Smart Switching Regulator

The main aim of this project is to design an adjustable converter that will supply the Apertus AXIOM camera system. This camera system will include multiple instances of this converter, each one working at a different output voltage. All of these converters are identical when it comes to hardware.

The key aspects of the converter are:

- Efficiency
- Accuracy
- Ripple

As can be seen in the main schematic (…), the system consists of the actual converter and the feedback mechanism (a comparator and a DAC). The DAC gives the reference for the comparator which provides the necessary information to control the switching. Two independent I2C devices will be present in this design: the DAC I2C, where the desired output voltage will be communicated and the converter I2C, where particular settings for the converter control will be set. While the DAC has its build in I2C slave interface (being a discrete chip), the control block of the converter needs one as long as it is a FPGA module and need a way to communicate with the Axiom main logical unit. Consequently, these two blocks (DAC and converter) were separately tested.

First, a computer to DAC interface was developed. Because prototyping required an uncomplicated way to adjust the DAC's settings, a way was developed to program the device from the computer serial console. The bridge consists of three main blocks: an **UART block** (*RX.v and TX.v modules*), that is responsible for the communication between the computer and FPGA, a **UART to I2C manager** (*UARTtoI2C.v*), that interprets the information received and splits it into packages, and a **I2C Master** (*I2C_Master_Rev2.v*) module that took these packages and sent them one by one to DAC. The files for this module are available in *UART to I2C Bridge* folder.

Example: Let's say that 2V are needed to be written in FPGA.

This is how the DAC is programmed:

Address	Ch1_MSbyte	Ch1_LSbyte	Ch2_MSbyte	Ch2_LSbyte	Ch3_MSbyte	Ch3_LSbyte	Ch4_MSbyte	Ch4_LSbyte

Where the MS and LS mean most significant, respectively least significant. 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 a 8 bit one. Concatenating those two, the 12-bit value is obtained for each channel.

So, returning to the example, SC0W07WFFW07WFFW07WFFW07WFFWs is sent from 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 2V for a 4V reference.

Second, the I2C Slave module was also implemented, but not finalized. This will take commands from the Axiom logic and will command the converter control block. It was not integrated with the other blocks because this will be done only after the converter is completely finalized.

The control algorithm that commands the hardware block is available at: *UART to I2C Bridge/PWM_Generator.v*

Mentions:

The experimentally status of the converter is described in: *Experimental status.pdf*

The first iteration of the PCB had a problem, whose solution is described here: *Layout Files\ How to fix Smart Switching Converter PCB problem.pdf*

The Layout project as well as the pdf schematic are available here: *Layout Files\PCB_Project*

Most significant remaining work:

- Improve the ripple voltage for the converter
- Finish the design for the I2C Slave