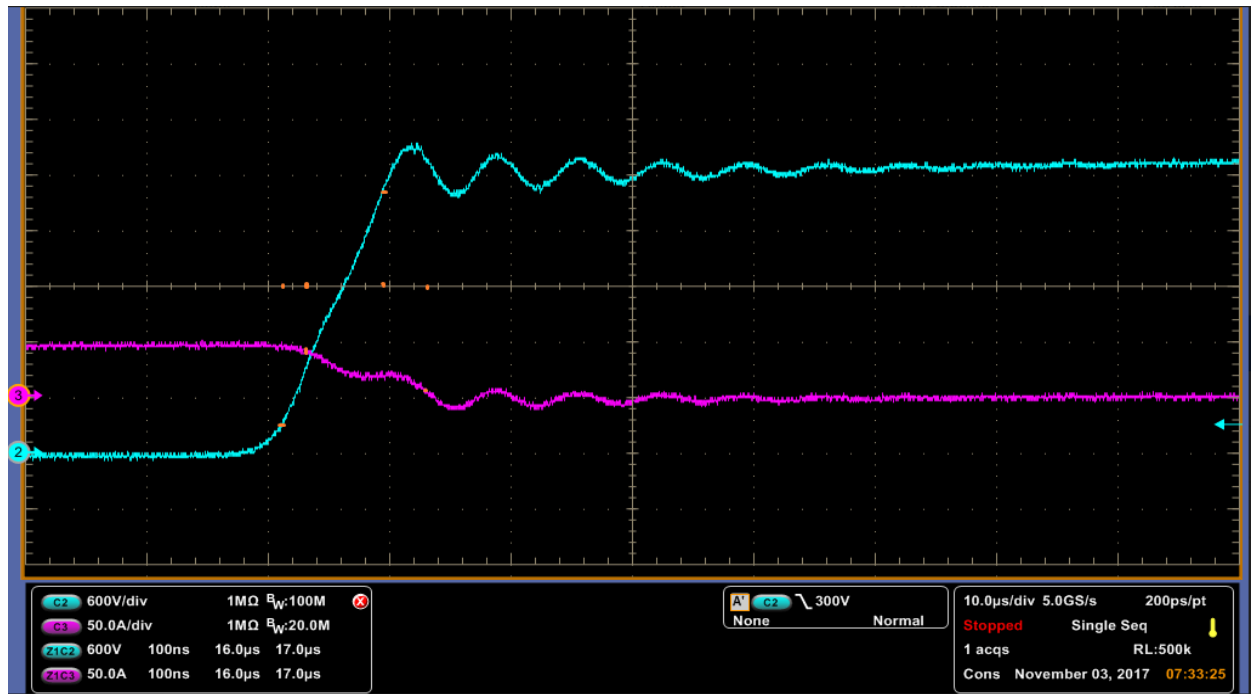


Figure 4.21: Turn off detail at 3kV 50A

The switching test result for SCPM S1 and SD1 at 3kV and 50A is shown in Fig. 35 and Fig. 36. The current rise time is 28ns and the current fall time is 100ns, corresponding to 1786A/us and 500A/us respectively.

## Chapter 5: Increasing Avalanche Energy Handling Capability

### 5.1: Necessity of avalanche energy handling capability

According to previously mentioned formula 2 and formula 4, substrate capacitive coupling switching energy loss and balancing network switching energy loss scale rapidly with stages, cubically and quadratically respectively, scaling a single SCPM (e.g. not multiple SCPMs connected in series) to very high voltage with a large number of stages is impractical. Generally speaking, starting from 4~8 stages the balancing network loss starts to severely limit operating frequency. Substrate capacitive coupling power loss can be greatly reduced if not eliminated with the use of advanced packaging and cooling techniques, such as direct dielectric spray cooling and spiral/serpentine layout without a baseplate. The direct dielectric spray cooling eliminates the need of having a highly thermally conductive baseplate <sup>[9]</sup>, thus almost completely eliminating substrate capacitive coupling power loss. Spiral/serpentine power device connections increase packaging density, reducing overall footprint and hence reducing stray capacitive coupling.

However, balancing network dynamic power loss for a single SCPM cannot be reduced by optimizing layout due to the capacitance is needed for the SCPM to function and is only related to QG of JFETs chosen for a given operation point. To allow more JFETs to be connected in series, the balancing network capacitor ladder must be broken down periodically in order to stop the quadratic term from growing indefinitely. In other words, to series connect a large number of JFETs, they must be grouped in strings of fewer JFETs in series, then the strings are connected in series to form another layer of cascode.

From the tests with previously mentioned manufactured demonstration SCPMs, a serious flaw of this design is revealed – the lack of avalanche energy handling capability. This may not be a considerable issue in a low inductance half bridge design, but if those SCPMs are series connected, voltage imbalance can cause avalanche breakdown during switching transitions.

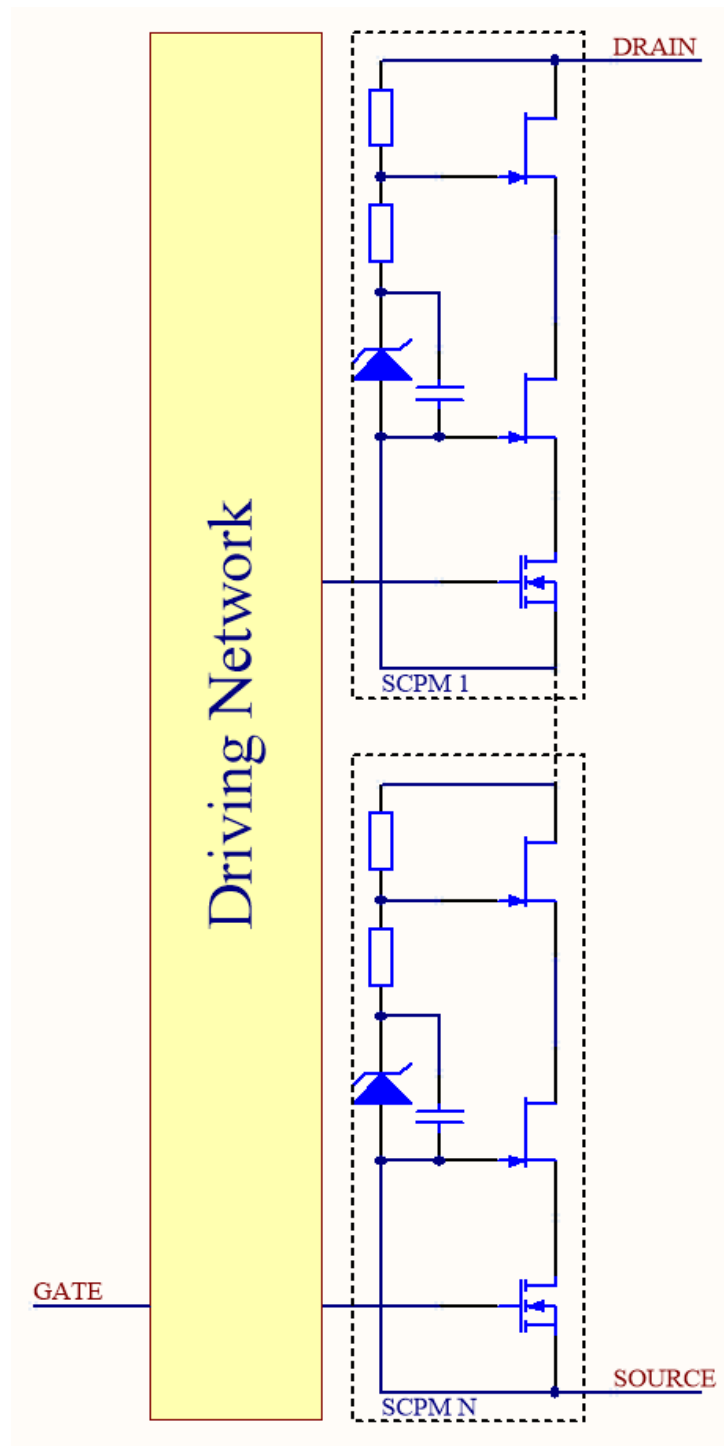


Figure 5.1: Series connecting SCPMs for higher blocking voltage

Fig. 37 shows a typical simplified series connection of SCPMs. In this case, if DC bus voltage is higher than maximum blocking voltage of either SCPM, then during switching transition, turn-on speed and delay mismatch can cause avalanche breakdown of the slower SCPM during turn-on

process. Similarly, turn-off speed and delay mismatch can cause avalanche breakdown of the faster SCPM during turn-off process. Therefore, it is important to find a way to greatly improve avalanche energy handling capability of SCPMs.

To increase avalanche energy handling capability, the JFETs must handle the majority energy in an avalanche event so that the avalanche diodes in the balancing network do not have to handle this energy. In addition, the D-S junction of a JFET can handle much more energy than the D-G junction. Unique to depletion mode power devices such as JFETs, due to the negative gate-to-source voltage during turn-off, D-G junction breaks down before D-S junction [10]. Therefore, to increase avalanche energy handling capability, two things must be done:

1. Divert avalanche current to JFETs from avalanche diodes
2. Limit JFET gate current during an avalanche breakdown

## 5.2: Directing avalanche energy from diodes to JFETs

To achieve the first goal, a resistor (RA1~RA5) can be added in series with each avalanche diode.

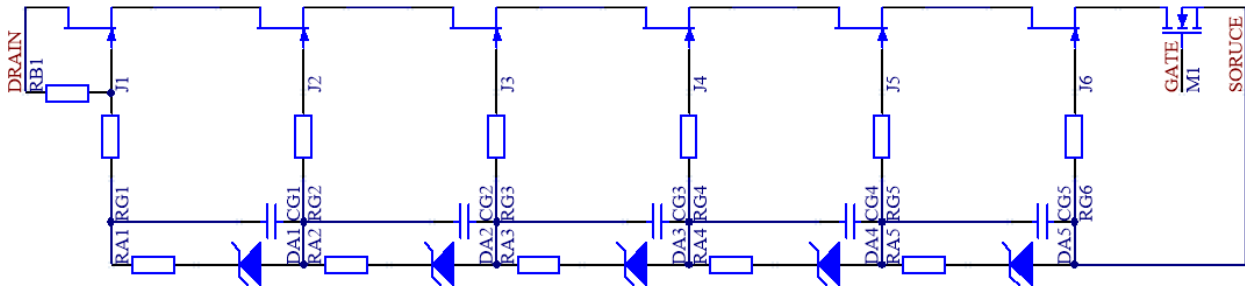


Figure 5.2: Topology with avalanche breakdown immunity showing diode resistors RA1~RA5

Since during normal static balancing operation, only very little current flows through those avalanche diodes, adding a resistor will not change their voltage by any significant margin. With UJN1202Z, bias current required for its balancing network is less than 1mA, therefore a 10kΩ resistor introduces no more than 10V error. However, during an avalanche breakdown event,

assuming the avalanche diodes break down at 1.3kV and JFETs breakdown at 1.5kV, only 200V is applied across each avalanche diode resistor. The 10kΩ resistor limits avalanche current to 20mA. Assuming the avalanche breakdown subsides in 20ns, the energy dissipated by the avalanche diode is:

$$E_D = I_D * V_D * t = 20\text{mA} * 1200\text{V} * 20\text{ns} = 0.48\text{uJ}$$

Assuming switching frequency is 100kHz, then the average power dissipation on each combined avalanche diode and resistor is:

$$P_D = E_D * f = 0.48\text{uJ} * 100\text{kHz} = 48\text{mW}$$

Without the series resistors, the diodes will have to dissipate very high average power. Assuming the avalanche diodes have 10Ω internal resistance, and the same conditions above, then the energy each avalanche diode will dissipate per switching cycle is:

$$E_D = I_D * V_D * t = (200\text{V}/10\Omega) * 1500\text{V} * 20\text{ns} = 600\text{uJ}$$

At 100kHz, each avalanche diode will dissipate an average power of:

$$P_D = E_D * f = 600\text{uJ} * 100\text{kHz} = 60\text{W}$$

Therefore, the addition of series resistors has increased avalanche handling capability of the avalanche diodes by a factor of 1250.

### **5.3: Simulation and verification**

To verify this theory, a simulation was carried out with LTSpice and 6\*UJN1202Z JFETs connected in series, switching an unclamped 5uH inductor at 4kV/100A. The simulation circuit is shown in Fig. 39. Simulation result shows great reduction of average avalanche diode power dissipation.

Table 5.1: Average power dissipation comparison

Energy Loss per Switching Cycle (Designator in Fig. 39)	Without Diode Resistors	With Diode Resistors
D1	266.12uJ	0.71uJ
D2	164.57uJ	0.67uJ
D3	60.27uJ	0.22uJ
D4	0.07uJ	0.04uJ
D5	0.04uJ	0.04uJ

Table 5 shows avalanche breakdown happens on the first 3 avalanche diodes, while D4 and D5 suffer from no avalanche breakdown. With higher or lower external inductance or DC bus fluctuation, more or less diodes may enter avalanche breakdown. For the first stage (J1, C1, D1), the addition of an avalanche diode resistor greatly reduces energy dissipation on avalanche diode. In this case, D1 dissipates 0.71uJ per switching cycle, which is 0.27% compared with energy dissipation of D1 without diode resistor R6. In other words, avalanche handling capability has been improved by 373 times.

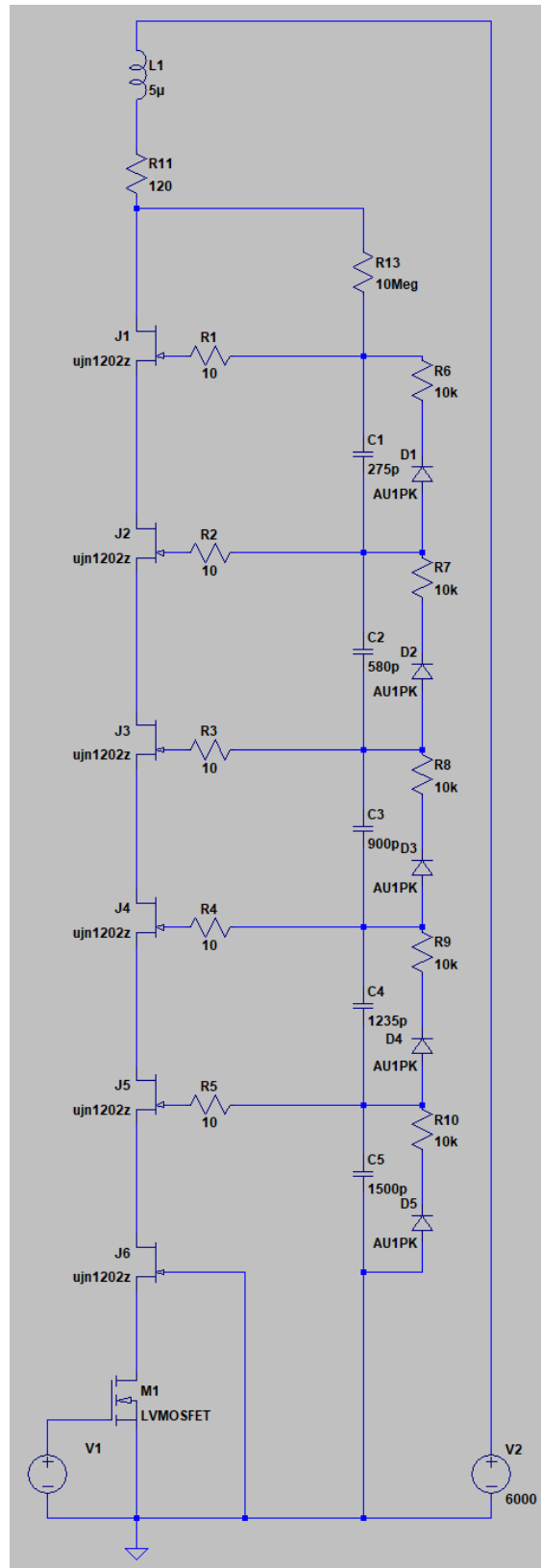


Figure 5.3: Simulation bench for avalanche current distribution

With the introduction of diode resistors, avalanche current is shifted from avalanche diodes to JFETs and set new balancing voltage distribution. If avalanche voltage is too high, higher than the  $V_{DS\_MAX}$  of all JFETs combined in an SCPM string, J6 will suffer from D-G junction avalanche because the gate of J6 is connected to module source, which is the lowest potential point in the SCPM. Under extreme condition (8.5kV, 4uH, 70A), J6 breakdown can be observed (Fig. 40).

