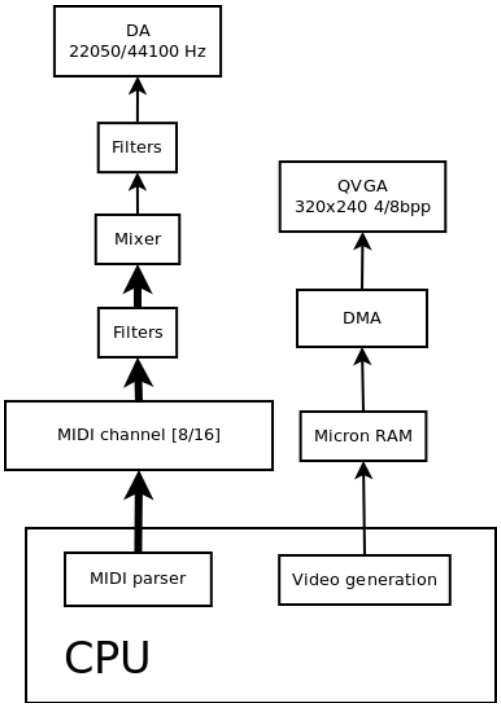
Project proposal
MIDI MONSTER

EDA385 - Design of embedded systems, advanced course
Advisor: Flavius Gruian

Johan Wennersten (dt08jw2@student.lth.se)
Arvid Lindell (dt08al6@student.lth.se)

1 Brief functionality



Figur 1: Intended system

1.1 Purpose

We intend to build a MIDI player with some sort of game attached. The game will be similar to Guitar Hero or Bit Trip Runner. The idea is to combine the timing of the music with a game.

1.2 Music

The music will be composed of square, triangle and sawtooth waves. These will be generated using DDS (Direct Digital Synthesis), which is a technique to playback a single sample (of for instance a single period of a waveform) at various speeds to create notes of different frequencies. DDS will be described in more detail in the final report.

MIDI-data will be parsed using the CPU. We are going to have 8 or 16 hardware-based channels, each attached with DDS functionality. The CPU will send commands to the channels, with information regarding what frequency is to be played, volume setting, and if the channel is active.

1.3 Video output

The game will be visualized using a monitor connected to the onboard VGA-connector. We are going to use a resolution of 320x240 pixels (QVGA) with

4/8bbp. A custom VGA controller will be constructed. In-game backgrounds will probably be generated using custom hardware.

1.4 Input

Input to the game is going to be delivered using a USB keyboard (the Nexys will convert the USB-signals to PS/2 in any case, so we will be constructing a custom PS/2 controller).

2 I/O processes

2.1 Sound

The hardware playback channels will be connected to the CPU using the PLB, they are to be controlled using software-accessible status registers. The resulting mix (of the 16 channels) will probably be converted to an analog signal using a simple R2R-ladder.

2.2 Video

The entire frame will be kept in Micron RAM. Video data will be sent from RAM to the VGA controller via the PLB (utilizing a FIFO), in turn using a Xilinx DMA controller.

2.3 Input

Data from the PS/2 controller will be accessible using a status register, with a couple of predefined keys reported in a bitfield. To clarify, the controller itself will keep track of which keys are currently pressed.

3 Memory

- The CPU will use a BRAM block for instructions, heap, static data and stack. The VGA FIFO will probably be placed in BRAM.
- As mentioned above, the video frame data will be placed in Micron RAM and will be accessed by the VGA controller using DMA.
- The DDS samples will be kept in distributed RAM for ease of access.
- MIDI files will probably be kept in non-volatile PCM memory. Another (worse) solution is to simply upload different files using the UART.

4 Feasibility

We have previous experience using DDS, PS/2 communication and VGA-timing. We have a rudimentary MIDI parser written in C, which should be fairly easy to port.

5 Time plan

Week	Activity
1–2	Basic VGA working. PS/2 controller complete. Rudimentary MIDI/DDS functionalty.
3–5	Integration, testing, report
6	Report