JACOBS
UNIVERSITY

Active Cell Balancing using Low Cost Components

by

Dorin Clisu

Bachelor Thesis in Electrical and Computer Engineering

Prof.Dr. Werner Bergholz, Uwe Pagel

Date of Submission: August 31, 2016

Jacobs University — School of Engineering and Science

Dorin Gabriel Clisu   Bremen, August 31, 2016

Abstract

Lithium-Polymer batteries consist of multiple cells stacked in series to deliver a specific voltage. Because minute differences in internal characteristics lead to unequal cell voltages during charge and discharge cycles, an electronic circuit is often used to balance the charge between cells in order to prevent damage. This thesis presents a novel circuit using inductors applying energy transfer between the cells. The ability to redistribute the energy from any subset of cells to the complementary subset in a single cycle minimizes the number of balancing cycles and thus increases the dynamic efficiency for an arbitrary cell voltage distribution. The circuit has been implemented using low cost parts with a microcontroller for reading the voltages, running the balancing algorithm and generating control PWM pulses. Balancing currents of up to 0.9A and static efficiency of up to 59% have been achieved using non-optimized gate drivers.

Contents

1 Introduction

Active cell balancing implies charge / energy transfer between cells, coordinated by a controller that knows the voltage / state-of-charge of the cells. Naturally, energy flows from higher voltage potential towards lower voltage potential, however that is impossible to accomplish within a series stack of cells using naive electrical connections. Therefore we need to transfer a small fraction of the higher voltage cell energy into a temporary element that will then transfer the energy to the lower voltage cell. The process would be repeated quickly such that there appears to be a continuous flow of energy. Two suitable elements are capacitors and inductors due to their high efficiency for short term energy storage. Capacitors store energy in the electric field of the charge while inductors store energy in the magnetic field of the current, as such their mode of usage differs significantly. A special type of inductor is the transformer which in turn can be of two types – forward and flyback.

Existing active balancing solutions include:

- Switched Capacitor [3]
- Adjacent Inductor [8] [7]
- Simplified Adjacent Inductor [1]
- Shared Dual Inductor [2]
- Shared-Primary Forward Transformer [3]
- Shared-Primary Flyback Transformer [5]
- Multiple Flyback Transformer [9]

This thesis brings a small modification to the Simplified Adjacent Inductor Topology([1]) in order to ease the harsh MOSFET control requirements. While the proposed circuit topology could theoretically work for an arbitrary cell number of any type (Fig. 1), this work is a case study on balancing 4 Li-Po cells.

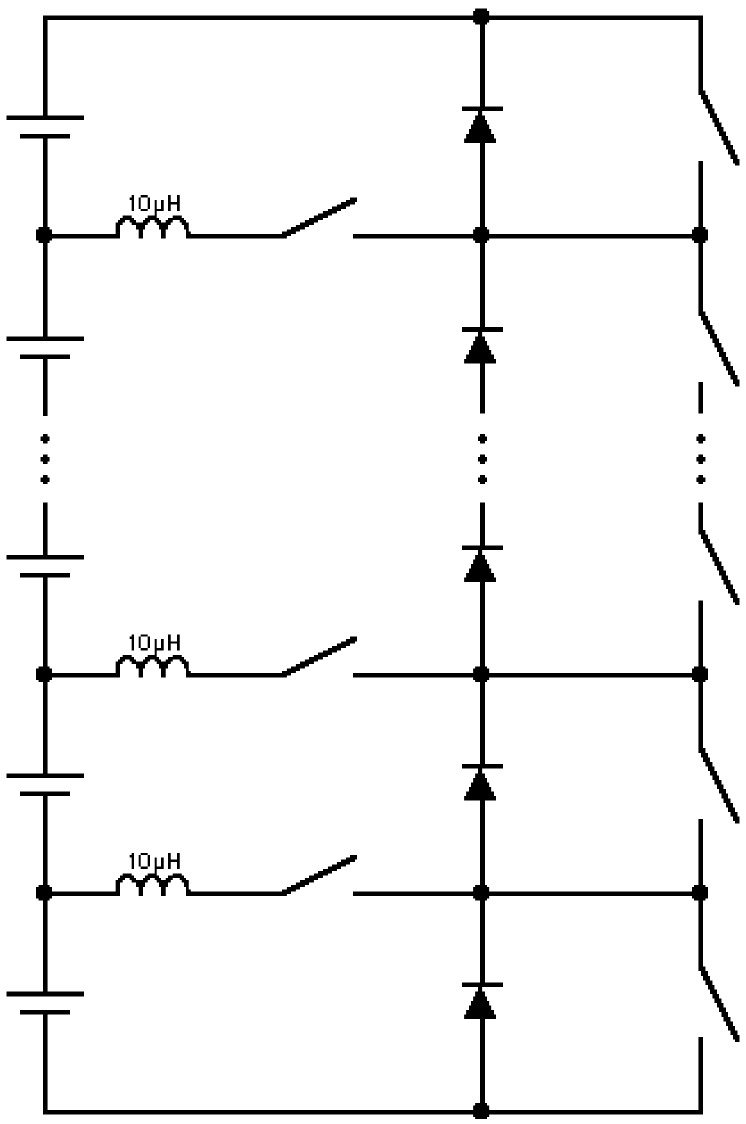


Figure 1: Novel cell balancing topology (simplified)

2 Circuit Design

2.1 Controller Selection

Optimum cell balancing requires closed loop control – the voltage of each cell is measured, the state of the system is determined, an algorithm determines the optimum correction, the controller applies the correction to the cells and the process is repeated.

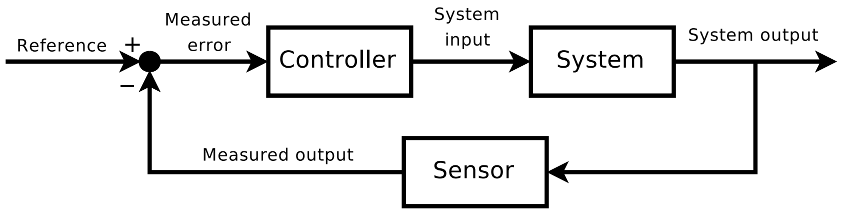


Figure 2: Typical single-input, single-output control feedback loop

Many solutions use separate ICs for voltage monitoring, switching control and higher level control (algorithms). The key idea in order to achieve low-cost is to use the integrated functionality of a microcontroller. Critical requirements are ADC channel count & resolution and output channel count & speed. Other important aspects are building tools that are easy to use, various peripherals (UART, CAN) for external communication and good general processing power for supporting complex algorithms.

The microcontroller of choice in this thesis is the STM32 F103 from STMicroelectronics. It is a popular MCU since its adoption on the LeafLabs platform, packing a good amount of functionality and speed for a unit price of 1-2 EUR. A small, inexpensive (4 EUR) development board is also available and the Arduino Hardware Abstraction Layer can be used for quick firmware setup and development, leaving low level C or register I/O only for the time critical output control.

Relevant specifications for our application are:

- 2 x 12 bit ADC at 1Msample/s, 8 channels each
- 31 x GPIO pins on the dev board with output speed up to 18MHz
- 12 x PWM pins on the dev board
- 3 x 16 bit general purpose timers with overflow interrupts
- 72MHz CPU with 20kB SRAM
- USART, USB, I2C, SPI, CAN interfaces

2.2 Novel Topology Details

A novel topology is proposed as a small modification to the existent adjacent inductor circuit [1]. The idea came while trying to solve the complicated system of equations describing the original circuit by splitting the problem into several smaller cases. Each

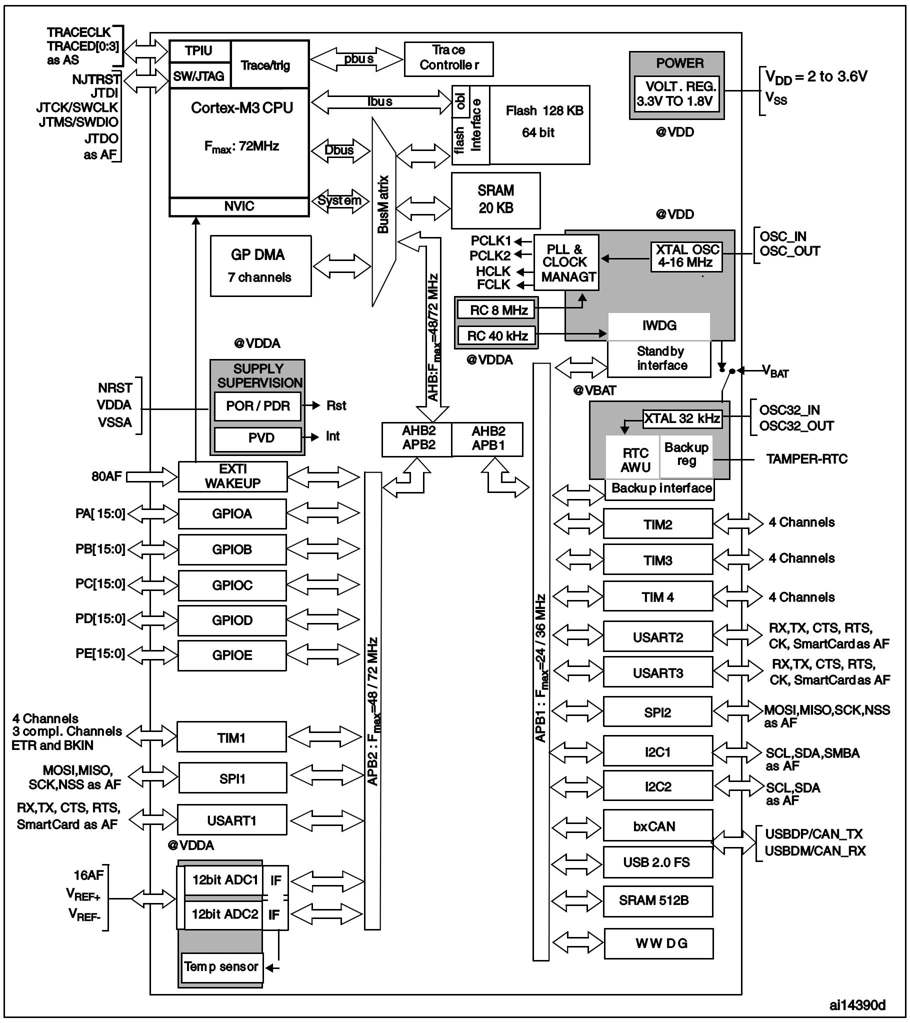


Figure 3: STM32F103xx performance line block diagram [12]

case represents an unique mode of operation consisting of transferring charge from a given subset of cells to the rest of the cells, and will be analyzed in the next section.

The modification consists of adding a bidirectional on/off switch on each of the inductor branches by using 2 back-to-back MOSFETs as shown in Fig. 4. This allows us to disconnect any unused inductor that would otherwise interfere with a desired charge transfer, because most of the time only 1 or 2 inductors need to be used. Moreover, the new topology has the potential to be more energy efficient for random charge transfer because by ”deactivating” the unused inductors for a specific mode, charge can be channeled more directly between distanced cells instead of being shuttled from inductor to inductor in a chain (until it reaches the destination) as it happens with the original circuit.

This would require 3 more outputs from the microcontroller but it is not an issue since we can use any GPIO (no need for precise timing and synchronisation). The cost impact of adding 6 more MOSFETs has been estimated (with preliminary market research) to be relatively low because the price of a suitable MOSFET ($44A, 10m\Omega$) is small (0.1-0.2EUR) compared to the cost of a suitable inductor (0.5-1EUR for $10\mu H, 5A$).

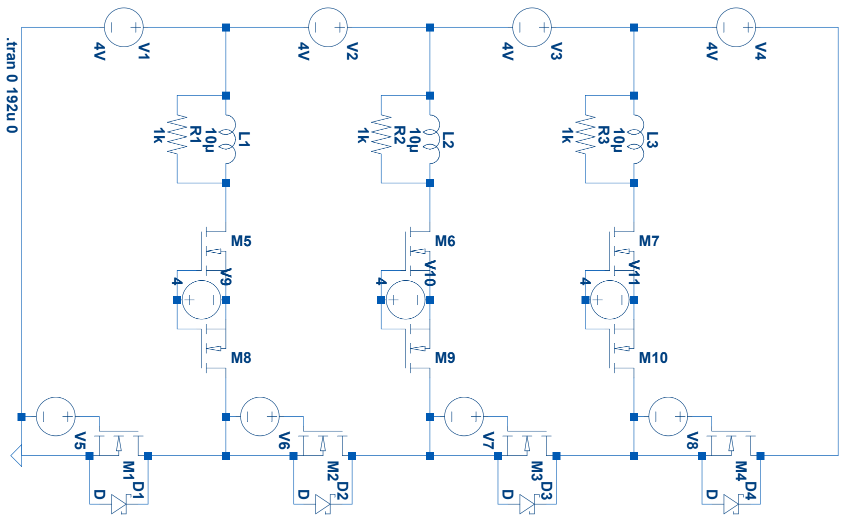


Figure 4: Proposed novel topology with inductor branch switches

2.3 Charge Transfer Modes

Reviewing basic inductor theory:

$$V = L \cdot \frac{dI(t)}{dt}$$

Assuming the voltage does not change significantly on a short timescale, the equation becomes:

$$I(t) = I(0) + \frac{1}{L}\int_0^t V\,dt = I(0) + \frac{V \cdot t}{L}$$

There are 2 modes in which an inductor can operate in a DC circuit [13]:

- Discontinuous mode - the inductor is energized up to a peak current and then it is discharged completely before starting a new cycle.
- Continuous mode - the inductor is re-energized before it is discharged completely, allowing it to carry higher average current with less ripple while under the same current peak limit.

