OTH REGENSBURG

Getting Started With *DCDCbuck* Board *Rev. 12*

Prof. Dr. Martin J. W. Schubert, Electronics Laboratory,
OTH Regensburg, Regensburg, Germany

Getting Started With *DCDCbuck* Board

Abstract. This communication presents the *DCDCbuck* converter board, which is used as daughter board to *Terasic's DE1-SoC* board.

1 Introduction

1.1 Objectives

This document introduces to the usage of the *DCDCbuck* board, performing DC/DC step-down conversion as daughter board of Terasic's *DE1-SoC* board. With an A/D converter available, the *DCDCbuck* board can also be used with other *DEx* boards with compatible user header, e.g. *DE2, DE2-70, DE2-115.*

1.2 Assumed Test Environment

1.2.1 *DE1-SoC* Hardware

This document assumes that you are familiar with the *Terasic's* [1] *DE1-SoC* board [2] or a similar *DEx* board with the same general-purpose input/output (GPIO) user header. The version of your *DE1-SoC* board can be identified at [3]. *DE1-SoC* board revisions *F* and *G* differ in a printed company label only. To get this information from the internet, navigate your browser to [4] and download *DE1-SoC_v.5.1.2_HWrevF_SystemCD.zip* [5] or a later version. It contains amongst other things important documents such as *DE1-SoC User Manual* [6] and *Schematic* [7]. On the computer system of OTH Regensburg you will also find the CD on drive K:\Sb\ [8]. Do not use any other manual revision to follow this documentation. The differences may be considerable.

1.2.2 *Quartus* [12] and *ModelSim* [13] Software Tools

It is assumed that you have *Intel's Quartus Lite [9]* software available, actual version in August 2022 is *Quartus 21.1*. To download this freeware for your private PC you have to sign in at *Intel*[14]. At OTH Regensburg's PC pools of faculty *EI* this software is installed. At faculty *EI,* also *Quartus II 8* is installed supporting the older *DE2* boards with *Cyclone II* FPGAs, which are no more supported for *Quartus* revision ≥ 13.1.

1.3 Contents

The organization of this document is as follows:

Section **1** is this **introduction**,
Section **2** presents the ***DCDCbuck*** board **hardware**,
Section **3** suggests some **tests and basic characterization** methods of the *DCDCbuck* board,
Section **4** details some measurement possibilities using the *ARM* core on the *FPGA*,
Section **5** draws relevant **conclusion** and
Section **6** offers **references.**
Section **7** offers some realization details, typically ignored during the standard practical training

1.4 Acknowledgements

The author would like to thank *Terasic Technologies* [1] Florian Schwankner [10] and Alexander Forster [11] for their support and authorization to use copies of their documents.

2 The *DCDCbuck* Daughter Board Hardware

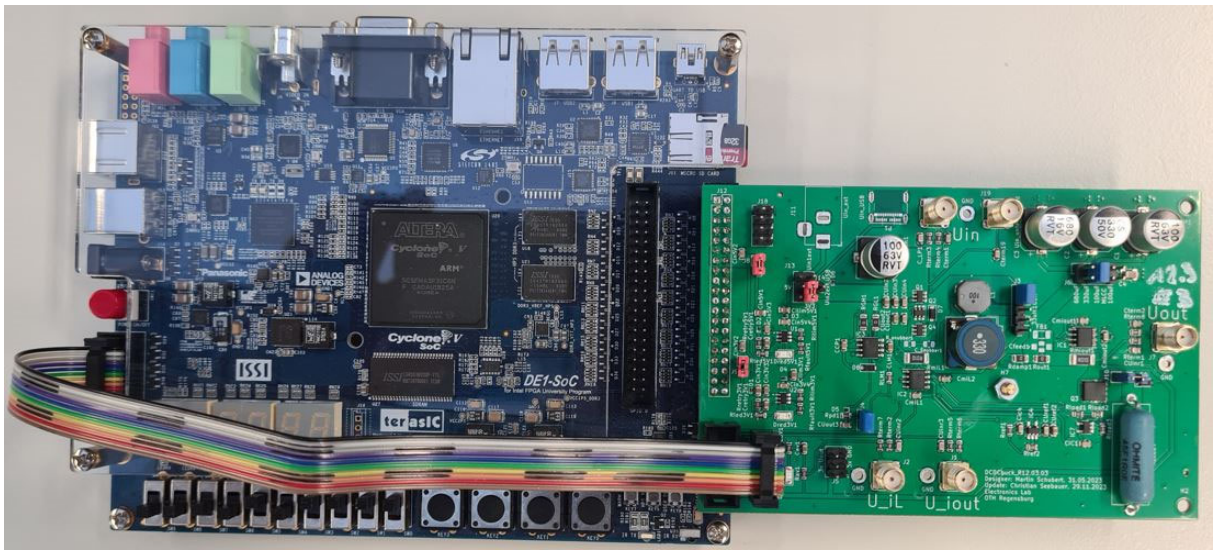


Fig. 2.1: *DE1-SoC* board (left) with plugged-in *DCDCbuck_R12.03* daughter board (right).

Fig. 2.1 shows the *DE1-SoC* board from Terasic [1-8] with *DCDCbuck_Rev10.02.06* daughter board fabricated by Florian Schwankner [10] in the Electronics Laboratory. The ribbon cable connects the input of the *DE1-SoC* on-board's *LTC2308* ADC [15] with output pins of the *DCDCbuck* boar

Bill of Materials

- Terasic's DE1-SoC board [1-8]
- DC/DC buck converter board labeled *DCDCbuck_Rev10.02.06* [10]
- 4-input-channel oscilloscope DSO-X 2024A 200MHz
- 1x 10-wire ribbon cable with 10-pin plugs of [16], (selfmade in OTH Rgbg. Electronics Lab)
- 2x cable BNC(male) – to – pin [17]
- 4x cable BNC(male) – to – SMA(male), ca. 50cm [18]
- 10x jumper [19]
- Multimeter featuring voltmeter, amperemeter and ohmmeter

(a) Board photo, jumpers:
red: power,
blu: settings,
grn: optional

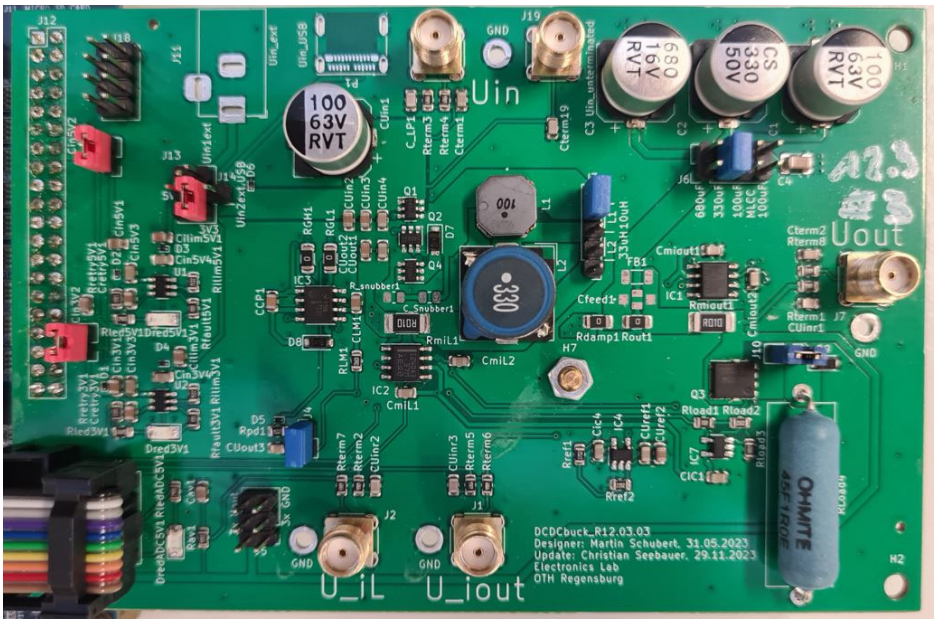


Fig. 2.2: daughter board *DCDCbuck Rev12.03.03*

Bord numbering scheme: *Rxx.yy.zz*, with
xx: Design/Designer, *yy*: No. of fabricated board, *zz*: No. of schematics update of design yy.

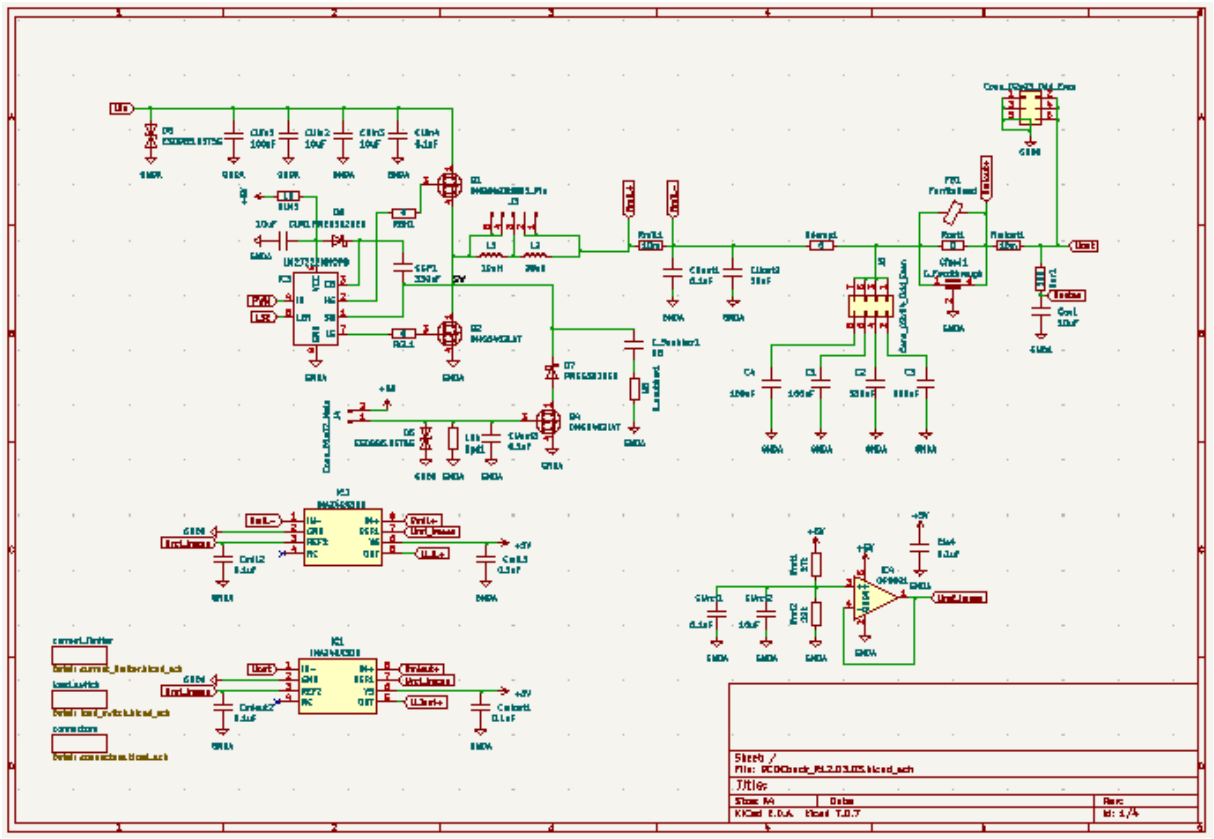
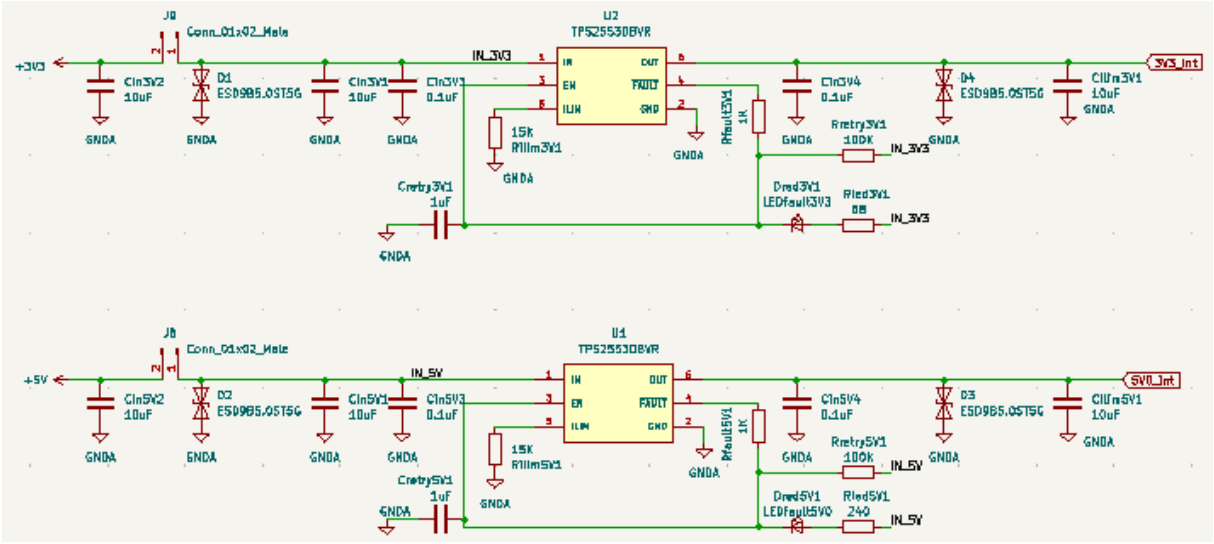
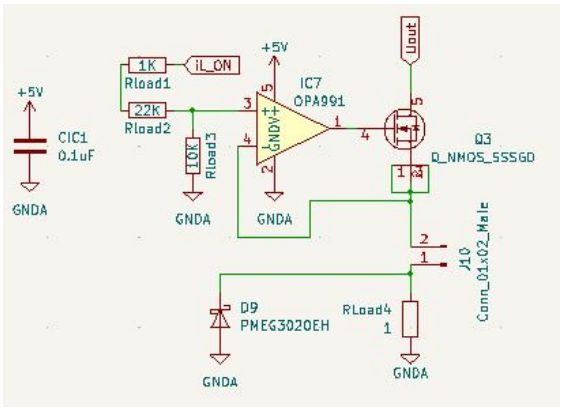


