Smart Switching Regulator

Student: Niculescu Vlad

Mentor: Herbert Pötzl

29th August 2017

Link to code: https://github.com/vladniculescu/GSOC

1 Introduction

The main goal of this project is to design an adjustable converter for the Apertus AXIOM Beta camera system. The AXIOM requires multiple voltages, so an adjustable solution is preferred over having multiple converters. Multiple instances of this converter will be utilized in the Beta, each one working at a different output voltage, but with identical hardware components. The performance criteria of the converter are efficiency, accuracy and ripple. The whole converter will represent an individual block, which will communicate with the central processing through I2C. The input supply voltage is 5V.

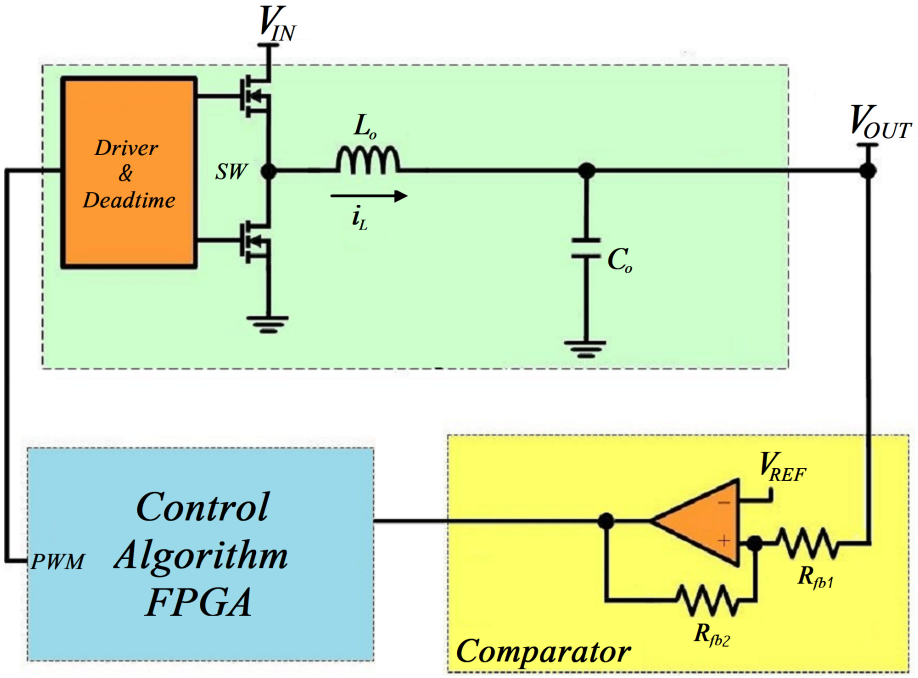

Figure 1: The Block Schematic

The synchronous configuration was preferred because of efficiency considerations. The chosen driver is controlled by a single PWM signal and process it in order to properly drive the two transistors – it generates two complementary signals and also adds deadtime. The output voltage is maintained constant due to the feedback mechanism, which This consists of a hysteresis comparator that indicates if the actual voltage is above or below a reference voltag. The control algorithm runs on an FPGA and controls the converter according to the information received from the comparator. Its main purpose is to maintain the output voltage as close as possible to the reference. This reference is generated by a 12-bit DAC an thus can be easily adjusted. The output voltage follows the

reference voltag regardless of the drawn current. Both the DAC and the control algorithm act as I2C slaves which allow the I2C master to modify essential parameters. While the DAC's I2C allows for a reference change, the converter's I2C gives control over the converter's parameters, like switching frequency or the speed of the feedback loop. Both slaves will be connected to the global I2C bus of the AXIOM system.

2 Designing the Converter

The first step in realizing this project was to design a schematic for a test setup as well as building a prototype to test with. A rendering of the designed prototype board can be seen in figure 2.