Because the continuous mode implies switching on/off non-zero currents, very fast MOSFET rise and fall times are needed (otherwise high switching losses occur) which pose a practical problem for the gate drivers. Additionally, closed-loop current control is needed which would greatly complicate the circuit. For these reasons, the discontinuous mode will be used, meaning I(0) = 0 so we can write:

$$I_{peak} = \frac{V_C \cdot t_C}{L} = \frac{V_D \cdot t_D}{L}$$

V_D is the sum voltage of the cells that are being discharged for time t_D and and V_C is the sum voltage of the cells being charged for time t_C, and L is either a single inductor L_1 or 2 series inductors with equivalent inductance $2L_1$. While t_D can be controlled by the switching MOSFETs, any peak current can be achieved by generating pulses as functions of inductance, current and cell voltage:

$$t_D = L \cdot I_{peak} \cdot \frac{1}{V_D} \quad \text{and} \quad t_C = L \cdot I_{peak} \cdot \frac{1}{V_C}$$

In discontinuous mode, t_C can only be partially controlled because the diodes will conduct current as long as any inductor still has energy to discharge, so while it is optional to generate pulses synchronised to the charging phases, this would increase the efficiency (details in the next sections).

The proposed circuit can transfer current from any subset of cells to the complementary subset. Each subset defines a mode of operation for the circuit, and there are $N(4)$ such modes, as shown in Tab. 1:

$$N(n) = \sum_{i=1}^{n-1} \binom{n}{i} = 2^n - 2 = 2^4 - 2 = 14$$

i	$mirror$	$mode$	$cell1$	$cell2$	$cell3$	$cell4$
1	9	CDDD	charg	disch	disch	disch
2	10	DCCC	disch	charg	charg	charg
3	11	CDCC	charg	disch	charg	charg
4	12	DDCD	disch	disch	charg	disch
5	13	DDCC	disch	disch	charg	charg
6	14	DCDC	disch	charg	disch	charg
7	7	CDDC	charg	disch	disch	charg
8	8	DCCD	disch	charg	charg	disch
9	1	DDDC	disch	disch	disch	charg
10	2	CCCD	charg	charg	charg	disch
11	3	CCDC	charg	charg	disch	charg
12	4	DCDD	disch	charg	disch	disch
13	5	CCDD	charg	charg	disch	disch
14	6	CDCD	charg	disch	charg	disch

Table 1: Charge transfer modes

From the 14 modes, 2 are symmetrical while the rest are mirrored pairs, leaving only $14 - (14 - 2)/2 = 8$ unique modes which will be analyzed.

A balancing cycle consists of a set of (parallel) discharging phases and a set of (parallel) charging phases. It must have a period long enough to ensure that the inductors remain in the discontinuous mode and short enough to have as little ripple current as possible.

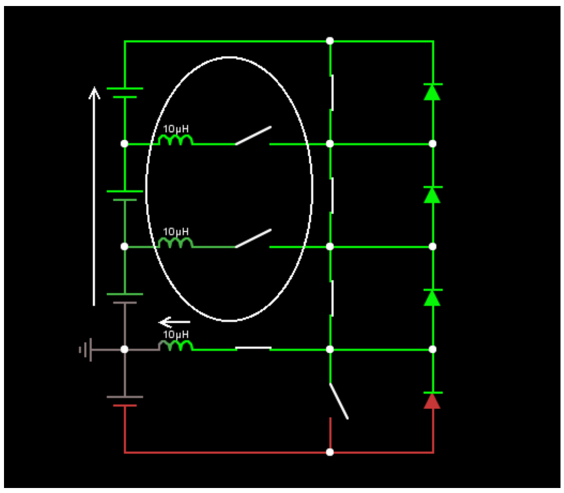
$$T_{cycle} = max(t_D(i)) + max(t_C(j)) \quad i, j = \overline{1, 4}$$

The maximum symbolic value of the t_D and t_C phases can be determined before runtime in most modes but not all ($V(i)+V(j) > V(k) \quad \forall i, j, k = \overline{1, 4}$ assuming $3V - 4.2V$ as valid voltage range for the cells), because the formulas depend on the runtime cell voltages. When generating the 4 pulses with widths given by t_D and t_C, they must be offset by relative delays $t_{delay}(i)$ because the transition from the discharging phases to charging phases must happen simultaneously for all the cells.

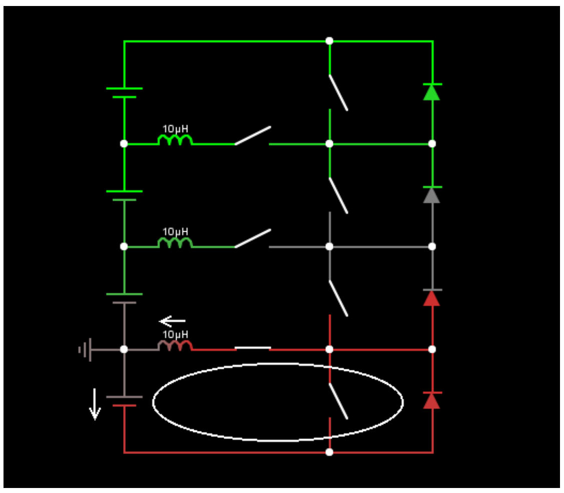
For each mode, the 3 branch switches are set such that charge transfer is optimized in the relevant direction.

For intuitive visualization, the circuits are drawn in Falstad Circuit Simulator, green / red colors representing + / - voltage gradients. The waveforms are example results from the LTSpice simulation ($V1 = V2 = V3 = V4 = 4V$*,* $L_1 = 10\mu H$ *and* $I_{peak} = 4.8A$*).*

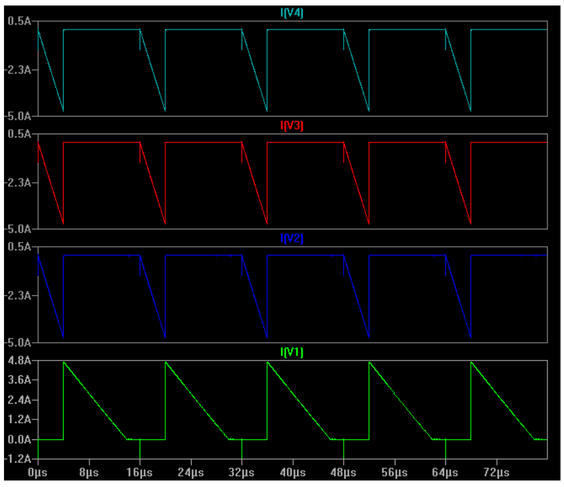
2.3.1 CDDD



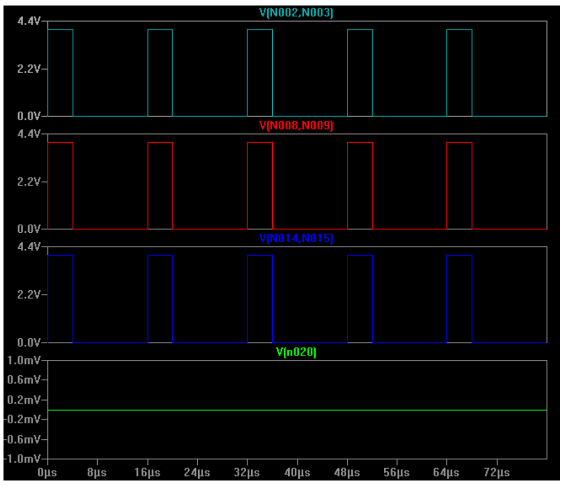
(a) Discharge phase



(b) Charge phase



(c) Cell currents



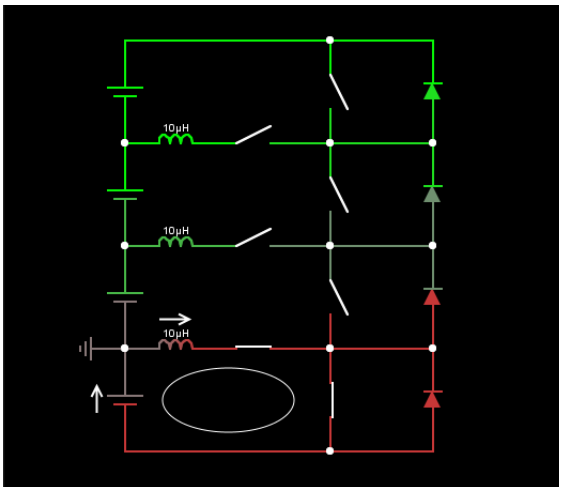
(d) MOSFET pulses

Figure 5: CDDD mode configuration

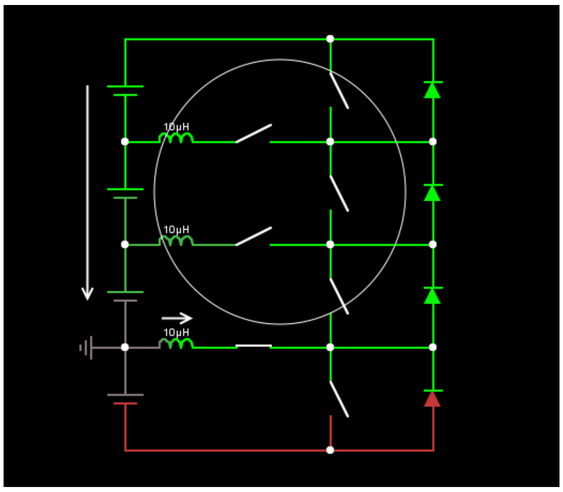
$t_D4 = L_1 I_{peak} \cdot \frac{1}{V2+V3+V4}$	$t_C4 = 0$	$t_{delay}4 = 0$
$t_D3 = L_1 I_{peak} \cdot \frac{1}{V2+V3+V4}$	$t_C3 = 0$	$t_{delay}3 = 0$
$t_D2 = L_1 I_{peak} \cdot \frac{1}{V2+V3+V4}$	$t_C2 = 0$	$t_{delay}2 = 0$
$t_D1 = 0$	$t_C1 = L_1 I_{peak} \cdot \frac{1}{V1}$	$t_{delay}1 = t_D2$
$T_{cycle} = t_D2 + t_C1 = L_1 I_{peak} \cdot \frac{V1+V2+V3+V4}{V1 \cdot (V2+V3+V4)}$		

Table 2: CDDD mode equations

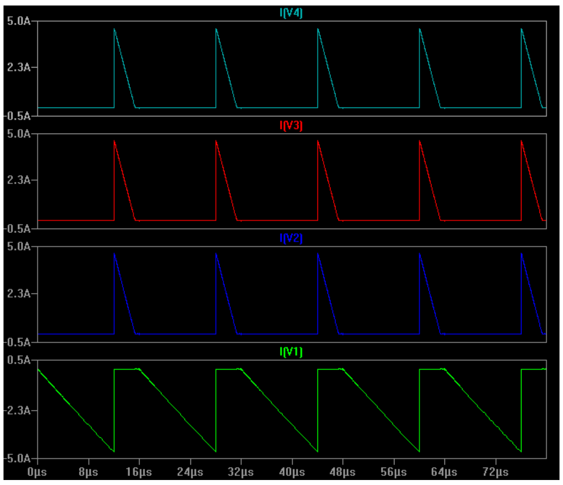
2.3.2 DCCC



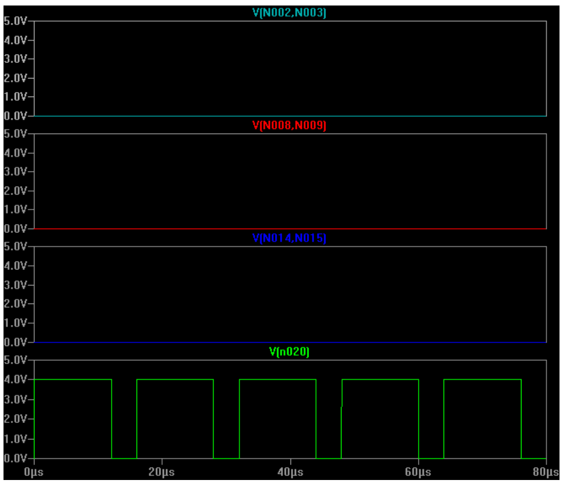
(a) Discharge phase



(b) Charge phase



(c) Cell currents



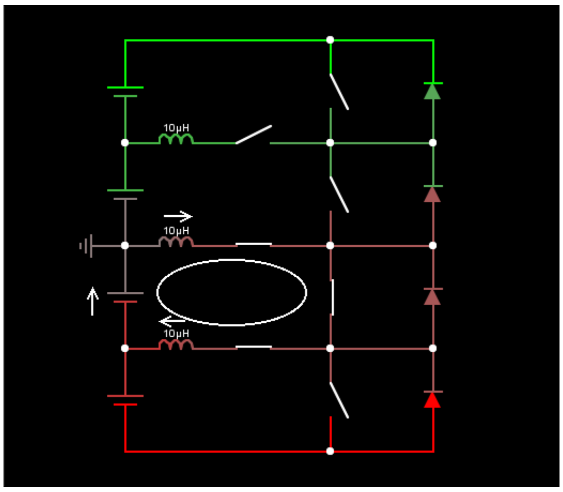
(d) MOSFET pulses

Figure 6: DCCC mode configuration

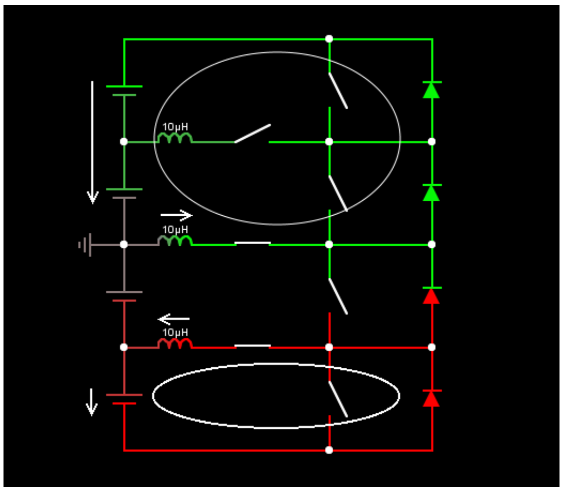
$t_D4 = 0$	$t_C4 = L_1 I_{peak} \cdot \frac{1}{V2+V3+V4}$	$t_{delay}4 = t_D1$
$t_D3 = 0$	$t_C3 = L_1 I_{peak} \cdot \frac{1}{V2+V3+V4}$	$t_{delay}3 = t_D1$
$t_D2 = 0$	$t_C2 = L_1 I_{peak} \cdot \frac{1}{V2+V3+V4}$	$t_{delay}2 = t_D1$
$t_D1 = L_1 I_{peak} \cdot \frac{1}{V1}$	$t_C1 = 0$	$t_{delay}1 = 0$
$T_{cycle} = t_D1 + t_C2 = L_1 I_{peak} \cdot \frac{V1+V2+V3+V4}{V1 \cdot (V2+V3+V4)}$		