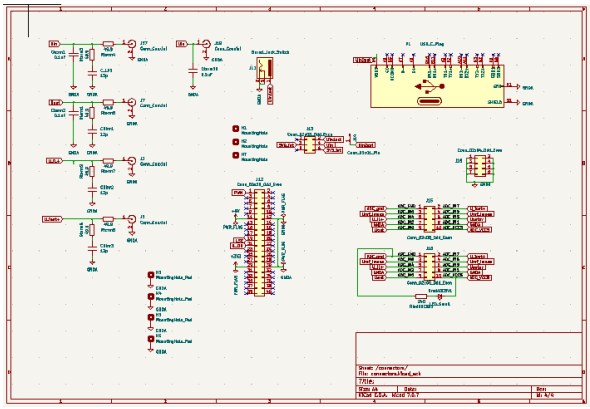
Fig. 2.3(a): *DCDCbuck_R12.03* board top level schematics (in *KiCad* software [19])



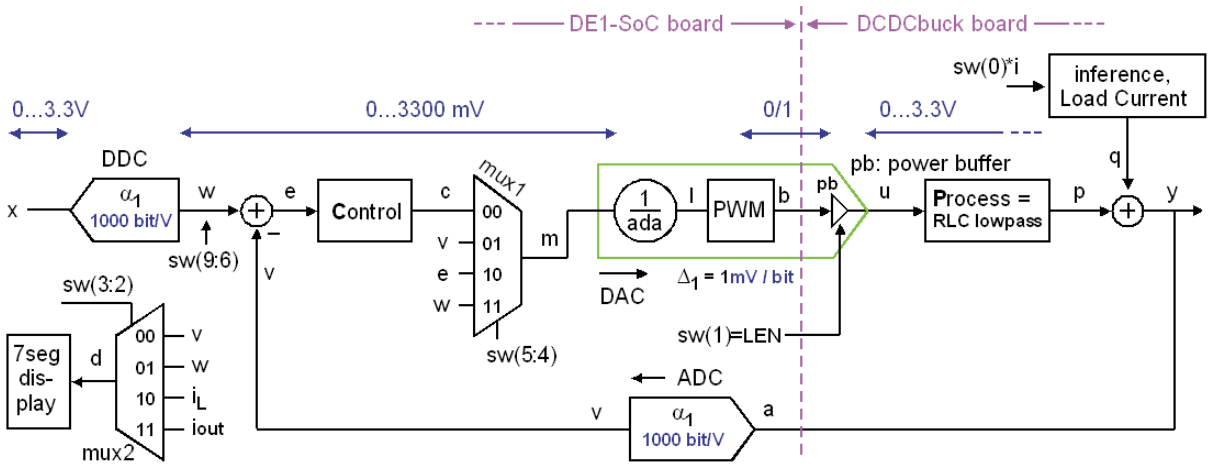
(b): *DCDCbuck_R12.03* board, subcircuit power supplies



(c) Subcircuit 1A Load Current



(d) Subcircuit Connectors



(e) Principal schematics of the DCDCbuck_R12.03 daughter board

Fig. 2.3: *DCDCbuck_R12.03* board, main circuit and subcircuits

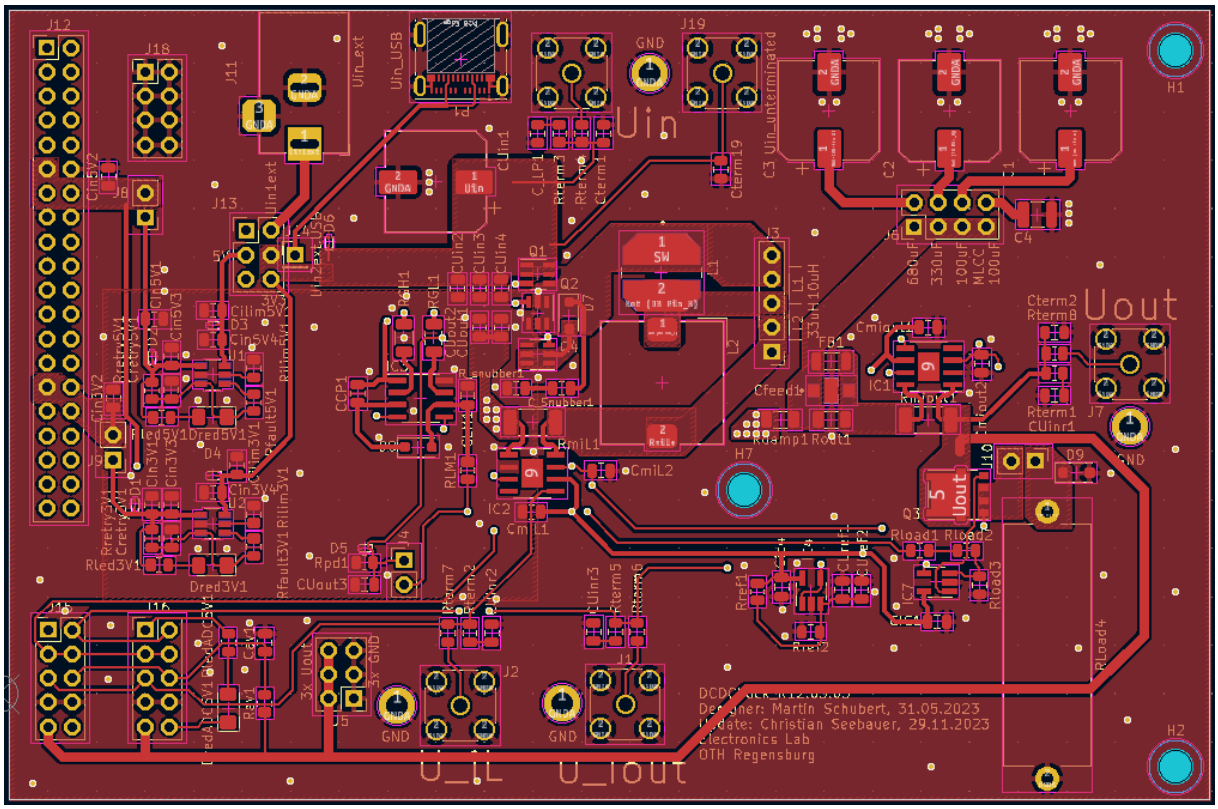


Fig. 2.4(a): *DCDCbuck_R12.03* front-side copper layer (*KiCad* [19])

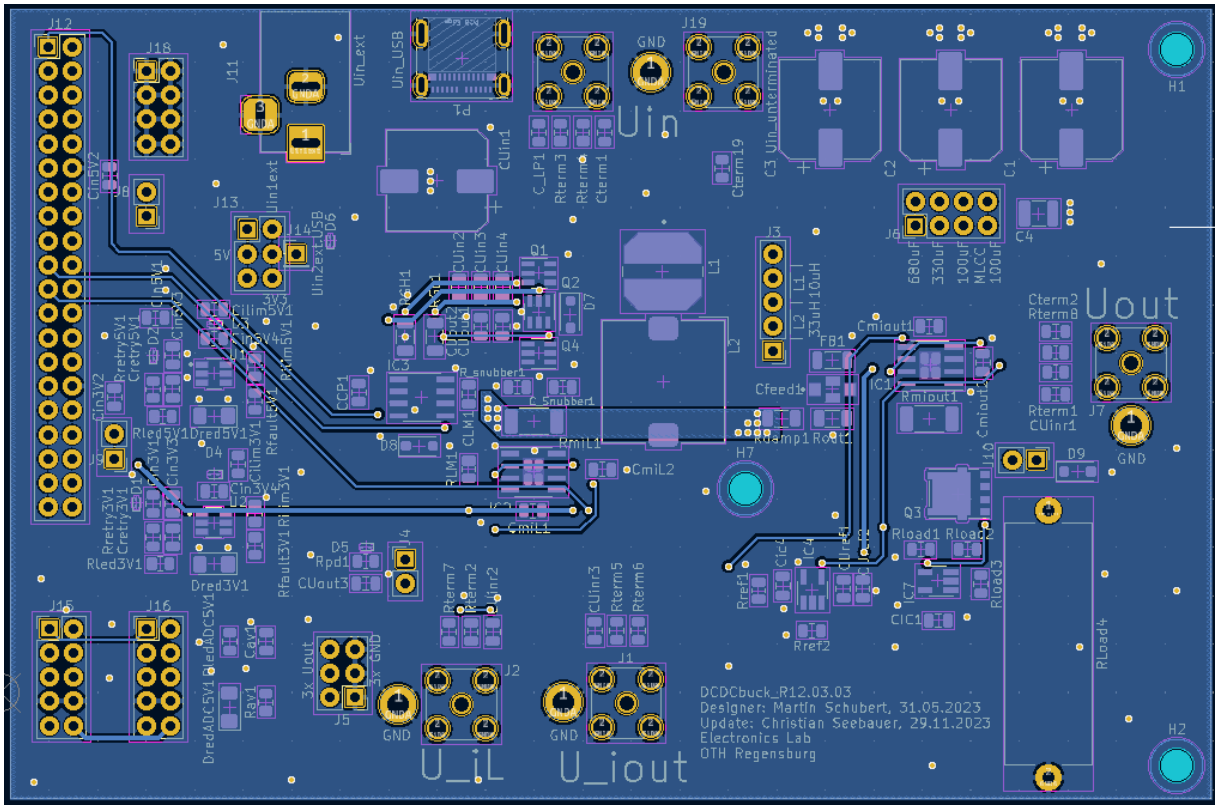


Fig. 2.4(b): *DCDCbuck_R12.03* back-side copper layer (*KiCad* [19])

Fig. 2.4 illustrates the layout (a) and photo (b) of the *DCDCbuck_Rev12.03.03* board. Vias connect different metal layers and may hold pins of plugs.

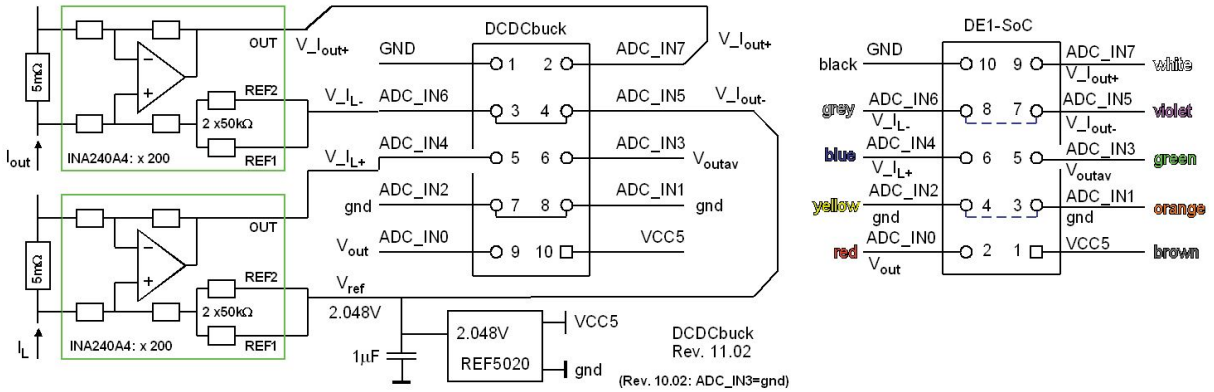


Fig. 2.5: Pin assignment of the 10-pin ADC input plug (connected by the 10 wire ribbon cable). It is a cross cable! Color code valid for $V_{CC}(ADC)$ = black. Numbers within the plug-box are the pin-numbers of the plug. Labels *ADC_IN#* (# = 1…8) indicate input channel number # of ADC *LTC2308* [15]. *ADC_IN3* is ground for board revisions *Rev* ≤ 11.01.

Fig. 2.5 illustrates the pin assignment of the 10-pin ADC input plug (connected by the 10 wire ribbon cable seen in the photo). Numbers within the plug-box are the pin-numbers of the plug. Label *ADC_IN#* (# = 1…8) indicates input channel number # of the ADC *LTC2308* [15]. *ADC_IN3* is ground for board revisions *Rev* ≤ 11.01.

3 *DCDCbuck* Board

3.1 Starting *DCDCbuck* Board

3.1.1 First Steps with Board and Equipment

Starting with *DE1-SoC* and *DCDCbuck* Hardware
Plug daughterboard *DCDCbuck* into user header *JP2* (=*gpio*_1) of *DE1-SoC* board as shown in Fig. 2.1.

On the *DCDCbuck* daughter board:
- Set jumpers as shown in Fig.2.1.
- Connect ribbon cable from *DCDCbuck* to *DE1-SoC* board's 10-pin plugs as.
- If a red LED lights up near red supply jumpers #, whereas # = 1,2, then the electronic fuse was activated by a current peak. You will have to unplug and re-plug the jumper *#.#V_SUPPLY1* to reset the respective current limiter.
- Set switches of the *DE1-SoC* board to all zero: *sw*(9:0) = "0000 00 00 00".

Starting with *Quartus* Software
Use *Quartus lite* software (Revision is 21.1 in Aug. 2022) to program *DE1-SoC* board with file *ci_de1soc_DCDCbuck.sof*, that you will get from Homepage Schubert [26].

