B00st converter

van Iterson, Arne
Student Number: 1798423

Selier, Tom
Student Number: 1808444

February 4, 2024

Abstract

This report describes the design, testing methods and behaviour of a DC-DC Boost converter using BS170 MOSFETs built by students of the University of Applied Sciences Utrecht.

1 Introduction

Students of the Electrical Engineering department of the University of Applied Sciences Utrecht are to design a DC-DC Boost converter in the fifth semester during the Hardware Design course.

The goal of the project is to design a DC-DC Boost converter on breadboard that can step up a voltage of around $3V$ to any voltage between $3.3V$ and $7V$. The converter should be able to deliver a current of $50mA$ at any given voltage. The entire system is to be controlled by a STM32F4 series microcontroller which should provide a way to control the output voltage through software.

An example circuit and a couple of components are provided and mandatory to use, such as the inductor, the capacitor and the MOSFETs. Apart from these components, the circuit can be altered and or extended to the student's liking to improve its function as long as the final circuit is able to deliver the required current and voltage.

2 Circuit Description

According to The Art of Electronics [1], the simplest implementation of a step-up or boost converter consists of just four components; An inductor, a switch, a diode and a capacitor connected as follows:

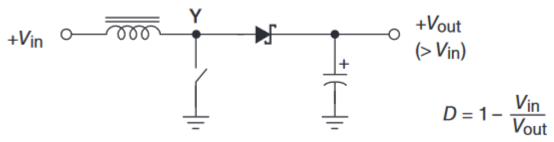


Figure 1: Basic boost converter

It describes the function of a boost converter as follows:

> During switch conduction (...) the inductor current ramps up; when the switch is turned off, the voltage at point Y rises rapidly as the inductor attempts to maintain constant current. The diode turns on, and the inductor dumps current into the capacitor. The output voltage can be much larger than the input voltage.

The image below shows the final circuit used. As shown, the circuit includes just a few modification to the example circuit provided.

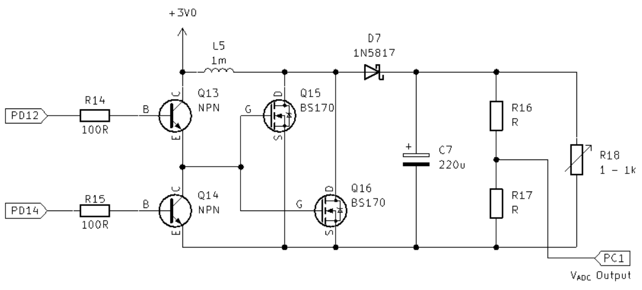


Figure 2: Final version of the boost converter

The circuit has been through several iteration throughout the project, however, the following key components have not been changed throughout:

- The inductor, a $1mH$ inductor
- The diode, a $1N5817$
- The capacitor, a $220\mu F$ capacitor
- The MOSFETs, BS170 series

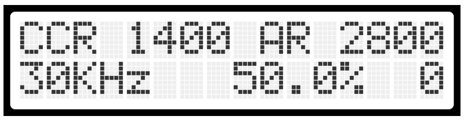
In the example circuit, there was only one MOSFET and it was directly controlled by the MCU. This circuit was able to reach a voltage of $7V$, but not at the required current. This is due to the fact that the BS170 series MOSFET has a fairly high $R_{DS(on)}$ of 1.2Ω [2]. To increase the output current, a second MOSFET was added to the circuit. This way, the $R_{DS(on)}$ is halved, effectively doubling the current at which the inductor can be charged if we neglect the resistance of the inductor itself. The downside of adding a second MOSFET is that it also doubles its input capacitance, however, with an additional circuit modification this can be overcome.

A second addition to the circuit is the push-pull driver on the gate of the MOSFETs. The driver consists of two NPN transistors that are controlled by inverted signals from the MCU, this way the MOSFET gates are quickly switched between $0V$ and $3.3V$. The push-pull driver was chosen over a pull-down resistor because it would cause several issues; A low resistor value would quickly discharge the gate capacitance, but it would also decrease the driving current when turning the MOSFET on. A high resistor value would have little effect on the gate capacitance. The push-pull driver solves both of these issues by quickly discharging the gate capacitance by connecting them to ground while not effecting the driving current when turning the MOSFET on. This way, the MOSFETs are switched faster, increasing the efficiency of the circuit.

The final addition is a voltage divider of $15k\Omega$ and $10k\Omega$ on the output, this simply lowers the output voltage to a level that can be read using the ADC on the MCU and is used for control loop feedback.