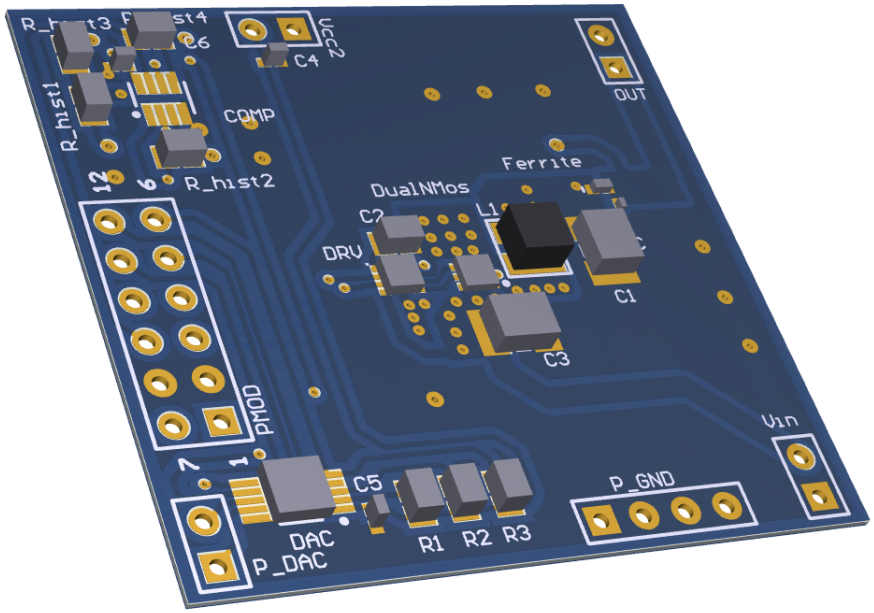

Figure 2: The PCB

Based on the design files, the PCB was fabricated at OSH Park and manually populated. Minor design problems could be easily fixed during initial testing.

3 The UART to I2C Bridge:

To simplify testing the hardware an UART to I2C bridge was designed to connect the system to a PC via serial port. This bridge allows real-time communication between computer and the converter which means that the converter's

parameters as well as the DACs output voltage can be adjusted from the serial console of the computer. The first step was to establish communication between computer and DAC to ensure that the bridge is working as intended. The FPGA development board used for this purpose was a Digilent Nexys Video, equipped with a Xilinx Artix 7. The data sent from PC is read by a UART module implemented in the FPGA. The received command is decoded and sent to the I2C bus controlling the DAC. Both UART and I2C Master are implemented as Verilog modules. The communication protocol consists of ASCII characters where numbers are sent as hexadecimal encoded strings specific characters are used for each command. For example, a sequence like "S07WFFW54Ws", writes the value 0x54 to register 0xFF for a device with the address of 0x07. "S" and "s" represent the start, respectively stop conditions on the I2C bus. After a command was successfully executed, the bridge responds with a "D" character as acknowledge.

The Bridge can also adjust the converter's parameters. The structure is very similar, but this time the first letter of the string command will not be "S".

Command	Effect	Length
S	Adjusts the DAC's settings	Variable
X	Modifies the switching frequency	16 bit
T	Adjusts the loop speed	16 bit

Table 1: The commands of the Bridge

Table 1Shows the adjustments that can be made using the Bridge. Those adjustments are done from the serial console.

The switching frequency can vary be adjusted between 0 and 780kHz with a 16 bit resolution. In order to modify it, a command like "Xxxxxs" should be sent from serial console, where "xxxx" represents a 16 bit number in hexadecimal format. Consequently, "XFFFFs" will set the frequency to its maximum value. The same mechanism works for the loop speed. A command like "Txxxxs" tells the algorithm how often to read the comparator. For a higher value of "xxxx" the loop will be slow.

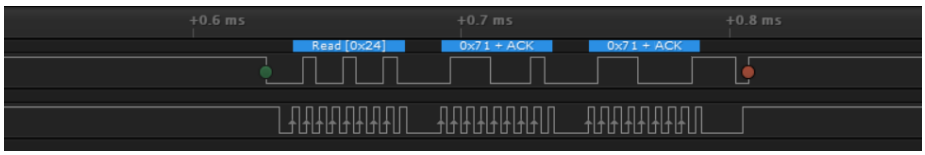

Figure 3: The I2C waveforms

A Saleae logic analyzer was used to test and debug the functionality of the I2C Master.

Example: Let's say that 2V are needed to be written in FPGA. This is how the DAC is programmed:

with								
Address	Ch1_MSb	Ch1_LSb	Ch2_MSb	Ch2_LSb	Ch3_MSb	Ch3_LSb	Ch4_MSb	Ch4_LSb

Table 2: The Package Structure for DAC configuration

Table 2 the structure of the package that has to be sent in order to configure the DAC.