- *Homepage Schubert > Offered Education > Laboratories > Labs with DE1-SoC Board and Daughter Boards > Model_files_complete,* download *Models_DCDCbuck_Rev##.zip*.
- Extract directory *Models_DCDCbuck* from the zip file.
- Navigate to directory .*Models_DCDCbuck\VHDL\Quartus\ci_de1soc_DCDCbuck*.
- Within directory …*ci_de1soc_DCDCbuck* click on file *ci_de1soc_DCDCbuck.qpf*. The *Quartus* software should start with *ci_de1soc_DCDCbuck* as working directory.
- Select from *Quartus* main menu: *Tools > Programmer*. The programmer window should open with *Hardware setup…* window showing string *DE1-SoC [USB-1]*. Otherwise, look at *Getting Started with DE1-SoC Board* from the author or any other reference.
- Select *Programmer* > subdirectory *output_files* > *ci_de1soc_DCDCbuck_Rev10.sof*.
- Click on programmer's button *Start* to program the FPGA. You should see a reaction at the LEDs of the *DE1-SoC* board.

Operating the Board
Set *sw*(2) = 1. On the 7seg- display, when *sw*(9:6)="0000", you should see something like "`i 1250`", which is the set point in millivolts (mV). The set voltage depends on *sw*(9:6).
Set *sw*(2) = 0. When *sw*(9:6)="0000", you will see output the voltage the measured by the ADC indicated on the 7seg display, something like "`U 12xy`", with *xy* being close to 50. Press push button *key*(3) to freeze the displayed number.

Got it running? Congratulations! You got the board working!

3.1.2 Load Current Switch and Avoid of Overheating

Read this page carefully to not destroy the *DCDCbuck* board!

Setting *sw*(0)=1 or pressing push button *key*(2) turns the load current on, when jumper *JP_Load* is set. Avoid long phases of load current flow, particularly with output voltages >1.25 V!

Output current of i_{out} = 1 A is achieved by 1V across the load resistor R_L = 1 Ω, which is the big blue resistor in Figs. 3.1.2 and 2.3(b). Consequently, we need output voltages U_{out} > 1 V to get I_{out} = 1A. The load resistor then dissipates P_R = 1 V·1 A = 1 W, while specified for 5 W.

Consequently, the voltage across the power FET (which regulates the resistor voltage to 1V) is U_{DS} = (U_{out} – 1 V),. The power dissipated by the FET is then P_{FET} = (U_{out} – 1 V)·1 A. Identify FET labeled *Q4* (e.g. Fig. 3.1.2), with its gate controlled by OpAmp *LM7301* and its source connecting jumper *JP_LOAD*. It is significantly smaller than the load resistor. At U_{out} = 2 V it has to dissipate the same heat.

We have already "killed" several of those power FETs!
Please , keep output voltage as low as possible at (1+*x*)V for load current ON
Here, we typically use U_{out} = 1.25 V.

Exercise 3.1.2(a): Let U_{out} = 2.5 V and load current on: I_L = 1 A. What is the total load power that must be translated to heat? Witch power shares are dissipated by R_L and the power FET?

Total output power $P_{L,tot} = U_{out} \cdot I_{out} =$ .....

Resistor's power dissipation $P(R_L) = U_{RL} \cdot I_{out} =$ .....

FET's power dissipation $P(FET) = (U_{out}-U_{RL}) \cdot I_{out} =$ .....

Exercise 3.1.2(b): Turn load current i_{out} OFF and set output voltage to U_{out} = 2 V, achived by sw = "1001 00 00 00", so that load resistor and load-current FET *Q4* dissipate the same amount of energy.

- Clean your fingers! Sweat contains salt which can cause creepage currents.
- Use 2 of your finger tips to check the temperature of R_L and Q_4, turn i_{out} from OFF → ON by setting *sw*(0)='1'. How long can you touch and which device becomes hotter? Turn i_{out} OFF then by *sw*(0) = '0' ! (Consider that the cooling effect of your finger is no more there now.)

Fig. 3.1.2: R_L, Q_4 and *JP_LOAD*

Solutions to the exercise 3.1.2(a):

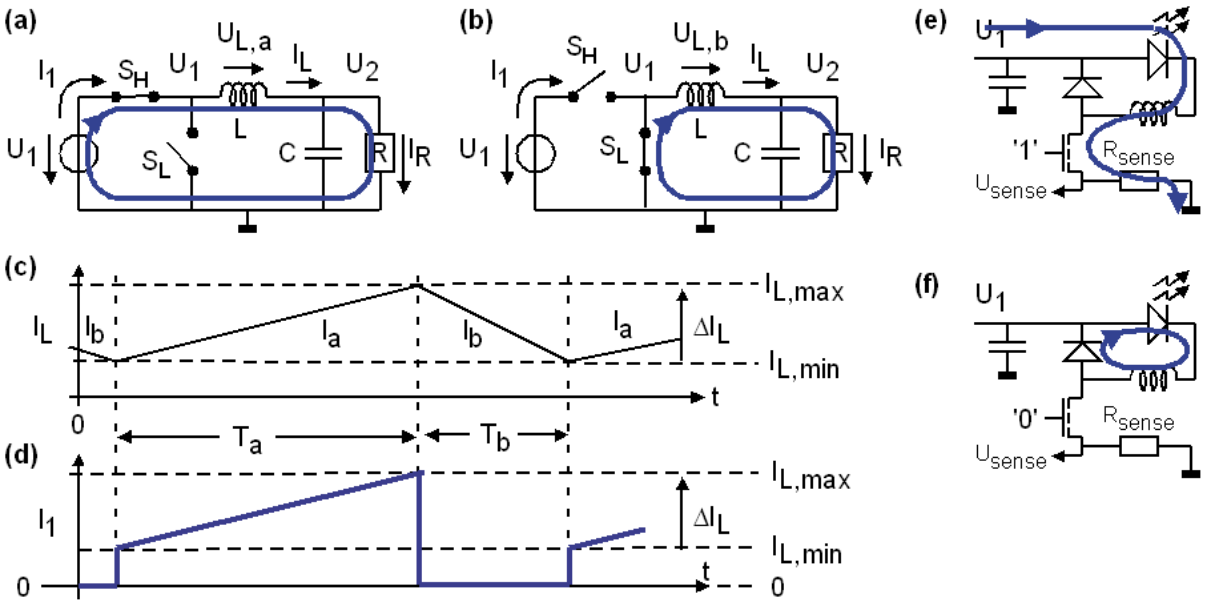
Total load power	$P_{L,tot}$	= $I_L \cdot U_{out}$ = 1A·2.5V = 2.5 W
Resistor's power dissipation	$P(R_L)$	= $I_L \cdot U(R_L)$ = 1A·1V = 1W
FET's power dissipation	P(FET)	= $I_L \cdot U(FET)$ = 1A·(U_{out}-1V) = 1.5 W

Solutions to the exercise 3.1.2(b): Can hold some 10 seconds, Power-FET Q4 becomes hotter.

3.1.3 Operating the DCDCbuck Board

Using the Oscilloscope

Look at the author's document *Quickstarting_DSoX2024_Oscilloscope.pdf* to handle the *DSO-X2024* oscilloscope. In the following, the expression "connect *pin$* to *channel#* [*CH#*]", stands for: connect the red and black ends of the *BNC*-to-pin cable to *pin$* and ground, respectively, and the *BNC* plug on the other end of the cable to channel # of the oscilloscope, whereas # = 1…4. As can be seen from Fig. 2.1, there is a group of 2x4 ground pins near the big blue load resistor.



(g) Switching noise on the output voltage (green) caused by high dI_1/dt.

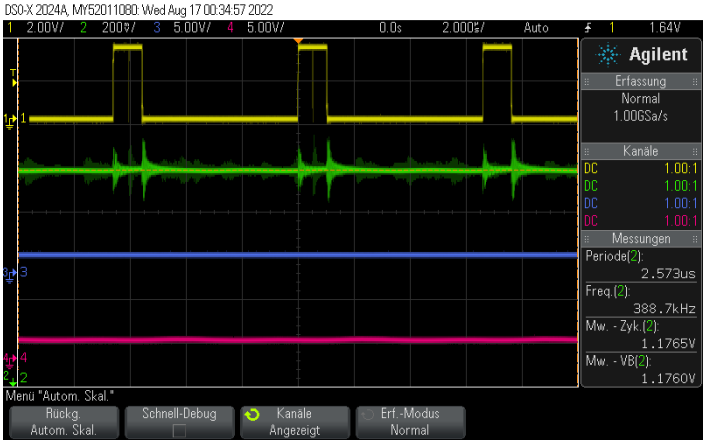


Fig. 3.1.3 : DC/DC buck converter. (a), (b): Operation mode is synchronous due to the operation of switch S_b instead of a diode. (e), (f): asynchronous due to diode. Switch S_H is

(a) for period T_a ideally conducting and

(b) for period T_b isolating. S_L operates inverted to S_H.

Figs. 3.1.3(a-d) illustrate synchronous operation with sufficiently high load current, so that the inductor current is always $I_L > 0$A. In this situation, synchronous operation (i.e. using low-side switch S_b) is superior to and independent of any diode.

When is Synchronous or Asynchronous Operation Advantageous?

When the inductor current becomes negative, i.e. $I_L < 0$A, synchronous operation becomes inefficient. In this case, asynchronous operation (i.e. using a diode instead of switch S_b) becomes more efficient. If low-side switch S_b, which is N-channel power MOSFET, is *OFF*, then its inevitably build-in body diode is available. However, a parallel Schottky diode is faster and more efficient due to its lower threshold voltage of 0.4…0.5V. The most efficient method is to operate the low-side switch S_b like a diode, which is called diode-emulation mode (*DEM*).

Interconnections to the *DSO-X 2024* oscilloscope:
Info: JP2==gpio_1 is occupied by the DCDCbuck board. A *VHDL* [22], [23], [24] statement "*gpio_0 <= gpio_1*" copies user header *gpio_1* to *gpio_0*. So signals at *gpio_0* and *gpio_1* are identical.

- PWM signal: Connect the osci's CH1 (yellow) to *pin 1* (upper left in Fig. 2.1) of *gpio*_0.
- Output voltage: Connect the osci's CH2 (green) to *V_OUT_PIN* of the *DCDCbuck* board.
- Output current: Connect the osci's CH3 (blue) to *BNC* plug *V_IOUT* of *DCDCbuck* board.
- Inductor current: Connect the osci's CH4 (red) to *BNC* plug *V_IL* of the *DCDCbuck* board.

Oscilloscope Auto Scaling and Saving the Screen to a JPG-File

Set sw(9:0) = "0000 00 00 00". Output voltage should be some 1250 mV.

- Switch on *CH1, CH2, CH3, CH4* of the oscilloscope. Then press hard key *Auto Scale*. You will get the image shown in Fig. 3.1.3(g).
- Save the screen on a USB memory stick. It must be *FAT32* formatted.

Triggering

- The oscilloscope triggers now the rising edge of CH1 (yellow), indicated by the little yellow triangles at the top and left edge of the screen.
- CH1 (yellow) shows the output of the pulse-width modulating (PWM) DAC.
- Change the setting of your oscilloscope such, that it triggers CH2 (green).
- Use the external trigger input: Connect *pin3* of *gpio_0* (directly "under" *pin1* in Fig. 2.1) to CH2 (green) of your oscilloscope: It is a short trigger-pulse indicating the rising edge of the PWM pulse. Unplug it from CH2 and plug it at the backside of the scope to the external trigger input labeled *EXT TRIG IN*. Then select trigger channel external (*EXT*) → oscillogram should be stable again.
- Display the output signal U_{out} on CH2 of the oscilloscope.

Oscilloscope Channel Scaling

Set *sw*(9:0) = "0000 00 00 00". Output and oscilloscope are still as shown in Fig. 3.1.3(g).

- Tune the osci settings such, that the screen looks like Fig. 3.1.4(a).
- Try the 8 possibilities illustrated in Figs. 3.1.4(a-h) for sw(1)=0/1, sw(0)=0/1, Schottky diode activated yes/no, i.e. jumper *JP_Schottky* plugged/unplugged.
- Explain the different pulse-widths of the PWM signal. Be aware, that a lower pulse width is more efficient for same output voltage and current.

Observing Some Effects

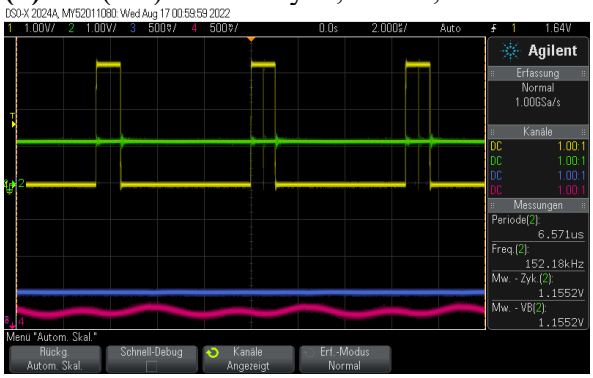
- Disconnect all black ground ends from the measurement cables from the ground pins on the DC/DC board and observe the impact of grounding. You should see more voltage overshoot on the oscilloscope. Then connect ground cables again.

3.1.4 Observing Important Output Voltages

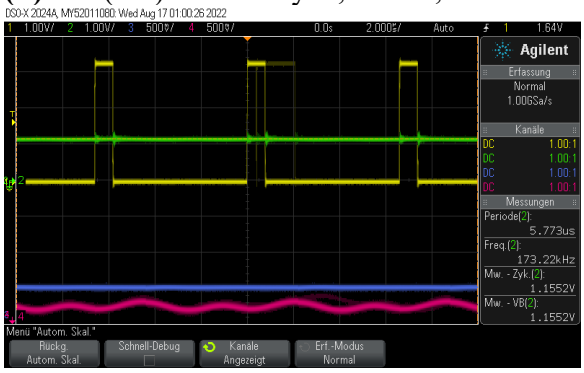
In Figs. 3.1.4(a-h) a higher efficiency results in lower pulse-width for the same output voltage and current.
The output current (blue) and inductor current (red) are measured as voltages over a 5mΩ shunt resistor, which are multiplied x 200 by the *INA240* amplifier. To allow for measurements of negative inductor currents, the output voltage of the *INA240* amplifiers have an offset of 2.048V generated by the *REF5020* reference-voltage IC. Consequently, currents are measured as voltages $V\_i_L = 1\Omega \cdot I_L + 2.048\text{V}$ and $V\_i_{out} = 1\Omega \cdot I_{out} + 2.048\text{V}$.

