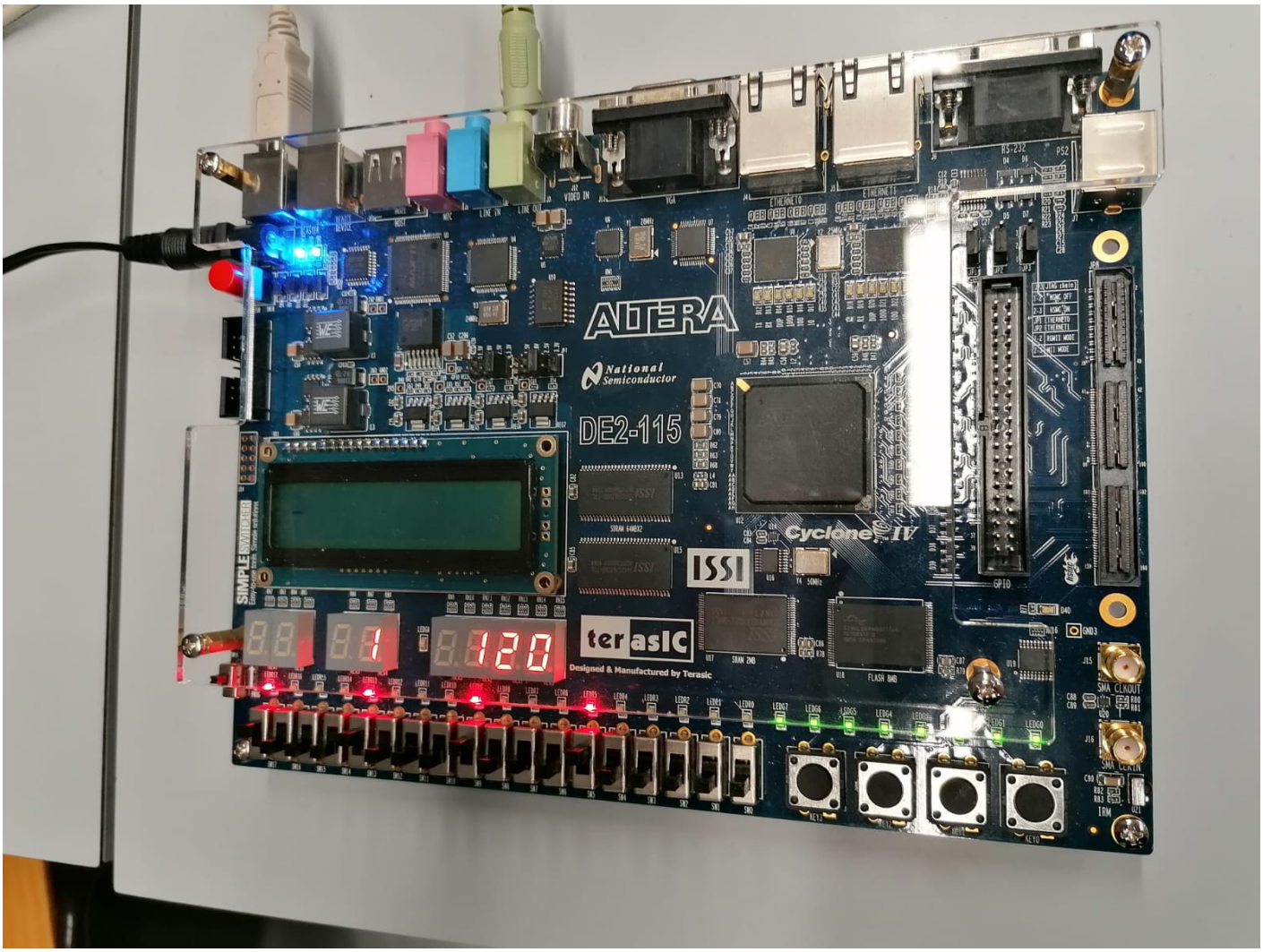
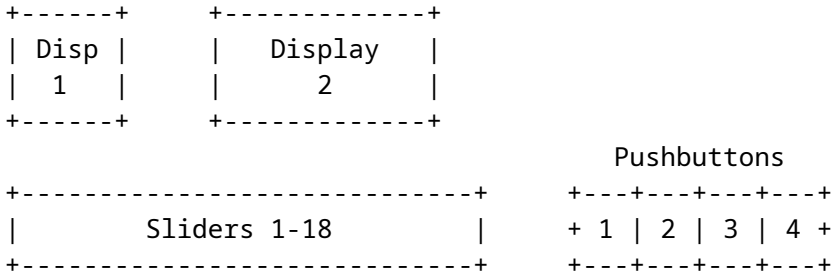
Drum Maschine

