BOARD FUNCTION

The Octa 18 Bit DAC with FPGA control has 8, 18 bit DACs. It is powered by + and – 15V (and ground). Onboard digital logic runs at 3.3V. The board is designed to support the Opal Kelly XEM7001 FPGA.

There is a +-10V reference (AD688) which has an error of 1mV meaning that there is a 1mV systematic error. This reference is buffered before being inputted into each DAC.

SOFTWARE FUNCTION

COMPUTER SIDE

The computer controller gives commands to the FPGA through the Opal Kelly Front Panel Interface. This code loads the bit stream (VHDL code) onto the FPGA on each startup. To change this code, the new bit-stream needs to be named “dac\_fpga\_bitstream.bit” and needs to be in the same directory as the dac\_frontpanel.py. If there was an error loading the bit-stream, an error will appear in the text box notifying this.

The computer side code also loads waveforms onto the FPGA and can control the DAC’s run state triggering to be manual or external (and, therefore, manually trigger it). See “Waveform Programming” section for more details.

FPGA SIDE

The FPGA has 4 states and 3 run-states. The 4 FPGA states are idle, load, ready, and run. These states are controlled by external sources (the computer and the trigger pin). Idle and load states will never become run state if the FPGA is in external triggering mode. If the FPGA is in the run state and is triggered, it will change to the ready state. The ready state is only useful for when the FPGA is externally triggered (not by the computer).

The 3 run states are not named and are changed by the system clock. In the first state (0), the new voltage data is loaded if the current voltage data has reached its endpoint. This state lasts for 10 system clock cycles (100ns for 100MHz). In the next state (1), the next voltage is calculated and the DAC is loaded with the next voltage’s serial data. The last two states are used for synchronization of the DAC. They make sure all 8 DAC’s trigger at the same time.

WAVEFORM PROGRAMMING

To program a waveform, a csv (comma separated value) file must be made. The maximum number of line entries in this csv file is 1025. There must always be two values for each line entry: the time count and the voltage step (separated by the comma). The time entry is the number of times the voltage increment will be repeated at a rate of 1MHz or 1us time steps (can be changed by changed by changing FPGA clock. This frequency will always be 1/100 of the FPGA clock. In the VHDL code, the FPGA clock is called “clk”). The csv file locations are loaded through the gui.

The time count value can be a 32-bit integer **in decimal**. Any bits over 32 will be knocked off. If this value is 0, the previous voltage level will be held and the DAC will output a constant. No further commands will be read.

The voltage step value can be an 18-bit integer **in decimal**. Any bits over 18 will be knocked off. If this value is 0, the voltage will not change but time will be counted. This allows a continuous voltage level for a defined period.

The first entry of the csv file is the initial voltage. The first value of the entry is the initial voltage (in decimal) with a maximum bit count of 18 (this entry can have any number for the second value). Any bits over 18 will be removed. **The DAC will output this voltage once the csv file is loaded and will default to it whenever the FPGA is not in its “run” state (see “dacboard\_state” in VHDL code).**

POSSIBLE SOFTWARE ADDITIONS

* Repeat Waveform bit command for csv programming. (Difficulty: easy)
* End all on reaching csv point (Difficulty: hard (will require dacs to communicate to controller.))
* Waveform “jumps” (Difficulty: moderate)
* Simple, pre-defined waveform “snippets” (Difficulty: depends on snippets)
* csv file writer (Difficulty: hard)
* csv file waveform previewer (Difficulty: moderate)

ISSUES/IMPROVEMENTS FOR BOARD REV000

* FPGA pins are slightly far apart and a bit too small.
* No onboard output filtering.
* LP filtering for voltage reference.
* Better voltage reference (if wanted)