- Observe that the voltages V_i_L and V_i_{out} depending on currents i_L and i_{out} rise by 1V, which corresponds to 1A, when *sw*(0) is pushed from position '0' to '1'. (Unfortunately, we observe considerable switching noise for this converter.)

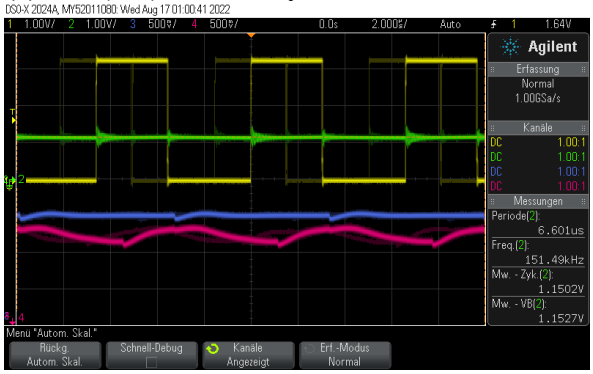
(a) sw(1:0) = '00': async, I_L=0A, D_S: Y



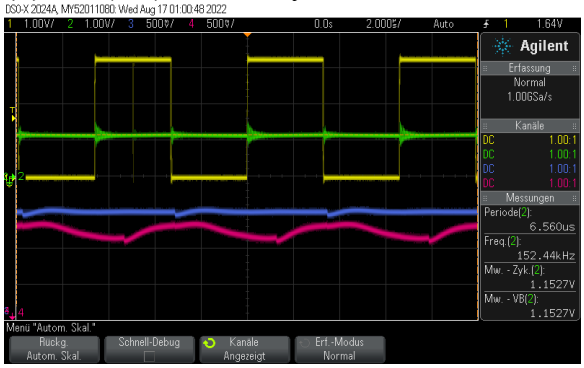
(b) sw(1:0) = '00': async, I_L=0A, D_S: N



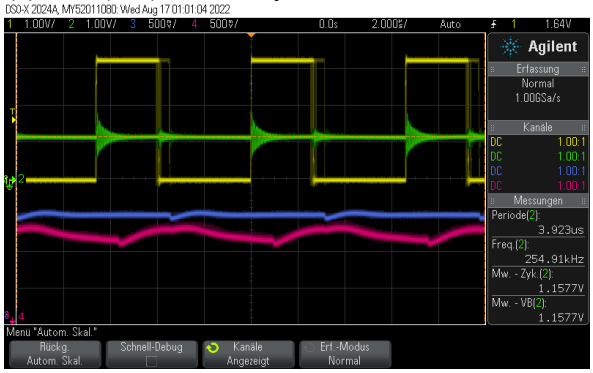
(c) sw(1:0) = '01': async, I_{out}=1A, D_S: Y



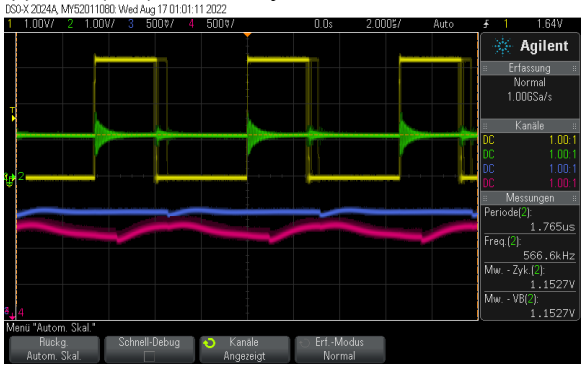
(d) sw(1:0) = '01': async, I_{out}=1A, D_S: N



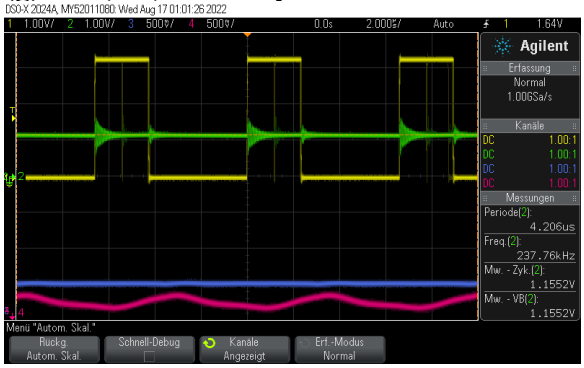
(e) sw(1:0) = '11': sync, I_{out}=1A, D_S: Y



(f) sw(1:0) = '11': sync, I_{out}=1A, D_S: N



(g) sw(1:0) = '10': sync, I_{out}=0A, D_S: Y



(h) sw(1:0) = '10': sync, I_{out}=0A, D_S: N

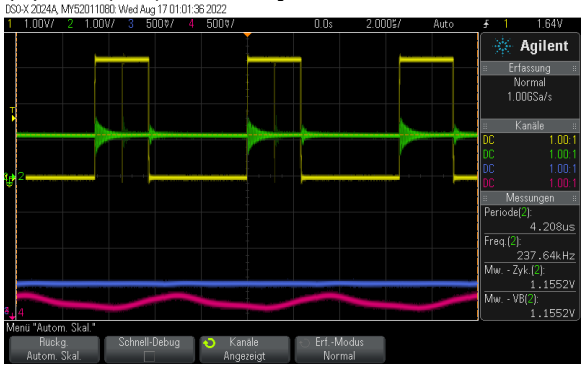


Fig. 3.1.4:. Osci screen copies for *sw*(1)=0/1: async./sync. operation, *sw*(0)=0/1: I_L=0/1A, D_S=Y/N: Schottky diode active: yes/no, i.e. jumper *JP_SCHOTTKY* plugged / unplugged; yellow: PWM signal (1V/div), green: output voltage (1V/div), blue: I_{out}, red: I_L

3.2 The System Setup

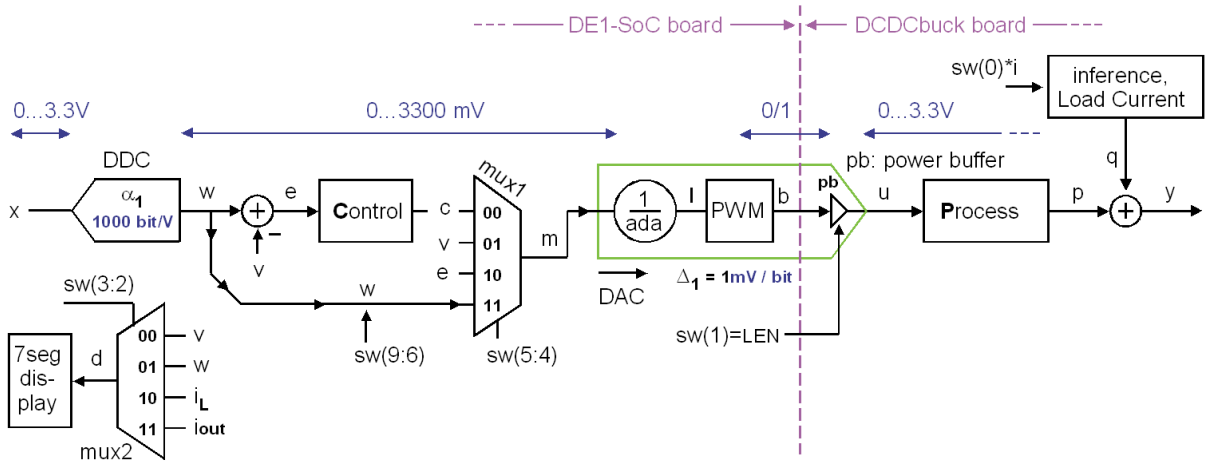


Fig. 3.2: DC/DC buck converter schematic showing voltage feedback only

Fig 3.2 illustrates the system with its analog and digital domains:

1. Analog: Physical voltage range being mainly defined by the two output voltage levels $U_{pwm,low}$, $U_{pwm,high}$ of the PWM DAC. Fig. 3.2 shows example voltages $U_{pwm,low} = 0.0V$, $U_{pwm,high} = 3.3V$ in blue letters.
2. Digital: ADC output range computed as $N_{ADCout} = \alpha_1 \cdot U_{ADCin} + \alpha_0$, with $\alpha_1 = 1000$ bit/V , $\alpha_0 = 0V$ given by ADC *LTC2308* with output range 0…4095 corresponding to 0…4.095V. Consequently, *v, w* and *e* are given in millivolts (mV) and voltage range 0…3.3V corresponding to integral number range 0…3300. (To compute input *w* from *x* the amplification of the ADC in the feedback branch must be compensated for by the preceding digital-to-digital converter (DDC) with same gain α_1 as the ADC.)

Factor *ada* is shown as a part of the PWM DAC in Fig. 3.2. It compensates for different amplification of ADC and DAC, so that a direct series of these modules delivers an amplification of 1.

In detail: The PWM DAC with input level *l* in Fig. 3.2 translates a series of 0 … *pwm_period* ones within a period of *pwm_period* bits into voltage range U_{low} - U_{high}. Consequently, its amplification is $\Delta_1 = (U_{high} - U_{low}) / pwm\_period$. In the example of Fig. 3.1 with $pwm\_period = 330$ this corresponds to a DAC input level range of $l = 0$ … *pwm_period* resulting in $l = 0…330$. Thus, one bit of DAC input level *l* corresponds to $\Delta_1 = 10mV$ of DAC output voltage *u*.

Seven-segment display, switch sw(3:2)

According to table 3.2, signals *v*, *w*, i_L, i_{out} can be selected by switches sw(3:2) to be shown on the 7-segment display in mV or mA depending on *sw*(3)='0' or '1', respectively.

System operation mode, switch sw(5:4)

According to table 3.2, signals fed to PWM input can be selected from *c*, *v*, *e, w* by switches *sw*(5:4).

Control Signals *pwmo, enc, LSE, I_{Load}_ON*

- *pwmo* is the pulse-width modulated output signal. Max. number of bits is *pwm_period*, which is by default 330, so that 3.3V correspond 330 bits → resolution is 10 mV/bit.
- *enc* is a flag having a width of T_{clock}=20ns used as sync signal for the oscilloscope.
- *LSE* (= Low-side Switch Enable) dis-/enables the low-side switching field effect transistor (FET) when 0/1.
- I_{Load_ON} switches a load current of I_L = 1A on when *ILoad_ON* = 1.
- *gpio_0 <= gpio_1*: User header *JP2* (= *gpio*_1) is covered by the *DCDCbuck* board, this statement copies all signals from *JP2* to *JP1*, where they can be observed.

Listing 3.2: *VHDL* [22] code lines of module *de1soc_DCDCbuck(rtl_de1soc_DCDCbuck)*.

```
-- DCDCbuck Daughter Board
  gpio_1(0)  <= pwmo;       -- PIN01 = PWM output
  gpio_1(2)  <= start_fast; -- PIN03 = PWM period start flag
  gpio_1(16) <= LSE;        -- PIN19 = LSE (Low-side Switch Enable): async/sync mode if 0/1
  gpio_1(18) <= ILoad_ON;   -- PIN21 = LoadCurrent_ON  -- miscellaneous
  -- As gpio_1 is occupied by the plug, log it on gpio_0
  gpio_0 <= gpio_1; -- this line delivers measurement points on gpio_0
```

Exercise 3.2.1: Signals *pwmo*, *enc*, *LSE*, I_{Load}_ON are driven to *gpio*_1 pins according to listing 3.2. However, these pins are covered by the main plug of the *DCDCbuck* board. Where can we measure these signals, and why? (For solutions → see next page)

. . . . . .

A/D and D/A Conversion

We model the D/A converter (i.e. *PWM DAC*) and A/D converter (i.e. *LTC2308*) as

$U_{PWM} = \Delta_0 + \Delta_1 \cdot l$ and $v = \alpha_0 + \alpha_1 \cdot y$

with integral PWM input number $l = l_{min} \ldots l_{max} = 0 \ldots pwm\_period = 0, 1, 2, \ldots 330$
and integral ADC *LTC2308* [15] output number $v = v_{min} \ldots v_{max} = 0, 1, 2, \ldots 4095$.

Exercises 3.2.2: Compute $\Delta_0, \Delta_1, \alpha_0, \alpha_1$ for $U_{ADCin,min}$ = 0V and $U_{ADCin,max}$ = 4.095V
(Solutions next page)

$\Delta_1 = (U_{PWM,high} - U_{PWM,low}) / pwm\_period =$ . . . . . .

$\Delta_0 = U_{PWM,low} - \Delta_1 \cdot v_{min} =$ . . . . . .

$\alpha_1 = (v_{max} - v_{min}) / (U_{ADCin,max} - U_{ADCin,min}) =$ . . . . . .

$\alpha_0 = v_{min} - \alpha_1 \cdot U_{ADCin,min} =$ . . . . . .

$ada = \alpha_1 \cdot \Delta_1 =$ . . . . . (ADC + DAC amplification compensation factor)

In which case do we get $U_{PWM,low}$ = -0.5V or -0.7V? / When do we get $U_{PWM,high}$ = 5V?