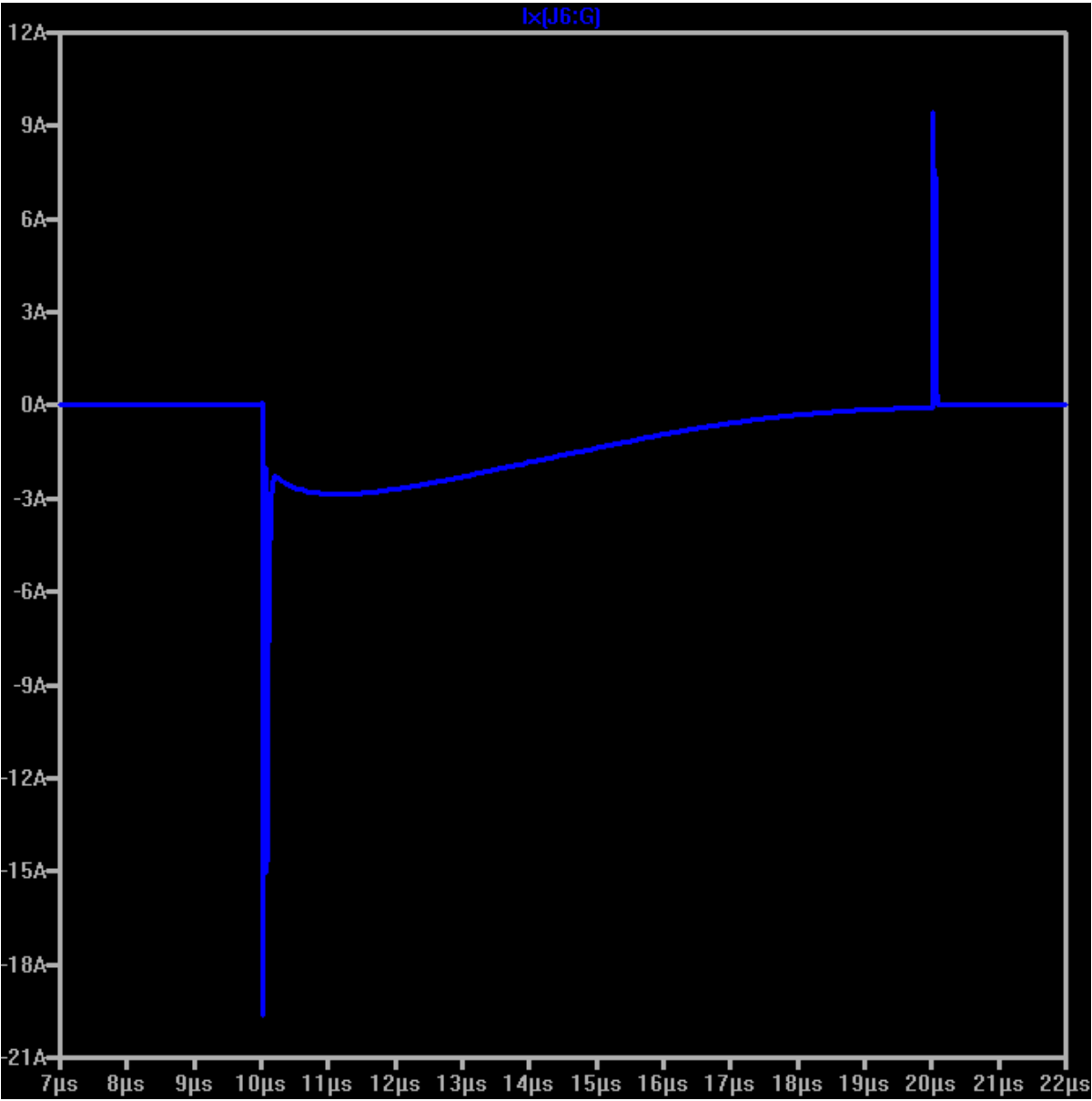


Figure 5.4: Gate current of J6 before and during avalanche breakdown



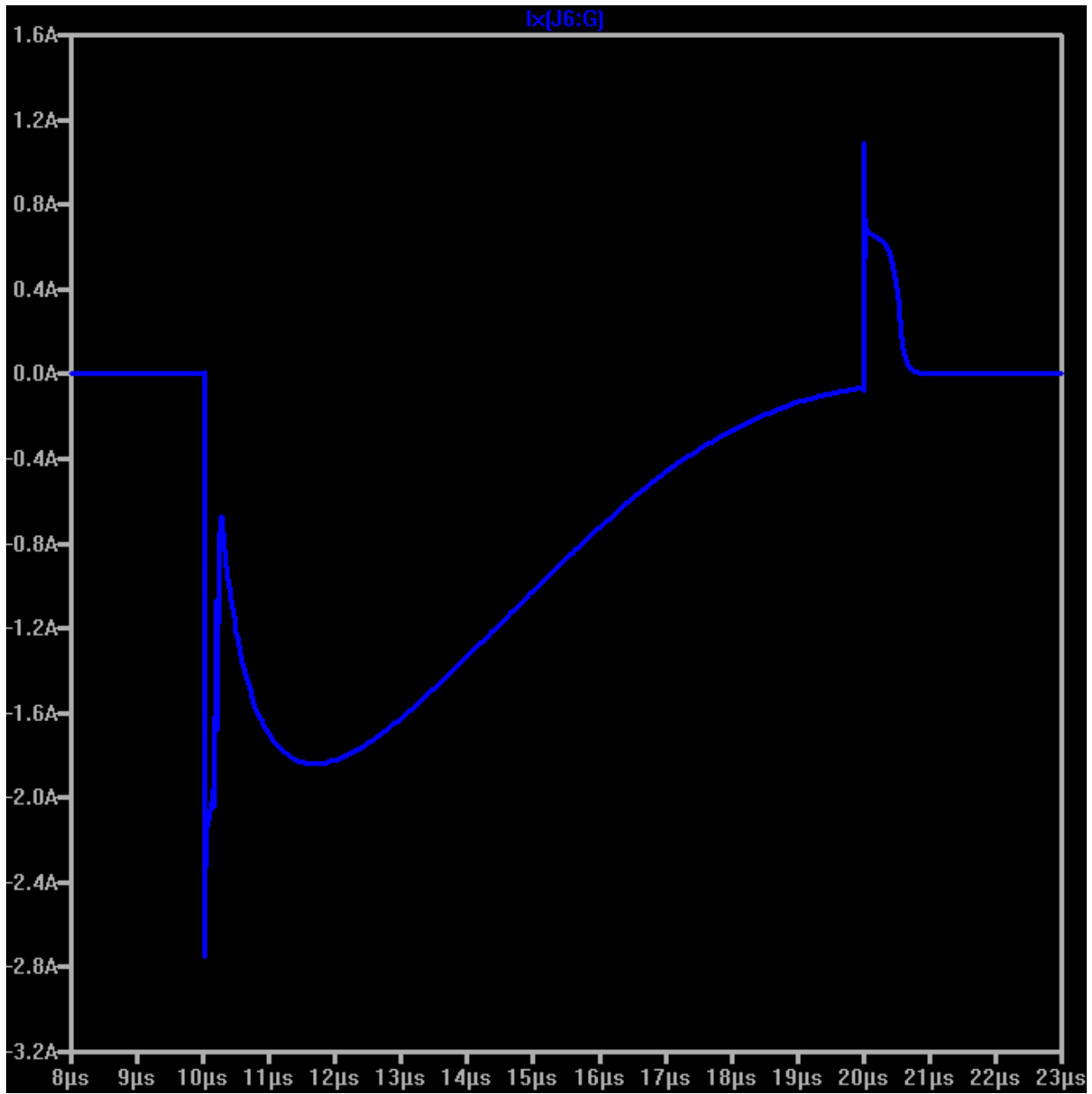


Figure 5.6: Gate current of J6 before and during avalanche breakdown with gate resistor

Fig. 42 shows the proposed SCPM topology can carry substantial avalanche energy. Combined with RA1~RA5, the topology shown in Fig. 41 can be used in harsh avalanche environments.

## Chapter 6: Scaling Multiple SCPMs

### 6.1: Using SCPMs as higher voltage JFETs

With individual SCPMs capable of handling avalanche energy, series connecting SCPMs becomes possible. As mentioned before, this research work focuses on reducing balancing network switching energy loss  $E_B$ . Due to the quadratic term in formula 4 (see below),  $E_B$  grows rapidly as number of stages  $N$  increases, hence a long string of JFETs must be periodically broken into shorter strings to reset  $N$ .

$$E_B = 1/4 * (Q_G - Q_D) * V_{DS} * (N^2 - N)$$

One approach to achieve this is by removing the bottom MOSFET and use the bottom JFET to control an SCPM stack, effectively using this JFET controlled SCPM as a higher blocking voltage JFET. This SCPM is referred as “Unit SCPM” in this paper.

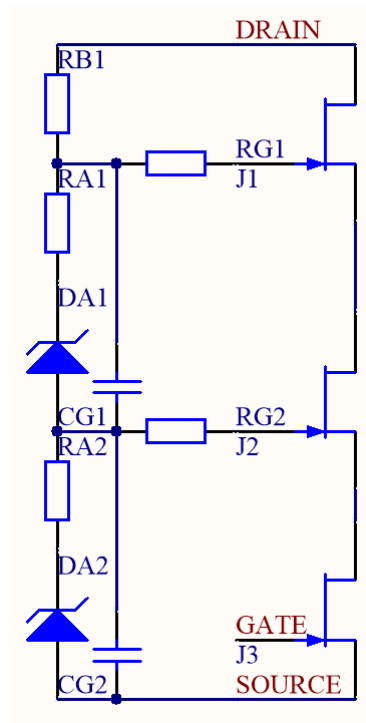


Figure 6.1: Unit SCPM structure of SCPM-J

Balancing network current flows through the drain of lower stages, instead of balancing network of lower stages (as seen in Fig. 44). By doing so, power loss of upper stages still has to be dissipated by lower stage JFETs, but the increasing of capacitance is broken periodically inside each Unit SCPM. This reduces the quadratic term in  $E_B$ . This connection is named "SCPM-J" since it uses a Unit SCPM as a JFET to create a larger cascaded SCPM.

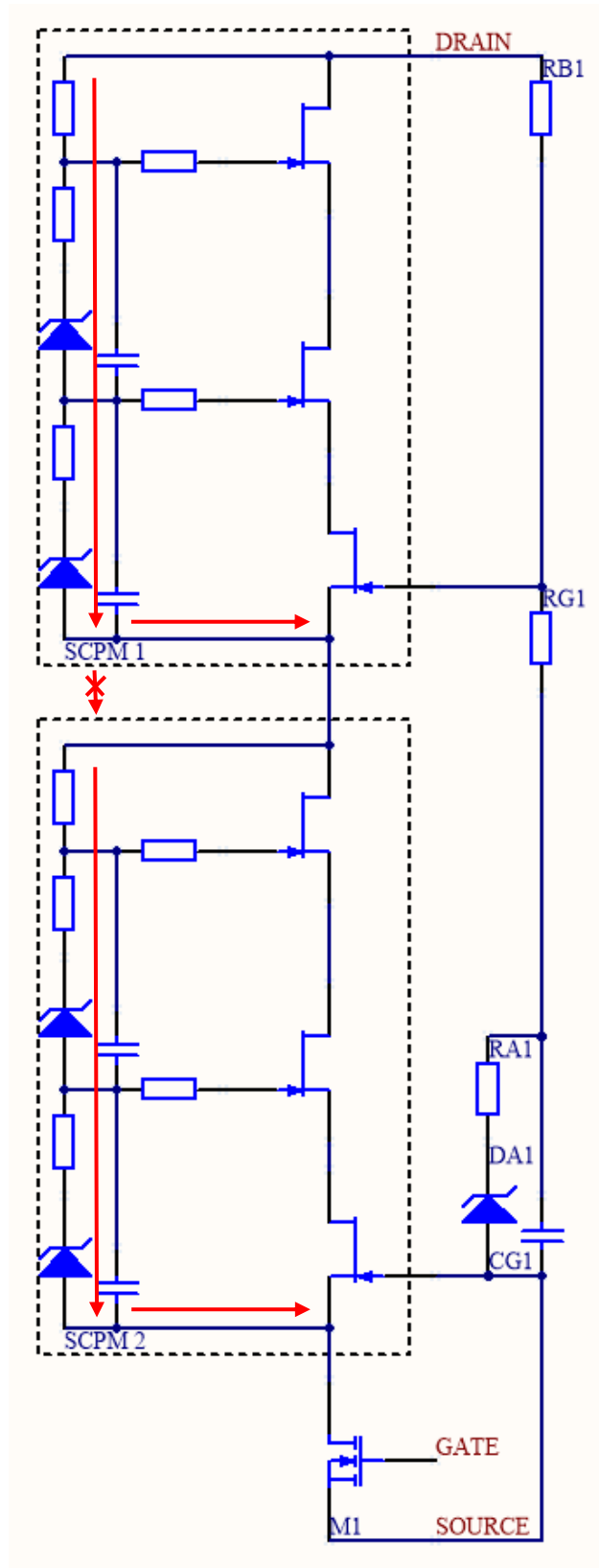


Figure 6.2: SCPM-J using individual Unit SCPMs as JFETs

However, each Unit SCPM must have their gates connected in another balancing network, which introduces quadratic terms again. Despite the re-introduction of quadratic term, because the number of capacitors needed per Unit SCPM as well as SCPM-J is their respective stage count  $N$  minus 1,  $N-1$  is introduced to quadratic function instead of  $N$ , therefore by using SCPM-J structure, overall  $E_B$  can be reduced.

To provide an example of series connecting 6 JFETs, refer to Fig. 44. Every 3 JFETs are series connected, forming a Unit SCPM. Two Unit SCPMs are series connected, forming the entire SCPM. Consider all JFETs to be perfectly balanced and each JFET blocks 1kV and has a gate charge of 300nC. Using formula:

$$C_n = (Q_G - Q_D) / V_{DS}$$

Omitting negligible  $Q_D$ , Gate capacitors required in each Unit SCPM are 300pF and 600pF.

Because each Unit SCPM can block 2kV, applying the abovementioned formula,  $C_{G1} = 150\text{pF}$ .

Therefore, for each Unit SCPM:

$$E_{B\_INT} = 1/2 * (300\text{pF} + 600\text{pF}) * 1000\text{V}^2 = 0.45\text{mJ}$$

For the external balancing network:

$$E_{B\_EXT} = 1/2 * 150\text{pF} * 2000\text{V}^2 = 0.30\text{mJ}$$

The total balancing network energy loss is:

$$E_B = 2 * E_{B\_INT} + E_{B\_EXT} = 0.45\text{mJ} * 2 + 0.30\text{mJ} = 1.20\text{mJ}$$

If the 6-JFET SCPM is implemented as one 6-stage SCPM, using formula 4:

$$E_B = 1/4 * Q_{DS} * V_{DS} * (N^2 - N) = 1/4 * 300\text{nC} * 1000\text{V} * (36 - 6) = 2.25\text{mJ}$$

As can be seen from the results shown above, the use of SCPM-J topology reduces  $E_B$  by 46.67%.

## 6.2: Optimizing series connected SCPM design

A generalized expression for SCPM-J with  $N_1$  stages per Unit SCPM and  $N_2$  Unit SCPMs per external cascode can be expressed as:

$$E_B = \sum(E_{C\_unit}) + \sum(E_{C\_out}) = E_{C\_unit} * N_2 + E_{C\_out}, \text{ where:}$$

$$E_{C\_unit} = 1/2 * (Q_{DS}/V_{DS}) * (1 + \dots + N_1 - 1) * V_{DS}^2 = 1/4 * Q_{DS} * V_{DS} * (N_1^2 - N_1), \text{ and}$$

$$E_{C\_out} = 1/2 * (Q_{DS}/V_{DS}/N_1) * (1 + \dots + N_2 - 1) * (V_{DS} * N_1)^2 = 1/4 * Q_{DS} * V_{DS} * N_1 * (N_2^2 - N_2)$$

Therefore, overall expression if  $E_B$  can be written as:

$$E_B = 1/4 * Q_{DS} * V_{DS} * [(N_1^2 - N_1) * N_2 + (N_2^2 - N_2) * N_1],$$

$$E_B = 1/4 * Q_{DS} * V_{DS} * N_1 * N_2 * (N_1 + N_2 - 2)$$

From the expression shown above, and for a given blocking voltage (determined by  $N_1 * N_2 * V_{DS}$ ), the minimum  $E_B$  occurs at  $N_1 = N_2$ . Because  $N_1$  and  $N_2$  must both be integer,  $N_1$  and  $N_2$  must be rounded to integer while still maintaining  $N_1 * N_2$  equal to the intended value. In this case,  $N_1 = 3, N_2 = 2$  and  $N_1 = 2, N_2 = 3$  both yield the optimum result.

The same concept can be further expanded to more than 2 layers of SCPMs. For a given total  $N$  series connected JFET structure, it can also be implemented with 3 layers of SCPMs or more.

For example, a 20kV SCPM made of UJN1202Z JFETs balanced at 1kV requires  $N = 20$  JFETs in series. It can be implemented with the single layer configuration ( $N_1 = 20, N_2 = 1$ ) as well as the following 2-layer configurations:

$$N_1 = 10, N_2 = 2$$

$$N_1 = 5, N_2 = 4$$

$$N_1 = 4, N_2 = 5$$

$$N_1 = 2, N_2 = 10$$

In addition, these 3-layer configurations are also possible:

$N_1=5, N_2=2, N_3=2$

$N_1=2, N_2=5, N_3=2$

$N_1=2, N_2=2, N_3=5$

Each configuration has its own  $E_B$  value, hence finding the lowest  $E_B$  yields the optimum design.

A computer program for breaking down a given  $N$  to possible configurations and calculating their  $E_B$  values is developed. The program then finds the combination(s) with lowest  $E_B$  value.

Finally, the program selects the configuration which requires minimum amount of higher voltage capacitors. The result is the optimum design. This program assumes all JFETs have the same balancing voltage, and it does not try to decrease JFET voltage to adjust  $N$  to a square number or cubic number. Therefore, this program can still be further improved, but that is not in the scope of this research.

In this case, for  $N=20$ , the program enumerates the following possible configurations:

Table 6.1:  $E_B$  value for all possible configurations at  $N=20$

Configuration	Total $E_B$ ( $Q_{DS}=300nC, V_{DS}=1000V$ )
$N_1=20, N_2=1$	28.5mJ
$N_1=10, N_2=2$	15.0mJ
$N_1=5, N_2=4$	10.5mJ
$N_1=4, N_2=5$	10.5mJ
$N_1=2, N_2=10$	15.0mJ
$N_1=5, N_2=2, N_3=2$	9.0mJ
$N_1=2, N_2=5, N_3=2$	9.0mJ
$N_1=2, N_2=2, N_3=5$	9.0mJ

The program also calculates the minimum cost solution with capacitor penalty factor linearly related to  $C \cdot V$  value. This cost model resembles a commercial MLCC economical and reliability cost model at high voltage.

Table 6.2: Total cost factor for all possible configurations at N=20

Configuration	Total Cost Factor ( $Q_{DS}=300nC$ , $V_{DS}=1000V$ )
N1=20, N2=1	57000nFV
N1=10, N2=2	27300nFV
N1=5, N2=4	13800nFV
N1=4, N2=5	12000nFV
N1=2, N2=10	16500nFV
N1=5, N2=2, N3=2	12900nFV
N1=2, N2=5, N3=2	9300nFV
N1=2, N2=2, N3=5	7500nFV

Therefore, after including the cost model, the optimum design is N1=5, N2=2, N3=2.

This program assumes the first priority is to reduce  $E_B$ . Even if one design has the lowest cost factor, as long as its  $E_B$  is not the lowest, it is not considered optimum. This may or may not be desired behavior in all applications. To obtain optimum design in all applications, a figure of merit is introduced based on reciprocal of weighted average of total  $E_B$  and total cost factor.

The new program was tested at respective weight of  $E_B$  and penalty factor,  $K_{EB}=1.0/mJ$  and  $K_{PF}=0.001/nFV$ . The result is shown below:

Table 6.3: Figure of merit for all possible configurations at N=20

Configuration	Figure of Merit ( $Q_{DS}=300nC$ , $V_{DS}=1000V$ )
N1=20, N2=1	85.5
N1=10, N2=2	42.3
N1=5, N2=4	24.3
N1=4, N2=5	22.5
N1=2, N2=10	31.5
N1=5, N2=2, N3=2	21.9
N1=2, N2=5, N3=2	18.3
N1=2, N2=2, N3=5	16.5

Under the new condition, N1=2, N2=2, N3=5 is the optimum design.