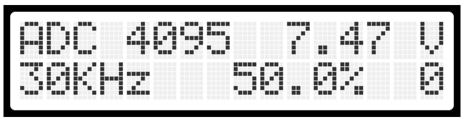
2.1 Control

The system is controlled by the STM32F407 on the HU development board. The board provides the required PWM signals to control the MOSFETs and the ADC to read the output voltage. The program provides a small interface in which the user can view and control the duty cycle and frequency of the switching and read the current output voltage. The program also implements a simple PI control loop to control the output voltage, however it is not yet fully functional and requires more tuning.



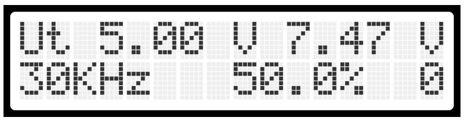
```
CCR 1400 AR 2800
30KHz    50.0%  0
```

Figure 3: Display readout of PWM Control registers, frequency and duty cycle



```
ADC 4095  7.47 V
30KHz    50.0%  0
```

Figure 4: Display readout of ADC value and converted voltage



```
Ut 5.00 V 7.47 V
30KHz    50.0%  0
```

Figure 5: Display readout of target and current output voltage

3 Methodology

To characterize the system, several tests have been performed. The characteristics of interest are the following:

1. Efficiency
2. Noise
3. Ripple characteristics
4. Start up

In this section a test or measurement will be described for each of the above characteristics.

Each of the characteristics have been tested at two different output voltages and various load currents. The different voltages are $7V$ and $3.3V$. The chosen load currents are 10, 20, 30, 40 and $50mA$. These values were chosen to characterize the circuit over a broad range of conditions. Lastly, the switching frequency (PWM frequency to the MOSFETs) of the STM was locked to $30kHz$. The switching frequency was found by testing the efficiency of the circuit.

For all tests, the data was handled in a simular way. For test 2 through 4, an oscilloscope was set up on the output voltage. The probe was set to 10x attenuation to minimize it's influence on the circuit. The oscilloscope's settings are set to get the signal full screen, and the acquire settings were adjusted so that it would store 20,000 points. Then, at each measurement .csv (comma seperated values) was stored on a USB drive.

After the measurements were collected, they were processed using a python script. The major functions are listed at each test.

3.1 Efficiency

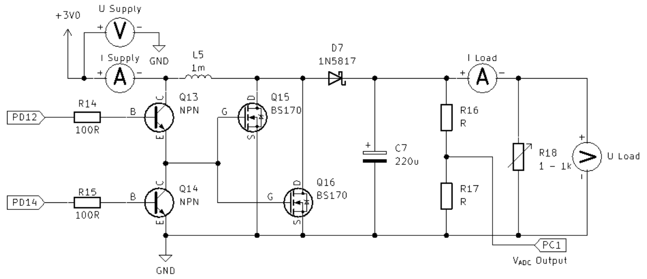


Figure 6: Circuit with multimeters

To measure the efficiency of the circuit, four measurements were taken. A current and a voltage measurement were taken at the supply and load, respectively. The measurements were taken as shown in figure 6. The energy used by the supply and the load can be calculated using equation 1. Then, using equation 2, efficiency can be calculated.

$$P[W] = U[V] \cdot I[A] \tag{1}$$

$$\eta[\%] = \frac{P_{load}[W]}{P_{supply}[W]} \cdot 100\% \tag{2}$$

3.2 Noise

To measure the noice of the circuit an oscilloscope probe was placed on the variable resistor in figure 2. Over the period of 1 millisecond, 20,000 points were measured.

Noise has several metrics in which it can be quantized. Two metrics were calculated, the standard devation (SD) and the peak to peak noise.

3.2.1 Peak to peak

Peak to peak is the simplest way to look at noise. The signal has a stationary mean over the period of 1 millisecond. Thus, the highest measured value can be subtracted from the lowest measured value.

```python
def PK_PK(all_data, ch):
  output_pk = []
  for data in all_data:
      maximum = max(data[ch + 1])
      minimum = min(data[ch + 1])
      pk_pk = maximum-minimum
      output_pk.append(pk_pk)
  return output_pk
```

Listing 1: Peak to peak function

3.2.2 Standard Deviation

The second metric used to measure noise was the standard deviation (SD). Unlike peak to peak, it gives a better impression of the average noise over a longer period. SD can be calculated using equation 3.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(x[i]-\mu)^2} \tag{3}$$