Table 3.2: Functionality of switches *sw*(9:0) and push buttons *key*(3:0)

9	8	7	6	5	4	3	2	1	0
Set point (wanted) *w* in mV				DAC inp. sel		7seg inp. sel		LSE	i_{Load}

Switches	
sw(9:6)	Select set point *x* in V
0000	Set point w = 1250
0001	Set point w = 0
0010	Set point w = 250
0011	Set point w = 500
0100	Set point w = 750
0101	Set point w = 1000
0110	Set point w = 1500
0111	Set point w = 1650
1000	Set point w = 1750
1001	Set point w = 2000
1010	Set point w = 2250
1011	Set point w = 2500
1100	Set point w = 2750
1101	Set point w = 3000
1110	Set point w = 3300
1111	Set point w : defined by the *HPS**) using *Linux* program *set_w*; Set-point values out of range 2mV … Vin will be modified to 1234 mV. *) Hard Processor System, embedded in the FPGA
sw(5:4)	Select quantity fed to the input of the PWM DAC
00	control mode => output c
01	control mode => output v
10	control mode => output e
11	control mode => output w
sw(3:2)	Select quantity displayed on 7-segment display
00	display v in mV : label U -> output voltage
01	display w in mV : label i(nput) -> wanted output voltage
10	display i_L in mA : label L -> sampled inductor current
11	display i_{out} in mA : label o -> sampled output current
sw(1)	LSE: Low-side Switch Enable
0	Asynchronous mode: Low-side power-MOSFET is always off.
1	Synchronous mode: Low-side switch is ready to operate
sw(0)	Load current switch
0	Load current OFF
1	Load current of 1A ON

Keys	(=push buttons)
key(0)	Global asynchronous reset, dominant over all other signals: all flipflops are reset to their reset-states
key(1)	Global enable: flipflops do not change state when *key*(1) pushed
key(2)	Load current ON: pushing *key*(2) has the same effect as *sw*(0)='1'; current flow stops when *key*(2) is released.
key(3)	hold 7-segment display: 7seg-display frozen while *key*(3) pushed

Mini keys	(small push buttons), functionality according to [24].
left	HPS reset, restarts the hard processor system
middle	HPS User button, restarts the hard processor system
right	Warm reset

Solutions 3.2.1: **We can measure *pwmo, enc, LSE, I_{Load}_ON* on gpio_0 due to the last line of listing 3.2**

Solutions 3.2.2: Compute Δ_0, Δ_1, α_0, α_1 for $U_{ADCin,min}$ = 0V and $U_{ADCin,max}$ = 4.095V

$\Delta_1 = (U_{PWM,high} - U_{PWM,low}) / pwm\_period =$ **3.3V – 0V / 330 = 10mV = 0.01V**
$\Delta_0 = U_{PWM,low} - \Delta_1 \cdot v_{min} =$ **0V - 10mV·0 = 0V**
$\alpha_1 = (v_{max} - v_{min}) / (U_{ADCin,max} - U_{ADCin,min}) =$ **(4095 – 0) / (4.095V – 0V) = 1/mV = 1000/V**
$\alpha_0 = v_{min} - \alpha_1 \cdot U_{ADCin,min} =$ **0 – 1/mV · 0V = 0 .**
A/D + D/A amplification compensation factor: $ada = \alpha_1 \cdot \Delta_1 =$ **1/mV · 10 mV = 10**
In which case do we get $U_{PWM,low}$ = -0.5V or -0.7V? / When do we get $U_{PWM,high}$ = 5V?
Asynchronous mode with Schottky or normal diode. / Change power jumper JP_3.3V to JP_5.0V.

3.3 Observing *DCDCbuck* Board Operation Details

3.3.1 *DCDCbuck* Board: Process and Load current Inference Realization

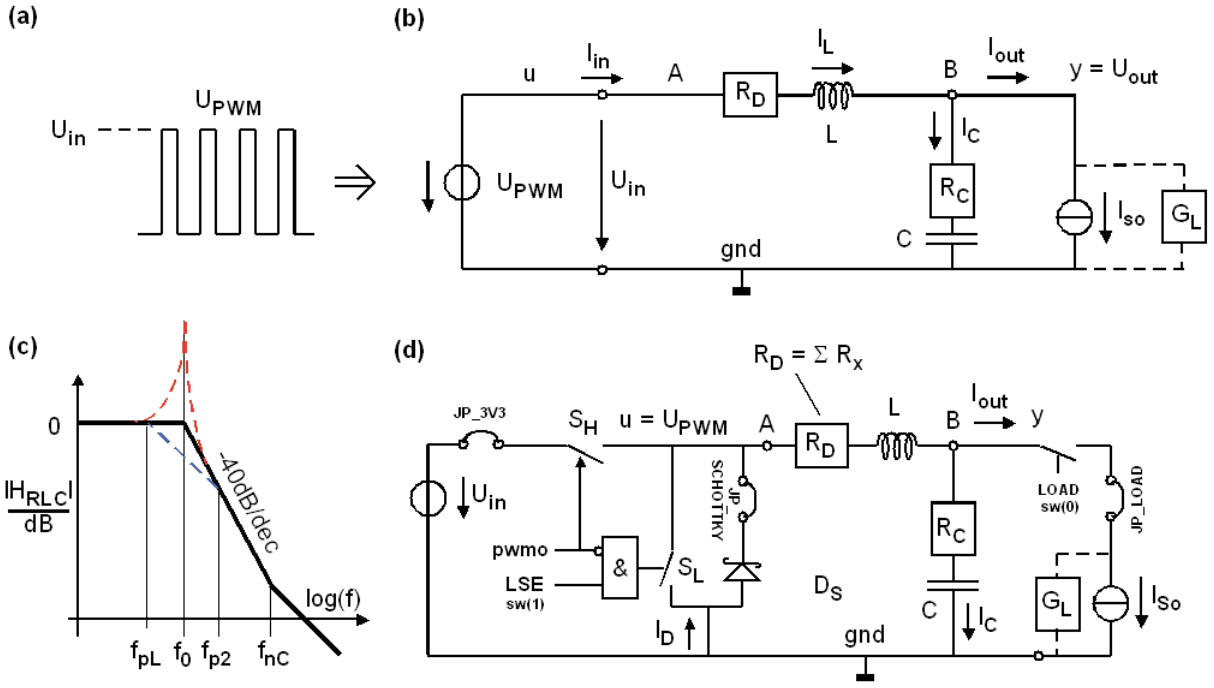


Fig. 3.3: (a) pulse-width modulated input voltage signal (U_{PWM}), **(b)** realization of boxes *process* and *load current inference* in Fig. 3.2: U_{PWM} is smoothed by an *LC* lowpass, damped by some resistors, **(c)** *RLC* function over frequency *f*, **(d)** more detailed view of part (b).

Fig. 3.3 details the boxes illustrated as *process* and *inference load current* in Fig. 3.2:

(a) Shows a pulse-width modulated input voltage (U_{PWM}) which is smoothed by an

(b) LC lowpass with unavoidable resistors R_D consisting of generator impedance *Rg* and wire resistance *Rw*, an equivalent serial resistor (R_C) of the capacitor and load conductance G_L. These resistors effect the damping of the LRLC lowpass, that is illustrating in the

(c) Bode diagram, showing asymptotes (**bold solid line**) over frequency *f*, with oscillating case indicated as red dashed line and aperiodic case shown as blue dashed line.

(d) Is a more detailed view of Fig. part (b), showing jumpers *JP_3.3V*, *JP_SCHOTTKY* and *JP_LOAD* being set. Consequently U_{PWM} oscillates up to 3.3V, Schottky diode D_S is ready for supporting asynchronous operation enforced *sw*(1) when *LSE* = 0, and a *load current inference* of 1A can be imposed setting the *LOAD* switch *sw*(0) = 1.

High-side switch S_H and low-side switch S_L are controlled by the digital logic signal *pwmo* generated by the *FPGA*. Chip *LM27222* [25] makes sure that S_H and S_L are operated non-overlapping, i.e. they never conduct at the same time, because this would open a direct current path from U_{in} to ground.

3.3.2 Operating the *DCDCbuck* Board

Set the set point for the output voltage to w = 1250mV and observe pulse with of *pwmo*, output voltage U_{out} and voltage over the 1Ω load resistor to know the current.

3.3.2.1 Operation Without Control Unit

We will now use fixed pulse-width for *pwmo* without control:

- Set *control mode => output w* (i.e. sw(5:4) = "11"). No feedback, as m=w in Fig3.2.
- Load current off: *sw*(0) = '0' → I_{out} = 0A.
- Set synchronous mode: *sw*(1) = *LSE* → 1.
- Show w = 1250mV (sw(3:2) = "01") on the 7-seg. display. Then show *v* (→ sw(3:2) = "00") to see the sampled output voltage, which should be near 1250mV.

Switch the load current on. The pulse-width should stay the same and the output voltage should drop, typically some 200mV. This is helpful to compute R_D in Fig. 3.3(d) according to $R_D = \Delta U_{out}/I_{Load}$. What do you measure?

With load current remaining ON, switch to the asynchronous mode. Switch S_L stays always open now. There should be an additional output voltage drop, because energy is lost within Schottky diode D_S.

Now switch the load current OFF. The output voltage begins to rise to an uncontrolled value. This is because there is no more means available to pull U_{out} down: S_L is out of operation, no output current and Diode D_S is reverse biased.

3.3.2.2 Check Controlled Operation Roughly

We will now use controlled pulse-width for *pwmo*:

- Set *control mode => output c* (i.e. sw(5:4) = "00"). Control is ON, as m=c in Fig3.2.
- Load current off: *sw*(0) = 0 → I_{out} = 0A.
- Set synchronous mode: *sw*(1) = *LSE* → 1.
- Show w = 1250mV (sw(3:2) = "01") on the 7-seg. display. Then show *v* (→ sw(3:2) = "00") to see the sampled output voltage, which should be some 1250mV, too.

Check output voltage roughly with and without load current, synchronous and asynchronous operation. The controller should now adapt the pulse-width (signal *pwmo*) such, that the output voltage constantly keeps at 1250mV. Push *key*(3) to get a single sample of the output voltage.

3.3.2.3 Controlled Operation in Asyncrounous Mode with Load-Current ON

Index: `(9876 54 32 10)`
Set `sw` = `"0000 00 00 01"`

The last 2 bits determine `sw(1) = '0'`: asynchronous mode, i.e. the lower PowerFET is off, but `sw(0) = '1'`: I_L = 1A load current is on. The controller can balance the desired output voltage by balancing the current flowing through the coil.

We now have the situation that output voltage is controlled by adapting pulse width. Less pulse-width at same output voltage and current corresponds to more efficiency.

3.3.2.4 Controlled Operation in Synchronous Mode with Load-Current OFF

Index: `(9876 54 32 10)`
Set `sw` = `"0000 00 00 10"`

The last 2 bits determine `sw(1) = '1'`: synchronous mode, i.e. the lower PowerFET operates, but `sw(0) = '0'`: I_L = 0A load current is off. The controller can balance the desired output voltage, but the situation is inefficient: The high-side PowerFET load charge into the output capacitor, then the low-side PowerFET discharges it versus ground.

3.3.2.5 Controlled Operation in Syncrounous Mode with Load-Current ON

Index: `(9876 54 32 10)`
Set `sw` = `"0000 00 00 11"`

The last 2 bits determine `sw(1) = '1'`: synchronous mode, i.e. the lower PowerFET operates and `sw(1) = '1'`: I_L = 1A load current is on. This is the most effient mode because the low-side PowerFET hat a lower voltage drop then its parallel diode.

3.3.2.6 Uncontrolled Operation in Asyncrounous Mode with Load-Current OFF

Index: `(9876 54 32 10)`
Set `sw` = `"0000 00 00 00"`

The last 2 bits determine `sw(1) = '0'`: asynchronous mode, i.e. the lower PowerFET is off and `sw(1) = '0'`: I_L = 0A: the load current is off. You will observe that the controller cannot fix the output voltage, which drifts up. Briefly set `sw(1)` or `sw(0)` to `'1'` for a short time: Immediately control is gained back. When both switches are `'0'`, the output voltage drifts up.

This is because we need a minimum pulse-width to keep the charge pump running, required to operates the high-side PowerFET. The short pulses continuously push charge into the output capacitor, but there is no way to discharge it, because both ways are blockt: Neither an output current I_L nor the low-side PowerFET are available to discharge the output capacitor. Consequently, the controller finds no means to pull down the output voltage, when it is too high.

4 Using the ARM Core Embedded on the FPGA

4.1 Embedded Exercises of Standard Practical Training

The *Cyclone V FPGA on the DE1-SoC* board *contains an 800MHz dual-core AMR Cortex-A9 MPCore* processor [35]. This chapter demonstrates how to use it for data transfer between the *FPGA* and a *PC*.

Acronyms:

ARM *Advanced RISC Machine*, a computer architecture family [27].
AXI *Advance eXtensible Interface* [28]. On-chip bus protocol developed by ARM.
HPS *Hard Processor System* [29]

Start the *Linux* [30] Server:

- Start *DE1-SoC* board
- Connect USB cable to Computer and *mini- USB B* [31] plug at the upper right corner of the *DE1-SoC* board
- Determine COM-Port number: Gerätemanager → Anschlüsse (COM & LPT) → USB serial Port (COM#)
- Start *PuTTY* [32]: set: serial, COM#, baud rate: 115200 [save session: hps], as shown in Fig. 4.1.1.
- Insert *microSD* [33] card into the SD card slot near the *USB mini-B* plug as shown in Fig. 4.1.2(c).
- Boot *Linux* system on *microSD* card by switching the *DE1-SoC* board's power OFF → ON or press the left or medium of the 3 little push buttons, which are located on the right hand side of the big push buttons.
- Program the *Cyclone-V FPGA* with the *Quartus* programmer
- **System prompt:** user input

```
socfpga login: root
<ci_de1soc_DCDCbuck software must run from here on the DE1-SoC board, on it set sw(9:0)=”0…0”>
root@socfpga:~# ls                                    // list directory
root@socfpga:~# ./hex_timer 5    // run hex timer for 5 seconds, observe 7-segment display!
root@socfpga:~# ./monitor 10      // read 10 data lines from HPS and print them
<make PuTTY window COM# wider using the mouse to avoid undesired line feeds, as shown in Fig. 4.1.3.>
root@socfpga:~# ./monitor 10  // print again with matched window
<On the DE1-SoC board set sw(9:6)="1111" to get HPS control over input w, the wanted output voltage>
root@socfpga:~# ./set_w 1234   // wanted input ( → sw(2)='1') is set to 1234
root@socfpga:~# ./monitor > target_file.log  // writes output to file target_file.log.
root@socfpga:~# cat target_file.log  // print the contents of target_file.log.
root@socfpga:~# rm target_file.log  // remove target_file.log.
```

Fig 4.1.1 illustrates how to figure out *COM#* of your serial port. In this figure it is *COM3*.