Simulation is done on this design with the circuit diagram shown in Fig. 45:

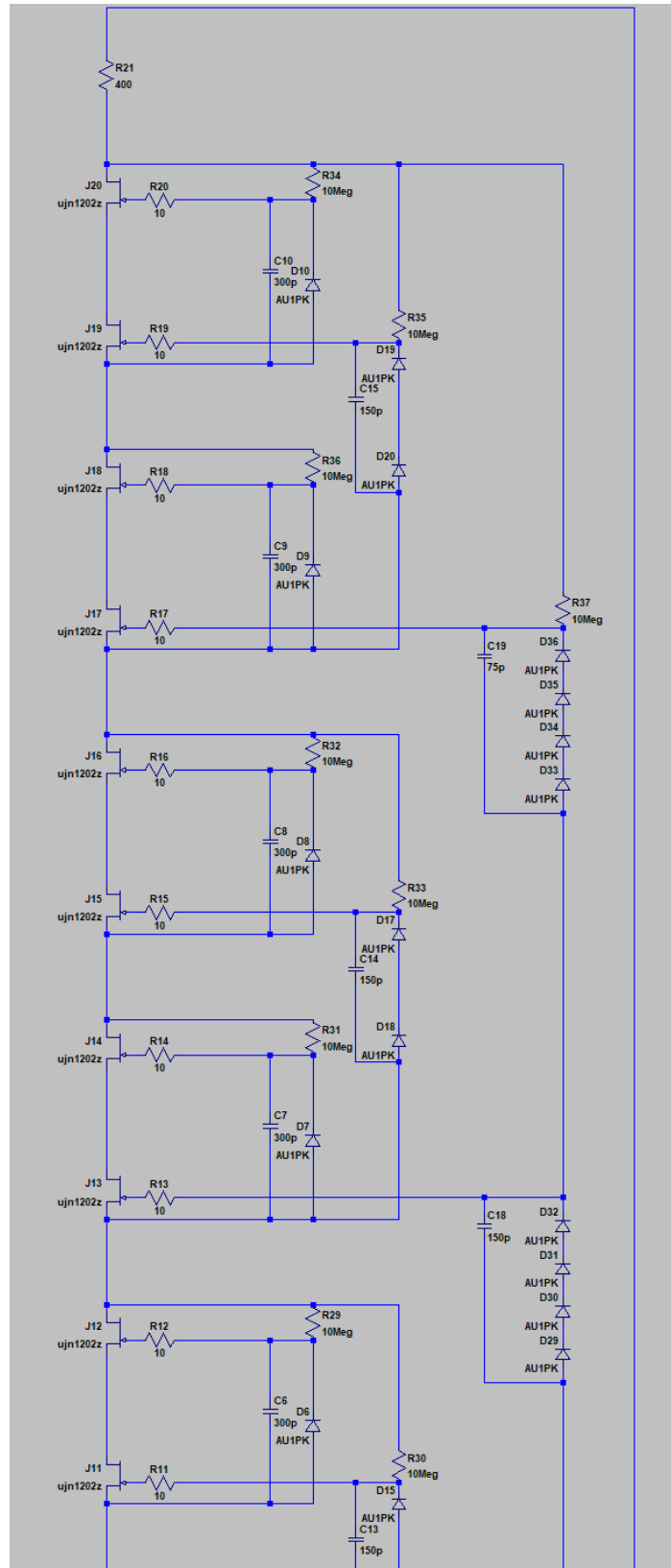


Figure 6.3a: Simulation bench of 2\*2\*5 SCPM-J (upper half)

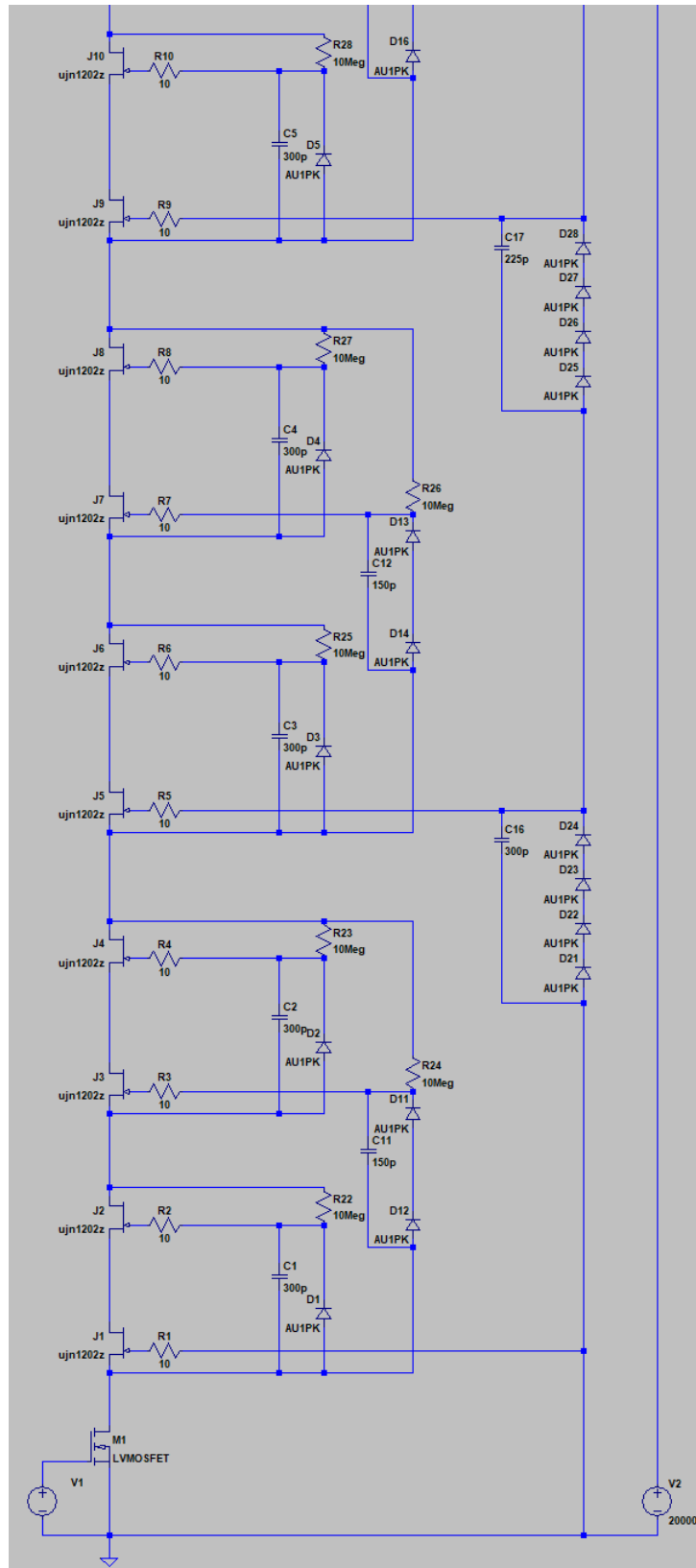


Figure 6.3b: Simulation circuit of 2\*2\*5 SCPM-J (lower half)

The simulation shows the proposed SCPM-J structure not only works, but also has lower switching loss than a 1-layer SCPM of same amount of JFETs connected in series.

Table 6.4: Simulation result of 2\*2\*5 2-layer SCPM-J compared with 20-stage SCPM

Configuration	JFET Switching Energy	Balancing Network Resistor Energy
N1=20, N2=1	37.52mJ	1955.73uJ
N1=2, N2=2, N3=5	30.40mJ	1514.76uJ
Loss Reduction	18.98%	22.55%

### 6.3: Using SCPMs as higher voltage MOSFETs

Not only Unit SCPMs, Single SCPMs and SCPM-Js (collectively called sub-SCPM) can also be connected in series, as shown in Fig. 46. Because this topology has MOSFETs in each sub-SCPM, this topology is called SCPM-M.

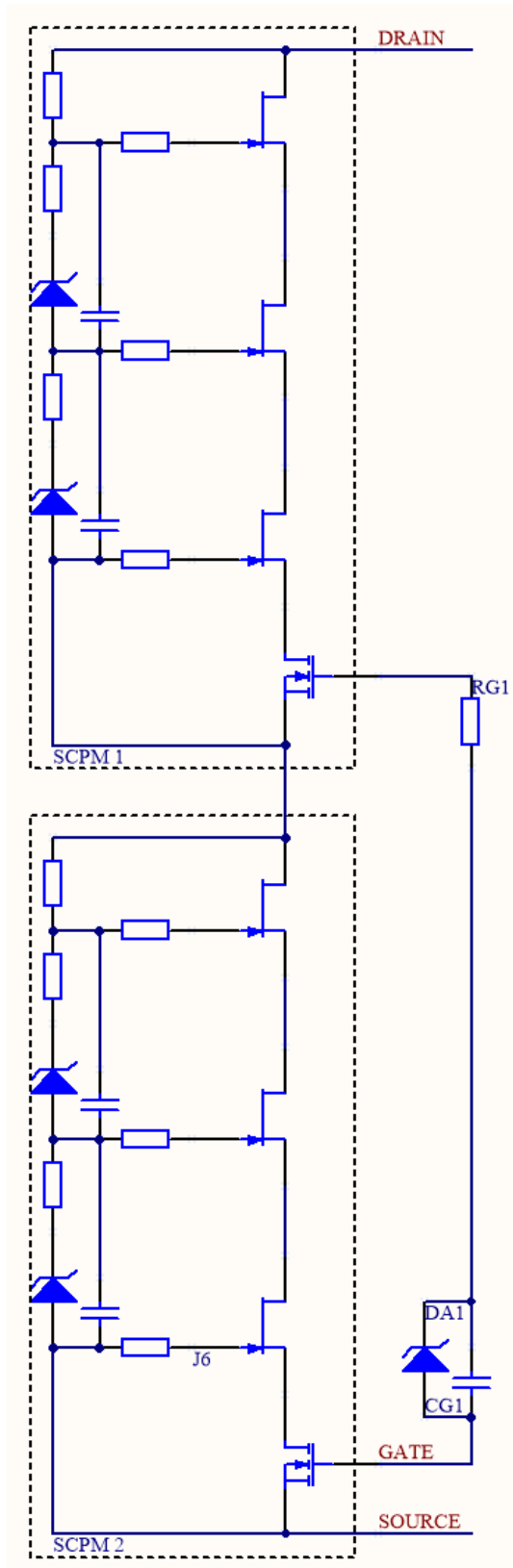


Figure 6.4: SCPM-M using individual Unit SCPMs as MOSFETs

Unique to SCPM-J, SCPM-M depends on avalanche diodes to charge MOSFETs in sub-SCPMs. Therefore, the charge flowing through external balancing network gate capacitors (CG1, etc.) cannot be directly modelled as  $Q_{G\_MOSFET}/V_{DS\_SINGLE\_SCPM}$ . In addition, to protect gate oxide, non-linear protection circuitry, such as Zener diodes or other clamping devices, needs to be added to each sub-SCPMs, this further complicates the calculation of external balancing network gate capacitors (CG1, etc.).

Furthermore, because positive gate driving current flows through external balancing network avalanche diodes (DA1, etc.), addition of series resistors to those diodes can drastically slow down switching speed of SCPM-M. Therefore, the avalanche energy handling capability improvement method discussed in chapter 5 cannot be used. As such, SCPM-M topology has far less avalanche handling capability compared with SCPM-J topology.

Finally, unlike SCPM-J which can be scaled to many stages at the cost of higher external balancing network power loss, maximum stage number of SCPM-M is limited by voltage drop on external balancing network avalanche diodes (DA1, etc.) as well as voltage drop ( $R_{DSON} * I_{DS}$ ) of JFETs and MOSFETs. The maximum allowable number of sub-SCPMs SCPM is defined as:

$$N_{MAX} = \text{floor}[(V_{GS\_MAX} - V_{GS\_MIN}) / (V_F * N_{Diode} + V_{DSON})] + 1$$

$$\text{Where } V_{DSON} = I_{DS} * (R_{DSON\_JFET} * N_{JFET} + R_{DSON\_MOSFET})$$

With typical MOSFETs (fully turned on at 8V, maximum gate voltage of 25V,  $R_{DSON} = 2\text{m}\Omega$ ), UJN1202Z JFET ( $R_{DSON} = 16\text{m}\Omega$ ,  $V_{DS} = 1\text{kV}$ ), typical diodes (3.3kV each,  $V_F = 1.7\text{V}$ , according to GeneSiC GAP3SLT33 datasheet), and a 6kV/50A per Single SCPM:

$$V_{DSON} = 50\text{A} * (16\text{m}\Omega * 6 + 2\text{m}\Omega) = 4.9\text{V}$$

$$N_{MAX} = \text{floor}[(25\text{V} - 8\text{V}) / (1.7\text{V} * 2 + 4.9\text{V})] + 1 = 2$$

Therefore, SCPM-M is only good for relatively low voltage applications. With 1.2kV JFETs, maximum voltage of the SCPM-M topology can achieve is 14.4kV, with 3.3kV JFETs, maximum voltage of the SCPM-M topology can achieve is 39.6kV.

So far, all SCPM topologies discussed are based on passive balancing, which is using only RCD networks to generate gate driving voltages. It is also possible to introduce active gate drivers to drive SCPMs. To power gate drivers, bootstrapping circuit is used. Active gate driving of SCPM-J is not discussed in this work because driving a JFET is more difficult than driving a MOSFET due to the negative voltage requirement, as well as the gate charge of a low voltage control MOSFET is much lower than the gate charge of a high voltage JFET of similar current rating.

A bootstrapping SCPM-M implementation is shown in Fig. 47. Similar to passive SCPM-M, the maximum number of stage is limited by voltage drop on bootstrapping diode (DB1, etc.) forward voltage drop,  $V_{DS(on)}$  of sub-SCPMs and allowable operation voltage range of gate drivers.

In addition, a bootstrapping SCPM-M suffers from slow turn-on after long period of turning off, due to the charge stored in bootstrapping capacitors (CB1, CB2, etc.) being dissipated by static current consumption of gate drivers.

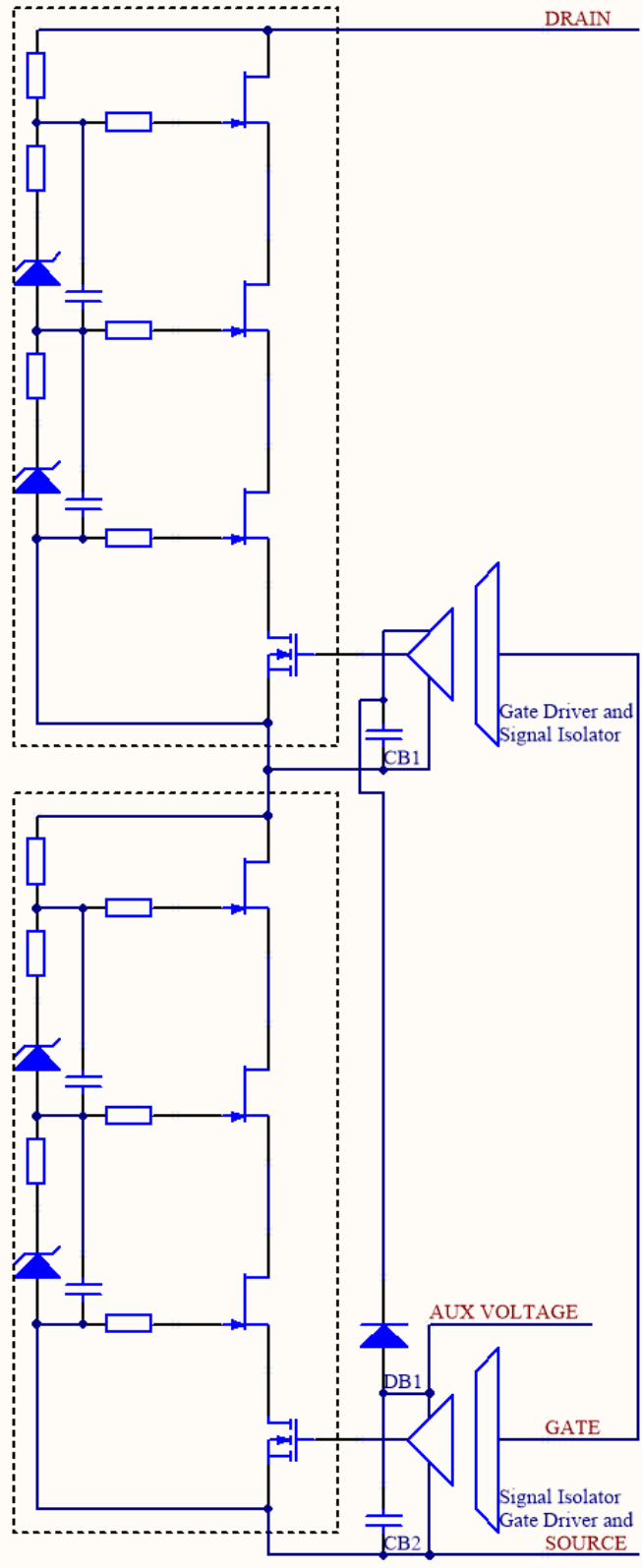


Figure 6.5: SCPM-M with bootstrapping gate drivers

#### **6.4: Triggering of series connected SCPMs through optical fiber**

To mitigate issues of bootstrapping SCPM-M discussed above, gate drivers with isolated power supply are introduced. Many forms of electrically isolation energy transfer mechanisms can be used, including magnetic, mechanical, photonic, etc. An optical fiber coupled gate driver is proposed in this work. Powering isolated gate drivers with DC/DC converter modules was also considered, but it was determined that the barrier capacitance ( $\sim 3\text{pF}$  on state-of-the-art commercial high isolation DC/DC modules, Recom RxxP2xx\_R series) contributes too much switching loss and heat dissipation at high voltage and high frequency applications. It also contributes a substantial amount of noise to the gate driver circuit, which can cause misfiring. The proposed optical fiber coupled gate driver consists of a transmitter, an optical transmission fiber and several receivers (Fig. 48). The transmitter modulates the gate signal and power to a beam of laser which is then coupled to a multi-mode optical fiber. The receiver converts laser energy back to electricity and detects modulated gate signal. The receiver then uses the recovered energy and gate signal to drive the gate of a Unit SCPM or a few Unit SCPMs in series. In a high voltage SCPM-M stack, the receiver is placed on every one or on a few Unit SCPMs to minimize receiver count (hence reducing cost) and still delivers enough gate voltage to the top Unit SCPM in a group of Unit SCPMs driven by the same receiver. Compared with existing power over fiber gate driver solutions<sup>[11]</sup>, the newly proposed solution combines signal and power and transmits both over a single optical fiber using a single wavelength. This allows for easier integration and less possibility of failure. In addition, the newly proposed solution features fail safe operation to accelerate receiver turn-off process upon transmitter or optical fiber damage.

