The VectorUGo-2 console

S. Morel, Zthorus-Labs, 2023-01-16

1 Introduction

This document describes the architecture of the VectorUGo-2 vector-display game console. This console uses the same hardware as the original VectorUGo: a Maximator board (based on an Altera MAX10 FPGA) and a dedicated electronic board featuring a pair of op-amp integrators (for the x and y axes) and a transistor switch (for the z axis, i.e. to turn on or off the beam of the display). A detailed description of this hardware can be found in the GitHub repository:
`zthorus/VectorUGo/VectorUGo_main_board_sch.pdf`

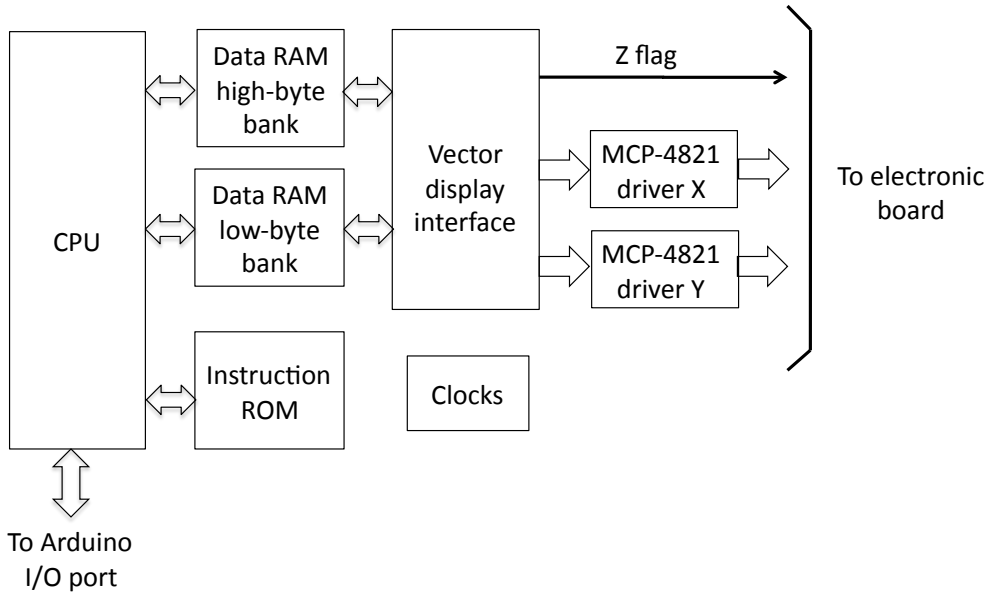
2 General characteristics

The VectorUGo-2 consists of the following components that are described by the indicated VHDL files:

- A special version of the ZTH1 CPU (16-bit RISC, Harvard architecture) with a 13-bit instruction address-bus and a 12-bit data address-bus. This version of the CPU includes the possibility to use the built-in analog-to-digital converter (ADC) of the MAX10 as an input for a "paddle" analog game-controller (not implemented yet). The VHDL file describing the CPU is: `zth1v_cpu.vhd`.
- A read-only memory (ROM) containing the instructions for the CPU, with a capacity of 8192 16-bit words of instructions. The ROM has been created from an IP core and consists of the files: `rom.cmp`, `rom.mif`, `rom.qip`, `rom.vhd`, `rom_inst.vhd`.
- A random-access memory (RAM) for the data, consisting of a bank of 4096 bytes for the "high-byte" part of the 16-bit words addressed by the CPU, and another bank of 4096 bytes for the "low-byte" part. The RAM can be simultaneously addressed by the CPU and by the vector-display interface by two separate access ports (an access port consists of: an address bus, a data bus, a read-enable signal and a write-enable signal). Because the high-byte and the low-byte banks usually contain different data at initialization, they are distinct entities and the files are: `ram_h.cmp`, `ram_l.cmp`, `ram_h.mif`, `ram_l.mif`, `ram_h.qip`, `ram_l.qip`, `ram_h.vhd`, `ram_l.vhd`, `ram_h_inst.vhd`, `ram_l_inst.vhd`.
- A vector-display interface that reads the content of the "vector table" stored in RAM (high-byte and low-byte) and sends out the corrected x and y vector components to the digital-to-analog converter (DAC) drivers. This interface also sends the z flag (beam on/off) to the electronic board. The VHDL file describing this interface is: `vector_display.vhd`
- Two instances (one for x, the other for y) of a driver which converts the data from the vector display interface into a SDI bit stream for the MCP-4821 DAC. The VHDL file describing this driver is: `mcp4821_drv.vhd`

- A PLL (phase-locked loop) that generates (from the oscillator on the Maximator board) the 10-MHz clock for the CPU, the 167-kHz clock for the vector-display interface, and the 2-MHz clock for the DAC drivers. This PLL has been created from an IP core and the files are: `clocks.cmp`, `clocks.ppf`, `clocks.qip`, `clocks.vhd`, `clocks_inst.vhd`.
- A description of the virtual printed-circuit board on which all the components above are connected and which is the top-level entity of the project: `vectorugo.vhd`.