User Manual





Classification of Components

Display 1: shows current Bank (Values 0-3) Display 2: shows current BPM (Values 60-240)

Sliders 1-16: Setting the current Pattern Slider 18: BPM Change direction

Pushbutton 1: Play/Pause Pushbutton 2: change BPM Pushbutton 3: change Bank and save current Pattern to current Bank Pushbutton 4: Reset

Example Workflow

1. Reset (Pattern should be clear, Bank 0, BPM 120)
2. Input a pattern with *Switches 1-16* (up=on; down=off)
3. Save Pattern to Bank with *Pushbutton 3*
4. clear Sliders (all to down position)
5. press *Pushbutton 3* **3** more times (should be Bank 0 again)
6. validate your Pattern with the Red Leds
7. change your disired BPM with *Slider 18* and *Pushbutton 2* (Slider up = BPM goes up, Slider down = BPM goes down)
8. Play your Pattern with the Play/Pause Button (validated by green LEDs)

System Description

The FPGA Drum Maschine is an stand-alone FPGA-based drum computer which can operate at a BPM Range between 60 and 240 BPM. The smallest sample size is 1/16 of a Bar. The System should, once flashed, work as a stand-alone solution with configurable UI elements through the sliders and pushbuttons on the FPGA. The computer should feature 4 Banks which can individually programmed to play different samples.

Hardware / Software partition reasons

We parted the Hardware (HW) and Software (SW) parts because of the processing order of our Solution. The 4 different Banks need to oparate at a parallel data structure, so we decided to let them be computed by HW. The IO and User Configuration needs to be not as fast as the Audio Processing so we decided to let the SW handle these kind of actions. With this speration, we can assure a maximized performance of our board.

Experiences whilst designing the Hardware

One of the main challenges whilst designing the hardware was finding documentation for the required tools. Since we had to use an outdated Version of Quartus II that still supported the cyclone 4e most of the documentation did not represent the actual workflow. The Intel FPGA youtube channel (https://www.youtube.com/user/alteracorp) prooved to be a very valuable resource, since most of their old videos form around 2011 covered the workflow of the used Quartus II version. The next step was to figure out how to use the audio CODEC of the DE2 115 board. Luckily its usage was covered in the excerise of our lecture. We modified the copy samples example to support audio IO with other entities. One of the biggest challenges was to find VHDL examlpes to see how the syntax is applied in a practical enviroment. Most engeneers, including altera seem to be using verilog instead. Hence most of the research took a lot of time. Most Documentation was found in the forums of microcontroller.net. Depending on how much knowledge in work with VHDL was already aquired for every team member research in understanding VHDL or sometimes even Verilog was crucial. At first we tried realising the communicaion between the Nios II processor and the hardware using a dual port ram. This would have enabled us to read samples from an sdcard using software and play them through hardware. After extensive research we noticed that althouh the NIOS II processor uses a RAM interface with 32 BIT wordwidth and 32 bit adresses, the board, and hence also the SRAM controller IP only uses 16 bit wordwidth and 20 bit adresses. The wordwidth could be worked around byusing the byteenable pins of the SRAM controller. However our time restrictions did not allow for deeper research into the adress translation happening between the NIOS II processor and the RAM module. Therefore we resorted to realize our communication between hard- and software using PIO registers.

Experiences whilst programming the Software

The HAL provided by Altera is easy to work with - once you understood everything. The documentation is very poorly provided and you have to figure out a lot of things by yourself. Once you find the few important

functions, the coding is very easy and straight forward.

We've learned a lot about direct communications from C to Hardware Ports and understood the concept of RAM pretty well. We've also written a Look-Up-Table for the 7 Segment Displays and resolved a lot of problems.

We think the software part of the Drum Maschine works user friendly and intuitive and We are satisfied with our work on the Project.

Lessons learned

I learned that you should not overestimate your programming skills based on the Language, but on the environment you are working with. Most of my working time went into researching all of NIOS components and the C libraries attached. The programming part went pretty well, but the lack of documentation did cost a lot of time.

The combination of the both parts was quite challenging, because we first needed to understand the systematic of integrating both project parts into one flashable system. The official Intel FPGA YouTube-Channel was very helpful because the had also used the 13sp1 Version of Quartus.

It became clear that either a system completely in Hardware or in Software is managable to construct in a matter of weeks. The communication between the two systems using a SRAM, which was the main task of this project took a lot of time and was underestimated in the end.