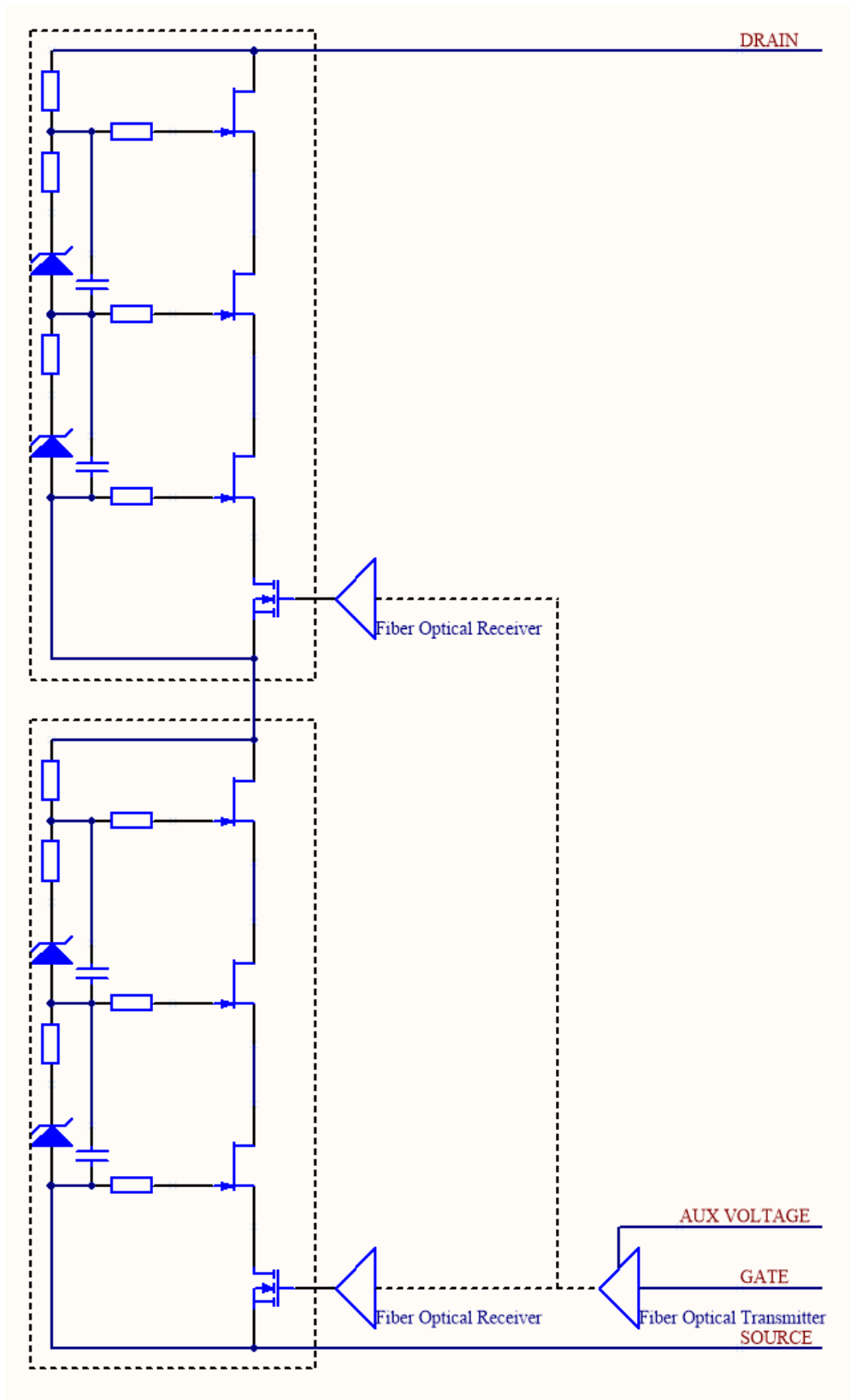


Figure 6.6: SCPM-M with optical fiber coupled gate drivers

The overall diagram of the proposed optical fiber coupled gate driver is shown below:

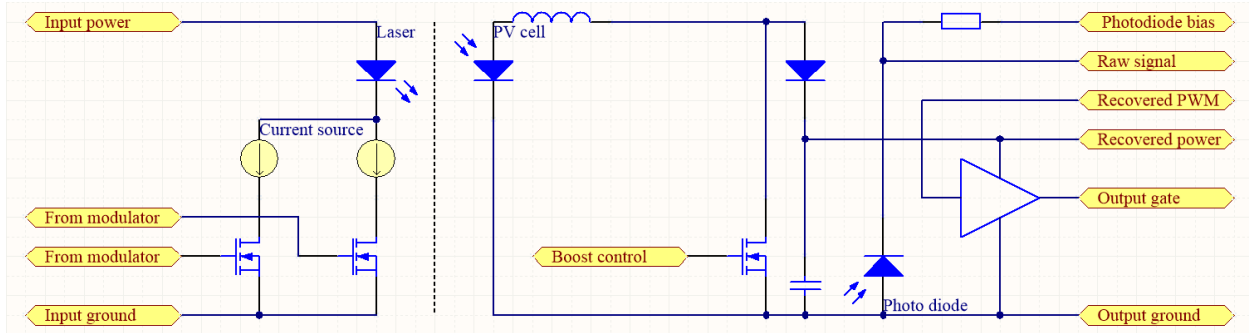


Figure 6.7: Proposed optical fiber coupled gate driver

The transmitter has a 3-level high current DAC that is capable of sourcing 0%, 50% and 100% of rated laser diode current. The transmitter normally operates at 100% output, and it briefly reduces its output to 50% to signal the receiver a turn-on signal, and it briefly reduces its output to 0% to signal the receiver a turn-off signal.

When power is lost or the optical fiber is damaged, the receiver detects a 100% to 0% transition and turns off the gate signal as a fail-safe protection measure against misfiring. Gradual declining of power caused by aging can be detected by transmitter by monitoring laser diode input power and heat dissipation, and laser diode heat dissipation can be calculated from TEC Peltier cooler/heater input electrical power and temperatures of both hot and cold sides.

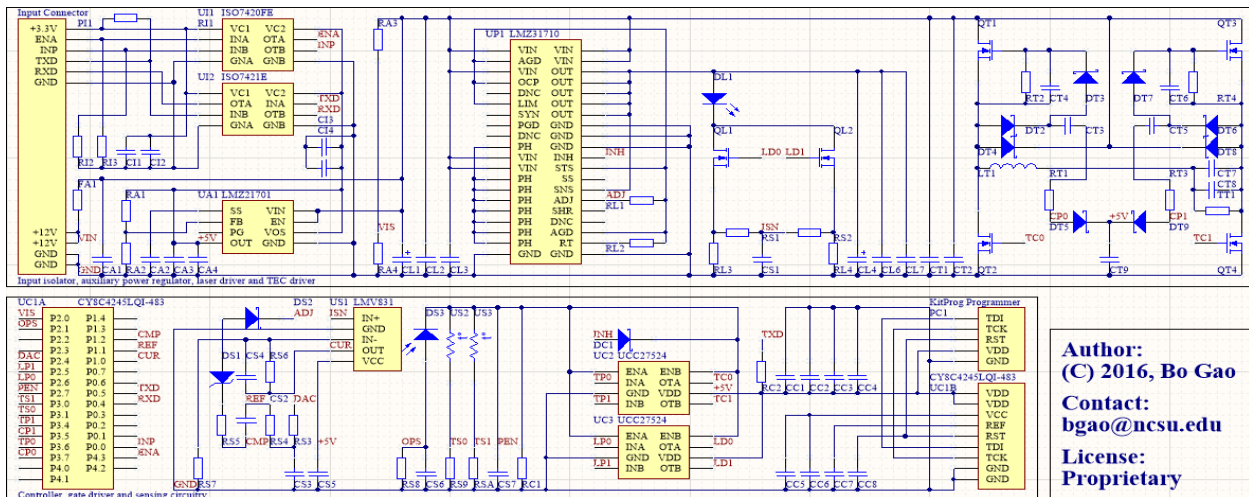


Figure 6.8: Transmitter circuit of proposed optical fiber coupled gate driver

**Author:**  
(C) 2016, Bo Gao  
**Contact:**  
bgao@ncsu.edu  
**License:**  
Proprietary

The transmitter is controlled by a Cypress PSoC 4 microcontroller with programmable logic. The microcontroller uses its programmable logic fabric to generate precise output timing for driving the high current DAC (implemented with 2 MOSFETs and 2 current limiting resistors).

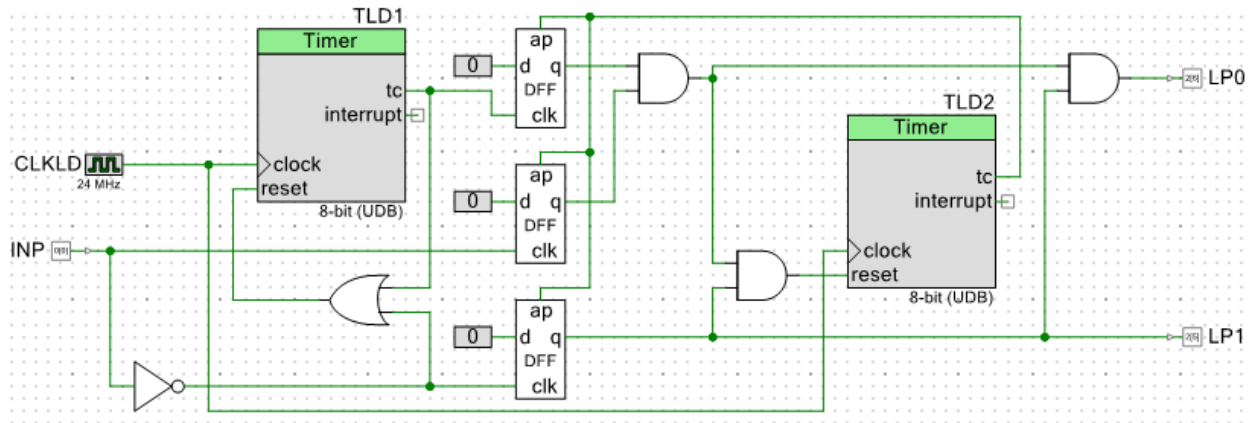


Figure 6.9: Hardware logic for generation of laser driving signals

A buck converter with cascode current feedback is used to power the high current DAC. Because the transmitter normally outputs 100% rated power and the transition frequency is considered very low, here we assume the average current is peak (100%) output current. Thus, by regulating average output current of buck converter, we can set peak current of the laser diode. The built-in OPAMP module of PSoC is used to implement this cascode current feedback buck converter. To allow wide operating temperature, a TEC cooler/heater based on full bridge buck topology is also designed to keep the laser diode at a constant temperature <sup>[12]</sup>. This is important for prolonging the service lifetime of the laser diode. This also regulates output wavelength of the laser diode, which is important for obtaining high electricity recovery efficiency for the receiver. LMT01 was chosen as remote temperature sensor for laser diode module. A protocol decoder was also developed to read its output and converts it to centigrade.

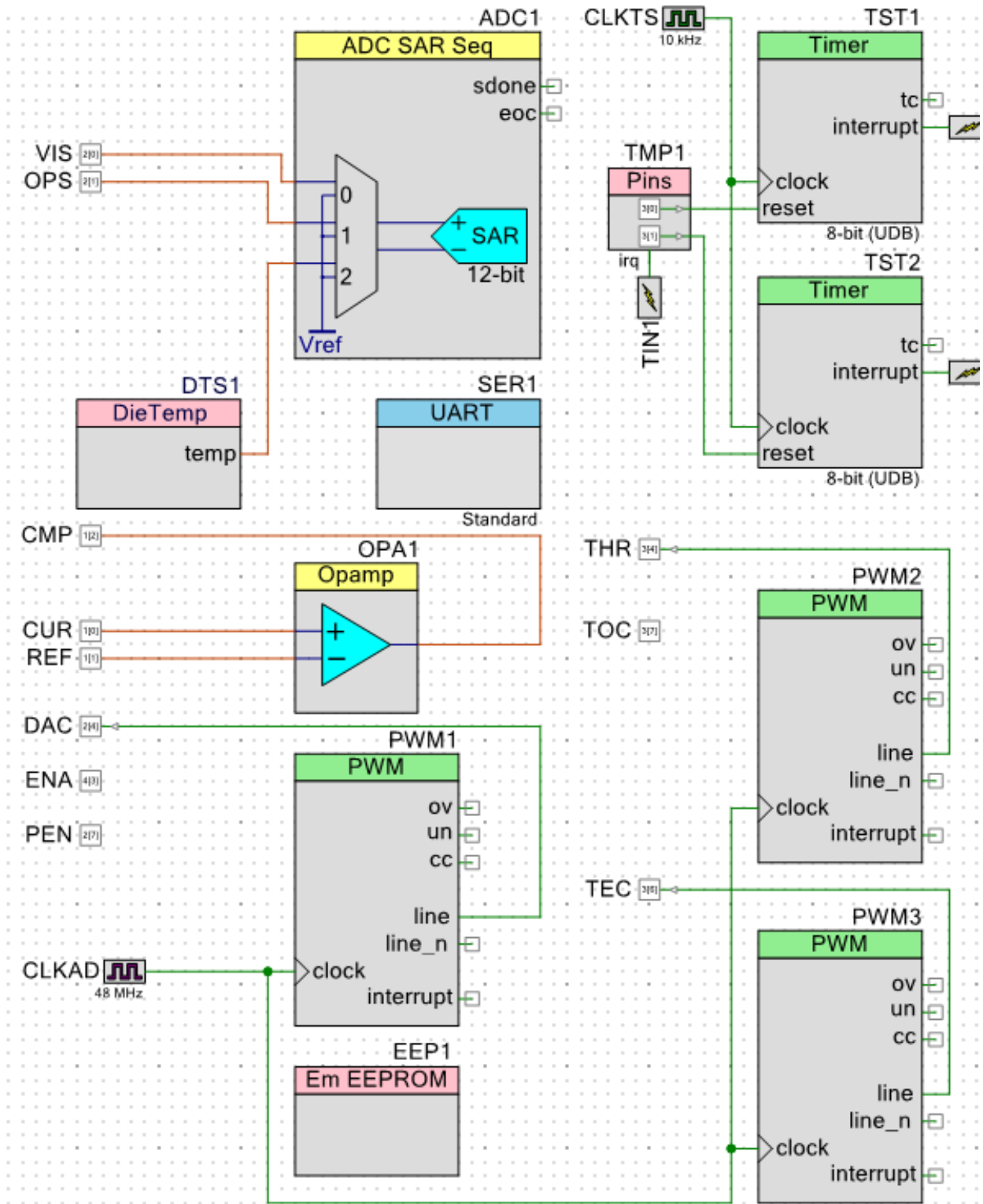


Figure 6.10: TEC controller, LMT01 decoder and system management function blocks

Two prototype gate driver transmitter modules were fabricated, one being tested, and result shows correct current modulation behavior. Due to the lack of optical measurement equipment, laser output was not able to be measured at the time this paper was written.

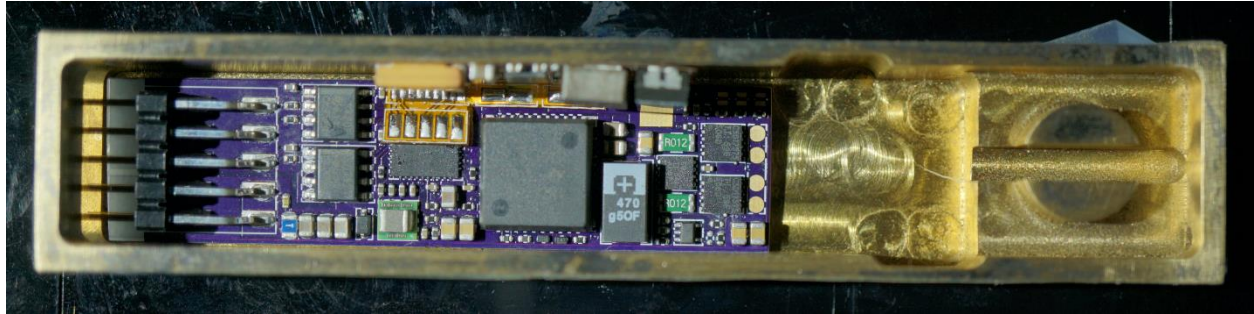


Figure 6.11: Prototype optical fiber coupled gate driver transmitter without laser diode

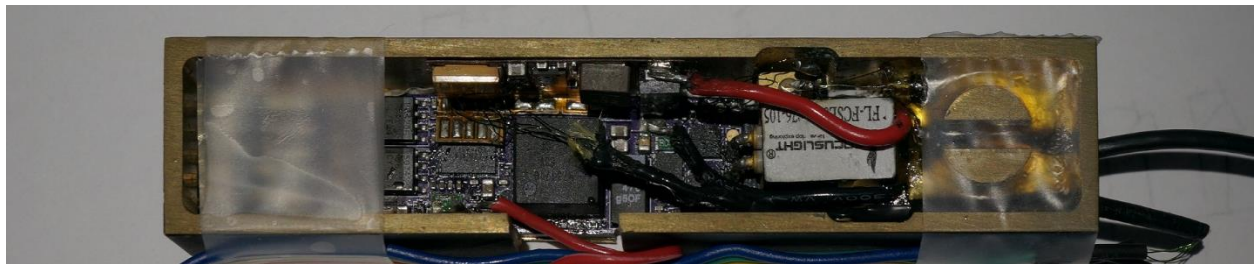


Figure 6.12: Prototype optical fiber coupled gate driver transmitter with laser diode

The modulated high current signal then is converted to laser using a fiber coupled laser diode module. The output laser is directly coupled into a 105/125 multi-mode optical fiber. The laser diode used in this design can output up to 8W of optical power. Assuming 40% optical to electrical recovery efficiency, this laser is enough to power 6 receivers driving  $4 \times 30\text{nC}$  load to 25V at 100kHz, even considering power loss in gate driver and receiver controller circuitry. Passive optical fiber splitters can be used to connect multiple receivers to one transmitter. The receiver has a PV cell stack for recovering electricity from laser, and a photodiode for recovering gate signal from laser.