The Intel Quartus Prime project to implement the VectorUGo-2 console on the Maximator board requires also the files: `vector3.qpf` and `vectorugo.qsf`.



Architecture of the VectorUGo-2 console (components).

3 Programming

A program for VectorUGo-2 consists of op-codes of CPU instructions in the ROM (stored in the `ROM.mif` file), and data in the RAM (stored in the `ram_h.mif` and `ram_l.mif` files). These files can be generated by the ZTH1 assembler running on Linux (see GitHub repository: `zthorus/ZTH1-Assembler`). Once created they have to be copied in the folder of the VectorUGo-2 Intel Quartus-Prime project. It is not necessary to fully recompile the project when updating any `.mif` file: just select in the "Processing" menu of Intel-Quartus Prime: "Update Memory Initialization File" and then, in the same menu: "Start" → "Start Assembler". Use the "Programmer" tool to upload the updated `.sof` file into the FPGA.

An OS for VectorUGo-2 has been developed. It mostly consists of routines for the vector display. The ZTH1-Forth compiler has been updated to compile programs for VectorUGo-2 using that OS. It is currently under test and will be available on GitHub when ready for release.

4 Organisation of the vector table

The vector-display interface reads out continuously line-by-line a table from the RAM. This table describes the vectors that are the elements of the displayed image. The vector table starts at address `x0000` and is organised as follows:

Address	**High-byte**	**Low-byte**
`x0000`	x DAC offset	y DAC offset
`x0001`	x zero-point	y zero-point
`x0002`	Number of vectors	
`x0003`	x vector 1	y vector 1
`x0004`	LFZ vector 1	-
`x0005`	x vector 2	y vector 2
`x0006`	LFZ vector 2	-
...	...	...

The DAC offsets for x and y (stored at `x0000`) are 4-bit values that are appended to the voltage settings of the MCP-4821 DACs by their drivers. These DACs convert a 12-bit input value to an analog voltage, but the vector-display interface sends 8-bit values. Therefore, the 4-bit DAC offset is the low part of the the 12-bit word. The tuning of these DAC offsets is required for a fine adjustment of the image.

The zero-point values for x and y (x_0 and y_0) are used by the vector-display interface to convert the 8-bit signed values of each vector component into unsigned values that are sent to the MCP-4821 drivers by calculating:

$$x_{out} = x_0 - x_{in} \quad ; \quad y_{out} = y_0 - y_{in}$$

The adjustment of the zero-points depends on the actual value of the voltage reference used by the op-amps of the integrators on the electronic board. Like for the DAC offsets, tuning the zero-point values is required to correct any image distortion.

The number of vectors forming the image is coded on 16 bits and stored at `x0002`. It can be modified at any time by the program. The size of the table is physically limited by the available amount of RAM, but the VectorUGo-2 OS (used by the ZTH1-Forth compiler) limits it to 1024 vectors.

Each vector in the table is defined by its 8-bit components x and y (ranging each one from -128 to +127) that are stored at an odd address, and by its 8-bit "length-factor and z" (LFZ) that is stored in the high-byte at an even address. The seven most significant bit form the length-factor of the vector, i.e., the waiting time of the vector-display between this vector and the next one, which has an influence on the vector tracing. It is however simpler to use the same length-factor for all the vectors. The least-significant bit of the LFZ is the z flag: if it is 0, the vector is invisible (beam off), if it is 1, the vector is visible (beam on).

5 Display setting

The display device of the VectorUGo-2 console can be an analog oscilloscope featuring an XY mode and a Z input. This oscilloscope would be directly connected to the output of the VectorUGo-2 electronic board. In the case of a Philips PM3215 oscilloscope, the settings shall be:

- Channel A: 2 V/div, DC.
- Channel B: 2 V/div, DC.
- Channel selection: A.
- Trigger selection: DC, B.
- Time/div: X DEFL.

We have found that the vector-table parameters giving the best image quality are:

- DAC offsets: 5 for x and 10 for y.
- Zero-points: 157 for x and 157 for y.

___oOo___