Where $x[i]$ is each voltage measurement, μ is the mean of the signal and N is the total amount of samples.

```python
def SD(all_data, ch):
  output_SD = []
  for data in all_data:
      N = len(data[ch + 1])
      MU = sum(data[ch + 1])/N
      x = 0
      for val in data[ch + 1]:
          x += (val-MU)**2
      SD = sqrt((1/N) * x)
      output_SD.append(SD)
  return [output_load, output_SD]
```

Listing 2: SD function

3.3 Ripple characteristics

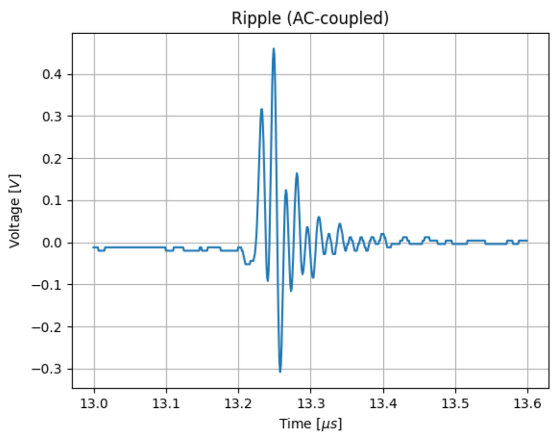


Figure 7: Close up of a ripple

A significant source of the noise was caused by a specific ripple, shown in figure 7. This ripple coincided with the MOSFETs opening or closing.

To further characterize this behaviour a close up measurement was taken. The oscilloscope was set to AC-coupling and the settigns were adjusted for the ripple to be full screen. Then, two additional characteristics can be calculated. The ripple's peak to peak voltage and the ripple's (most prevalent) frequency. The peak to peak value can be calculated using the method described in section 3.2.1.

To measure the frequency of the signal using an FFT, it had to be pre-processed first using a Hamming window, this eliminates sharp edges at the edge of the measurement, causing unwanted frequencies to appear in the frequency domain.

$$w(i) = 0.54 - 0.46 \cdot cos\left(2\pi\frac{i}{N}\right) \tag{4}$$

Where i is the current sample and N is the total amount of samples. Each sample in the signal can be multiplied by the corresponding value in the window, preparing the signal for the FFT.

```python
def window(y):
  N = len(y)
  hamming = []
  for n in range(N):
      hamming.append(
        0.54 - 0.46 * cos(2*np.pi*(n/N)))
  y = [hamming[i]*x for i, x in enumerate(y)]
  return y
```

Listing 3: Hamming function

```python
def Freq(all_data, ch):
output_freq = []
for data in all_data:
    # FFT
    # using realfft,
    # because the signal only has real parts
    y = np.fft.rfft(data[ch + 1])

    # Window
    y = window(y)

    # Calculate the bins
    dt = data[1][1] - data[1][0]
    x = np.fft.rfftfreq(len(data[ch+1]), dt)

    # find the maximum, max() and np.argmax()
    # are not playing nice with
    # imaginary numbers
    maximum = 0
    max_idx = 0
    offset = 1  # Skip the first bin, DC offset
    for idx, val in enumerate(y[offset:]):
        mag = sqrt(val.real**2 + val.imag**2)
        if mag > maximum:
            maximum = mag
            max_idx = idx + offset

    # get the frequency of the maximum
    output_freq.append(x[max_idx])

return output_freq
```

Listing 4: FFT function

3.4 Start up

The last characteristics is the start up, specifically the different rise times under load. The voltage was measured at the output as the supply was turned on.

Different rise times can be defined. First off, τ and 2τ were defined as 63% and 95% respectively. Further more, 'rise time' was defined as 90%, a metric used often in control theory.

A problem that occured during the measurements, is that the aforementioned ripples and noise would cause erroneous readings. As such, the signal was filtered using a low pass filter ($\tau = 0.2ms$), reducing the high frequencies from the signal.

```python
initial = data[3][0]
dt = data[1][1] - data[1][0]

x_filter = [initial]
for i in range(1, len(data[3])):
    x_dot_filter = \
      (data[3][i] - x_filter[i-1])/0.0002
    x_filter.append(
      x_filter[i-1] + x_dot_filter*dt)
```

Listing 5: LPF snippet

4 Results

In this section the results from section 3 will be discussed, as well as discuss some probable causes for unknown or unintended results.