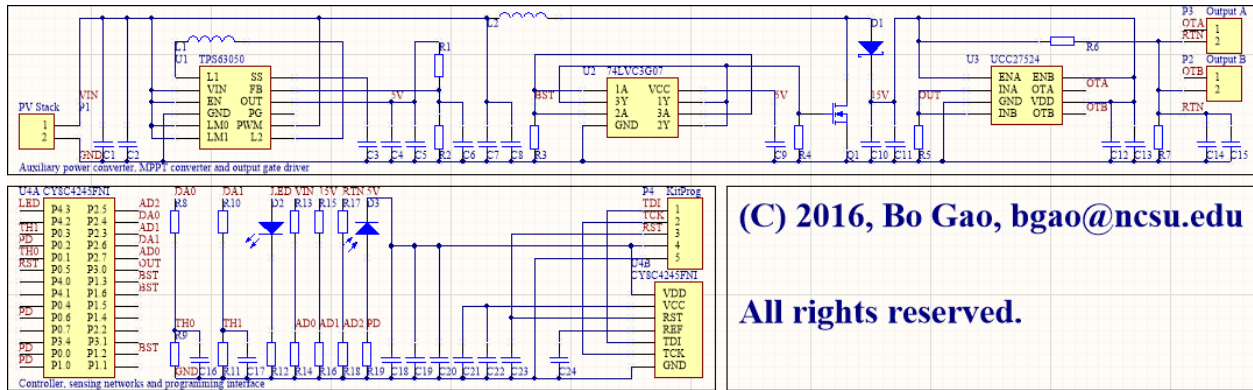


Figure 6.13: Receiver circuit of proposed optical fiber coupled gate driver

The receiver is also based on PSoC 4 microcontroller. Two built-in threshold comparators are used to detect 100% laser power, 50% laser power and 0% laser power from output of photo diode. The threshold values are dynamically adjusted based on average laser output power based on output of photo diode, so that this receiver can operate with any laser source of same wavelength, within a certain range of output power and follows the same modulation scheme. This also mitigates aging effect and laser diode output power/PV stack recovered power fluctuation with temperature and optical fiber condition.

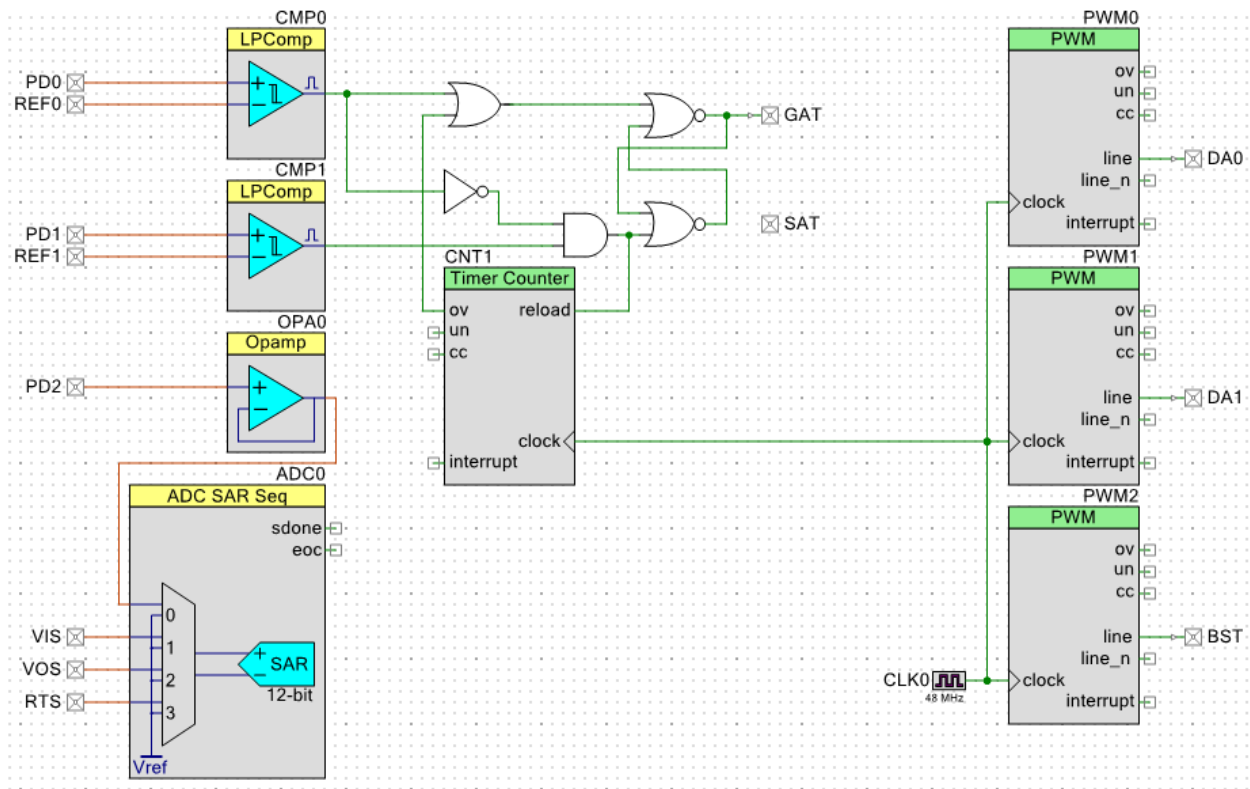


Figure 6.14: Hardware logic for recovering gate signal from modulated laser

The PSoC also serves as a digital controller of a boost converter that converts output voltage of PV stack to a stable higher voltage for driving the MOSFET gates. Here, an EPC2014C GaN eHEMT transistor is used.

A prototype module was fabricated and tested to validate the electrical behavior of the proposed gate driver.

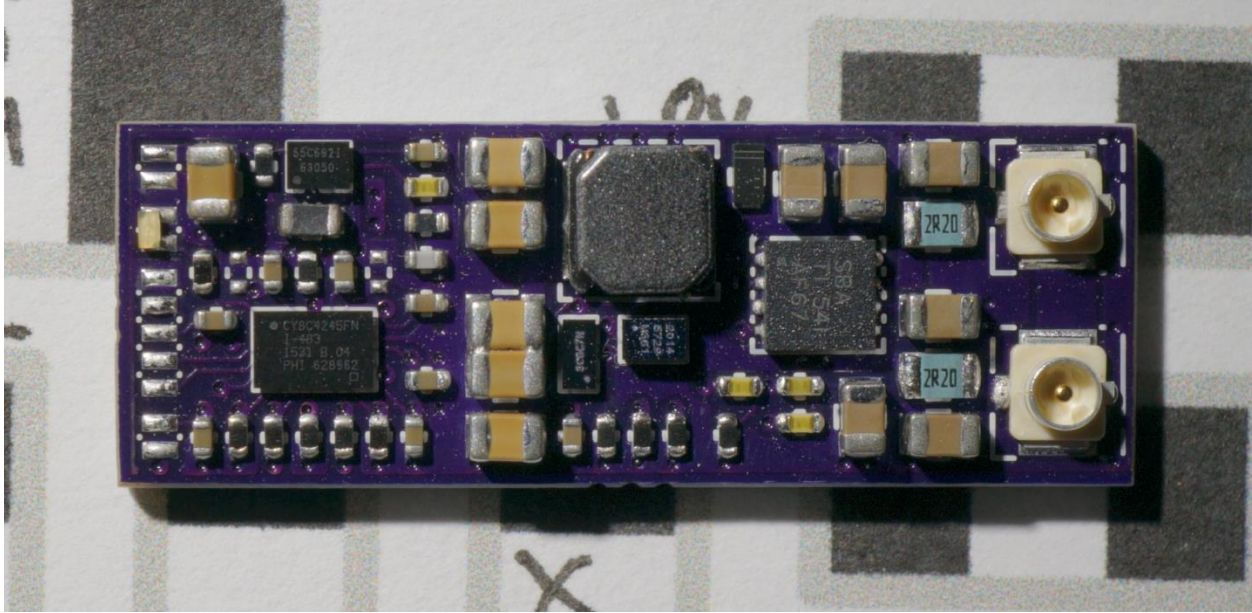


Figure 6.15: Prototype of optical fiber coupled gate driver receiver PCB

According to test result, the proposed receiver module is able to recover gate signal from an artificial stimulation that resembles photo diode output.

With SCPM-M, balancing network power loss changes from SCPM-J, therefore an SCPM-J converted to SCPM-M may not be the optimal design. To obtain optimal SCPM-M designs, a new computer program was designed based on the existing SCPM-J optimization program. The new SCPM-M design program assumes the external layer is driven by optical gate drivers, and the internal layers are either Single SCPM or SCPM-J. Because there is no external balancing network, overall  $E_B$  only contains  $E_B$  of internal sub-SCPMs. Instead, a second cost factor ( $K_{dr}$ ) was introduced per optical fiber coupled gate driver receiver to reflect their cost. Overall figure of merit is defined as:

$$FoM=1/(E_B * K_{eb} + CF * K_{cf} + N_{EXT} * K_{dr})$$

A 20kV sample design using UJN1202Z was generated using this program with  $V_{DS}=1kV$ ,  $Q_G=300nC$ ,  $K_{eb}=1.0/mJ$ ,  $K_{cf}=0.001/nFV$ ,  $K_{dr}=1.0/stage$ . Based on the calculation of the program, by using an optimum SCPM-M topology, the balancing network switching energy loss

$E_B$  can be reduced by 66.67% from the optimum SCPM-J topology, which on its own has a 68.42%  $E_B$  reduction from a conventional 1-layer SCPM topology.

Table 6.5: Calculated  $E_B$  value of SCPM topologies compared in this work

Configuration	Balancing Network Switching Energy Loss
20kV optimum SCPM-M, N1=2, N2=2, N3=5	3.0mJ
20kV optimum SCPM-J, N1=2, N2=2, N3=5	9.0mJ
20kV optimum 1-layer SCPM, N1=20, N2=1	28.5mJ

## **Chapter 7: Conclusion and Future Work**

### **7.1: Conclusion of research**

This dissertation focuses on the analysis, optimization, implementation and testing of super cascode power modules.

In chapter 2, existing SCPM structure is reviewed, including its principle of operation, existing implementations and limitations. In addition, several common applications are reviewed and discussed. The discussion of applications provides guidance to the new SCPM topologies and optimizations discussed later on.

In chapter 3, a new Single SCPM topology is proposed which aims to improve electrical performance, and manufacturability. A generalized switching power loss model is proposed, which is essential to future research on voltage scaling.

In chapter 4, the SCPM topology proposed in chapter 3 is implemented, with full electromagnetic-thermal-mechanical analysis. The implementation focuses on providing a reliable and high-performance replacement of existing E3-packaged IGBT modules. Several copies of the implementation are then fabricated and tested. A few other power modules are also designed and fabricated for the testing of the implementation. Because of the existence of two balancing mechanisms, a special double pulse test signal generator is designed and used for testing SCPMs.

In chapter 5, the necessity of high avalanche energy handling capability is discussed, and a new topology for greatly increasing avalanche energy handling capability is proposed. The new

topology is then verified by simulation. This enables the series connection of multiple SCPMs, allowing for voltage scaling with lower switching power loss.

In chapter 6, several methods of voltage scaling are discussed. Two new topologies are proposed for series connecting SCPMs, namely SCPM-J and SCPM-M. The SCPM-J uses a modified Single SCPM (called Unit SCPM) as higher voltage JFETs, and then cascode the Unit SCPMs to form higher voltage SCPMs. The new higher voltage SCPM can be further modified to a Unit SCPM and be used to form even higher voltage SCPMs. Based on loss model proposed in chapter 3, analysis is done on SCPM-J topology, and this provides guidance for numerical optimization of the design process of any SCPM-J. Computer programs are developed to automatically find the optimum design of an SCPM-J for a given set of constraints. Simulation confirms that SCPM-J has lower balancing network power dissipation as well as lower overall module power dissipation. Other gate driving mechanisms for series connected SCPMs are also discussed and was concluded that with current power semiconductor device technology, the most feasible and highest performance method is to drive Single SCPMs or SCPM-J modules (collectively called sub-SCPMs) with a galvanic isolated gate driver, this new SCPM scaling is called SCPM-M. Among several isolation mechanisms, fiber optical isolation is used to carry both power and signal to gate drivers. A fiber optical gate driver is designed and tested. Finally, a computer program is developed to automatically find the optimum design of an SCPM-M for a given set of constraints. Calculation shows great reduction of energy network switching energy loss compared with the best SCPM-J design of same switching capacity.

This work contributes to the body of knowledge with the following:

1. Systematic analysis of principle of Single SCPMs, its loss model and its limitations
2. New topologies of Single SCPMs to improve performance, reliability and manufacturability, enabling SCPMs to be a practical high-performance replacement of Si-IGBTs and SiC-MOSFETs
3. Techniques for scaling SCPMs to higher voltage while maintaining low switching power loss and high reliability.

### 7.2: Application on device testing of future semiconductor devices

As this work is being concluded, two new applications that are enabled by the new SCPM are being proposed. The new SCPM enables high voltage, high speed, high performance and high current switching at a very low cost compared with SiC-MOSFET. This allows the new SCPMs to be used in energy recirculating converters (Fig. 59) as control switches for testing faster semiconductor switches.

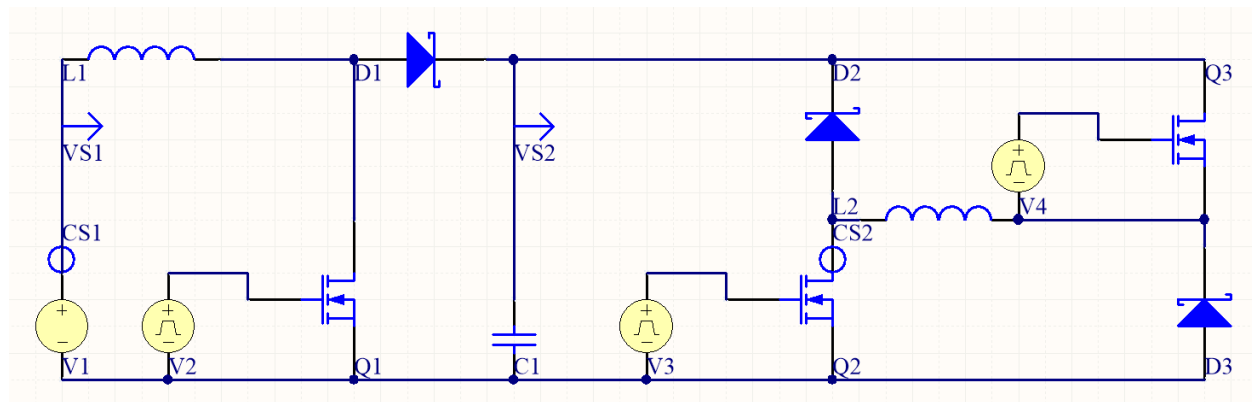


Figure 7.1: H-bridge energy recirculation converter with boost converter pre-regulator

In Fig. 59, Q3 is the control switch, and Q2 is the device under test. Ripple current in main inductor L2 is highly dependent on the lowest switching frequency of both switches in asynchronous operation mode, hence with faster control switch, for the same ripple current requirement, inductance of L2 can be lower. This reduces parasitic winding capacitance, hence increases maximum frequency Q2 can switch at. Therefore, a fast control switch can be used to test faster devices under test. With an SCPM as control switch, this energy recirculation converter allows for the characterization of the latest power semiconductors, thus helps the development of next generation power semiconductor technology.

### **7.3: Application on solid state circuit breakers and power controllers**

Another application being proposed is to use the new SCPM as breaking element in solid state circuit breakers (SSCBs) and solid-state power controllers (SSPCs). SCPM topology uses multiple JFETs to block voltage, therefore heat is spread across a larger area compared with using a monolithic high voltage power switch. For high energy absorption applications such as SSCBs and SSPCs, inductive energy stored in the transmission line during a fault current surge can be dissipated by multiple JFETs spread over a larger area. Furthermore, a fail-short JFET in an SCPM stack will not compromise the entire module (except for the control switch in each Unit SCPM), unlike a fail-short MOSFET in a parallel power module, as long as the total blocking voltage of the rest JFETs is still higher than the voltage being interrupted. This allows for using massive amount of JFETs to achieve very high blocking voltage and passing current while still maintaining high reliability that is required by such applications.

## REFERENCES

1. Power electronic converters for HVDC renewable energy applications, Pat Wheeler, et al, IEEE CHILECON 2015, DOI: 10.1109/Chilecon.2015.7400412.
2. Comparative Evaluation of 6kV Si and SiC Power Devices for Medium Voltage Power Electronics Applications, Xiaoqing Song, et al, IEEE WiPDA 2015, DOI: 10.1109/WiPDA.2015.7369289.
3. Design Comparison of High Power Medium-Voltage Converters based on 6.5kV Si-IGBT/Si-PiN diode, 6.5kV Si-IGBT/SiC-JBC diode, and 10kV SiC MOSFET/SiC-JBC diode, Hesam Mirzaee, et al, IEEE ECCE 2011, DOI: 10.1109/ECCE.2011.6064090.
4. Stacked high voltage switch based on SiC VJFETs, P. Friedrichs, et al, IEEE ISPSD 2003, DOI: 10.1109/ISPSD.2003.1225249.
5. Balancing Circuit for a 5-kv/50-ns Pulsed-Power Switch Based on SiC-JFET Super Cascode, Juergen Biela, et al, IEEE TPS 2011, DOI: 10.1109/TPS.2011.2169090.
6. Medium voltage power switch based on SiC JFETs, Xueqing Li, et al, IEEE APEC 2015, DOI: 10.1109/APEC.2016.7468286.
7. 15kV/40A FREEDM Super-Cascode: A Cost Effective SiC High Voltage and High Frequency Power Switch, Xiaoqing Song, et al, IEEE ECCE 2016, DOI: 10.1109/ECCE.2016.7854643
8. Investigation of Potential Benefits of MOSFETs Hard-Switching and Soft-Switching Converters at Cryogenic Temperature, Hui Li, et al, IEEE TAS 2005, DOI: 10.1109/TASC.2005.849672.
9. Two-Phase Spray Cooling of Hybrid Vehicle Electronics, Issam Mudawar, et al, IEEE TCPT 2009, DOI: 10.1109/TCAPT.2008.2006907.

10. A comprehensive analysis of breakdown mechanisms in 4H-SiC MOSFET and JFET, A. Mihaila, et al, IEEE ISC 2000, DOI: 10.1109/SMICND.2000.890214.
11. A Gate Drive With Power Over Fiber-Based Isolated Power Supply and Comprehensive Protection Functions for 15-kV SiC MOSFET, Xuan Zhang, et al, IEEE JESTPE 2016, DOI: 10.1109/JESTPE.2016.2586107.
12. Reliability of laser diode modules in temperature-uncontrolled environment, M. Ciappa, et al, IEEE RPS 1994, DOI: 10.1109/RELPHY.1994.307798.