Table 3: DCCC mode equations

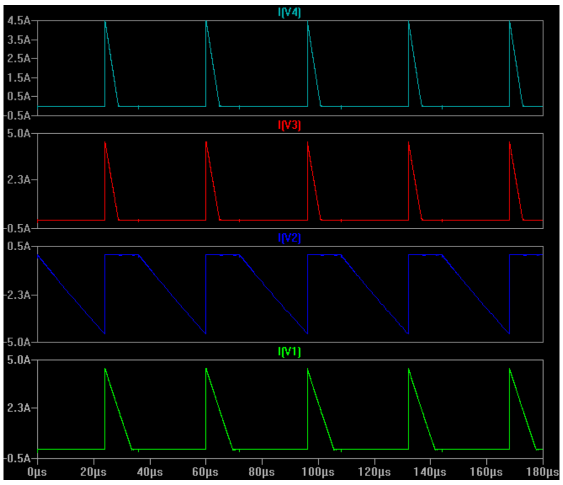
2.3.3 CDCC



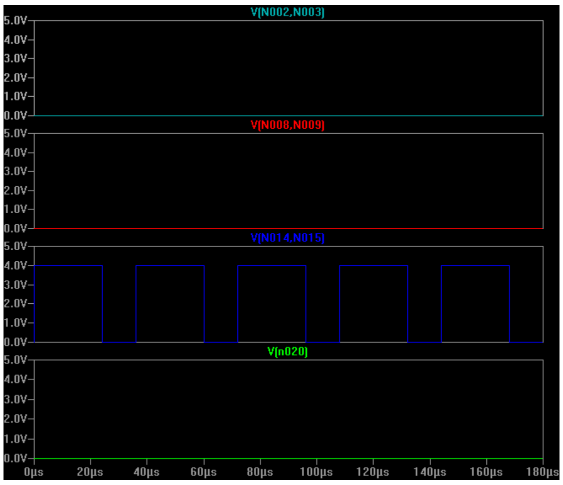
(a) Discharge phase



(b) Charge phase



(c) Cell currents



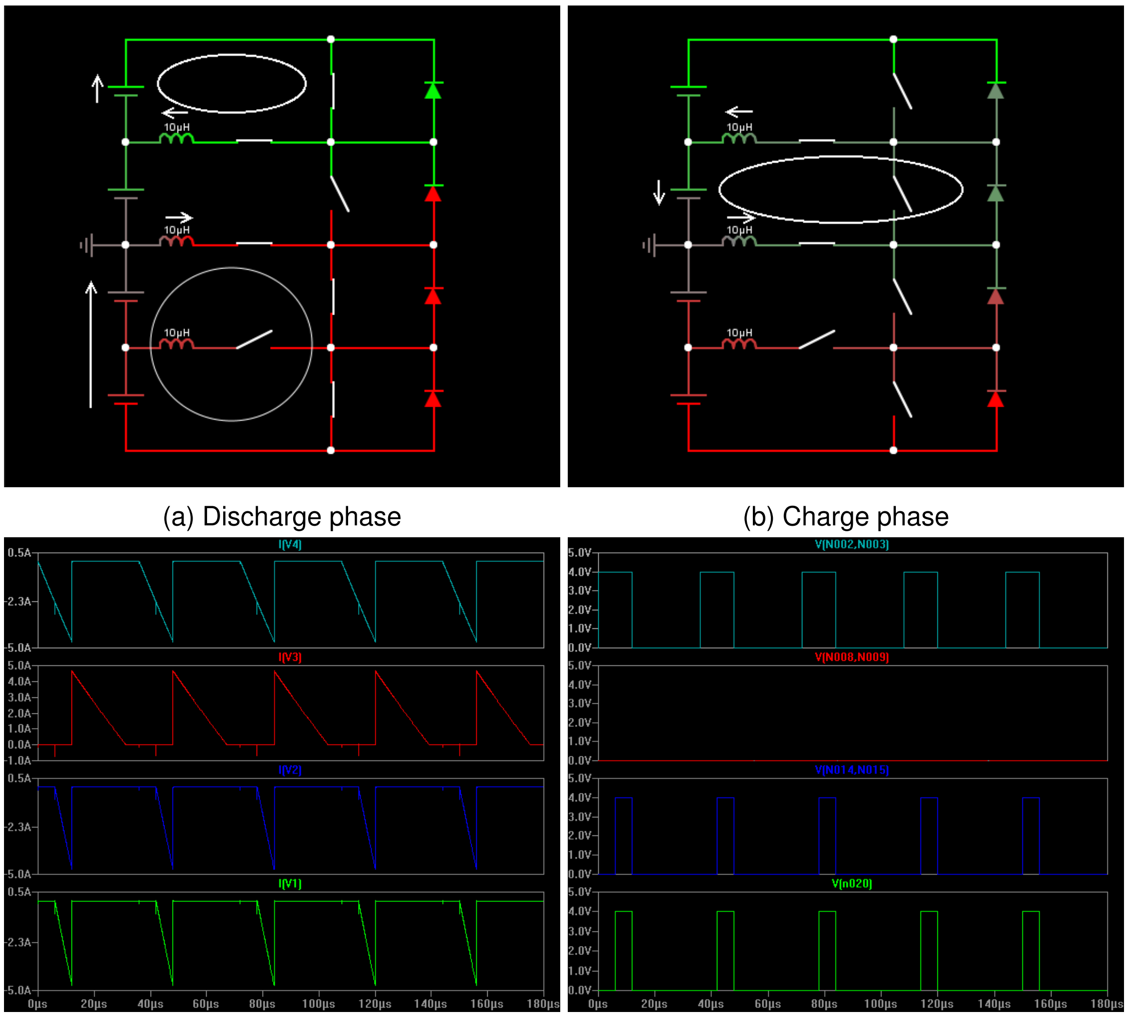
(d) MOSFET pulses

Figure 7: CDCC mode configuration

$t_D4 = 0$	$t_C4 = L_1 I_{peak} \cdot \frac{1}{V3+V4}$	$t_{delay}4 = t_D2$
$t_D3 = 0$	$t_C3 = L_1 I_{peak} \cdot \frac{1}{V3+V4}$	$t_{delay}3 = t_D2$
$t_D2 = L_1 I_{peak} \cdot \frac{2}{V2}$	$t_C2 = 0$	$t_{delay}2 = 0$
$t_D1 = 0$	$t_C1 = L_1 I_{peak} \cdot \frac{1}{V1}$	$t_{delay}1 = t_D2$
$T_{cycle} = t_D2 + t_C1 = L_1 I_{peak} \cdot \frac{2V1+V2}{V1 \cdot V2}$		

Table 4: CDCC mode equations

2.3.4 DDCD



(a) Discharge phase (b) Charge phase

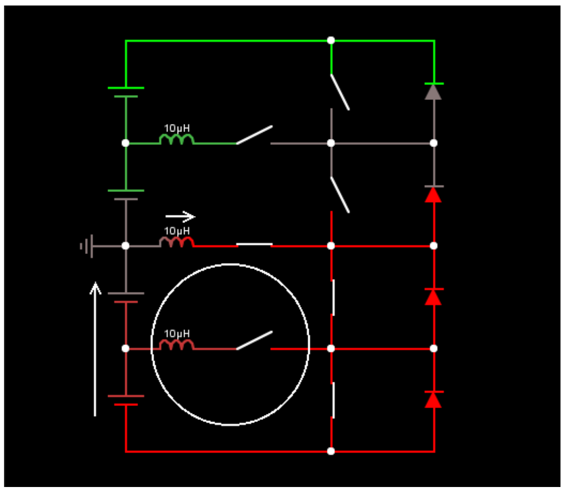
(c) Cell currents (d) MOSFET pulses

Figure 8: DDCD mode configuration

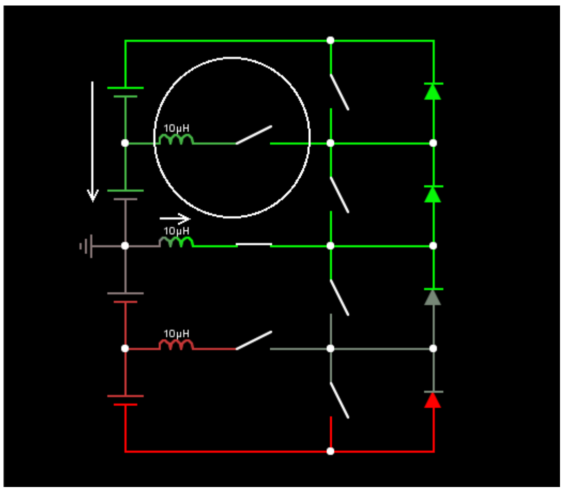
$t_D4 = L_1 I_{peak} \cdot \frac{1}{V4}$	$t_C4 = 0$	$t_{delay}4 = 0$
$t_D3 = 0$	$t_C3 = L_1 I_{peak} \cdot \frac{2}{V3}$	$t_{delay}3 = t_D4$
$t_D2 = L_1 I_{peak} \cdot \frac{1}{V1+V2}$	$t_C2 = 0$	$t_{delay}2 = t_D4 - t_D2$
$t_D1 = L_1 I_{peak} \cdot \frac{1}{V1+V2}$	$t_C1 = 0$	$t_{delay}1 = t_D4 - t_D1$
$T_{cycle} = t_D4 + t_C3 = L_1 I_{peak} \cdot \frac{V3+2V4}{V3 \cdot V4}$		

Table 5: DDCD mode equations

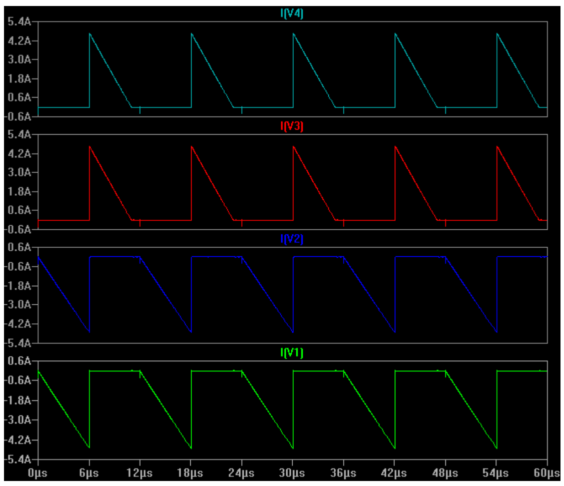
2.3.5 DDCC



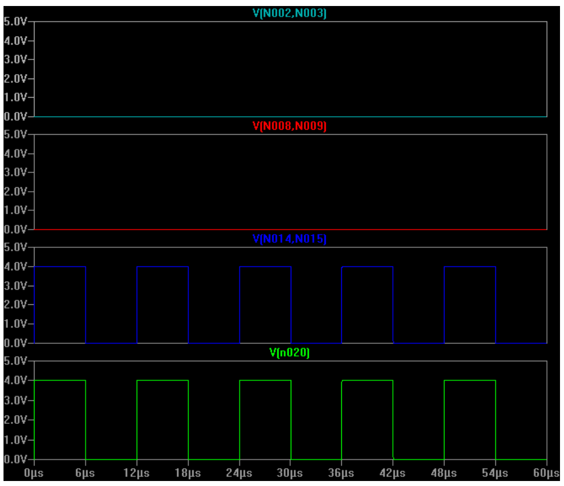
(a) Discharge phase



(b) Charge phase



(c) Cell currents



(d) MOSFET pulses

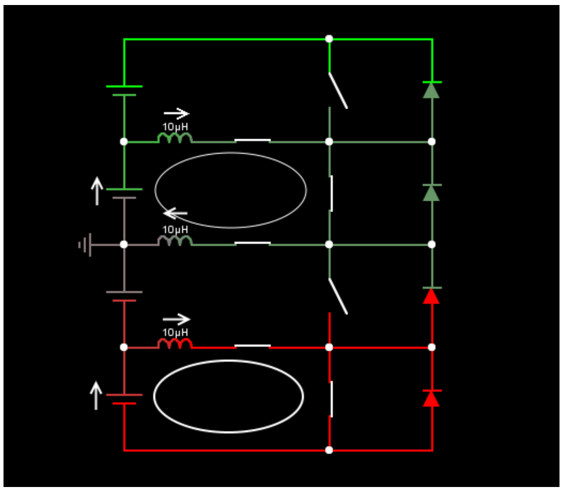
Figure 9: DDCC mode configuration

$t_D4 = 0$	$t_C4 = L_1 I_{peak} \cdot \frac{1}{V3+V4}$	$t_{delay}4 = t_D1$
$t_D3 = 0$	$t_C3 = L_1 I_{peak} \cdot \frac{1}{V3+V4}$	$t_{delay}3 = t_D1$
$t_D2 = L_1 I_{peak} \cdot \frac{1}{V1+V2}$	$t_C2 = 0$	$t_{delay}2 = 0$
$t_D1 = L_1 I_{peak} \cdot \frac{1}{V1+V2}$	$t_C1 = 0$	$t_{delay}1 = 0$