Hold your mouse on the *Windows* sign (), click right mouse button → *Device Manager* (dt.: *Geräte-Manager*) to get into this menu.

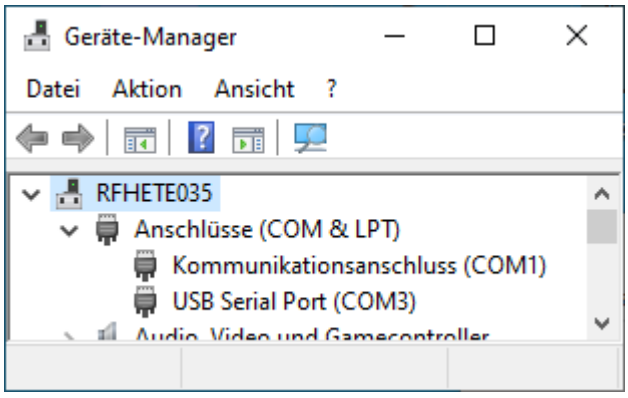


Fig. 4.1.1: Determine the COM port number of your serial connection via the *mini-USB B* plug.

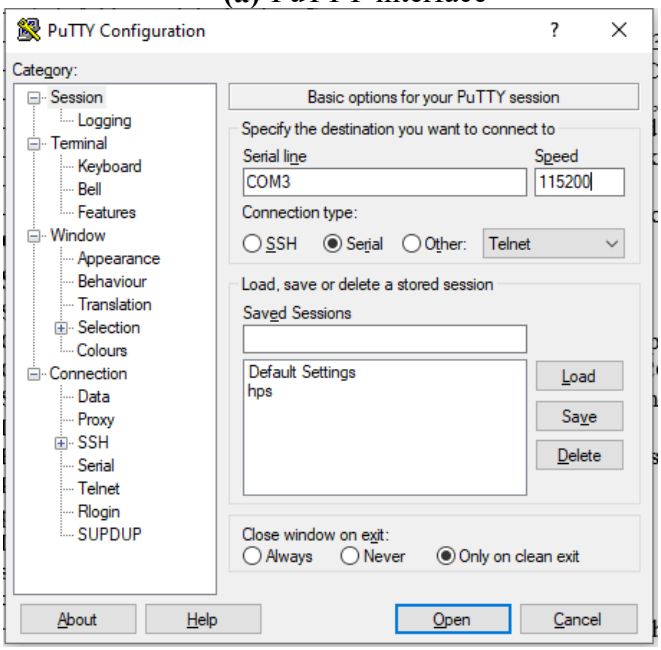
Fig. 4.1.2 illustrates how to connect the DE1-SoC board with a PC using a mini-USB-B [31] cable and where to insert the microSD card with the Linux.

Fig 4.1.3 illustrates the PuTTY window printing data.

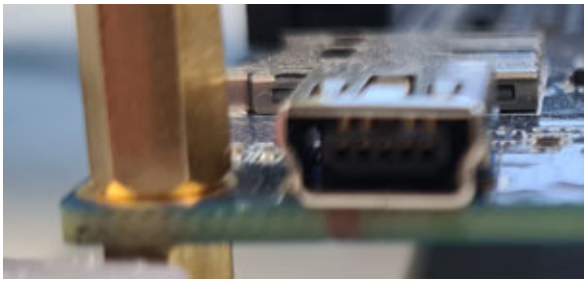
PuTTY interface (right):
Connection type: serial
Serial line: *COM#*
Speed: 115200 baud
[Save configuration as session *hps*]

To figure out #, use the *MS Window's* device manager (dt. Gerätemanager) > Anschlüsse > USB Serial Port (COM#).

(a) PuTTY interface



(b) *mini-USB B* [30] plug at *DE1-SoC* board



(c) *mini-USB B* and *microSD* [32] card

Fig. 4.1.2: *USB* connection between PC and *DE1-SoC* board:

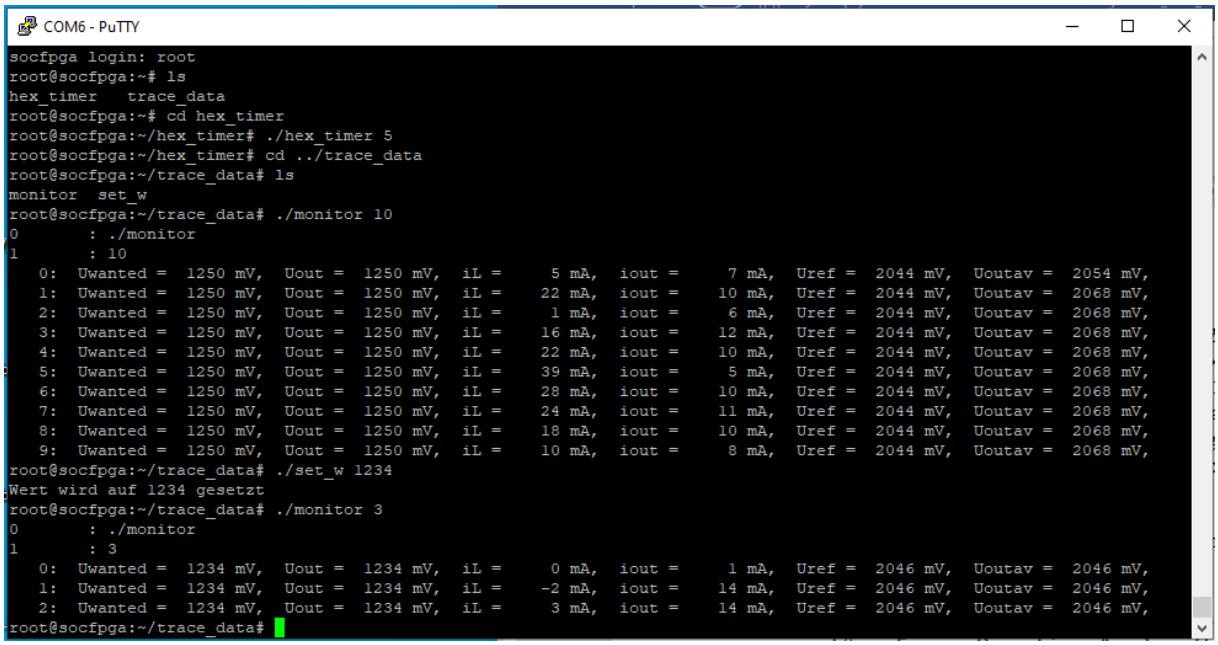


Fig. 4.1.3: Window "*COM# - PuTTY*" showing measured data of signals *w* (u_{wanted}), *v* (u_{ADC}, sampled ADC out), *i_L* (sampled inductor current i_L), *i_out* (sampled output current i_{out}), U_{ref} (reference voltage for current +/- measurement), $U_{out,av}$ (average output voltage measured as output of the RC lowpass (R_a, C_a) on the *DCDCbuck_Rev.10.02* board.)

5 Conclusions

DCDCbuck board was operated as daughterboard of *DE1-SoC* board and basic digital functionality was tested. Some embedded (hardware / software codesign) aspects were demonstrated, reading measured data from VHDL signals in the PFGA to an external PC.

6 References

[1] Available: https://www.terasic.com.tw.
[2] Available: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=836
[3] Available: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=886
[4] Available: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=836&PartNo=4
[5] CD ROM *DE1-SoC CD-ROM (rev.F Board) Version 5.1.2 of 2910-01-28* from [4]
[6] *DE1*-SoC User Manual, Ref. F, taken from [5], available: https://hps.hs-regensburg.de/~scm39115/homepage/education/labs/Lab_ElectronicBoards/DE1-SoC_UserManual.pdf
[7] *DE1*-SoC Schematic, Ref. F, taken from [5], available: https://hps.hs-regensburg.de/~scm39115/homepage/education/labs/Lab_ElectronicBoards/DE1-SoC_Schematic_revF.pdf
[8] K:\SB\Sources\EDA\Terasic\Hardware\
[9] Bode 100 Vector Network Analyzer, Omicron Labs, Available: https://www.omicron-lab.com/products/vector-network-analysis/bode-100/
[10] Florian Schwankner, *Hardware-Erweiterung einer DC/DC-Buck-Konverter-Platine zur Messung des mittleren Spulen- und Ausgangs-Stromes mittels eines ADC*, MEI-Master Projektarbeit, Elektroniklabor der OTH Regensburg, Januar 2022.

[11] Alexander Forster, *Development of a Python to VHDL Compiler Applied to Interface an ARM Processor to an FPGA*, MEI-Master Thesis, Elektroniklabor der OTH Regensburg, September 2022.
[12] Available: https://en.wikipedia.org/wiki/Intel_Quartus_Prime
[13] Available: https://en.wikipedia.org/wiki/ModelSim
[14] Av. https://www.intel.com/content/www/us/en/programmable/downloads/download-center.html
[15] Linear Technology, *LTC2308, Low Noise, 500ksps, 8-Channel, 12-Bit ADC*, available Aug. 2022 : https://www.analog.com/media/en/technical-documentation/data-sheets/2308fc.pdf
[16] Ribbon cable plugs: Sullins Headers SFH11-PBPC-D05-ST-BK, availble Aug. 2022 : https://www.digikey.de/en/products/detail/sullins-connector-solutions/SFH11-PBPC-D05-ST-BK/1990087
[17] MP770803, BNC breakout to pin, available Aug. 2022: https://de.farnell.com/multicomp-pro/mp770803/bnc-plug-sq-pin-socket-x-2-blk/dp/3703639?ost=bnc+to+pin&cfm=true
[18] BNC coaxial cable, available Aug 2022 : https://www.farnell.com/datasheets/179130.pdf
[19] Jumper *M50-1900005*, available : https://cdn.harwin.com/pdfs/M50-19X.pdf
[20] Eagle 7.2, Autodesk, available Aug. 2022 : http://eagle.autodesk.com/eagle/software-versions/7
[21] Eagle 9.6, Autodesk, availble Aug. 2022 : https://www.autodesk.com/products/eagle/blog/autodesk-eagle-9/
[22] Available: https://edg.uchicago.edu/~tang/VHDLref.pdf
[23] Available: https://www.mimuw.edu.pl/~marpe/pul/card_vhdl.pdf
[24] Available: https://www.mimuw.edu.pl/~marpe/pul/card_1164.pdf
[25] Texas Instruments, *LM27222 High-Speed 4.5A Synchronous MOSFET Driver*, available: http://www.ti.com/lit/ds/symlink/lm27222.pdf
[26] Homepage Schubert, Available Aug. 2022: https://hps.hs-regensburg.de/~scm39115/index.htm
[27] ARM: Advanced RISC Machine, avail.: https://en.wikipedia.org/wiki/ARM_architecture_family
[28] AXI: Advanced eXtensible Interface, available: https://en.wikipedia.org/wiki/Advanced_eXtensible_Interface
[29] HPS: Hard Processor System, available: https://www.intel.com/content/www/us/en/docs/programmable/683717/current/hard-processor-system-hps.html
[30] *Linux*, available: https://de.wikipedia.org/wiki/Linux
[31] *Mini-USB B*, Avalable: https://www.hb-digital.de/USB-20-Kabel-USB-A-Stecker-auf-Mini-USB-Stecker
[32] [putty] PuTTY, avaiable: https://www.putty.org/
[33] *microSC* Card, Avalable: https://de.wikipedia.org/wiki/MicroSD
[34] *Linux* image*, Altera, DE1_SoC_SD.img*, Available : http://www.terasic.com/downloads/cd-rom/de1-soc/linux_BSP/DE1_SoC_SD.zip
[35] *Altera*, *DE1-SoC Getting Started Guide*, Available: http://www.ee.ic.ac.uk/pcheung/teaching/E2_experiment/DE1-SoC_Getting_Started_Guide.pdf
[36] Rufus, available: https://rufus.ie/de/
[37] *Microsoft, Windows 10* Operating System Available: https://de.wikipedia.org/wiki/Microsoft_Windows_10
[38] Standard C++ Library reference, available*:* https://cplusplus.com/reference/
[39] *Texas Instruments, "REF5020, 2.048-V, 3-µVpp/V noise, 3-ppm/°C drift precision series voltage reference"*, Available Sep. 2022: https://www.ti.com/product/REF5020
[40] Kwang Liang Chong, "*Design, Manufacturing, Optimization and Characterization of a DC/DC Buck Converter Board Driving Up to 30V/4.5A for FPGA and µC Control*", BA Thesis, *OTH Regensburg, Electronics Lab*., Feb. 2019.
[41] Christoph Mayer, " *Reduzierung von Noise bei einer DC/DC-Buck-Konverter-Platine*", MEI-Master Projektarbeit, Elektroniklabor der OTH Regensburg, April 2022.

7 Appendix: Maintenance Details

This section with detailed realization and maintenance information is not intended for students in the standard practical training.