## APPENDICES

## Appendix A Fabrication of E3-Packaged Single SCPMs

Totally 4 E3-packaged SCPMs were fabricated, names as S0-6500-50, S1-6500-100, S2-6500-100 and S3-6500-100. S0 has only half devices populated (6 JFETs in series, no parallel connections). S1 and following SCPMs all have all devices populated (6 JFETs in series, 2 strings in parallel). In addition, S0 does not have proper dynamic balancing capacitors populated, hence was only used for static balancing testing.

All E3-packaged SCPMs made have the same baseplate, DBC and wire bonding design. After simulation, optimal design was drawn in CAD software, and sent to baseplate and DBC suppliers.

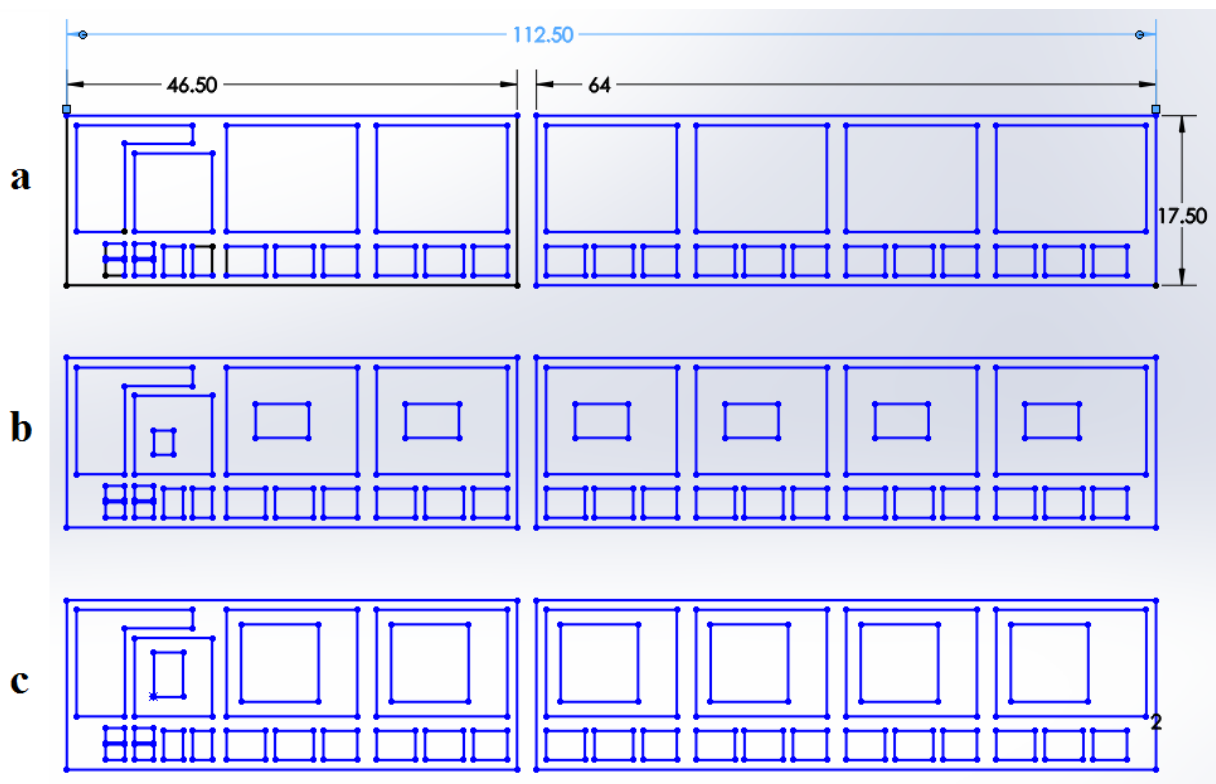


Figure A1: DBC design of E3-packaged SCPMs

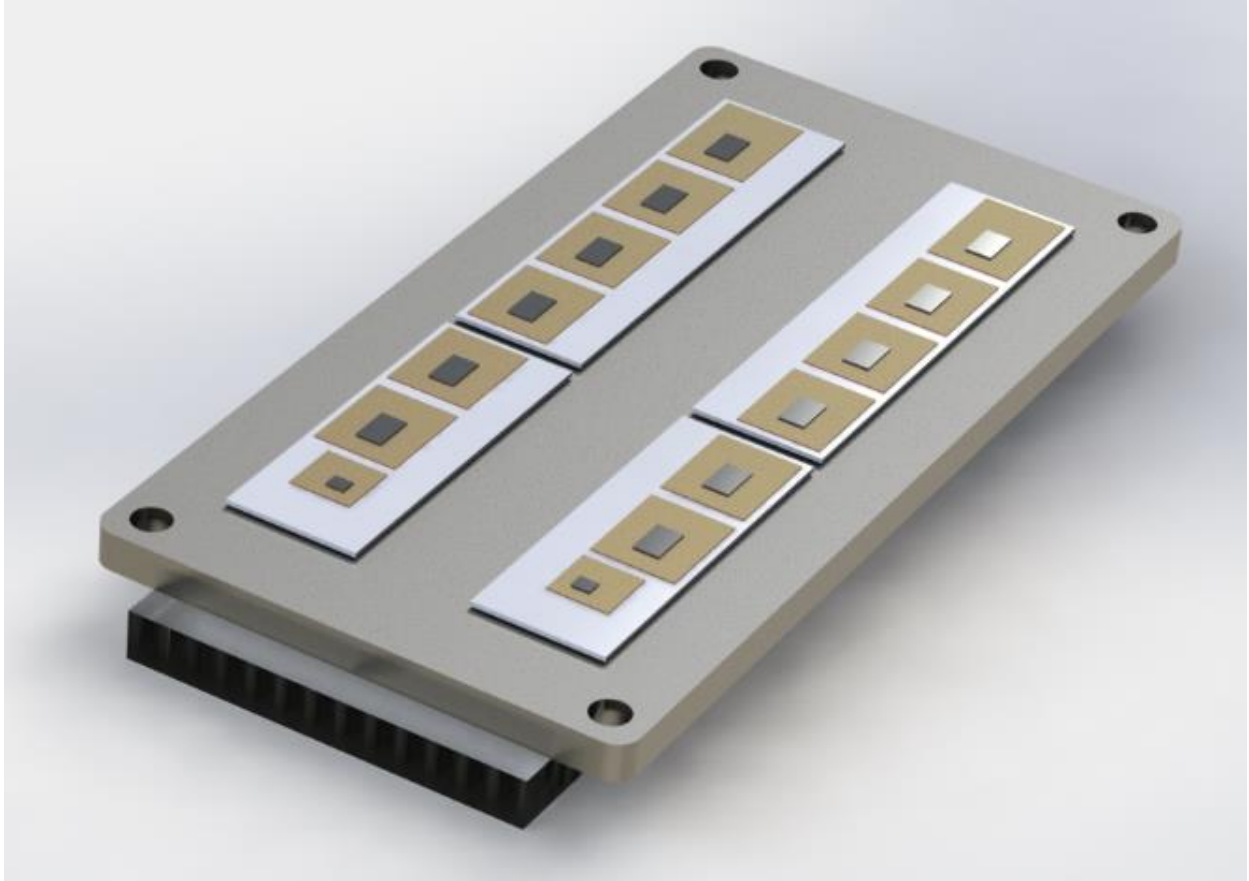


Figure A2: DBC layout on baseplate, center space is for balancing network PCB

All E3-packaged SCPMs started as a baseplate, a few pieces of DBC and some plastic resin precursor. First, we 3D printed the housing part using an SLA/DLP 3D printer with digital ABS material. The final module uses PEEK material, but we used digital ABS for prototyping due to the fast turnaround time and technical issues with our PEEK printer.



Figure A3: Digital ABS housing in a jig

Then, we soldered down JFETs on DBC pieces, and soldered down 4 DBC pieces per module. 3 different types of solder alloys were used to allow hierarchical reflowing. Sn5Pb95 was used for attaching JFETs to DBCs, Sn95Sb5 solder was used for attaching DBCs and balancing network spacer to baseplate, and Sn63Pb37 solder was used to attach busbars to DBCs as well as to solder components on balancing network. Balancing network and its spacer were attached with high temperature epoxy.

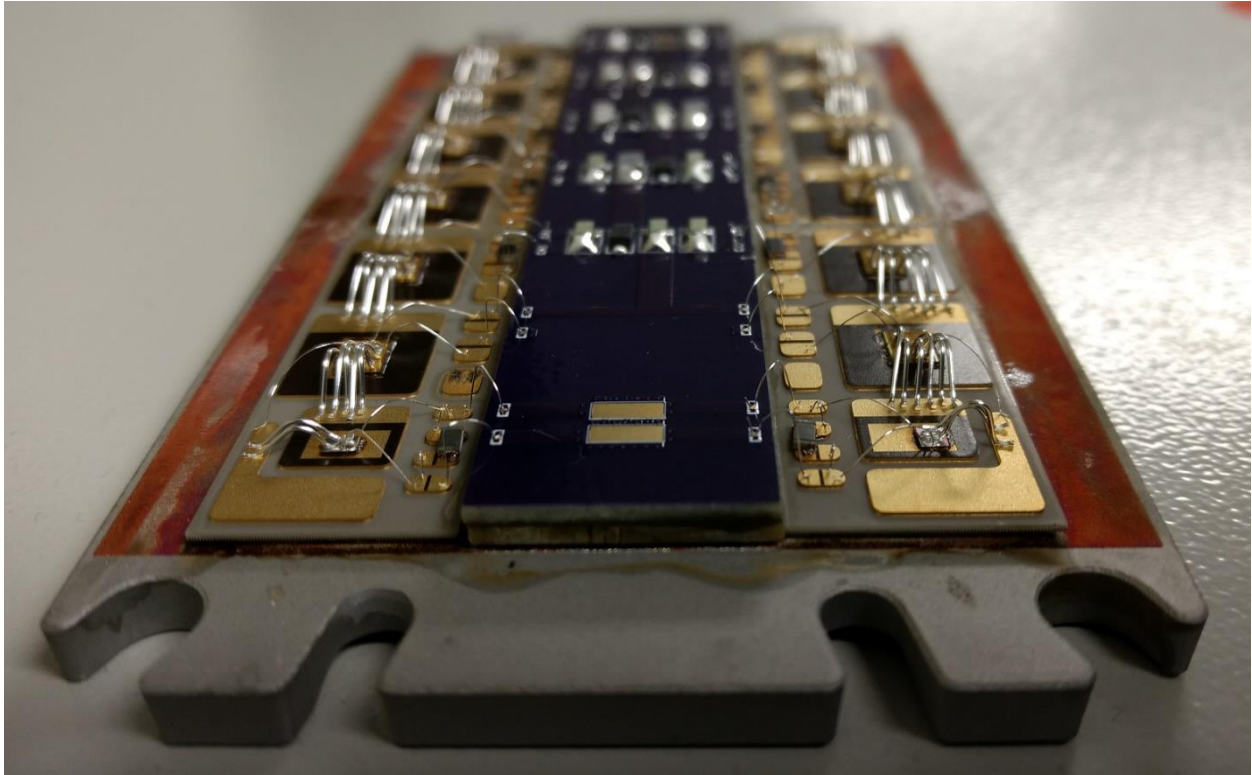


Figure A4: Side view of E3-packaged SCPM before busbar soldering, housing attachment and encapsulation

Form the above picture we can see that the standoff height of DBCs is not uniform, that is because the particular unit was an early prototype (S1) and the soldering process had not fully stabilized yet. Later units have better soldering quality.

S1 was the first functional SCPM being made.



Figure A5: SCPM S1 photo before encapsulation

S2 was scrapped during fabrication due to excessive leakage current and MOSFET breakdown. It was later determined that the low yield was caused by ESD and improper wire bonding parameter setup. Too little ultrasonic energy and force will not make reliable bonds on DBC metallization, too much ultrasonic energy and force will crush JFET and MOSFET dice.

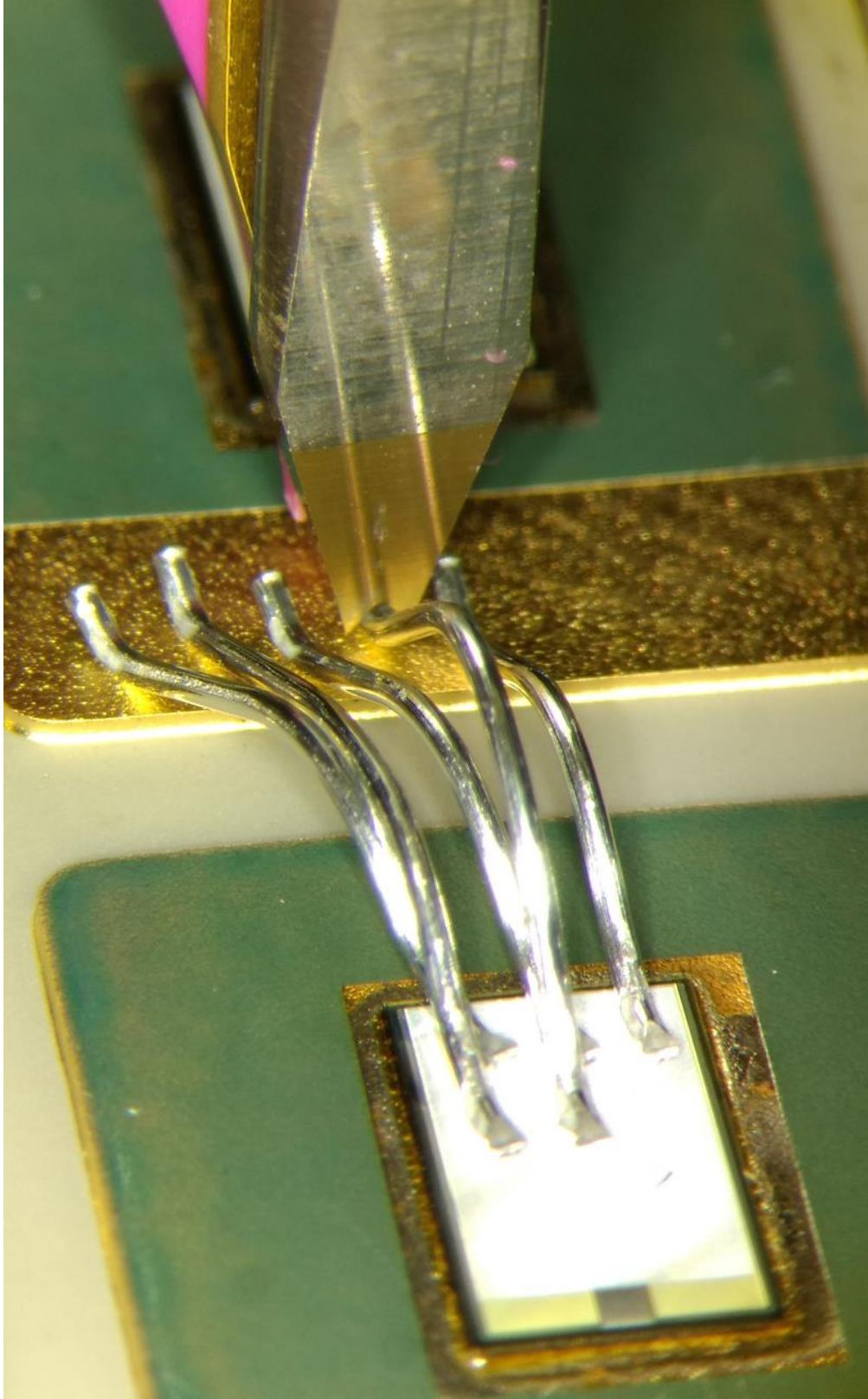


Figure A6: Bonding head on DBC metallization

After several trials and errors, as well as new ESD protocol, including the use of ionizing fans numerous tweaking to wire bonding process, S3 was able to be fabricated successfully. Also, leakage current was measured on all JFETs used in S3, to make sure there were no damaged chips being populated.

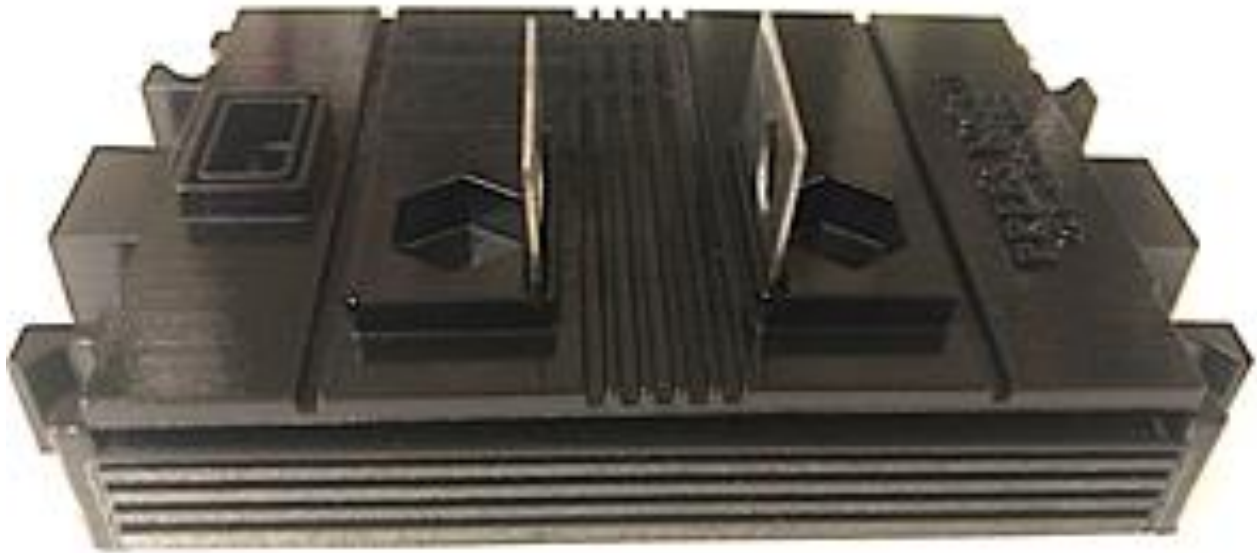


Figure A7: Finished SCPM S3 with black digital ABS housing

Despite the baseplate and all components on it turned out to be functioning, we encountered encapsulation problems with S3. The encapsulants failed to cure under specified condition. A diode module had encountered the same issue using the same partially cured digital ABS material. It was later found out that the partially cured digital ABS plastic had interfered the curing process of the silicone gel encapsulant. A test was carried out using fully cured digital ABS (exposed under UV light for 24 hours), and the result shows that the silicone gel was able to be cured properly.