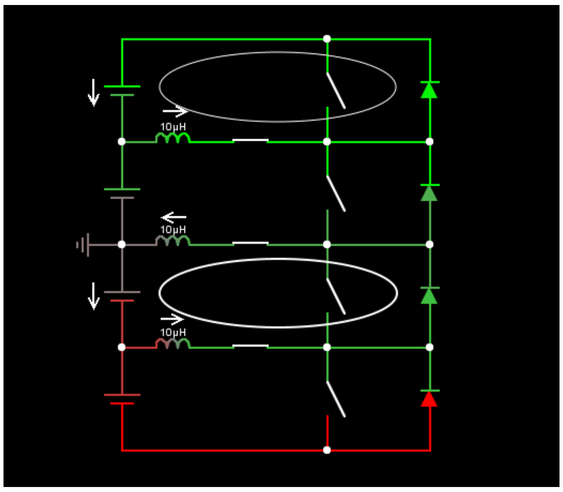
$$T_{cycle} = t_D1 + t_C3 = L_1 I_{peak} \cdot \frac{V1+V2+V3+V4}{(V1+V2)\cdot(V3+V4)}$$

Table 6: DDCC mode equations

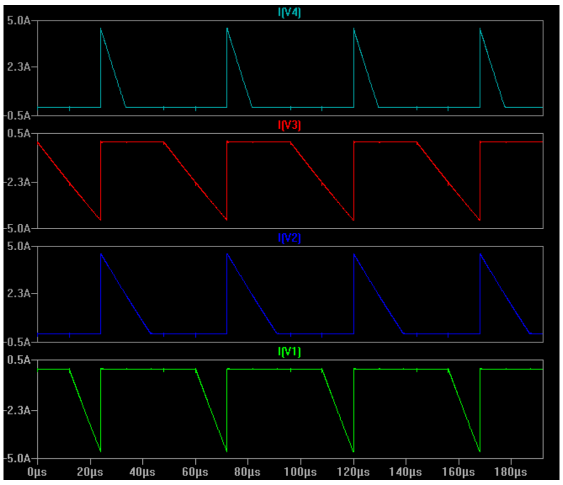
2.3.6 DCDC



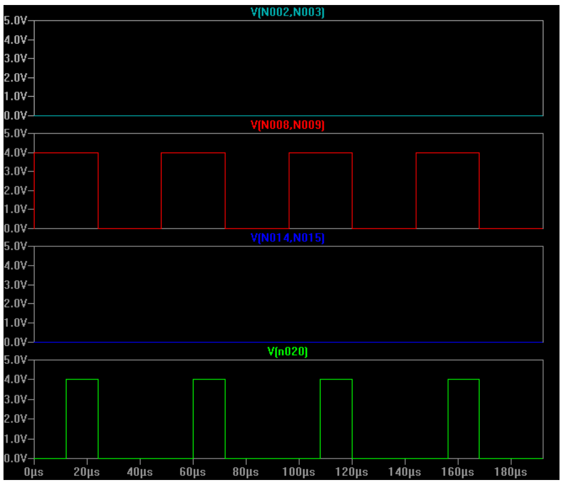
(a) Discharge phase



(b) Charge phase



(c) Cell currents



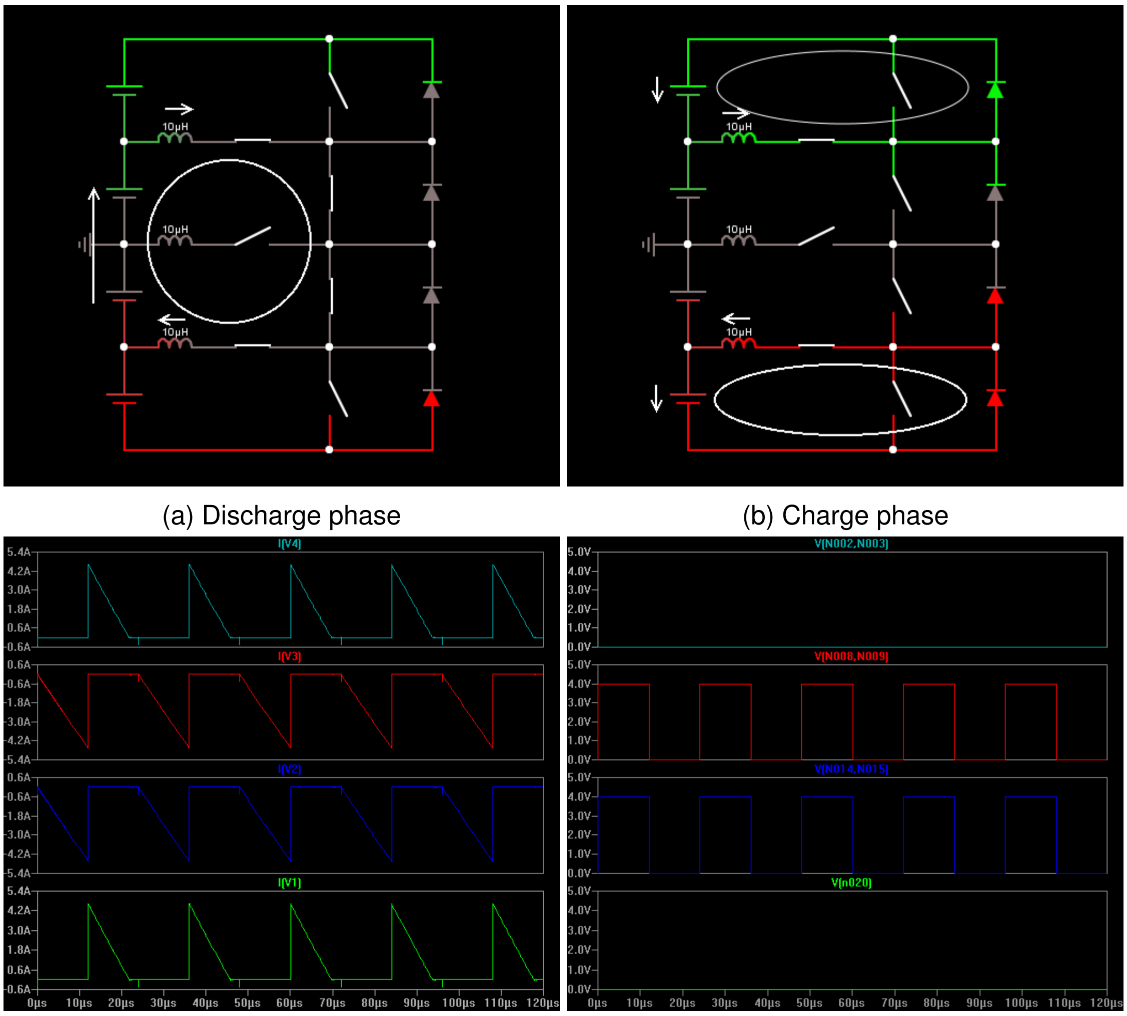
(d) MOSFET pulses

Figure 10: DCDC mode configuration

$t_D4 = 0$	$t_C4 = L_1 I_{peak} \cdot \frac{1}{V4}$	$t_{delay}4 = t_D3$
$t_D3 = L_1 I_{peak} \cdot \frac{2}{V3}$	$t_C3 = 0$	$t_{delay}3 = 0$
$t_D2 = 0$	$t_C2 = L_1 I_{peak} \cdot \frac{2}{V2}$	$t_{delay}2 = t_D3$
$t_D1 = L_1 I_{peak} \cdot \frac{1}{V1}$	$t_C1 = 0$	$t_{delay}1 = t_D3 - t_D1$
$T_{cycle} = t_D3 + t_C2 = L_1 I_{peak} \cdot \frac{2V2+2V3}{V2 \cdot V3}$		

Table 7: DCDC mode equations

2.3.7 CDDC



(a) Discharge phase (b) Charge phase

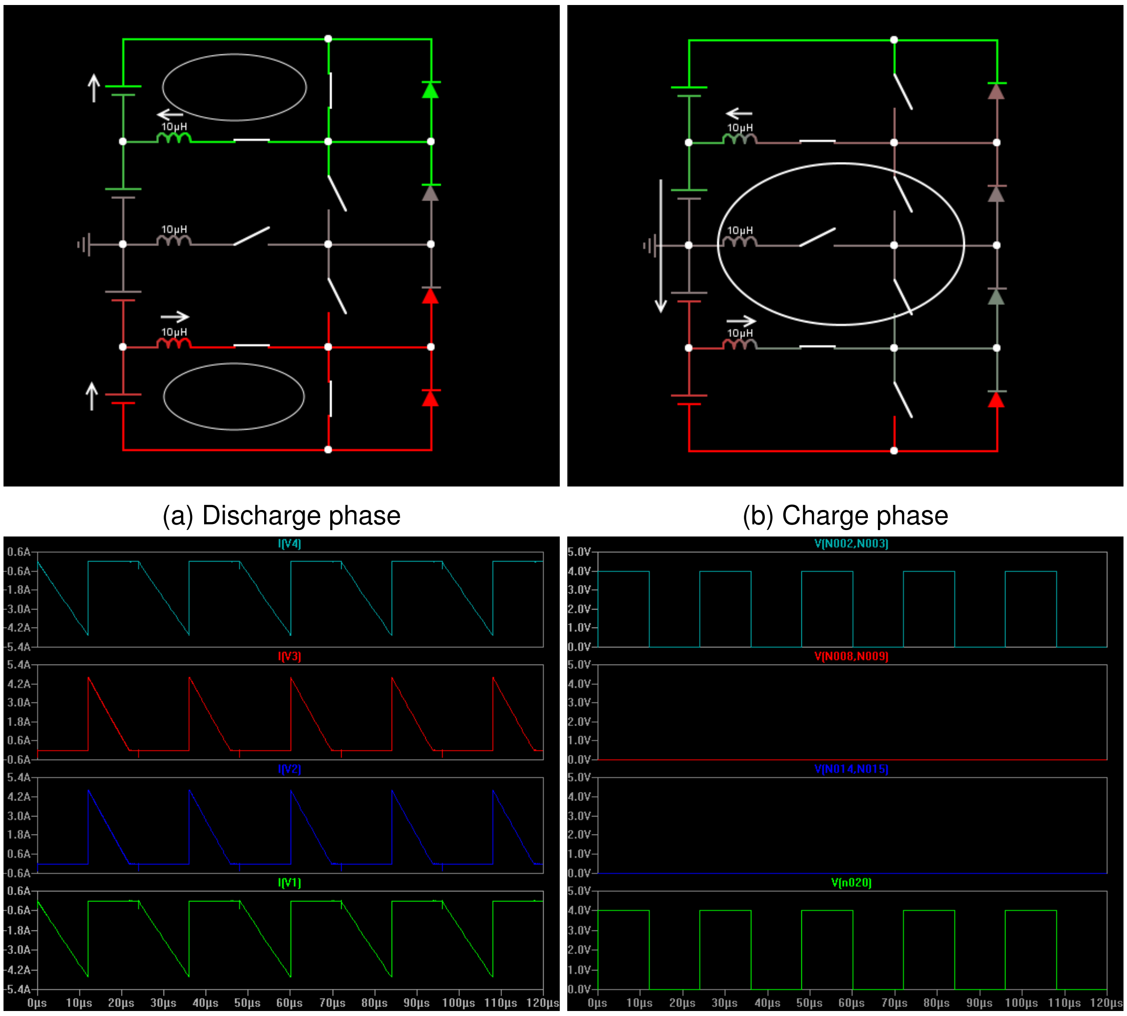
(c) Cell currents (d) MOSFET pulses

Figure 11: CDDC mode configuration

$t_D4 = 0$	$t_C4 = L_1 I_{peak} \cdot \frac{1}{V4}$	$t_{delay}4 = t_D2$
$t_D3 = L_1 I_{peak} \cdot \frac{2}{V2+V3}$	$t_C3 = 0$	$t_{delay}3 = 0$
$t_D2 = L_1 I_{peak} \cdot \frac{2}{V2+V3}$	$t_C2 = 0$	$t_{delay}2 = 0$
$t_D1 = 0$	$t_C1 = L_1 I_{peak} \cdot \frac{1}{V1}$	$t_{delay}1 = t_D2$
$T_{cycle} = runtime\ computation$		