7.1 *DCDCbuck_Rev10.02* Properties

Table 7.1: Data used within the *VHDL* code

Identifier	Data Object	Data Type	Default	Units	Comments
Take-over of	**7-segment**	**displays by the**	***HPS's***	***AXI***	**interface**
test_key *within module ci_de1soc_ DCDCbuck*	CONSTANT	std_logic_vector (31 downto 0)	0xEF...F		*std_logic_vector*(31:0):="1110 1111 … 1111": If register *reg_readwrite_7 = test_key*, then the 7-seg display is controlled by the *AXI* interface of the *HPS* and displays by *hex# = reg_readwrite_#*(6:0), # = 0…5, which is used for example for the *hex_timer*. Otherwise the 7-seg displays are controlled by VHDL signals according to: *hex# <= hx#.*
fast counting	**loop**				**within module *de1soc_DCDCbuck***
start_fast	SIGNAL	std_logic	'0'		= '1' when *count_fast* = 0, else '0'
count_fast	SIGNAL	POSITIVE	1		1…*pwm_period*, counts clocks of 50MHz-clock signal *clock_50*.
smpl_v	CONSTANT	POSITIVE	330*1/6		phase position of *v(out)*-sampling within interval 1…*pwm_period*.
smpl_iL	CONSTANT	POSITIVE	330*3/6		phase position of *iL*-sampling within interval 1…*pwm_period*.
smpl_iout	CONSTANT	POSITIVE	330*5/6		phase position of *iout*-sampling within interval 1…*pwm_period*
slow counting	**loop**				**within module *de1soc_DCDCbuck***
start_slow	SIGNAL	std_logic	'0'		= '1' when *count_slow* = 0, else '0'
count_slow	SIGNAL	POSITIVE	1		1…*SMPL_period*, counts PWM samples of length *pwm_period*..
SMPL_vref	CONSTANT	POSITIVE	5		phase position of *vref*-sampling within interval 1…*SMPL_period*.
SMPL_v_av	CONSTANT	POSITIVE	15		phase position of *v_av*-sampling within interval 1…*SMPL_period*.
SMPL_period	CONSTANT	POSITIVE	20		count_slow=1…SMPL_period, *count_slow*++ if *count_fast =-0*.
Values	**to be**	**sampled**			**within module *de1soc_DCDCbuck***
v	SIGNAL	NATURAL	1250	mV	output voltage
iL	SIGNAL	INTEGER	0	mA	inductor current
iout	SIGNAL	INTEGER	0	mA	output current
vref	SIGNAL	NATURAL	2048	mV	Sampled instead of *iout* in sampling intervals of *SMPL_period* when *SMPL_vref* ≤ *SMPL_period*, else default 2048 acc. to [39].
v_av	SIGNAL	NATURAL	0	mV	Sampled instead of *iout* in sampling intervals of *SMPL_period* when *SMPL_v_av* ≤ *SMPL_period*, else 0. For *DCDCbuck* Board revisions that cannot sample *ADC_IN3* (Rev ≤ 11.01), we sample *v_av = v*(at phase position of *iout*).

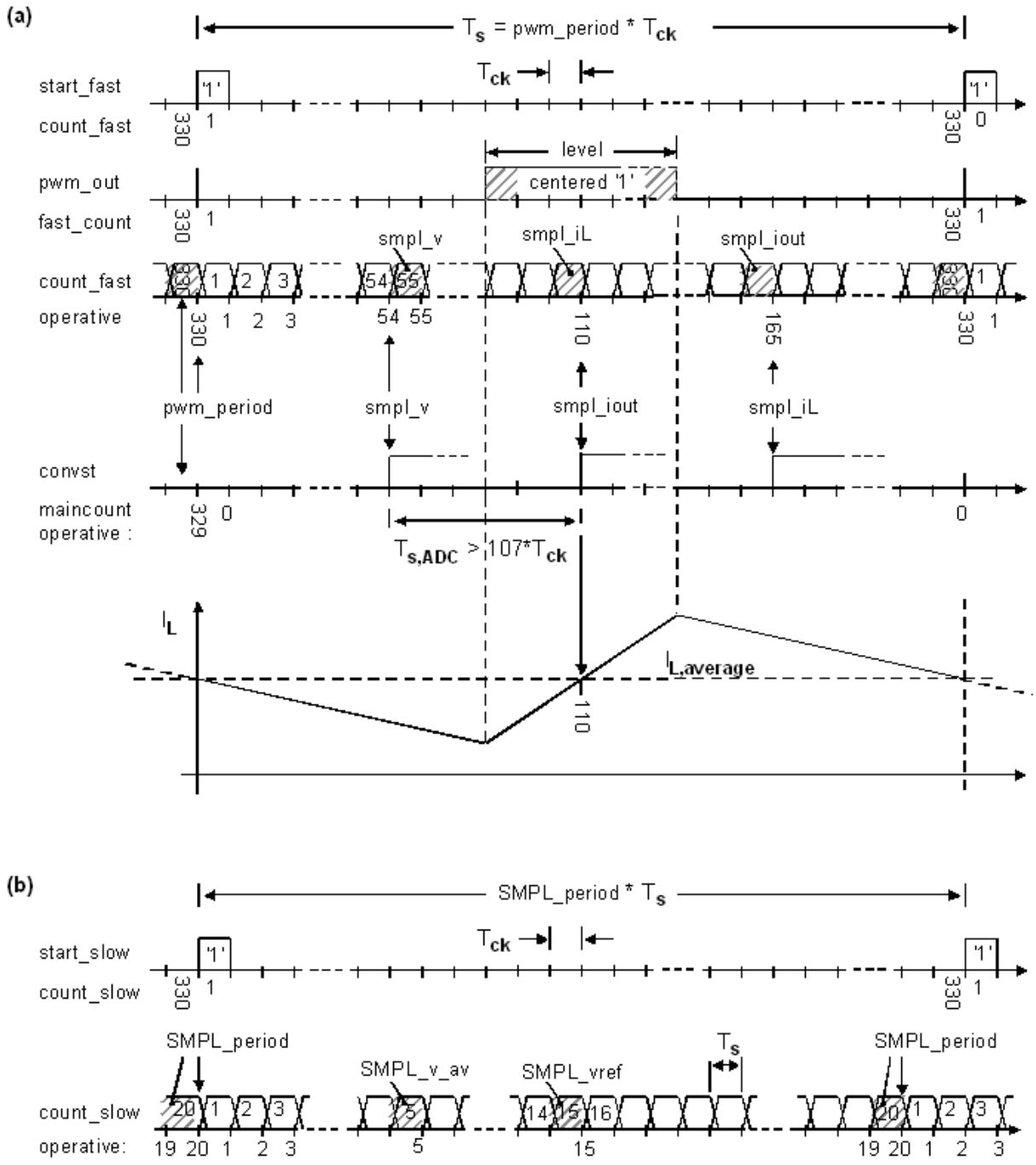


Fig. 7.1.1: Timing diagram with starting-'1' output, T_{ck} = 1 / f_{ck} = 1/ 50MHz. Sampling coil durrent I_L in the middle of the High- or Low-phase delivers its average value in CCM.

7.2 Preparations of Embedded Exercises for Practical Training

This part is typically not done during the standard practical training.

7.2.1 Modifications to be done at the VHDL code to include the HPS:

(This is done for files of the practical training)

1. Copy file AxiInterface.vhd to directory ...\Models_DCDCbuck\VHDL\Quartus\ci_de1soc_DCDCbuck_hps
2. Copy entity instantiation i_AxiInterface: entity work.AxiInterface(arch_AxiInterface) PORT MAP(...) into ci_de1soc_DCDCbuck.vhd
3. Add file hps.vhd to Quartus project (Project > Add/Remove File in Project > hps.vhd > Apply > OK
4. Add file AxiInterface.vhd to Quartus project (Project > Add/Remove File in Project > AxiInterface.vhd > Apply > OK
   Sequence in Window of: Project > Add/Remove Files in Project >
   1. AxiInterface.vhd, 2. hps/ip/hps/synthesis/hps.vhd, 3. hps/ip/hps/synthesis/hps.qip
   (a) Clear entry in library properties of hps/ip/hps/synthesis/hps.vhd
   (b) compile -> error within fitter
   (c) run tcl script *hps_sdram_p0_pin_assignments.tcl* using Tools > Tcl Scripts >
   (d) compile again (should work now)
5. signals reg_w, reg_v, reg_iL, reg_iout, reg_vref, reg_voutav into the system:
   (a) Within directory ...\Models_DCDCbuck\VHDL\Quartus\ci_de1soc_DCDCbuck_hps,
   architecture rtl_ci_de1soc_DCDCbuck OF ci_de1soc_DCDCbuck, add
   (i) component declaration of de1soc_DCDCbuck, add the 4 port signals:
   reg_w,reg_v,reg_iL,reg_iout: OUT std_logic_vector(31 downto 0));
   (ii) Add signal declaration:
   "SIGNAL reg_w,reg_v,reg_iL,reg_iout,reg_vref,reg_voutav: std_logic_vector(31 DOWNTO 0);" to ci_de1soc_DCDCbuck
   (iii) Add the 4 signal in component instantiation of de1soc_DCDCbuck
   i_fpga:de1soc_DCDCbuck ... PORT MAP(...,reg_w,reg_v,reg_iL,reg_iout);
   (b) Within directory ...\Models_DCDCbuck\VHDL\VHDL\rtl\de1soc
   add the 4 signals into the PORT of entity de1soc_DCDCbuck:
   ENTITY de1soc_DCDCbuck IS ...
   PORT(...,reg_w,reg_v,reg_iL,reg_iout:OUT std_logic_vector(31 downto 0));

7.2.2 Preparation of a *microSD* Card.

We have to copy a *Linux* [30] image on the *microSD* card [33]. First download the file *DE1_SoC_SD.zip* (66 495 KB) from [34] und unpack the required image named *DE1_SoC_SD.iso* (1 899 724 KB) as detailed in the "*DE1-SoC Getting Started Guide*" [35]. Then a Software like *Rufus* [36] can be used on the *Microsoft Windows 10* [37] operating system (OS) to copy the image bit-accurate on the *microSD* card as illustrated in Fig. 7.1.2. Use button "*Auswahl*" to select the image *DE1_SoC_SC.img*. Click on button *START* to write the image on the *microSD* card, click on *SCHLIESSEN* to quit the *Rufus* Software.

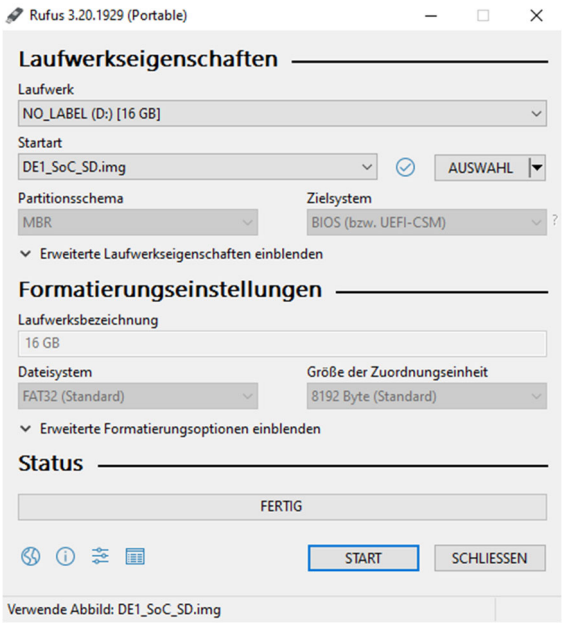


Fig. 7.1.2: Using *Rufus* [34] under Windows 10 to copy an image on a *microSD* card.

7.2.3 First *Linux* Operations on the ARM Processor

We assume that the *monitor* and *set_w* executable programs are not yet on the *microSD* card.

Listing 7.1.3: first *Linux* operations on the *ARM* processor

```
root@socfpga:~# ls                    // list directory
root@socfpga:~# ls -al ls             // list directory with all properties
root@socfpga:~# mkdir trace_data      // create directory trace_data/
root@socfpga:~# cd trace_data         // go into directory trace_data/
root@socfpga:~# ls -al                // go into directory trace_data/
root@socfpga:~# mkdir bin             // create directory bin/
```

7.2.4 Preparing the *microSD* card on a *Linux* Computer

Checking the *microSD* card on a *Linux* computer, we will see that it contains 2 partitions now with a size of 589 MB and 1100 MB. The latter contains the directories *home/* and *home/root/*. Within the latter we find our selfmade directory and file *home/root/trace_data/monitor*.

Some basic knowledge for operation within Linux environment

- Getting the terminal window path: *right mouse button* in window → *open in Terminal*
- "~/" is root directory, "./" is this and "../" is parent directory
- **`mkdir`** and **`rmdir`** are make and remove directory.
- Permission problems: precede a command by **`sudo`** (*super-user do*)
- Rename something: **`$ mv <name1> <name2>`** or **`$ sudo mv <name1> <name2>`**
  Example with permission: **`$ sudo mv meas trace_data`**
- Grant to all directory contents read/write/exe rights: **`$ sudo chmod 777 ./`**
- Remove recursively (i.e. also directory contents): **`$ sudo rm -r <filename>`**
- Make directory: $ **`mkdir trace_data`**
- Change directory: **`$ cd trace_data`**

Create executable monitor.