Two diode modules, each containing 20 Wolfspeed 1.7kV 50A JBC diodes connected in 5-series, 4-parallel configuration were fabricated for use in DPT test setup and for future full power test platform test setup. The fabrication process is similar to the fabrication process of E3-packaged SCPMs. The diode modules have the same baseplate, DBC, terminal and housing design.

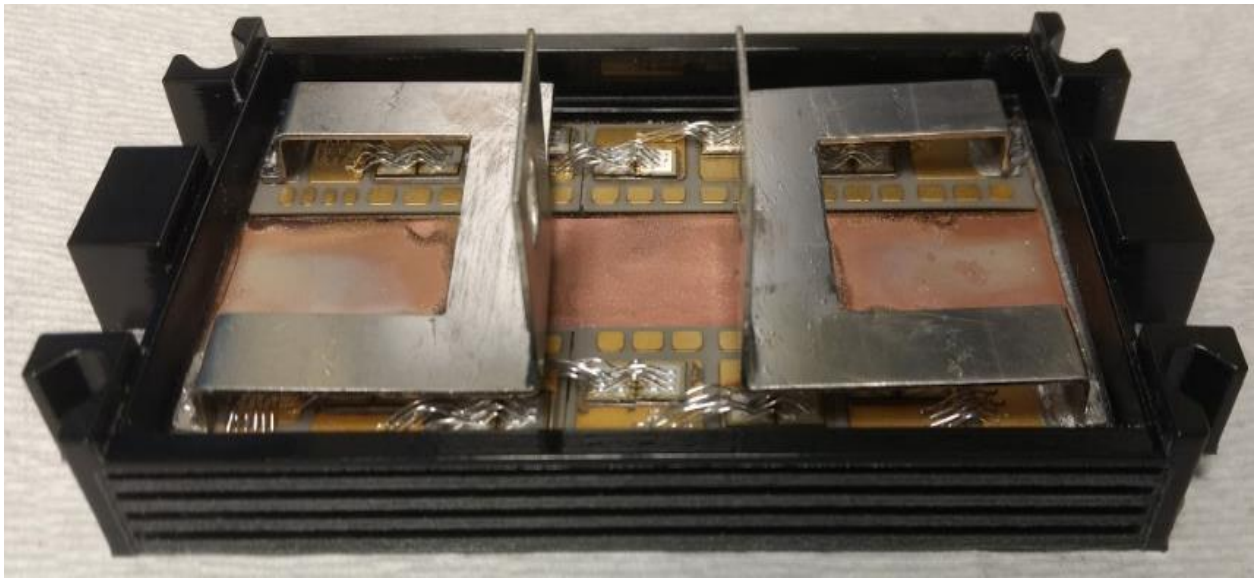


Figure A8: 8.5kV/200A Schottky diode module before encapsulation

## **Appendix B MCU Design for DPT Tester Controller**

Due to the nature of SCPMs, they have 2 balancing modes, dynamic balancing and static balancing. To test the modules under the condition that resembles real world application condition, the DPT controller generates a number of pulses before the main DPT pulses in order to pre-charge and pre-equalize the static balancing network. To make the DPT signal generator a truly universal DPT test signal generator, the number and duration of pre-pulses as well as the main pulses are all user programmable.

The DPT controller was implemented with a PSoC 5LP MCU. Programmable hardware logic was used to handle trigger button debouncing, so that a single user trigger input will not be registered multiple times. To protect operator, the DPT sequence can also be triggered through USB communication, and therefore can be isolated with USB isolators and/or wireless network.

The system diagram consists of two major parts: USB communication and trigger button interface. The USB communication part exposes a USB serial port which can be accessed by host computer through PuTTY or any serial port console. The trigger button interface debounces trigger button input and generates an interrupt to allow MCU program to handle this event.

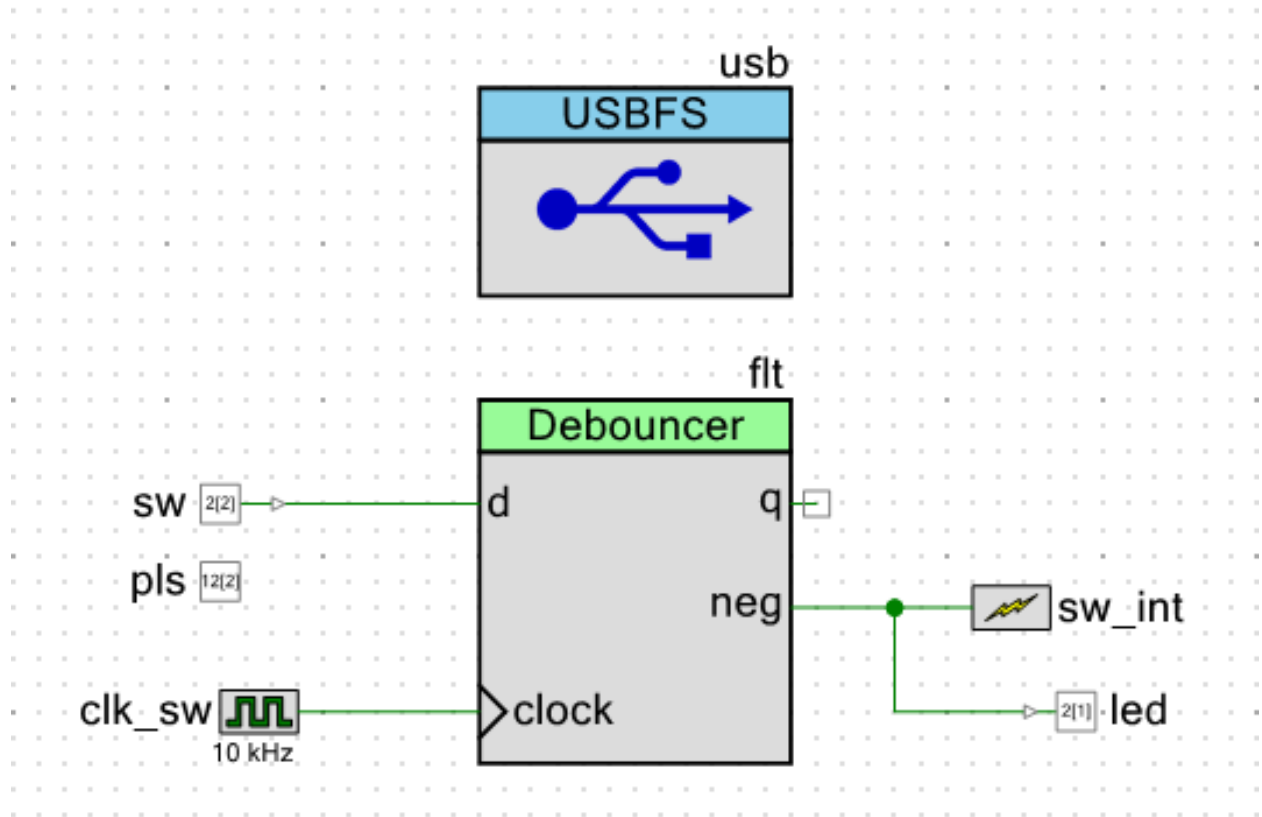


Fig. B1: Block diagram of DPT signal generator

The firmware program contains 4 major parts:

1. Initialization code
2. USB communication handling code
3. Trigger button handling code
4. Test signal generator

The initialization code enables interrupts and starts USB communication interface, then it enters a dead loop waiting for events. Upon USB event, the USB code tries to reconstruct entire command line from data packets, then delimits receiving buffer by carrier return or new line.

Once a carrier return or new line is detected, the USB code sends the content of the buffer to a parser. The parser then tries to extract test parameters from command line. If the parser is able to

extract meaningful parameters, it will copy the received parameters to test parameter memory space, and set test pending flag. If the parser is not able to extract meaningful parameters, the parser copies an error message to USB return buffer.

Once the parser function returns, the USB code writes return buffer back to host computer, which then gets displayed on operator's screen.

The main loop not only handles USB events, it also examines test pending flag. If test pending flag is set, it generates a test signal sequence based on parameter set by operator.

The trigger button can also trigger a test signal sequence. When a trigger event is detected, an interrupt is generated. The interrupt handler examines test parameters, and if the test parameters are yet to be set by operator through USB interface, the trigger button interrupt handler will ignore this event. If valid test parameters are set by operator through USB interface, the trigger button interrupt handler will set test pending flag and wait for the main loop to carry out the test sequence generation process.

Source code of main program is attached here:

```
#include <project.h>
#include <stdio.h>

//Limits maximum pulse time in us
#define T_MAX 100000

int n1, n2, n3, n4, n5;
int testing=-1;
int tested=0;

void parse(const char *buf, char *buf_out)
{
    int nn1, nn2, nn3, nn4, nn5;
    if(testing==1)
    {
        strcat(buf_out, "Testing pending, try later.\n\r");
        return;
    }
}
```

```

//sscanf() returns number of anticipated elements matched from input
if(sscanf(buf, "test %d %d %d %d %d", &nn1, &nn2, &nn3, &nn4, &nn5)!=5)
{
    strcat(buf_out, "Invalid command.\n\rSyntax: test [t_pre] [n_pre] [t_ps1] [t_hld]
[t_ps2]\r\n");
    return;
}
if(nn1<0 || nn2<0 || nn3<=0 || nn4<=0 || nn5<=0)
{
    strcat(buf_out, "Pulse width shorter than minimum.\n\r");
    return;
}
if(nn1*nn2*2+nn3+nn4+nn5>T_MAX)
{
    strcat(buf_out, "Time longer than maximum.\n\r");
    return;
}
n1=nn1;
n2=nn2;
n3=nn3;
n4=nn4;
n5=nn5;
testing=1;
}

```

```

int main()
{
    char buf[128], buf_out[128], buf_pkt[64], buf_pkt_out[1024]={0};
    char c;
    char s[2]={0};
    unsigned int p1=0, p2, n, i;
    int ii;
    pls_Write(0);
    //Enable global interrupts
    CyGlobalIntEnable;
    //Start trigger interrupt
    sw_int_Start();
    //Initialize USB driver
    usb_Start(0, usb_5V_OPERATION);
    //Wait until USB plugged
    while(!usb_GetConfiguration());
    //Initialize USB serial port driver
    usb_CDC_Init();
    //Main dead loop
    while(1)
    {

```

```

//Handles testing pulse generation
if(testing==1)
{
    if(n1*n2>0)
        for(ii=0;ii<n2;ii++)
            {
                pls_Write(1); //On and off times are same
                CyDelayUs(n1);
                pls_Write(2);
                CyDelayUs(n1);
            }
    pls_Write(1); //First main DPT pulse
    CyDelayUs(n3);
    pls_Write(0); //Turn off
    CyDelayUs(n4);
    pls_Write(1); //Second main DPT pulse
    CyDelayUs(n5);
    pls_Write(0); //Turn off
    testing=0;
    tested=1;
}
//Handles multiple interface request
if(usb_IsConfigurationChanged())
{
    while(!usb_GetConfiguration());
    usb_CDC_Init();
    continue;
}
//Reads endpoint buffer
if(usb_DataIsReady())
    n=usb_GetAll((uint8*)buf_pkt);
else
    n=0;
//Puts buffer content to command line
for(i=0;i<n;i++)
{
    c=buf_pkt[i];
    //New line
    if(c=='\r' || c=='\n')
    {
        buf[p1]='\0';
        p1=0;
        strcpy(buf_out, "\r\n");
        parse(buf, buf_out);
        strcat(buf_pkt_out, buf_out);
    }
}

```

```

//Back space
else if(c==0x7f)
{
    if(p1>0) buf[--p1]='\0';
    //Remove last character, equal to backspace
    strcat(buf_pkt_out, "\x7f");
}
//Other characters
else if(isprint((int)c))
{
    buf[p1++]=c;
    if(p1>sizeof(buf)-2) p1=sizeof(buf)-2;
    s[0]=c;
    strcat(buf_pkt_out, s);
}
}
if(tested==1)
{
    //Delete everything in current line, requires TTY support.
    strcat(buf_pkt_out, "\r\033[KTest signal generated.\r\n");
    buf[p1]='\0';
    p1=0;
    tested=0;
}
//Writes back result to USB
n=strlen(buf_pkt_out);
for(p2=0;p2<n;p2+=64)
{
    while(!usb_CDCIsReady());
    usb_PutData((uint8*)(buf_pkt_out+p2), n-p2);
}
//If last packet is 64 bytes, send another empty one
if(p2==n)
{
    while(!usb_CDCIsReady());
    usb_PutData(NULL, 0);
}
buf_pkt_out[0]='\0';
}
}

```

Abridged source code of trigger button interrupt handler is attached here:

```

extern int testing;
CY_ISR(sw_int_Interrupt)
{

```

```
#ifdef sw_int_INTERRUPT_INTERRUPT_CALLBACK
    sw_int_Interrupt_InterruptCallback();
#endif /* sw_int_INTERRUPT_INTERRUPT_CALLBACK */

/* Place your Interrupt code here. */
/* `#START sw_int_Interrupt` */
//testing would be -1 before test parameters are received through USB
//testing would be 1 if an active testing is in progress
if(testing==0)
    testing=1;
sw_int_ClearPending();
/* `#END` */
}
```

## Appendix C Program for Optimizing 2-Layer SCPM-J Designs

```
#include <stdio.h>
#include <stdlib.h>
#include <float.h>

//Enable verbose output
#define TESTMODE

//One type of configuration
typedef struct
{
    //How many layers
    int n_layer;
    //How many cells per layer (cell=JFET for Unit JFET)
    //Layer 0 being Unit SCPM, last layer being outer layer
    int *n_stage;
}config;

//List of configurations
int n_config=0;
config *configs;

//Parameters
float ntotal; //Total JFET number of entire SCPM
float vds; //Vds of individual JFET
float qds; //Qds in nC
float keb; //Keb factor
float kcf; //Kcf factor

//Calculate Eb of one SCPM topology
float calculate(int index)
{
    int i, n;
    //Per-cell voltage
    float v=vds;
    //Overall Eb
    float eb=0.0;
    //Overall cost factor
    float cf=0.0;
    //Memory boundary check
    if(index<0 || index>=n_config)
        return -1.0;
    //Allocate memory for Eb of each layer
    float *eb_layer; //in mJ
    eb_layer=malloc(sizeof(float)*configs[index].n_layer);
```

```

//Memory allocation failed
if(!eb_layer)
    return -1.0;
//Allocate memory for CF of each layer
float *cf_layer; //in nFV
cf_layer=malloc(sizeof(float)*configs[index].n_layer);
//Memory allocation failed
if(!eb_layer)
{
    free(eb_layer);
    return -1.0;
}
//Iterate through all layers, from inner to outer
for(i=0;i<configs[index].n_layer;i++)
{
    //Calculate Eb and cost factor per layer
    n=configs[index].n_stage[i];
    eb_layer[i]=0.25*qds*v*(n*n-n);
    cf_layer[i]=0.5*qds*(n*n-n);
    //Increase per-cell voltage variable
    v=v*n;
}
//n is used as cell count accumulator
n=1;
//Iterate through all layers, from inner to outer
for(i=0;i<configs[index].n_layer;i++)
{
    //Accumulate all eb_layer values
    n=n*configs[index].n_stage[i];
    eb=eb+eb_layer[i]*ntotal/n;
    cf=cf+cf_layer[i]*ntotal/n;
}
free(eb_layer);
free(cf_layer);
return eb*keb/1000000.0+cf*kcf;
}

//Display one configuration
void display(int n)
{
    int i;
    float tcf;
    //Print SCPM configuration index
    printf("SCPM %d:\n", n);
    for(i=0;i<configs[n].n_layer;i++)
        printf("N%d=%d\t", i+1, configs[n].n_stage[i]);
}

```

```

//Calculate and print total cost factor
tcf=calculate(n);
//Error handling
if(tcf==-1.0)
{
    printf("Unexpected memory error.\n");
    exit(-1);
}
printf("TCF=%.2f\n", tcf);
}

//List all layers
void list(void)
{
    int i;
    //Enumerate all configurations
    for(i=0;i<n_config;i++)
        display(i);
}

//Factorize an integer
//Result stored in an array, starting with number of factors found
int *fact(int n)
{
    int *factors;
    //Factor counter
    int n_factors=0;
    int i, j;
    //Calculate all factors
    for(i=1;i<=n;i++)
        if(n%i==0) n_factors++;
    //First elements is n_factors
    factors=malloc(sizeof(int)*(n_factors+1));
    if(!factors) return NULL;
    factors[0]=n_factors;
    //Calculate all factors
    j=1;
    for(i=1;i<=n;i++)
        if(n%i==0) factors[j++]=i;
    return factors;
}

//Free configs structure
void freeall(void)
{
    int i;

```

```

for(i=0;i<n_config;i++)
{
    //Do not free a freed pointer
    if(configs[i].n_stage)
        free(configs[i].n_stage);
}
free(configs);
}

//Entry point of program
int main(int argc, char **argv)
{
    //Argument check
    if(argc!=6)
    {
        printf("Syntax: %s [n_stage] [vds (V)] [qds (nC)] [keb] [kcf]\n", argv[0]);
        return -1;
    }
    //Read arguments
    ntotal=atoi(argv[1]); //total stage count
    vds=atof(argv[2]); //voltage per JFET
    qds=atof(argv[3]); //charge per JFET (Qds-Qd)
    keb=atof(argv[4]); //Keb
    kcf=atof(argv[5]); //Kcf
    //Generate 2 layer designs
    int n1, n2;
    int i;
    //Find all factors of total stage count
    int *factors;
    factors=fact(ntotal);
    if(!factors)
    {
        printf("Unexpected memory error.\n");
        return -1;
    }
    config *temp;
    //Generate designs
    //Enumerate through all factors
    for(i=0;i<factors[0];i++)
    {
        n1=factors[i+1];
        n2=ntotal/n1;
        n_config++;
        //Buffer pointer to prevent memory leakage
        temp=configs;
        if(!configs)

```

```

        configs=malloc(sizeof(config)*n_config);
else
    configs=realloc(configs, sizeof(config)*n_config);
if(!configs)
{
    printf("Unexpected memory error.\n");
    free(factors);
    free(temp);
    return -1;
}
//Only do a 2-layer design
configs[n_config-1].n_layer=2;
configs[n_config-1].n_stage=malloc(sizeof(int)*2);
if(!configs[n_config-1].n_stage)
{
    printf("Unexpected memory error.\n");
    free(factors);
    freeall();
    return -1;
}
//Assign stages per layer
configs[n_config-1].n_stage[0]=n1;
configs[n_config-1].n_stage[1]=n2;
}
free(factors);
//Print all designs
#ifdef TESTMODE
list();
#endif
//Find best solution
float tcfmin=FLT_MAX, tcf;
for(i=0;i<n_config;i++)
{
    tcf=calculate(i);
    if(tcf<tcfmin)
        tcfmin=tcf;
}
//Display minimum TCF design
printf("\nThe optimal designs are:\n");
for(i=0;i<n_config;i++)
{
    if(calculate(i)==tcfmin)
        display(i);
}
freeall();
return 0;

```

```
}
```

The program was tested with gcc 7.2.0 on MSYS2 platform.