Where the MSb and LSb mean most significant, respectively least significant bit. There are four channels because the DAC is a 12-bit four channel type. The ChX_MSbyte is a 4-bit number and ChX_LSbyte is an 8 bit one. Concatenating those two, the 12-bit value is obtained for each channel. So, returning to the example, "SC0W07WFFW07WFFW07WFFW07WFFWs" is sent from the serial console. S represents the start command, C0 the device address and W written after each two-number group says that the number has to be written (R is for read). The numbers are sent in hexadecimal format. Concatenating 07 and FF for each channel, 2047 is obtained. This is 2.05V for a 4.096V reference. This I2C Master module was designed just for development. It won't be necessary when the converter will be integrated in the AXIOM System, because it will be driven by this one through the I2C Slave module. Next, the Slave I2C for converter was implemented. This writes and reads numbers in a register file. Some registers store the parameters of the control algorithm.

4 The Comparator

The hysteresis comparator acts like a feedback mechanism which gives the input for the control algorithm. The hysteresis was introduced to reduce the switching frequency of its output. However, the hysteresis width was set to 80mV. A higher value would generate a higher ripple, too.

Figure 4 The comparator's output In order to obtain the desired value, a 62 ratio of the resistors was needed – in theory. To test if this value is also obtained in practice, a triangular signal was applied at the non-inverting input of the comparator, while a fixed voltage was applied at the inverting input. Figure 4 illustrates the triangular signal and also the comparator's output where the two thresholds are obvious.

5 The control algorithm

The control algorithm uses a fixed switching frequency and varies only the duty cycle of the PWM control signal. The parameters of this algorithm are the interval at which the duty cycle is updated and the switching frequency. The switching frequency was experimentally determined to be 780kHz. A counter is incremented whenever the output voltage is higher than the reference and decremented when this is smaller than the reference. When the update time

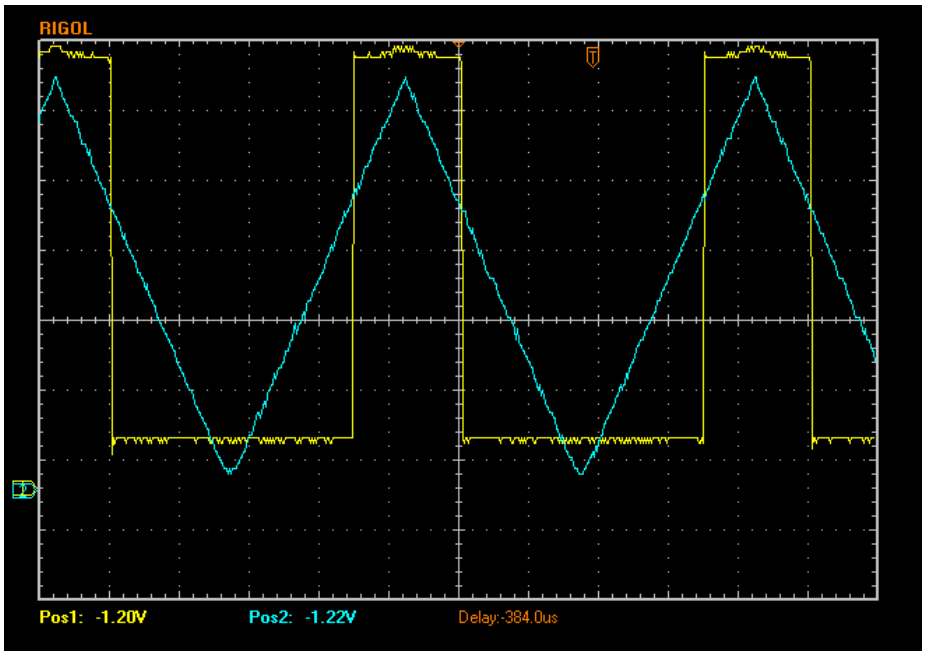

Figure 4: The comparator's output

comes, the counter is evaluated and the duty cycle is increased or decreased by 1, according to the value of this counter.

```
always @(posedge clock) begin
    ramp_prev <= ramp;

    if (ramp == 0 && ramp_prev == 63) begin
        test <= !test;
        if (fb) begin
            if (decimal == 65000 - fb_interval) begin
                decimal <= 65000;
                if (duty > 0)
                    duty <= duty - 1;
            end
            else
                decimal <= decimal - 1;
        end
        else begin
            if (decimal == 65000 + fb_interval) begin
                decimal <= 65000;
                if (duty < 63)
                    duty <= duty + 1;
```

```
            end
            else
                decimal <= decimal + 1;
        end
    end
end
```

Tuning the algorithm: The ripple was measured using the oscilloscope, measuring the peak-peak amplitude of the AC component. It has a high frequency component, which comes from the switching frequency and a lower frequency component generated by the action of the control loop. This is usually between 1 and 30 kHz. If the feedback loop actions too slow, the ripple will be higher, because a low frequency component won't be rejected by the LC filter.