- create directory *trace_data* and go into this directory
- copy *main.c* and *MakeFile* into *trace_data/* and create there an empty folder *bin/*
- in window *trace_data/* click *right mouse button* → *open in terminal*
  type command **`make`**.
  If **`make`** does not work properly, install *C* compiler for *ARM* using
  `$ sudo apt install gcc-arm-linux-gnueabihf`
- Make should create in *bin/* files *monitor* and *main.o*; whereas *main.o* may be deleted
- **`$ ls -al`** // do my files have sufficient rights to be copied?
- **`$ chmod 777 *`** // grant all rights to all files in the actual directory.
- Copy

7.2.5 Testing the Own C Program *microSD* card on *Linux* on *ARM*

Listing 7.1.5: Run *Linux* on the *ARM* processor to monitor data

… *DCDCbuck_Rev10* board must be connected and run at this point to deliver the required monitoring data
 # ./monitor // print 50 measurement lines, *DCDCbuck_Rev10* must run at this point!
<make *PuTTY* window COM# wider using the mouse to avoid undesired line feeds>
 # ./monitor 10 // example 10 is the number of desired measurement lines
 # ./monitor 10 > monitor.log // write output into file *monitor.log*.
 # ./cat monitor.log // print the contents of *monitor.log* in the PuTTY window.
 # ./set_w 1234 to set *w*=1234mV, set *sw*(9:6) ="1111" to make that value of *w* active.

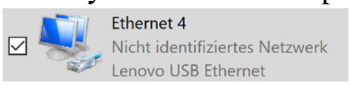
7.2.6 Install Ethernet Connection to *DE1-SoC* Board

Linux operations in the PuTTY window

- Open Putty: **cd /etc/network**
- **vi interfaces** // open file *interfaces* with editor **vi**
- **d** <to delete the 2 lines containing string *eth0*>
- **i** → *Enter* to insert a line // key 'i' for insert
- insert the following 3 lines:
  iface eth0 inet static
  <tab> **address 10.0.0.2** //region 10, board addressed by trailing 2, PC by 1
  <tab> **netmask 255.255.255.0**
- *ESC* **:wq** → *Enter* // write (save) file and quit
- **cat interfaces** // print file interfaces to check its contents
- **ifup eth0** // interface upload

Windows operations

- Hit *Windows key* , type **Ethernet** → *Return* → *click on Adapteroptionen ändern*
- Identify the *Ethernet* adaptor connected to the *DE1-SoC* board, left-click on it



- Window *Status of Ethernet#* opens →click on *Eigenschaften* (*Properties*)
- Click on *Internet-Protokoll, Version 4* →
- Set *IP-Adresse 10.0.0.1*, *Subnetz-Maske 255.255.255.4*.

(a) status window (b) adaptor properties (c) TCP/IPv4 settings

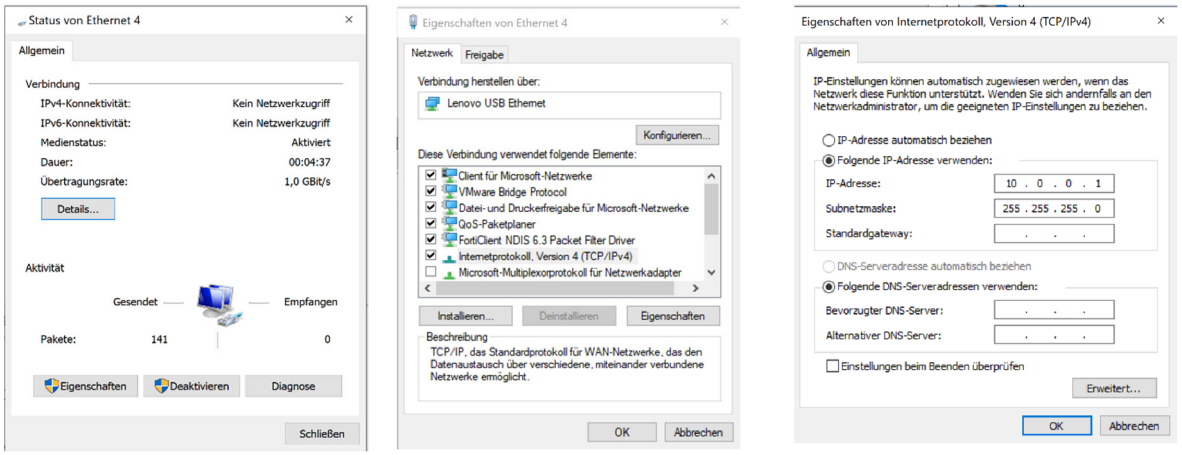


Fig. 7.2.6: Ethernet setting windows

Linux operations in PuTTY window, after every start on the *microSD* card:

- **`ifup eth0`** // upload interface setup
- **`ping 10.0.0.1`** → *CTRL+C* // ping must reach target, *CTRL+C* stops ringing
- User: **`root`** <enter> → Password: **`root`** <enter>

Window operations to see the microSD card:

- Open *WinScp* → *neues Verbindungsziel*:
  SFTP, *Serveradresse 10.0.0.2*, *Port 22*, *User=root*, *Pwd=root*, *Verbindung vertrauen* → *ja*
- Now you can exchange data between *Windows* ⇔ *microSD* card using *WinSCP*.

7.2.7 Save and Duplicate an Image-File Using *Linux Ubuntu*

- Select *Open in Terminal* on *Linux* desktop background
- **`ls /dev`** // list devices
  sda = memory of the virtual machine (VM) with *Linux*
  sdb = memory of *microSD* card
- Commands used below: **`dd`**: disc dump, **`if`**: input file, **`of`**: output file, **`bs`**: block size, **`count=`**#: number of block to be copied, **`status=progress`** displays operation progress.
- Copy an image of the *microSD* card to a file in Ubuntu (will be 2500 MB big!)
  `sudo dd if=/dev/sdb of=microSD_ref_0 bs=1M count=2500 status=progress`
- Copy the image in Ubuntu to the *microSD* card (2500 MB = all → may be omitted)
  `sudo dd if=microSD_ref_0 of=/dev/sdb bs=1M count=2500 status=progress`
- Attention: if accidently sda is used, the *Linux* within the virtual machine is overwritten!

7.3 Board Revision History

Board revisions are noted as *Rev##.xx.yy* whereas ## corresponds to the type of the board (typically a thesis product), *xx* is the number of the board fabricated and *yy* its design software update.

- Quang Liang Chong, an opto-electronics physicist, designed and assembled *DCDCbuck Rev5* board shown in Fig. 7.3.1 as his bachelor thesis [40]. It was the first board in the *DCDCbuck* series that was suitable and duplicated for use in practical training.

VHDL code was written by the author, except the *AxiInterface*, which was coded by Alexander Forster [11]. He also delivered the *C*-Code and the techniques to interconnect the *DE1-SoC* board and its hard-processor system (*HPS*) via *USB* and *Ethernet* to the outer world.

7.3.1 *DCDCbuck_Rev5* of Kwang Liang Chong

Kwang Liang Chong produced in his bachelor thesis the board *Rev5* [40], which was the first to be duplicated for student use in practical trainings of the *OTH Regensburg's Electronics Lab*. A single feedback wire to input channel IN1 of the ADC is used for output voltage feedback.

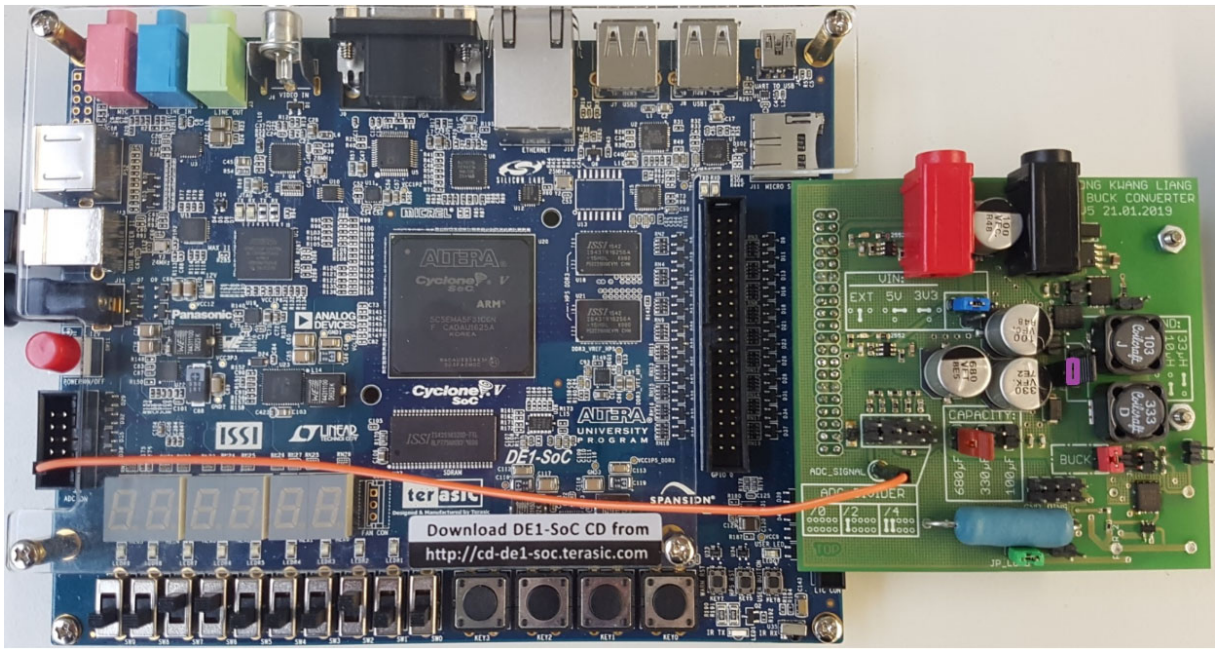


Fig. 7.3.1: Board *DCDCbuck_Rev5* of Kwang Liang Chong [40] on DE1-SoC board. A single feedback wire to the *ADC*'s input channel *IN1* is used for output voltage feedback.

7.3.2 *DCDCbuck_Rev10* of Florian Schwankner

Florian Schwankner produced in his *MEI* master project thesis [10] the Rev10 board illustrated in Fig. 2.1, which was the first board including inductor and output current measurements. To allow for positive and negative inductor currents – particularly required for the inductor –, these measurements employ differential techniques. Feedback to the digital side occurs via the 10-wire ribbon cable seen in Fig. 2.1. This board still features relatively high switching noise. It was the second board to be used in practical training of the *Electronics Lab*.

7.3.3 *DCDCbuck_Rev11* of Christoph Mayer

Christoph Mayer produced in his *MEI* master project thesis board *Rev11* [41], illustrated in Fig. 7.1.3. Its main goal was to reduce switching noise. Chucks were introduced and the board was constructed over the *DE1-SoC* carrier board to get the feedback wires as short as possible. Switching noise suppression success was moderate. This board features some mechanical and measurement problems and was not yet used in practical trainings of the *Electronics Lab*.

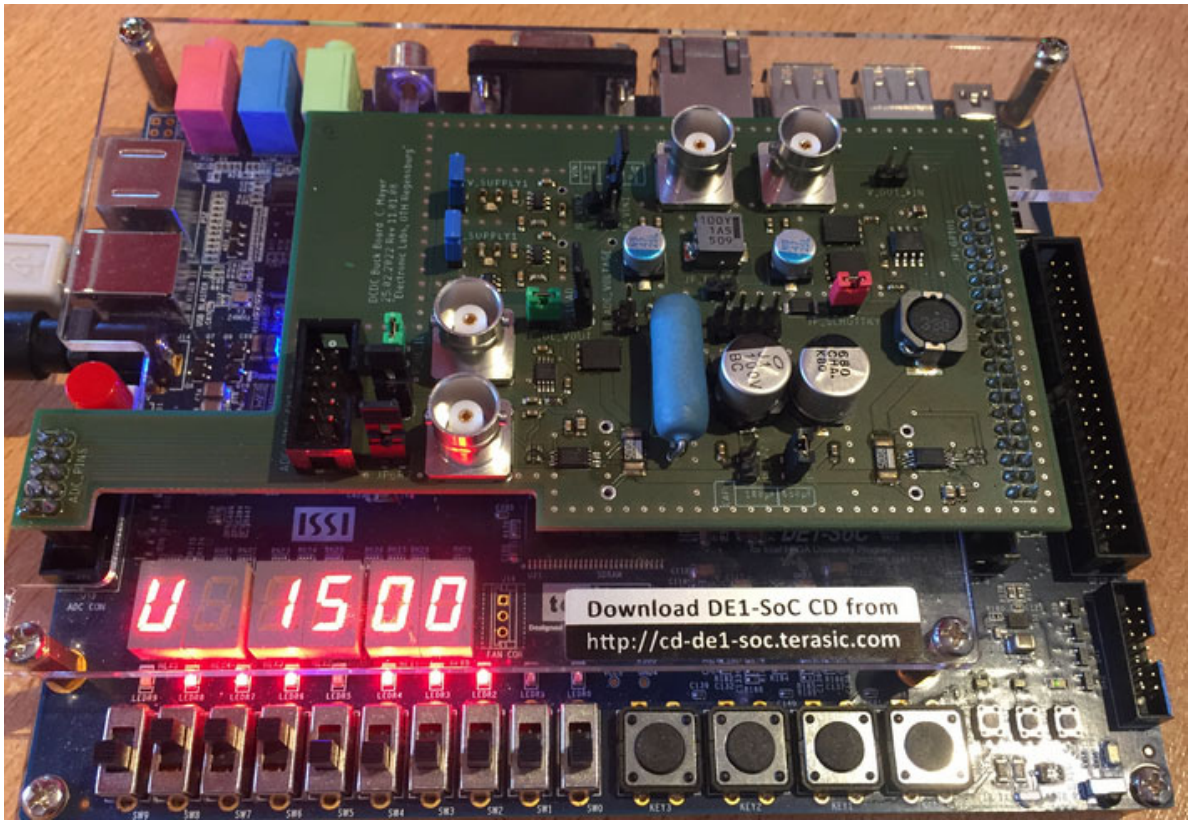


Fig. 7.3.3: Board *DCDCbuck_Rev11* of Christoph Mayer [41], constructed over the *DE1-SoC* board to avoid long feedback lines and switching noise.