Table 8: CDDC mode equations

2.3.8 DCCD



(a) Discharge phase (b) Charge phase

(c) Cell currents (d) MOSFET pulses

Figure 12: DCCD mode configuration

$t_D4 = L_1 I_{peak} \cdot \frac{1}{V4}$	$t_C4 = 0$	$t_{delay}4 = t_D1 - t_D4$
$t_D3 = 0$	$t_C3 = L_1 I_{peak} \cdot \frac{2}{V2+V3}$	$t_{delay}3 = t_D1$
$t_D2 = 0$	$t_C2 = L_1 I_{peak} \cdot \frac{2}{V2+V3}$	$t_{delay}2 = t_D1$
$t_D1 = L_1 I_{peak} \cdot \frac{1}{V1}$	$t_C1 = 0$	$t_{delay}1 = 0$
$T_{cycle} = runtime\ computation$		

Table 9: DCCD mode equations

2.4 Modes Overview

Considering $V1 = V2 = V3 = V4 = V$ for simplification, an overview with the modes and their switch configurations, pulse timings and cell currents can be generated (Tab. 10). Exact timings are computed during runtime.

i	$mode$	$sw1$	$sw2$	$sw3$	T	$t1$	$t2$	$t3$	$t4$	$d1$	$d2$	$d3$	$d4$
1	CDDD	on	off	off	4/3	1	1/3	1/3	1/3	1/3	0	0	0
2	DCCC	on	off	off	4/3	1	1/3	1/3	1/3	0	1	1	1
3	CDCC	on	on	off	3	1	2	1/2	1/2	2	0	2	2
4	DDCD	off	on	on	3	1/2	1/2	2	1	1/2	1/2	1	0
5	DDCC	off	on	off	1	1/2	1/2	1/2	1/2	0	0	1/2	1/2
6	DCDC	on	on	on	4	1	2	2	1	1	2	0	2
7	CDDC	on	off	on	2	1	1	1	1	1	0	0	1
8	DCCD	on	off	on	2	1	1	1	1	0	1	1	0
9	DDDC	off	off	on	4/3	1/3	1/3	1/3	1	0	0	0	1/3
10	CCCD	off	off	on	4/3	1/3	1/3	1/3	1	1	1	1	0
11	CCDC	off	on	on	3	1/2	1/2	2	1	2	2	0	2
12	DCDD	on	on	off	3	1	2	1/2	1/2	0	1	1/2	1/2
13	CCDD	off	on	off	1	1/2	1/2	1/2	1/2	1/2	1/2	0	0
14	CDCD	on	on	on	4	1	2	2	1	2	0	2	1

Table 10: Normalized timings overview; $T = \frac{T_{cycle}V}{L_1 I_{peak}}$; $t(i) = \frac{t_{D/C}V}{L_1 I_{peak}}$; $d(i) = \frac{t_{delay}V}{L_1 I_{peak}}$;

After the pulse timings have been determined for given cell voltages, inductance and peak current, the actual cell currents must be computed in order to properly control the charge balancing process. By convention, I_{peak} is the absolute value of the peak current, the discharging current I_D is negative and the charging current I_C is positive. Averaging over a cycle period we get for discharging:

$$I_D = \frac{-1}{T_{cycle}} \cdot \int_0^{T_{cycle}} I(t)dt = \frac{-1}{T_{cycle}} \cdot \int_0^{t_D} \frac{t \cdot I_{peak}}{t_D} dt = I_{peak} \cdot \frac{-t_D}{2T_{cycle}} = I_{peak} \cdot \frac{-duty_D\%}{2}$$

Similarly for charging:

$$I_C = I_{peak} \cdot \frac{duty_C\%}{2}$$

Under the same assumption that all cell voltages are equal (they would practically not be allowed to go too far apart), the duty cycles (ratio of pulse width to pulse period) are fixed for a given mode. Controlling the balancing consists of choosing the right mode and scaling the balancing currents with respect to I_{peak} which is determined by the cycle period (i.e. switching frequency).

An overview of the balancing currents can be seen in Tab. 11. Exact values are computed during runtime.

2.5 Optimum Algorithm

Based on the measured cell voltages, the best mode must be selected. In order to solve the problem, we need to model the dynamics, set a goal and an associated loss function.

i	$mode$	$I1$	$I2$	$I3$	$I4$
1	CDDD	3/8	-1/8	-1/8	-1/8
2	DCCC	-3/8	1/8	1/8	1/8
3	CDCC	1/6	-1/3	1/12	1/12
4	DDCD	-1/12	-1/12	1/3	-1/6
5	DDCC	-1/4	-1/4	1/4	1/4
6	DCDC	-1/8	1/4	-1/4	1/8
7	CDDC	1/4	-1/4	-1/4	1/4
8	DCCD	-1/4	1/4	1/4	-1/4
9	DDDC	-1/8	-1/8	-1/8	3/8
10	CCCD	1/8	1/8	1/8	-3/8
11	CCDC	1/12	1/12	-1/3	1/6
12	DCDD	-1/6	1/3	-1/12	-1/12
13	CCDD	1/4	1/4	-1/4	-1/4
14	CDCD	1/8	-1/4	1/4	-1/8

Table 11: Normalized currents overview; $I(i) = \frac{I_{D/C}}{I_{peak}}$

A battery cell can be modeled as a large capacitor with capacity C:

$$V(t) = V(0) + \frac{1}{C} \cdot \int_0^t I(t)dt$$

Because $1/C$ is relatively small, the integral can be approximated by Riemann summation with T = fixed time for which a mode (constant balancing current) is applied:

$$V[n] = V[0] + \frac{1}{C} \cdot \sum_{i=0}^{n=t/T} I[i] \cdot T$$

$$V[k+1] = V[k] + \frac{T}{C} \cdot I[k]$$

The relation applies to each of the cells so we can write it in vector form:

$$\mathbf{V}_k = \langle V1[k], V2[k], V3[k], V4[k] \rangle$$

$$\mathbf{I}_{mode[k]} = \langle I1[k], I2[k], I3[k], I4[k] \rangle$$

$$\mathbf{V}_{k+1} = \mathbf{V}_k + constant \cdot \mathbf{I}_{mode[k]}$$

Because the sum of the $I(i)$ components is null (conservation of charge, neglecting conversion losses), the sum of the voltage components is invariant. The goal is reaching the balanced voltage vector:

$$\widetilde{\mathbf{V}} = \langle \tfrac{V1+V2+V3+V4}{4}, \tfrac{V1+V2+V3+V4}{4}, \tfrac{V1+V2+V3+V4}{4}, \tfrac{V1+V2+V3+V4}{4} \rangle$$

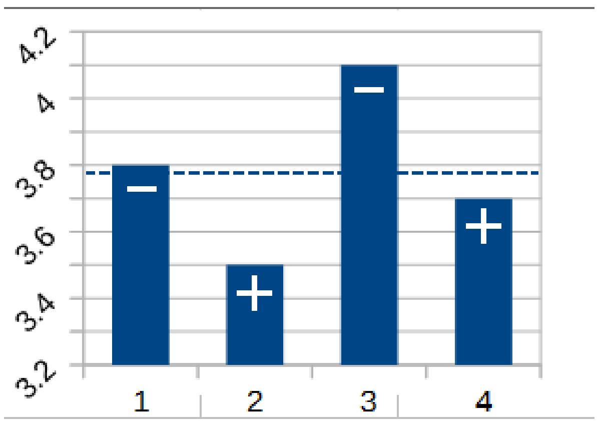
One approach is to reach the balanced state as quickly as possible. In this case the loss function is the number of steps n of duration T (which lead to a balancing time of $t = nT$) that needs to be minimized by choosing the sequence $mode[i]$. Finding the global optimum of this function would require brute forcing with 14^n computations which is impossible in terms of available computing power.

However, finding the local optimum for every step is easy and should provide a good enough solution. This involves choosing the mode which will lead to a minimum residual voltage error (example Fig. 13):

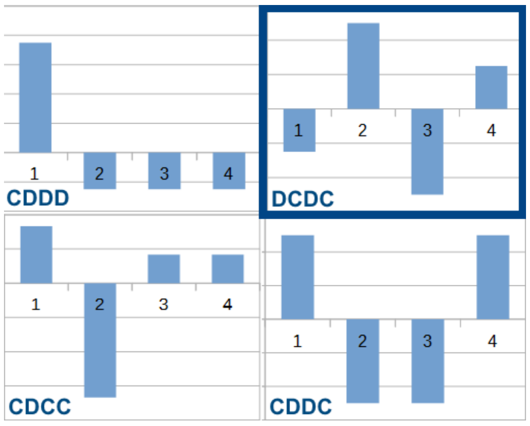
$$\text{Find } mode \text{ for which } error_{mode} = minimum, \quad mode = \overline{1, 14}$$

$$error_{mode} = \|\mathbf{V}_{k+1} - \widetilde{\mathbf{V}}\|^2 \;=\; \|\mathbf{V}_k + constant \cdot \mathbf{I}_{mode} - \widetilde{\mathbf{V}}\|^2$$

This is a form of least squares optimization. The exact value of the $constant$ should not affect the result of the algorithm as long as it is small enough. In reality $constant = \frac{T}{C}$ is very small but in floating point arithmetic this would pose precision problems, so we can take a value of 10^{-3} for example.



(a) Cell voltages



(b) DCDC current mode is optimum

Figure 13: Example voltage and current vectors

Practically, the control loop consists of:

- Read voltage vector $\mathbf{V}_k$ from ADC sensor
- Compute average voltage vector $\widetilde{\mathbf{V}}$
- Compute the mode currents (as in Tab. 11)
- Compute $error_{mode}$ for all 14 modes
- Choose $mode$ for minimum error and apply it
- Wait a period of time T and repeat the cycle (read voltage again, etc.)

All currents need to be rescaled back to I_{peak} and this value must be computed such that the system is stable and does not oscillate (more on this later).

3 Microcontroller Firmware

In this section the firmware controlling the A/D Converter and the Timer Hardware is developed and the performance of the microcontroller is measured. A small and breadboard-friendly PCB hosting the MCU is used (Fig. 14).

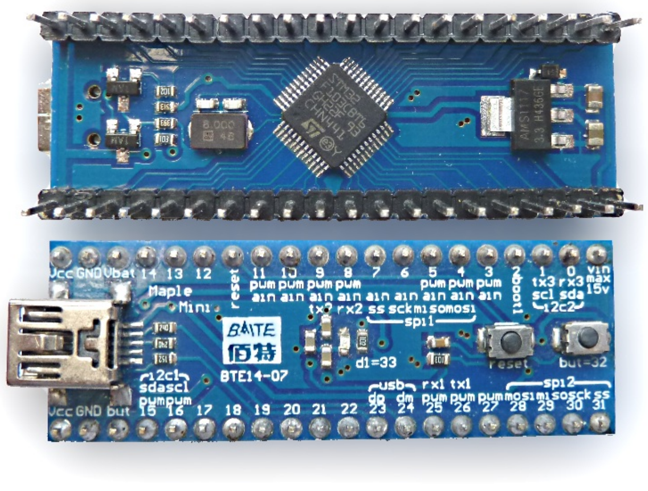


Figure 14: STM32F103 CBT6 development board

Basic functions can be easily implemented and uploaded to the MCU via USB using Arduino IDE (1.6.5) and a STM32 hardware core (available at http://stm32duino.com/). For the more advanced features needed, direct register access code is inlined with arduino code, according to the registers description in the STM32 Reference Manual.

3.1 ADC Accuracy

The maximum sample rate of the ADC is 1MHz, however the voltage of the battery cells cannot significantly fluctuate so fast therefore a sampling period in the range of milliseconds is enough. Single conversions using analogRead() are made.

The resolution of the ADC is 12bit i.e. 4096 steps from 0 to 3.3V (default V_{ref}). In order to increase the resolution and the Signal-to-Noise Ratio (SNR), oversampling with averaging is used. By taking 2^{2n} successive samples and averaging them, n bits of resolution are gained [10]. In this application we average 256 samples to gain 4 additional bits and obtain a 16bit ADC (65535 steps from 0 to 3.3V means $50\mu V$ step). Input samples are placed in a ring buffer such that the last 256 samples can be efficiently summed up with a recursive formula.

$$sum[n] = sum[n-1] + sample[n] - sample[n-256]$$

After practically eliminating quantization errors, the problem of ADC offset and differential linearity error still exists and it can be up to ± 5 LSB units [12] meaning 4mV which would be the major bottleneck in achieving satisfactory ADC accuracy. In order to minimize these errors, we use linear interpolation for the voltage range of interest.

$$val_{calibrated} = ref_1 + (val_{raw} - mref_1) \cdot \frac{ref_2 - ref_1}{mref_2 - mref_1}$$

where ref_1 and ref_2 span the range of interest and $mref_1$ and $mref_2$ are the measured values of the ref_1 and ref_2 references during the calibration procedure.

Computing the voltage from the ADC result is straightforward:

$$voltage_{mV} = \frac{3300 \cdot val_{ADC}}{4096}$$

A set of 10 measurements was made by sending the voltage computed with the aforementioned techniques via USB to display on a serial monitor (Tab. 12). The calculation has been calibrated using 1000mV and 3000mV as voltage range endpoints, referenced from the output of a potentiometer voltage divider adjusted to value within 0.1mV with the aid of a multimeter.