Figure 5: The output ripple for a slow feedback loop

Figure 5 highlights the output signal when such situation occurs. The transitions between levels present a quite square form, which indicates a poor filtering effect. In this case, the feedback loop speed should be increased. Its proper value was found to be 96.

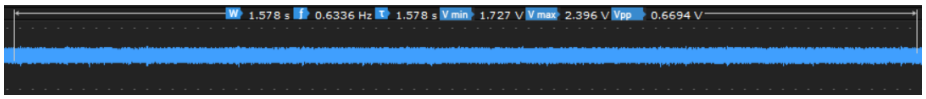

Figure 6: The output ripple for a fast feedback loop

However, if the feedback loop speed is set to a high value, its effect will be harmful, too. Because the algorithm uses only a proportional component, if this one is too aggressive will determine an increase in the output voltage at every iteration. Consequently, it will somehow oscillate between the average position will a high amplitude. Because the comparator's output turned out to be a little noisy, an averaging filter was introduced to reduce the noise.

In Figure 7, the effect of the filter is obvious, because the narrow high frequency spikes are absent. Finally, adjusting the switching frequency at 780kHz, it was obtained the waveform in Figure 8.

Figure 7: Filtered vs unfiltered comparator

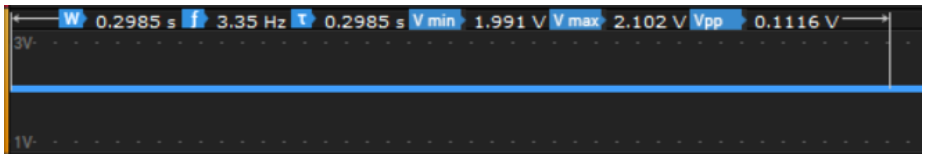

Figure 8: The output signal – 110mV ripple

The obtained ripple is around 110mV. This value is still poor for the requirements of this project. In order to reduce this ripple, an additional filter was introduced. This one is consisted of a ferrite bead and an 1uF capacitor. Figure 9 presents the output ripple after this additional circuit was added. The ripple was reduced to 30mV.

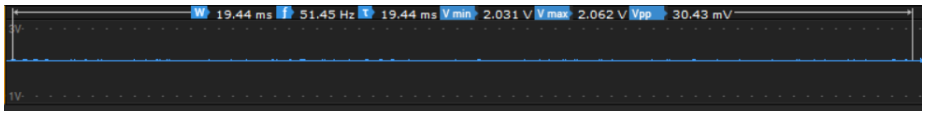

Figure 9: The output ripple after the second filter was added

Figure 10 displays the same waveform exported from a Rigol DS1102E oscilloscope set to 50mV/div.

6 Measuring the efficiency

The efficiency was measured as the ratio between the output power and the input one. All the efficiency measurements were done for a 2V output voltage. An amperemeter was put in series with the power supply and another one was put in series with the load. For different load resistance, the input current, input voltage, output current and output voltage were measured. The efficiency was calculated with the formula:

$$Efficiency = \frac{U_{out} \cdot I_{out}}{U_{in} \cdot I_{in}} \tag{1}$$

7 Measuring the accuracy

Table 3 shows that the difference between reference voltage and output voltage is more than 40mV.