## Appendix D Program for Optimizing Any-Layer SCPM-J Designs

```
#include <stdio.h>
#include <stdlib.h>
#include <float.h>

//Enable verbose output
#define TESTMODE

//One type of configuration
typedef struct
{
    //How many layers
    int n_layer;
    //How many cells per layer (cell=JFET for Unit JFET)
    //Layer 0 being Unit SCPM, last layer being outer layer
    int *n_stage;
}config;

//Factor list
typedef struct
{
    int n_factor;
    int *factors;
}flist;

//Factor list array
typedef struct
{
    int n_flist;
    flist *lists;
}farray;

//List of configurations
int n_config=0;
config *configs;

//Parameters
float ntotal; //Total JFET number of entire SCPM
float vds; //Vds of individual JFET
float qds; //Qds in nC
float keb; //Keb factor
float kcf; //Kcf factor

//Calculate Eb of one SCPM topology
float calculate(int index)
```

```

{
    int i, n;
    //Per-cell voltage
    float v=vds;
    //Overall Eb
    float eb=0.0;
    //Overall cost factor
    float cf=0.0;
    //Memory boundary check
    if(index<0 || index>=n_config)
        return -1.0;
    //Allocate memory for Eb of each layer
    float *eb_layer; //in mJ
    eb_layer=malloc(sizeof(float)*configs[index].n_layer);
    //Memory allocation failed
    if(!eb_layer)
        return -1.0;
    //Allocate memory for CF of each layer
    float *cf_layer; //in nFV
    cf_layer=malloc(sizeof(float)*configs[index].n_layer);
    //Memory allocation failed
    if(!eb_layer)
    {
        free(eb_layer);
        return -1.0;
    }
    //Iterate through all layers, from inner to outer
    for(i=0;i<configs[index].n_layer;i++)
    {
        //Calculate Eb and cost factor per layer
        n=configs[index].n_stage[i];
        eb_layer[i]=0.25*qds*v*(n*n-n);
        cf_layer[i]=0.5*qds*(n*n-n);
        //Increase per-cell voltage variable
        v=v*n;
    }
    //n is used as cell count accumulator
    n=1;
    //Iterate through all layers, from inner to outer
    for(i=0;i<configs[index].n_layer;i++)
    {
        //Accumulate all eb_layer values
        n=n*configs[index].n_stage[i];
        eb=eb+eb_layer[i]*ntotal/n;
        cf=cf+cf_layer[i]*ntotal/n;
    }
}

```

```

    free(eb_layer);
    free(cf_layer);
    return eb*keb/1000000.0+cf*kcf;
}

//Display one configuration
void display(int n)
{
    int i;
    float tcf;
    //Print SCPM configuration index
    printf("SCPM %d:\n", n);
    for(i=0;i<configs[n].n_layer;i++)
        printf("N%d=%d\t", i+1, configs[n].n_stage[i]);
    //Calculate and print total cost factor
    tcf=calculate(n);
    //Error handling
    if(tcf==-1.0)
    {
        printf("Unexpected memory error.\n");
        exit(-1);
    }
    printf("TCF=%.2f\n", tcf);
}

//List all layers
void list(void)
{
    int i;
    //Enumerate all configurations
    for(i=0;i<n_config;i++)
        display(i);
}

//Factorize an integer
//Result stored in an array, starting with number of factors found
int *fact(int n)
{
    int *factors;
    //Factor counter
    int n_factors=0;
    int i, j;
    //Calculate all factors
    //Excluding 1
    for(i=2;i<n;i++)
        if(n%i==0) n_factors++;
}

```

```

//First elements is n_factors
factors=malloc(sizeof(int)*(n_factors+1));
if(!factors) return NULL;
factors[0]=n_factors;
//Calculate all factors
//Excluding 1
j=1;
for(i=2;i<n;i++)
    if(n%i==0) factors[j++]=i;
return factors;
}

//Push an item into factor list
int fpush(flist *list, int number)
{
    int *temp;
    if(list->n_factor==0)
        temp=malloc(sizeof(int));
    else
        temp=realloc(list->factors, sizeof(int)*(list->n_factor+1));
    if(!temp)
        return -1;
    list->factors=temp;
    list->factors[list->n_factor++]=number;
    return 0;
}

//Add a factor list to an array
int fadd(farray *array, flist list)
{
    flist *temp;
    if(array->n_flist==0)
        temp=malloc(sizeof(flist));
    else
        temp=realloc(array->lists, sizeof(flist)*(array->n_flist+1));
    if(!temp)
        return -1;
    array->lists=temp;
    array->lists[array->n_flist++]=list;
    return 0;
}

//Delete a flist array
void ffree(farray *array)
{
    if(!array) return;

```

```

if(array->n_flist==0 || !array->lists) return;
int i;
for(i=0;i<array->n_flist;i++)
    if(array->lists[i].factors)
        free(array->lists[i].factors);
free(array->lists);
array->n_flist=0;
}

//Recursive core of factall() function
int fcore(int n, flist *list, farray *array)
{
    int *factors;
    int n1, n2;
    factors=fact(n);
    if(!factors) return -1;
    int i, j;
    flist list2;
    //Reached end
    if(factors[0]==0)
    {
        free(factors);
        return 0;
    }
    for(i=0;i<factors[0];i++)
    {
        n1=factors[i+1];
        n2=n/n1;
        //Add (... , n1, n2)
        list2.n_factor=0;
        for(j=0;j<list->n_factor;j++)
            if(fpush(&list2, list->factors[j])) return -1;
        if(fpush(&list2, n1)) return -1;
        if(fpush(&list2, n2)) return -1;
        fadd(array, list2);
        //Add (... , n1, fcore())
        list2.n_factor=0;
        for(j=0;j<list->n_factor;j++)
            if(fpush(&list2, list->factors[j])) return -1;
        if(fpush(&list2, n1)) return -1;
        if(fcore(n2, &list2, array)) return -1;
        free(list2.factors);
    }
    free(factors);
    return 0;
}

```

```

//Find any factor combination of an integer
//Result stored in an array, starting with number of factors found
farray factall(int n)
{
    farray array;
    array.n_flist=0;
    //Recursively find factor combinations
    flist temp;
    temp.n_factor=0;
    if(fcore(n, &temp, &array)==-1)
    {
        ffree(&array);
        return array;
    }
    //Only n2 is entered to recursion, hence no duplications are generated
    return array;
}

//Free configs structure
void freeall(void)
{
    int i;
    for(i=0;i<n_config;i++)
    {
        //Do not free a freed pointer
        if(configs[i].n_stage)
            free(configs[i].n_stage);
    }
    free(configs);
}

//Entry point of program
int main(int argc, char **argv)
{
    //Argument check
    if(argc!=6)
    {
        printf("Syntax error.\nSyntax: %s [n_stage] [vds (V)] [qds (nC)] [keb] [kcf]\n",
argv[0]);
        return -1;
    }
    //Read arguments
    ntotal=atoi(argv[1]); //total stage count
    vds=atof(argv[2]); //voltage per JFET
    qds=atof(argv[3]); //charge per JFET (Qds-Qd)
    keb=atof(argv[4]); //Keb
}

```

```

kcf=atof(argv[5]); //Kcf
//Generate 2 layer designs
int n1, n2;
int i, j;
//Find all factor combos of total stage count
farray array;
array=factall(ntotal);
if(array.n_flist==0)
{
    printf("Unexpected memory error.\n");
    return -1;
}
config *temp;
//Generate designs
//Add 1-layer design
configs=malloc(sizeof(config));
configs[0].n_layer=1;
configs[0].n_stage=malloc(sizeof(int));
configs[0].n_stage[0]=ntotal;
n_config=1;
//Enumerate through all factor combos
for(i=0;i<array.n_flist;i++)
{
    temp=realloc(configs, sizeof(config)*(n_config+1));
    if(!temp)
    {
        printf("Unexpected memory error.\n");
        freeall();
        free(&array);
        return -1;
    }
    configs=temp;
    configs[n_config].n_layer=array.lists[i].n_factor;
    configs[n_config].n_stage=malloc(sizeof(int)*array.lists[i].n_factor);
    if(!configs[n_config].n_stage)
    {
        printf("Unexpected memory error.\n");
        freeall();
        free(&array);
        return -1;
    }
    for(j=0;j<array.lists[i].n_factor;j++)
        configs[n_config].n_stage[j]=array.lists[i].factors[j];
    n_config++;
}
free(&array);

```

```

//Print all designs
#ifdef TESTMODE
list();
#endif
//Find best solution
float tcfmin=FLT_MAX, tcf;
for(i=0;i<n_config;i++)
{
    tcf=calculate(i);
    if(tcf<tcfmin)
        tcfmin=tcf;
}
//Display minimum TCF design
printf("\nThe optimal designs are:\n");
for(i=0;i<n_config;i++)
{
    if(calculate(i)==tcfmin)
        display(i);
}
freeall();
return 0;
}

```

The program was tested with gcc 7.2.0 on MSYS2 platform.

## Appendix E Program for Optimizing SCPM-M Designs with Fiber Optical Gate Drivers

```
#include <stdio.h>
#include <stdlib.h>
#include <float.h>

//Enable verbose output
#define TESTMODE

//One type of configuration
typedef struct
{
    //How many layers
    int n_layer;
    //How many cells per layer (cell=JFET for Unit JFET)
    //Layer 0 being Unit SCPM, last layer being outer layer
    int *n_stage;
}config;

//Factor list
typedef struct
{
    int n_factor;
    int *factors;
}flist;

//Factor list array
typedef struct
{
    int n_flist;
    flist *lists;
}farray;

//List of configurations
int n_config=0;
config *configs;

//Parameters
float ntotal; //Total JFET number of entire SCPM
float vds; //Vds of individual JFET
float qds; //Qds in nC
float keb; //Keb factor
float kcf; //Kcf factor
float kdr; //Kdr factor

//Calculate Eb of one SCPM topology
```

```

float calculate(int index)
{
    int i, n;
    //Per-cell voltage
    float v=vds;
    //Overall Eb
    float eb=0.0;
    //Overall cost factor
    float cf=0.0;
    //Optical driver cost
    int dr;
    //Memory boundary check
    if(index<0 || index>=n_config)
        return -1.0;
    //Allocate memory for Eb of each layer
    float *eb_layer; //in mJ
    eb_layer=malloc(sizeof(float)*configs[index].n_layer-1);
    //Memory allocation failed
    if(!eb_layer)
        return -1.0;
    //Allocate memory for CF of each layer
    float *cf_layer; //in nFV
    cf_layer=malloc(sizeof(float)*configs[index].n_layer-1);
    //Memory allocation failed
    if(!eb_layer)
    {
        free(eb_layer);
        return -1.0;
    }
    //Iterate through all inner layers, from inner to outer
    for(i=0;i<configs[index].n_layer-1;i++)
    {
        //Calculate Eb and cost factor per layer
        n=configs[index].n_stage[i];
        eb_layer[i]=0.25*qds*v*(n*n-n);
        cf_layer[i]=0.5*qds*(n*n-n);
        //Increase per-cell voltage variable
        v=v*n;
    }
    //n is used as cell count accumulator
    n=1;
    //Iterate through all inner layers, from inner to outer
    for(i=0;i<configs[index].n_layer-1;i++)
    {
        //Accumulate all eb_layer values
        n=n*configs[index].n_stage[i];
    }
}

```

```

        eb=eb+eb_layer[i]*ntotal/n;
        cf=cf+cf_layer[i]*ntotal/n;
    }
    free(eb_layer);
    free(cf_layer);
    //Calculate number of optical gate driver
    //Here we assume all external stages require optical gate driver
    if(configs[index].n_layer>1)
        dr=configs[index].n_stage[configs[index].n_layer-1];
    else
        dr=1;
    return eb*keb/1000000.0+cf*kcf+dr*kdr;
}

//Display one configuration
void display(int n)
{
    int i;
    float tcf;
    //Print SCPM configuration index
    printf("SCPM %d:\n", n);
    for(i=0;i<configs[n].n_layer;i++)
        printf("N%d=%d\t", i+1, configs[n].n_stage[i]);
    //Calculate and print total cost factor
    tcf=calculate(n);
    //Error handling
    if(tcf==-1.0)
    {
        printf("Unexpected memory error.\n");
        exit(-1);
    }
    printf("TCF=%.2f\n", tcf);
}

//List all layers
void list(void)
{
    int i;
    //Enumerate all configurations
    for(i=0;i<n_config;i++)
        display(i);
}

//Factorize an integer
//Result stored in an array, starting with number of factors found
int *fact(int n)

```

```

{
    int *factors;
    //Factor counter
    int n_factors=0;
    int i, j;
    //Calculate all factors
    //Excluding 1
    for(i=2;i<n;i++)
        if(n%i==0) n_factors++;
    //First elements is n_factors
    factors=malloc(sizeof(int)*(n_factors+1));
    if(!factors) return NULL;
    factors[0]=n_factors;
    //Calculate all factors
    //Excluding 1
    j=1;
    for(i=2;i<n;i++)
        if(n%i==0) factors[j++]=i;
    return factors;
}

//Push an item into factor list
int fpush(flist *list, int number)
{
    int *temp;
    if(list->n_factor==0)
        temp=malloc(sizeof(int));
    else
        temp=realloc(list->factors, sizeof(int)*(list->n_factor+1));
    if(!temp)
        return -1;
    list->factors=temp;
    list->factors[list->n_factor++]=number;
    return 0;
}

//Add a factor list to an array
int fadd(farray *array, flist list)
{
    flist *temp;
    if(array->n_flist==0)
        temp=malloc(sizeof(flist));
    else
        temp=realloc(array->lists, sizeof(flist)*(array->n_flist+1));
    if(!temp)
        return -1;
}

```

```

    array->lists=temp;
    array->lists[array->n_flist++]=list;
    return 0;
}

//Delete a flist array
void ffree(farray *array)
{
    if(!array) return;
    if(array->n_flist==0 || !array->lists) return;
    int i;
    for(i=0;i<array->n_flist;i++)
        if(array->lists[i].factors)
            free(array->lists[i].factors);
    free(array->lists);
    array->n_flist=0;
}

//Recursive core of factall() function
int fcore(int n, flist *list, farray *array)
{
    int *factors;
    int n1, n2;
    factors=fact(n);
    if(!factors) return -1;
    int i, j;
    flist list2;
    //Reached end
    if(factors[0]==0)
    {
        free(factors);
        return 0;
    }
    for(i=0;i<factors[0];i++)
    {
        n1=factors[i+1];
        n2=n/n1;
        //Add (... , n1, n2)
        list2.n_factor=0;
        for(j=0;j<list->n_factor;j++)
            if(fpush(&list2, list->factors[j])) return -1;
        if(fpush(&list2, n1)) return -1;
        if(fpush(&list2, n2)) return -1;
        fadd(array, list2);
        //Add (... , n1, fcore())
        list2.n_factor=0;
    }
}

```

```

        for(j=0;j<list->n_factor;j++)
            if(fpush(&list2, list->factors[j])) return -1;
        if(fpush(&list2, n1)) return -1;
        if(fcore(n2, &list2, array)) return -1;
        free(list2.factors);
    }
    free(factors);
    return 0;
}

//Find any factor combination of an integer
//Result stored in an array, starting with number of factors found
farray factall(int n)
{
    farray array;
    array.n_flist=0;
    //Recursively find factor combinations
    flist temp;
    temp.n_factor=0;
    if(fcore(n, &temp, &array)==-1)
    {
        ffree(&array);
        return array;
    }
    //Only n2 is entered to recursion, hence no duplications are generated
    return array;
}

//Free configs structure
void freeall(void)
{
    int i;
    for(i=0;i<n_config;i++)
    {
        //Do not free a freed pointer
        if(configs[i].n_stage)
            free(configs[i].n_stage);
    }
    free(configs);
}

//Entry point of program
int main(int argc, char **argv)
{
    //Argument check
    if(argc!=7)

```

```

    {
        printf("Syntax error.\nSyntax: %s [n_stage] [vds (V)] [qds (nC)] [keb] [kcf] [kdr]\n",
argv[0]);
        return -1;
    }
//Read arguments
ntotal=atoi(argv[1]); //total stage count
vds=atof(argv[2]); //voltage per JFET
qds=atof(argv[3]); //charge per JFET (Qds-Qd)
keb=atof(argv[4]); //Keb
kcf=atof(argv[5]); //Kcf
kdr=atof(argv[6]); //Kdr
//Generate 2 layer designs
int n1, n2;
int i, j;
//Find all factor combos of total stage count
farray array;
array=factall(ntotal);
if(array.n_flist==0)
{
    printf("Unexpected memory error.\n");
    return -1;
}
config *temp;
//Generate designs
//Add 1-layer design
configs=malloc(sizeof(config));
configs[0].n_layer=2;
configs[0].n_stage=malloc(sizeof(int)*2);
configs[0].n_stage[0]=ntotal;
configs[0].n_stage[1]=1;
n_config=1;
//Enumerate through all factor combos
for(i=0;i<array.n_flist;i++)
{
    temp=realloc(configs, sizeof(config)*(n_config+1));
    if(!temp)
    {
        printf("Unexpected memory error.\n");
        freeall();
        free(&array);
        return -1;
    }
    configs=temp;
    configs[n_config].n_layer=array.lists[i].n_factor;
    configs[n_config].n_stage=malloc(sizeof(int)*array.lists[i].n_factor);

```

```

    if(!configs[n_config].n_stage)
    {
        printf("Unexpected memory error.\n");
        freeall();
        free(&array);
        return -1;
    }
    for(j=0;j<array.lists[i].n_factor;j++)
        configs[n_config].n_stage[j]=array.lists[i].factors[j];
    n_config++;
}
free(&array);
//Print all designs
#ifdef TESTMODE
list();
#endif
//Find best solution
float tcfmin=FLT_MAX, tcf;
for(i=0;i<n_config;i++)
{
    tcf=calculate(i);
    if(tcf<tcfmin)
        tcfmin=tcf;
}
//Display minimum TCF design
printf("\nThe optimal designs are:\n");
for(i=0;i<n_config;i++)
{
    if(calculate(i)==tcfmin)
        display(i);
}
freeall();
return 0;
}

```

The program was tested with gcc 7.2.0 on MSYS2 platform.