4.1 Efficiency

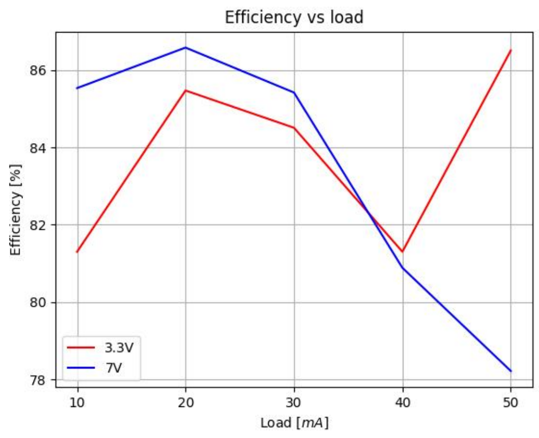


Figure 8: Efficiency vs load

The results for the efficiency measurements, as described in section 3.1 are displayed in figure 8. The $7V$ measurements follow a predictable curve, however, the $3.3V$ makes an unexplained jump back to a higher percentage.

4.2 Noise

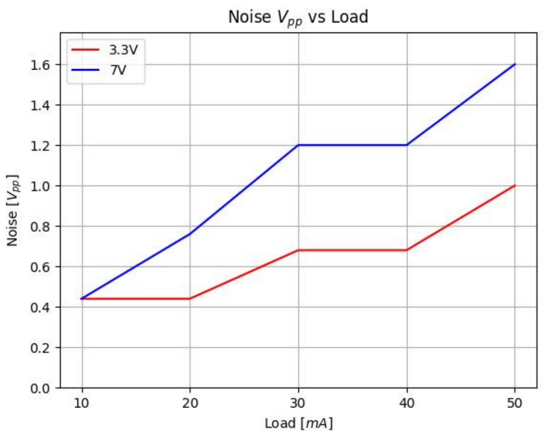


Figure 9: Noise V_{pp} vs load

The results for the noise measurements, as described in section 3.2.1 are displayed in figure 9. The peak to peak voltage is a significant fraction of the output voltage, with $3V$ peaking at 33%. It seems there is a relation between peak to peak voltage and the output voltage as well, as $7V$ has more noise than $3.3V$

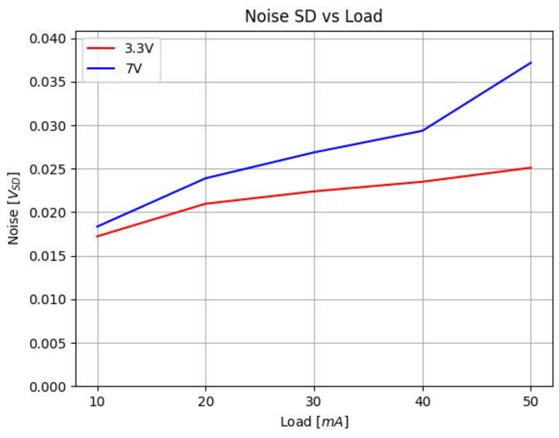


Figure 10: Noise SD vs load

The results for the noise measurements, as described in section 3.2.2 are displayed in figure 10. Although the voltage peaks are high, the standard deviation of the noise is in the range of millivolts. The trend that a higher output voltage has more noise is continued in this graph.

4.3 Ripple

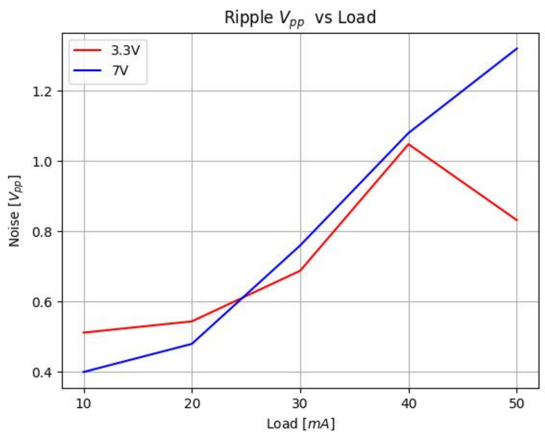


Figure 11: Ripple V_{pp} vs load

The results for the ripple measurements, as described in section 3.3 are displayed in figure 11. The voltage level in the graph seems to confirm that the peak to peak noise, seen in section 4.2 is caused by the ripple.