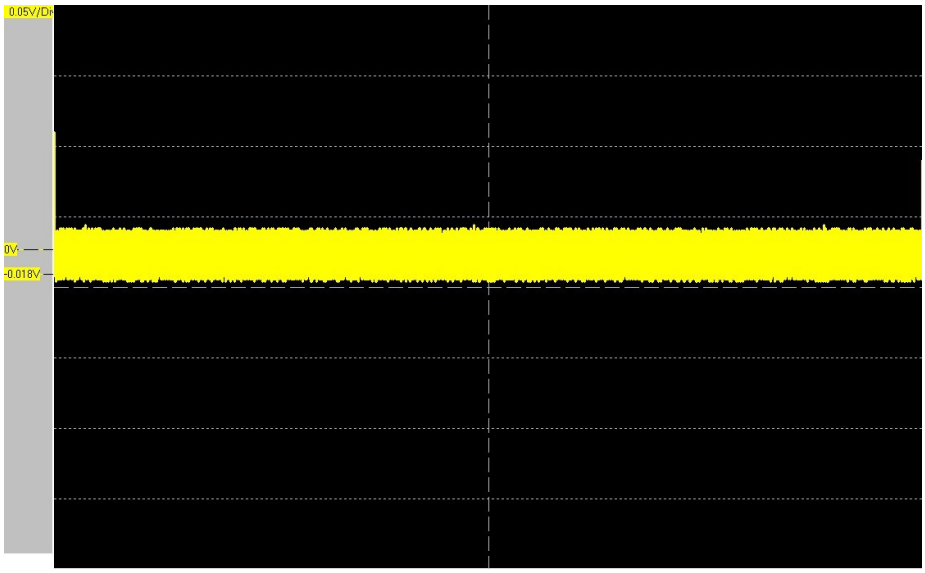

Figure 10: The output voltage measured with an oscilloscope

8 Area occupied

The area occupied by each functional block is described below:

Table 4 shows the area occupied by each element of the buck converter.

Important: These parts were chosen for prototyping. For a final solution, the capacitors in a 1210 package can be replaced with a 0805 one.

Table 5 shows the area occupied by each element of the DAC's circuit.

Important: The 0805 resistances can be resistances can be replaced with some in 0402.

Table 6 shows the area occupied by each element of the Comparator's circuit.

Important: The 0805 resistances can be resistances can be replaced with some in 0402.

DAC voltage [V]	Output voltage [V]
1.5	1.47
1.753	1.749
2.05	2.08
2.249	2.229
2.49	2.53
2.99	3.03
3.49	3.53
3.99	3.988

Table 3: Reference Voltag vs Output Voltage

Part type	Area [mm2]	Code
Driver	4.6	NCP81151B
Dual MOS	4.6	NTLUD4C26N
2 x Capacitors 1210	25.2	
Capacitor 0805	4.05	
Inductor	12.21	
Ferrite bead	1.3	742792023
Capacitor 0603	1.3	
Total	53.2	
Total are on PCB	148	

Table 4: Area Occupied by Buck Converter

Part type	Area [mm2]	Code
DAC	11.1	MCP4728
3 x Resistance 0805	12.1	
1 x Capacitor 0402	0.3	
Total	23.5	
Total are on PCB	60	

Table 5: Area Occupied by DAC's circuit

Part type	Area [mm2]	Code
Comparator	9	MAX9107
4 x Resistance 0805	16.1	
1 x Capacitor 0402	0.3	
Total	25.4	
Total are on PCB	73.5	

Table 6: Area Occupied by Comparator's circuit

Figure 11: Variation of efficiency with current

9 Future improvements

When it comes to future improvements, I think there are still many other things to be improved, especially for ripple and efficiency: Replacing the comparator with an ADC might bring some advantages. Basically, the comparator is a 1-bit ADC and the information provided by it is very poor. Consequently, the complexity of the control algorithm is limited due to the poor feedback information. By knowing exactly the value of the output voltage, a more precise control can be achieved. However, a research should be done here, because those advantages might not justify the additional price and space occupied by an ADC

Integrating a current measurement circuit. The output ripple and efficiency are powerfully dependent on the load current. By knowing the value of the instantaneous current, the algorithm's parameters can be adapted even during running. A look-up table can be developed inside the FPGA, so the algorithm will know what parameters to use depending on the load current.

References

[1] Texas Instruments, *Application Report* **SLVA477B–December 2011–Revised August 2015**, "Basic Calculation of a Buck Converter's Power Stage".

[2] Texas Instruments, *Application Report* **SLVA390–February 2010**, "Calculating Efficiency".

[3] Texas Instruments, *Application Report* **SLVA630A–January 2014–Revised October 2014**, "Output Ripple Voltage for Buck Switching Regulator".

[4] Analog Devices, *Application Note* **AN-1144**, "Measuring Output Ripple and Switching Transients in Switching Regulators".

[5] https://e2e.ti.com/support/power_management/simple_switcher/w/simple_switcher_wiki/2243.understanding-measuring-and-reducing-output-voltage-ripple