Reference [mV]	1000	2000	3000
	999.4	2000.4	3000.4
	999.6	2000.2	2999.7
	1000.01	1999.9	3000.5
	999.3	2000.2	2999.4
Samples [mV]	998.9	2000.8	2999.8
	999.5	2000.2	3000
	999.7	2000	3000.3
	999	1999.8	2999.7
	1000.02	20001.2	3000.2
	999.2	2001	3000.1
Average [mV]	999.46	2000.37	3000.01
Max error [mV]	-1.1	1.2	-0.6
Avg error [mV]	-0.54	0.37	0.01
σ error [mV]	0.38	0.47	0.35

Table 12: ADC Measurements

Although linear interpolation works better for smaller intervals, we have obtained satisfactory results even for an interval spanning 60% of the ADC range. With an error of under 1mV, the real error when measuring cells through a 6:1 divider ($3.3V \cdot 6 = 19.8V > 16.8V = 4.2V \cdot 4$) would be 6mV which is acceptable for detecting significant differences between cells.

3.2 PWM Speed and Resolution

The microcontroller has 4 timers for general use (besides a systick timer used to keep the on-board time and 2 watchdog timers optionally used to ensure system integrity). Each of the 4 timers is a complex independent hardware block with various counting, pin input/output and interrupt features.

Timer 2, Timer 3 and Timer 4 structure is shown in Fig. 15:

- Counter Register (CNT) is a 16bit register that counts up/down from a clock source, which can be the main clock (72MHz) divided by a 16bit clock prescaler (Fig. 17).
- Auto-Reload Register (ARR) is a 16bit register that defines the overflow value of CNT i.e. when CNT == ARR, CNT is reset to 0 (in up-counting mode) (Fig. 17).
- 4x Capture/Compare Registers (CCR1-4) are 16bit registers that can trigger an interrupt or switch the corresponding output pin when CNT == CCRx (Fig. 18).

- Master/Slave Trigger Controller which defines the signals that start/stop the counter and the signals generated by counter events. This can be used for synchronisation between the timers (more on this later).
- Various Enable/Control/Mode Registers that define the interaction between the timer components.

Timer 1 has additional features and structure is shown in Fig. 16:

- Each of the 4 Output Compare Channels has a complementary output with configurable dead-time insertion.
- Output Channels can be reset to a preconfigured state on an external pin event (BKIN). This is useful in transistor drives which have an overcurrent detection circuit that would signal the microcontroller through the BKIN pin.
- Corresponding Enable/Control/Mode Registers for the additional functions.

All registers can be read/written by the CPU for configuring the desired mode, after which the timers work in parallel as independent hardware entities while the CPU is completely free to do other processing jobs. The timers will be used to generate 4 independent pulses of arbitrary frequency, width and phase-shift, one for each of the MOSFETs controlling the inductor circuit (4 cells). PWM mode will ensure precise and glitch-free timing that is required for efficient and safe power transfer.

Each timer has 4 output compare channels and pins so there are 16 PWM outputs but not all of them are useful for the application due to the requirement of arbitrary phase-shift of the PWM pulses – within each timer, the 4 compare outputs are either edge aligned (rising edge for positive output polarity or falling edge for negative output polarity) or center aligned. Therefore arbitrary phase-shift between the channels of a timer is not possible.

However, the timers are can be offset relative to each other with any value by writing to their CNT registers and have 1 output channel from each timer. Since the output channels would be edge aligned with their counter and the ARR value would be the same for all timers (charge transfer in periodic cycles), the outputs would maintain the offset of their timers until the next CNT overwrite.

Careful consideration needs to be taken when overwriting the CNT registers. The CPU has no atomic write for all of them so they would be written one at a time. If the prescaler is small then the speed at which CNT is counting (updated by counter circuitry) is on par with the speed of the CPU (which can have unpredictable overheads depending on compiler optimizations) therefore this naive method is not reliable.

The solution is to pause the counters, update their value and restart them at the same time. Since starting the counters is done by the CPU by setting the CEN (counter enable) bit in their control register, the same problem of non-atomicity remains. In a naive implementation, they would be started sequentially and by the time the last timer is started, the first one has already advanced in counting by an unpredictable value. Fortunately, the timers are not completely independent and they share trigger lines. By configuring the Master/Slave Trigger Controllers such that the enable signal on one master timer is routed as enable signal for the other slave timers, starting all timers at the same time is possible by only setting the CEN bit in the control register of the master timer.

The HardwareTimer class from the arduino hardware core provides methods for setting

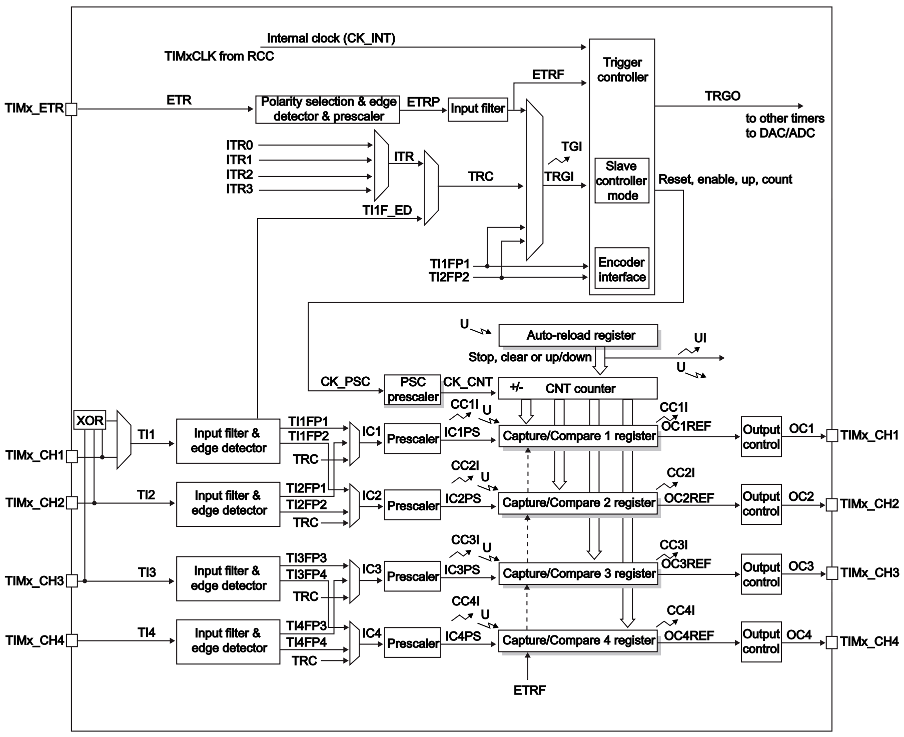


Figure 15: STM32F10x General-Purpose Timer Architecture [11]

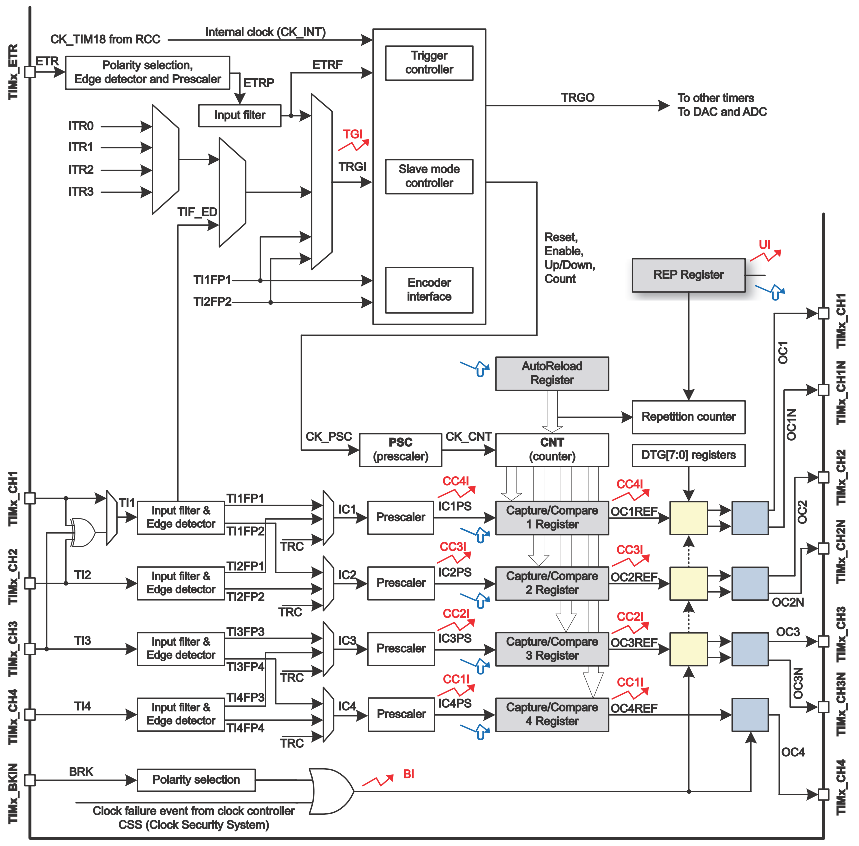


Figure 16: STM32F10x Advanced Timer Architecture [11]

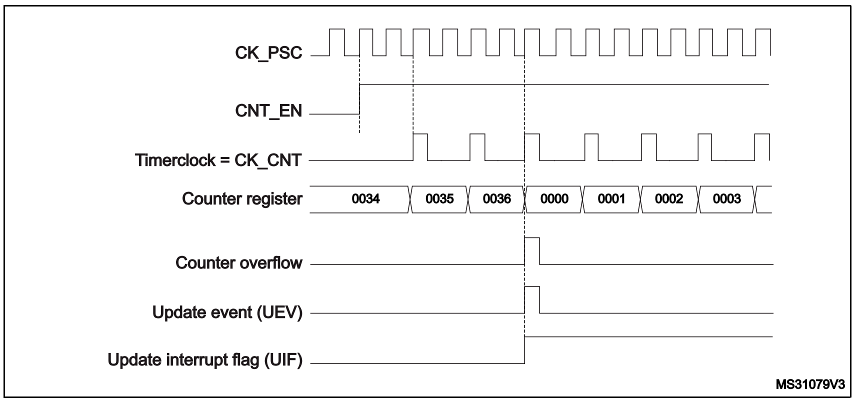


Figure 17: Basic counter operation, clock PSC=2, ARR=36 [11]

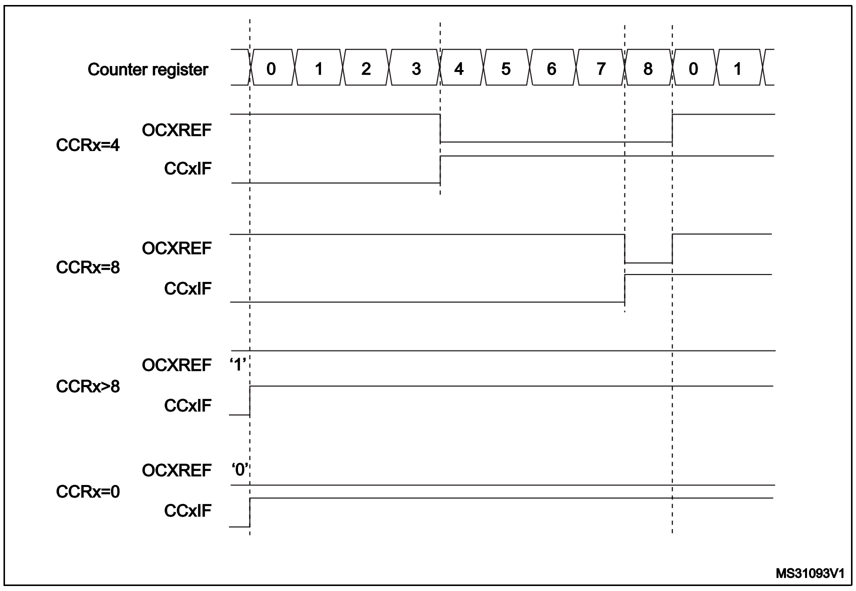


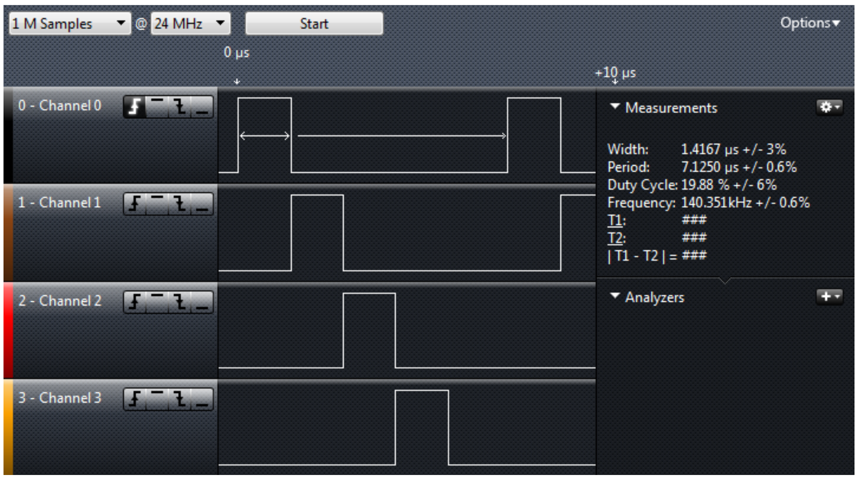
Figure 18: Edge-aligned PWM, ARR=8, different compare values [11]

the PSC, CNT, ARR and CCRx registers but no method for timer synchronisation, therefore direct access to the Control Register 2 (CR2) and Slave Mode Control Register (SMCR) has been inlined.