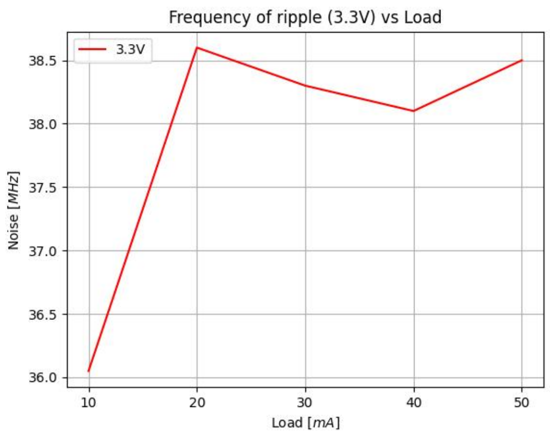


Figure 12: Frequency ripple vs load

The frequency of the ripple is roughly $38MHz$ and independant of the load. To figure out if this ripple is caused by the combination of the inductor and the capactitor the following equation can be used.

$$f = \frac{1}{2\pi\sqrt{LC}} \tag{5}$$

Using the values from figure 2, the resonating frequency of the circuit should be around $27kHz$. Thus, this cannot be the cause of the high frequency. As the frequency of the ripple is magnitudes higher than the LC-circuit's resonant frequency, what is seen is most likely the Self Resonating Frequency (SRF) of the inductor. Typically the SRF is $> 10MHz$, so that could be a probable source of the high frequencies.

4.4 Start Up

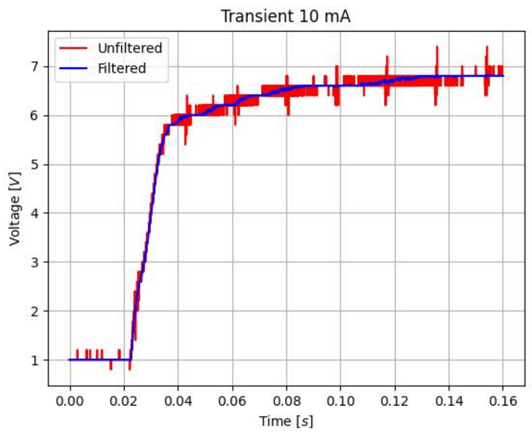


Figure 13: Start up 10 mA

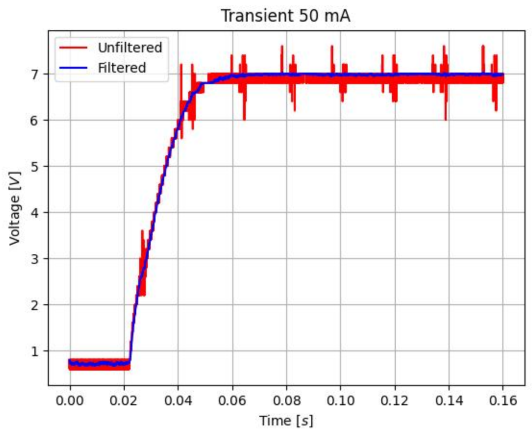


Figure 14: Start up 50 mA

Table 1: $10mA$

Metric	τ	2τ	Rise time
Percentage [%]	63	95	90
Time [s]	0.031	0.075	0.053

Table 2: $50mA$

Metric	τ	2τ	Rise time
Percentage [%]	63	95	90
Time [s]	0.033	0.048	0.043

The results for the start up measurements, as described in section 3.4 are displayed in figure 13 and 14, and table 1 and 2.

Counterintuitively, the rise time is shorter with a higher load.

5 Conclusion

In conclusion, the circuit does what it's supposed to do. It regulates well between $7V$ and $3.3V$ and it's able to supply the required loads. The efficiency of the circuit is respectable, with a peak efficiency of 86% and a lowest efficiency of 78%. The output noise is a problem, and it would be too high for normal use. Lastly, the PI controller works, but can easily be saturated and become unstable.

5.1 Recommendations

The circuit can be improved in several ways. The most significant improvement would be to replace the breadboard in favour of a soldered PCB, this would significantly improve the stability of the system and reduce noise. During testing, it was found that Heisenberg's uncertainty principle was enough to influence the behaviour of the circuit. Simply observing the output voltage was enough to make it fluctuate.

Using MOSFETs with a lower $R_{DS(on)}$ compared to the BS170 would allow the inductor to charge quicker, increasing the efficiency of the circuit.

The current push-pull driver could be replaced with a dedicated gate driver, such as the IR2125 IGBT Driver IC. This would allow for a more stable and faster switching of the MOSFETs, while reducing the amount of IO used on the MCU.

As mentioned in section 2.1, the control loop can be improved. The current implementation of the PI controller is rather unstable and requires more tuning.

References

[1] P. Horowitz and W. Hill, *The Art of Electronics.* Cambridge University Press, 2022.

[2] Onsemi, "Bs170 data sheet," Apr 2022. [Online]. Available: https://www.onsemi.com/pdf/datasheet/mmbf170-d.pdf