Operation of the timers has been verified and measured with a 24MHz USB Logic Analyzer. Because the minimum period that can be measured is 42ns, long enough duty cycles have been applied and verified to comply within a reasonable error margin. Each analyzer channel is the output from channel 1 of each timer (Fig. 19).

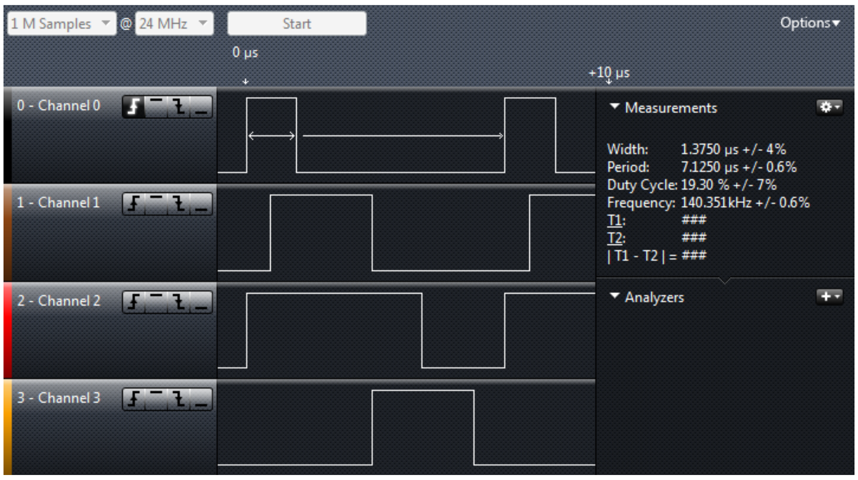
The closed-loop feedback control for balancing the cells means that the output control sequence is modified according to cell voltage, which is expected to refresh once every few milliseconds. The dead-time during a PWM sequence refresh is illustrated in Fig. 20. Since the dead-time is only about $10\mu s$ with a period of no less than a few milliseconds, its influence on the system operation is negligible.

Now the system is characterized such that the timing constrains can be used later for deriving the control scheme. All timer registers are 16bit, therefore the dynamic range is 65536:1. With no prescaler (PSC=1) the timer is driven with 72MHz therefore the



Width: 1.4167 µs +/- 3%
Period: 7.1250 µs +/- 0.6%
Duty Cycle: 19.88 % +/- 6%
Frequency: 140.351kHz +/- 0.6%
T1: ###
T2: ###
| T1 - T2 | = ###

(a) Consecutive PWM pulses



Width: 1.3750 µs +/- 4%
Period: 7.1250 µs +/- 0.6%
Duty Cycle: 19.30 % +/- 7%
Frequency: 140.351kHz +/- 0.6%
T1: ###
T2: ###
| T1 - T2 | = ###

(b) Arbitrary PWM pulses

Figure 19: Example PWM configurations

time step is $1/72MHz = 13.8ns$. Such a small step means that we can adjust the shift between different pulses fall and rise edges to be distanced enough with respect to the fall and rise times (10-100ns) of the MOSFETs and avoid current shoot-through. The maximum interval before overflow is $65536/72MHz = 910\mu s$.

	Min [ns]	Max [ns]	Step [ns]
period	13.9	$910 \cdot 10^3$	13.9
width	0.0	$period$	13.9
offset	0.0	$period - 13.9$	13.9

Table 13: PWM pulse period / width / offset ranges and resolution

Tab. 13 provides the important characteristics of the PWM. Everything is relative to the clock prescaler, therefore having a prescaler of N instead of 1 will multiply all the values in the table by N. Most likely a prescaler of 1 will be used because the $910\mu s$ period is already longer than enough for inductor values under $100\mu H$ and desired currents under $10A$. *Rule of thumb: $1V$ across a $1\mu H$ inductor will increase the current with $1A$ per microsecond.*

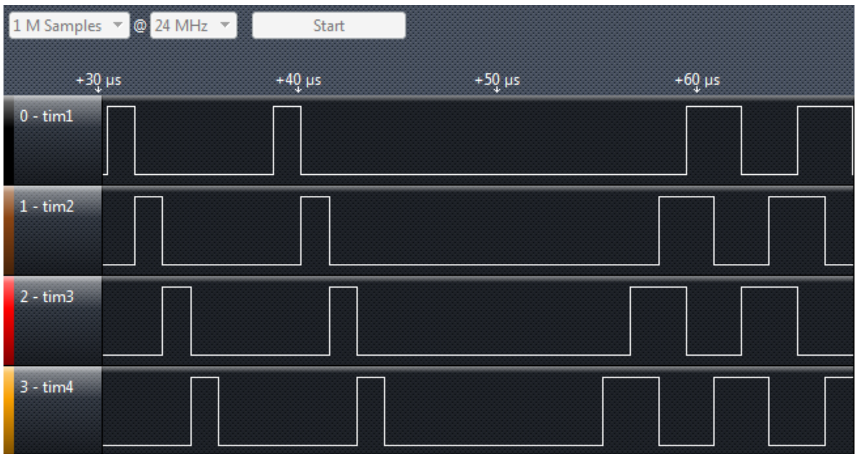


Figure 20: PWM sequence change between 2 arbitrary states

4 Circuit Implementation

4.1 Practical details

Up to this point, only a minimalist circuit has been displayed and simulated in order to focus on the core functionality. Aside from additional trivial components required to implement the circuit such as power circuitry associated with the controller and voltage dividers for sensing, the MOSFET gate drivers represent a crucial component that needs some attention. Previous simulations used parametrized power supplies for driving the MOSFET gates, each with complete common-mode voltage separation. In reality, these ideal power supplies are representing the microcontroller outputs which are all referenced to ground, therefore there is no common-mode separation by default. Since the transistor source (S) terminals are either floating in the circuit or getting swinged high and low by nearby transistors, there needs to be common-mode separation between all the gate drives. A gate driver is a circuit that provides common mode separation between a microcontroller output and a MOSFET source-gate port. It is often also used to buffer a weak MCU output with a high current drive that switches the gate voltage faster in order to minimize switching losses.

Specialized gate driver IC's have been studied, however they were found to cost more than the main switching transistors and inductors combined, therefore it was decided they cannot be part of a low cost solution.

A simple gate driver circuit (often used in low cost brushless motor controllers) can be made from a capacitor, a diode, a BJT transistor, and a couple of resistors. In the switched-on mode the capacitor keeps the gate charged above the threshold voltage, while in the switched-off mode the open-collector BJT transistor is activated, pulling the gate voltage to an absolute low. During the switched-off mode, the diode is positive-biased and keeps the capacitor charged and ready to drive the gate high after the transistor is released. After the MOSFET source terminal swings up, the diode-becomes reverse biased in order to maintain the capacitor charge. See the updated circuit with gate drivers in Fig. 21.

Capacitor and resistor values have been optimized empirically for the working frequency to maximize efficiency. This involved finding a trade-off between shorter rise/fall times

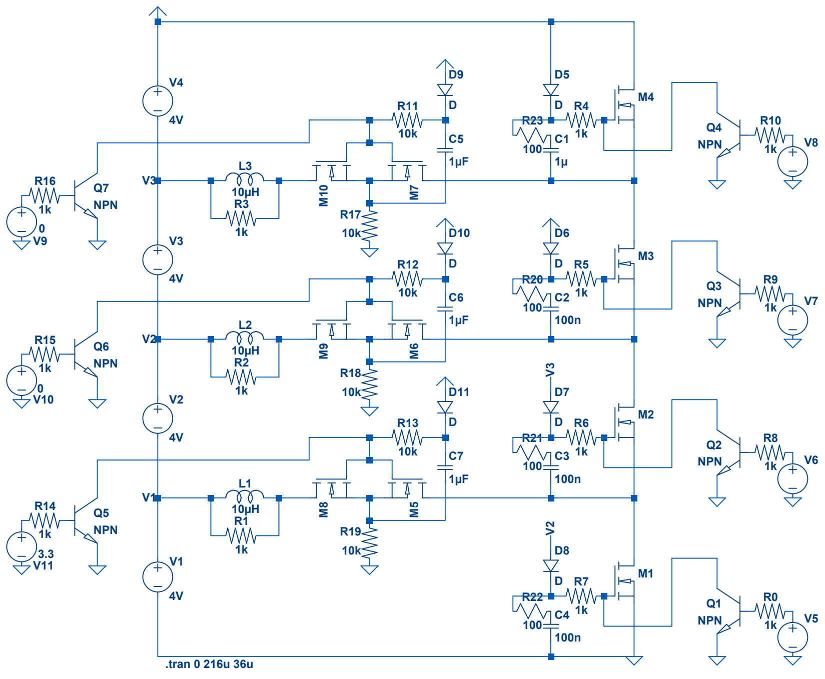


Figure 21: Ground-referenced power supplies represent actual controller outputs

with higher leakage currents and longer rise/fall times with higher switching losses. Leakage currents represent wasted power in the driver resistors while switching losses represent wasted power in the MOSFETS when they are not fully turned on and have a high series resistance.

4.1.1 Key Components

With a 5A final peak current requirement (for approximately 1A balancing currents), parametrized searches on farnell.com and digikey.com were made in order to find the best available parts.

Desirable inductor properties:

- High saturation current
- Low series resistance
- Low cost, weight and size

Desirable MOSFET properties:

- Low on-state resistance (correlated with high current capability)
- Short rise and fall times
- Low threshold voltage
- Low cost

After numerous trade-off decisions, the following key parts were selected:

- MGV1004100M-10 inductor, $10\mu H,\ 36m\Omega,\ 6.8A,\ 0.5-1.5EUR$

- NTMFS4926NT1G n-channel MOSFET, $5.6 m\Omega$, $44A$, $30V$, $5{-}40ns$, $0.15{-}0.25EUR$

While the exact cost of the circuit is difficult to tell as it heavily depends on the volume of parts purchased, we can safely state that the bill of materials is under 15 EUR even at a very low production output.

4.2 PCB Layout

The layout has been designed in Altium CircuitMaker - a free, community based version of Altium Designer (Fig. 23 and Fig. 24). The schematic has also been re-written to include all the necessary details (Fig. 22).

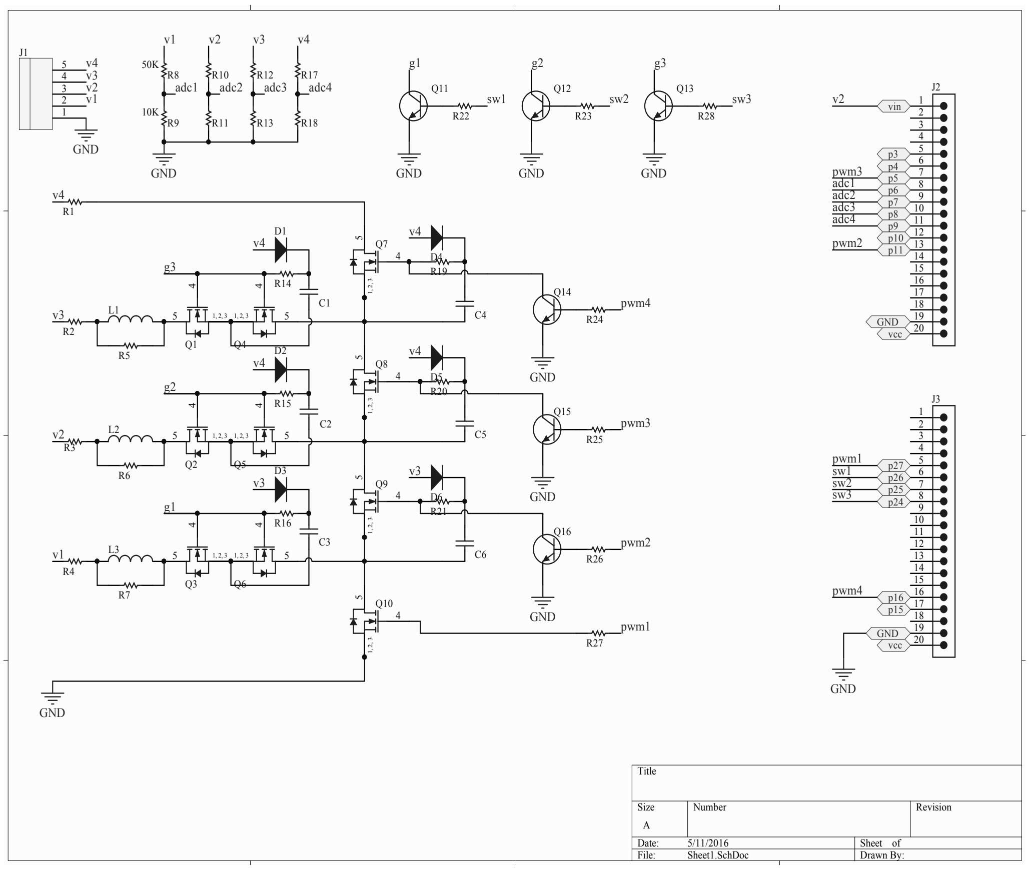


Figure 22: PCB schematic

4.3 Assembly

The PCB was CNC milled and the components manually soldered (Fig. 25a and Fig. 25b).

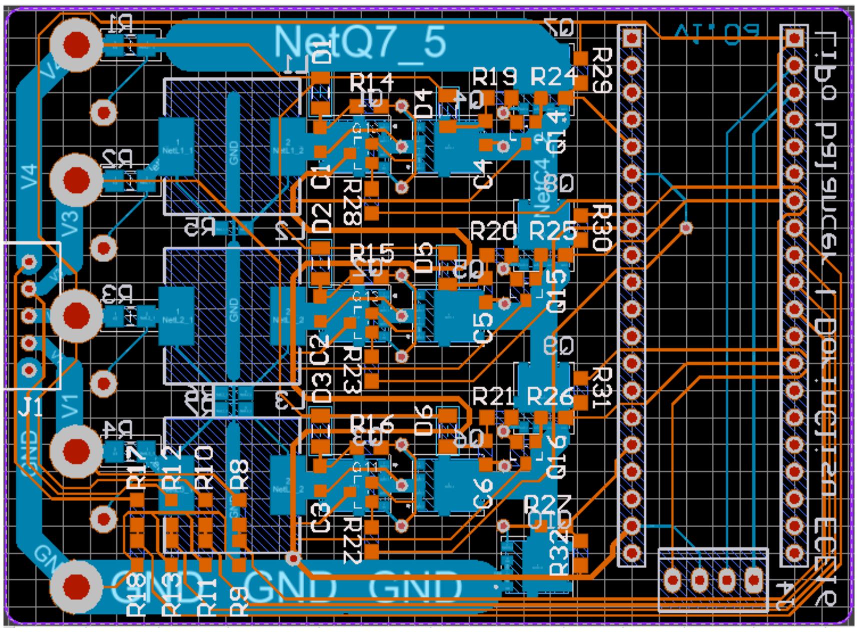
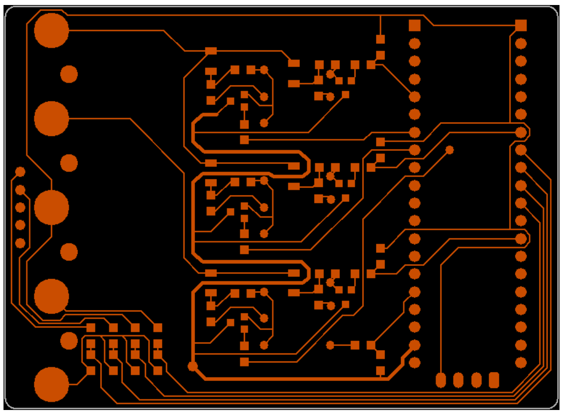
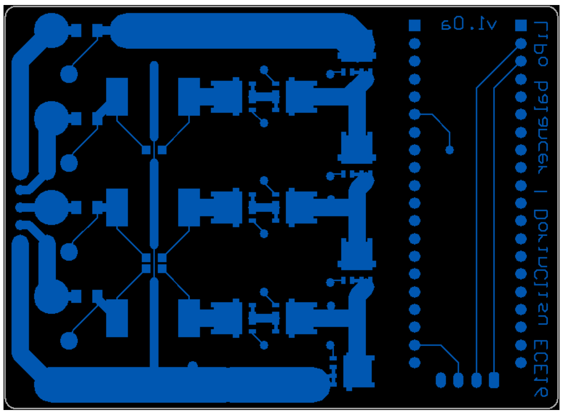


Figure 23: PCB project view

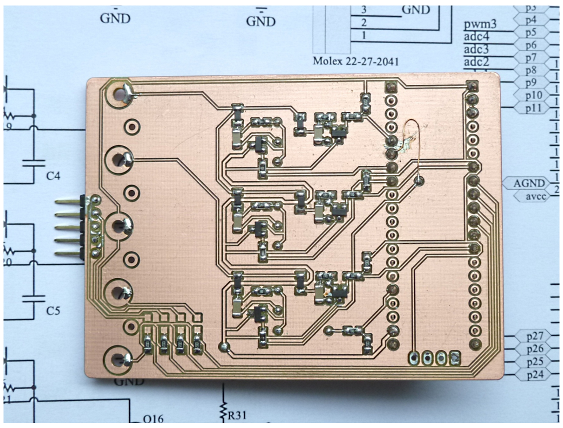


(a) top

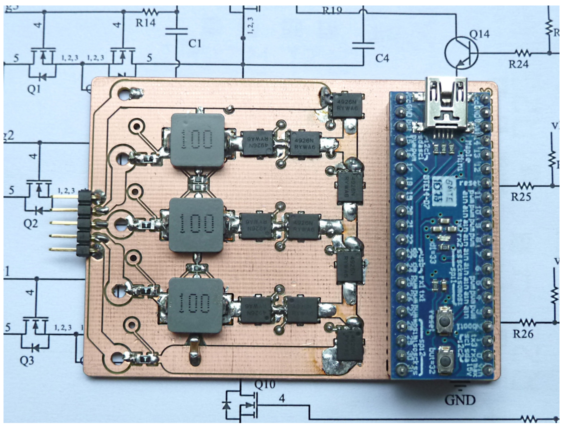


(b) bottom

Figure 24: PCB gerber masks



(a) top



(b) bottom

Figure 25: Assembled PCB

5 Performance Testing

5.1 Static measurements

Measurements were performed with 4 current limited power supplies acting as battery cells. A power resistor was put in parallel with the balancing circuit, draining a constant offset current in order to prevent negative current injection into the power supplies (Fig. 26). Gate drive pulses were probed with the oscilloscope in Fig. 27. Given that the voltages were all adjusted and regulated to 4V, the port currents are equivalent to transferred power, so we can compute the power efficiency as:

$$\eta = \frac{\sum I_{charge}}{\sum I_{discharge}}$$

Average currents were measured in each of the 4 ports in two different modes and at different duty cycles (Fig. 14). The measured power efficiency is lower than the simulated values in the 70-85% interval. The cause for this are switching loses at levels higher than expected, due to a number of reasons that required additional time to correct:

- gate drive transistors were not dimensioned properly and overheated
- gate drive diodes overheated for similar reasons
- gate drive pull-up resistors overheated

The overheated junctions threw the characteristics outside their intended range, resulting in increased MOSFET gate rise and fall times that determine switching losses.

Although parts rated for higher currents can solve the overheating problem and bring the efficiency closer to the simulation, a better solution would be to use transistor-transistor gate drivers instead of transistor-resistor charge pump gate drivers. The new solution would involve multiple BJT transistors for each gate driver.

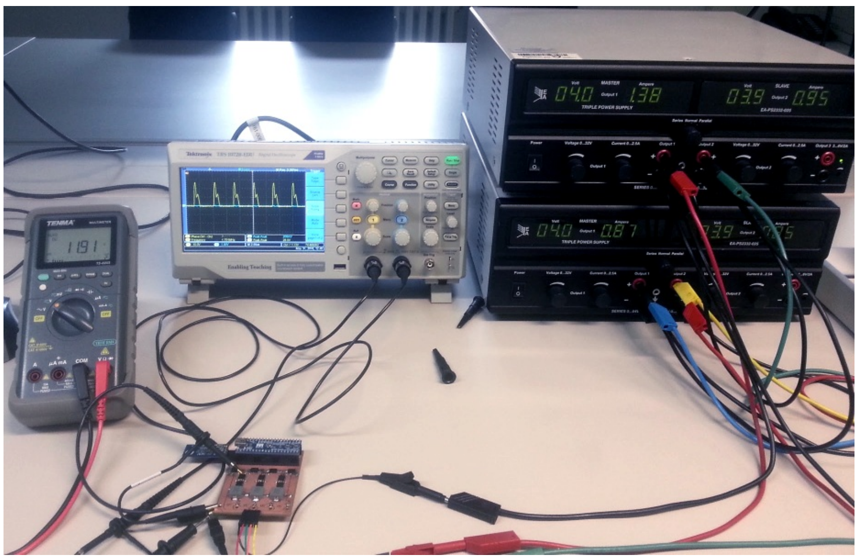
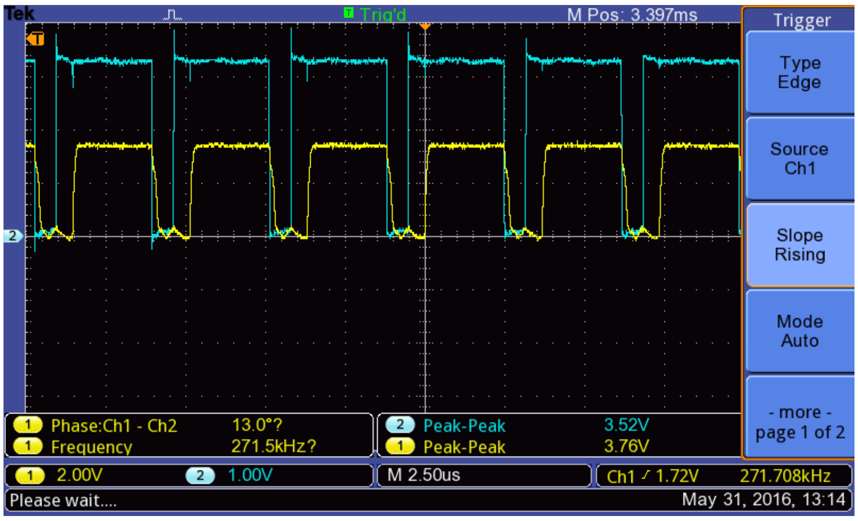


Figure 26: Lab measurement setup



Phase:Ch1 - Ch2 13.0°?
Frequency 271.5kHz?
Peak-Peak 3.52V
Peak-Peak 3.76V
2.00V 1.00V M 2.50us Ch1 ⟋ 1.72V 271.708kHz
Please wait.... May 31, 2016, 13:14

Figure 27: Control pulses with complementary activation and deadtime

i	$mode$	$I1$	$I2$	$I3$	$I4$	I_c	I_d	η
1	DCCC	-0.39	0.05	0.05	0.06	0.16	0.39	41%
2	DCCC	-0.90	0.14	0.13	0.15	0.42	0.90	46%
3	CDDC	0.11	-0.20	-0.20	0.11	0.22	0.40	55%
4	CDDC	0.41	-0.69	-0.69	0.41	0.82	1.38	59%

Table 14: Balancing currents

6 Conclusions

Balancing currents up to 90% of the designed values have been achieved at efficiencies of 40-50% compared to 70-80% in the simulation. Future work can apply certain improvements to the implementation and achieve the intended efficiency while also possibly extend the controller logic to support more than 4 battery cells. Despite providing only a rough analysis, the thesis proved that the new balancing topology is a promising low-cost solution for efficiently balancing Li-Po battery packs of up to 4 cells, and can be considered for further development.

A APPENDIX

A.1 Firmware code

https://gitlab.com/dorinclisu/active-cell-balancing/tree/master

A.2 Acronyms

A/D	Analog/Digital
ADC	Analog to Digital Converter
BJT	Bipolar Junction Transistor
CAN	Controller Area Network
CPU	Central Processing Unit
FET	Field Effect Transistor
GPIO	General Purpose Input Output
I/O	Input/Output
IC	Integrated Circuit
I2C	Inter-Integrated Circuit Interface
LSB	Least Significant Bit
MOSFET	Metal Oxide Semiconductor FET
MCU	Microcontroller
PCB	Printed Circuit Board
PWM	Pulse Width Modulation
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
USART	Universal Sync/Async Receiver/Transmitter
USB	Universal Serial Bus

References

[1] Pablo Cassani, *Modeling, Design, and Implementation of a Novel Battery Cell Equalizer for Electric, Hybrid Electric, and Plug-In Hybrid Electric Vehicles*, Concordia University, Canada, 2009.

[2] Haifeng Dai, Xuezhe Wei, Zechang Sun, Daizhuang Wang, *A novel dual-inductor based charge equalizer for traction battery cells of electric vehicles*, International Journal of Electrical Power & Energy Systems, Vol.67, 2015.

[3] M. Zahran, *Charge Equalization Unit for a NiCd Battery of Small Earth Observation Satellite EPS Simulation*, 7th WSEAS International Conference on Power Systems, Beijing, 2008.

[4] Matteo Zanibellato, *Analysis of Charge Balancer System*, University of Applied Sciences Regensburg, Germany, 2012.

[5] Carl Bonfiglio, Werner Roessler, *A Cost Optimized Battery Management System with Active Cell Balancing for Lithium Ion Battery Stacks*, IEEE 978-1-4244-2601-0/09, 2009.

[6] Sihua Wen, *Cell balancing buys extra run time and battery life*, Texas Instruments Analog Applications Journal, 2009.

[7] James Moran, *PowerPump^{TM}Balancing*, Texas Instruments Application Report, 2009.

[8] Stanislav Arendarik, *Active Cell Balancing in Battery Packs*, Freescale Semiconductor Application Note, 2012.

[9] Linear Technology Corporation, *High Efficiency Bidirectional Multicell Battery Balancer*, LTC3300-1 Datasheet, 2013.

[10] Atmel Corporation, *AVR121: Enhancing ADC resolution by oversampling*, Rev. 8003A-AVR-09/05.

[11] STMicroelectronics N.V., *STM32F103xx advanced ARM®-based 32-bit MCU Reference Manual RM0008*, DocID13902 Rev 16, 2015.

[12] STMicroelectronics N.V., *STM32F103xx Datasheet*, DocID13587 Rev 17, 2015.

[13] Everett Rogers, *Understanding Buck-Boost Power Stages in Switch Mode Power Supplies*, Texas Instruments Application Report, SLVA059A - March 1999.