VT100 – Project Proposal

EDA385 – Computer Science – LTH

Mattias Jernberg, D06 (dt06mj4@student.lth.se)
Björn Uusitalo, D06 (tn06bu0@student.lth.se)
Andreas Lord, D08 (dt08al1@student.lth.se
Handledare: Flavius Gruian (Flavius.Gruian@cs.lth.se)

September 13, 2010

Contents

1 Overview

This projects intends to recreate a VT100 compatible terminal on the Digilent Nexys2 board. The VT100 is a serially connected terminal displaying text from and accepting input to a host computer. In order to support additional features such as colour and cursor movement it supports a set of in-band signalling codes (so called *escape sequences*). This project aims to support a subset of these codes to allow the connection of a Linux host computer and displaying the terminal on a screen.

2 Connectivity

In order to simulate the basic terminal there is a need to connect the FPGA-board to a computer using a standard serial port. A 16550 UART will be used to allow for communication and flow control. In standard usage the intended configuration is the 9600 8N1 mode, 9600 bits per second, 8 bits data, 1 bit parity and no flow control. This is a basic configuration allowing for connection to most systems, however it might be slow for receiving large amounts of text in a short interval.

User input will be accepted via the PS/2 port. No special configuration is intended as the keyboard controls data transfer itself and an existing IP core is used for serving the communication.

The terminal will be rendered on a VGA display. The actual output to the screen will be 640x480 pixels at 60Hz. This results in a clock frequency of roughly 25 Mhz for generating the output signal to the screen. The VGA controller will be character based and using a font with 8x16 pixels will allow for 80x30 characters to be shown. The actual resolution might be decided to be smaller (for instance 80x25) if it is deemed necessary.

Finally an audio output will be provided to support the bell instruction with which the terminal can be instructed to generate a tone with a specific frequency and duration.

3 Hardware requirements

Memory demands in this implementation are quite modest as a full frame of 80x30 characters using 2 bytes per character will amount to 4800 bytes total memory. 2 bytes are required for supporting one character and 4 bits of color (both foreground and background). Additionally a ROM for storing the font is needed. A maximum of 256 characters results in $256 \cdot 8 \cdot 16 = 32$ Kbits for storing the font. Finally 32 Kbyte will be available for software on the MicroBlaze CPU. The total amount of memory for this implementation is $4.8 \cdot 8 + 32 \cdot 8 + 32 \approx 326$ Kbits. As the Spartan 3E 1200K — which is our target device — has 448 Kbits of BRAM there is no problem fitting the

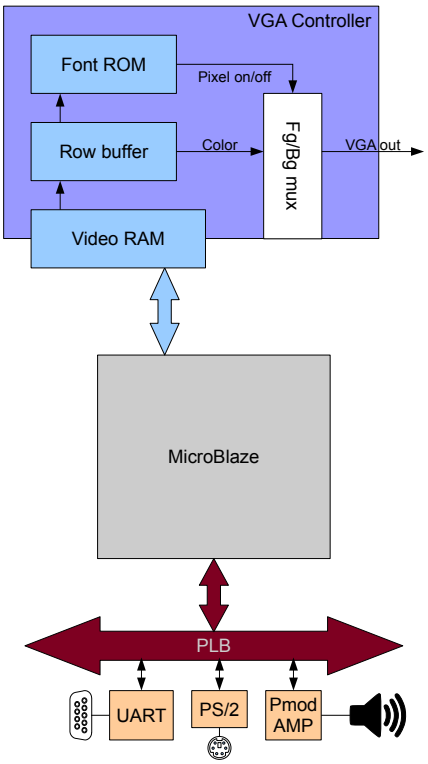

Figure 1: Proposed architecture for the implementation

whole memory inside BRAM circuits and this also leaves enough RAM to create a full framebuffer copy of the screen data to avoid any distortions while rendering the screen however this is not part of the proposed design (figure 1) where instead only the current row is cached to save memory and memory copying logic. Caching of at least the current row is necessary to avoid rendering one row of pixels as one character and the next row as a second character in the event that a certain position changes value in the middle of rendering.

Timing requirements should be minimal as characters will be updated as they arrive and do not have a timing requirement in regard to the VGA controller. Assuming that the VGA output operates at half the system clock and that a character is 8 pixels wide there are 16 clock cycles for fetching each succeeding part of the current row and store it in flip-flops. It should be quite trivial to meet these timing requirements. More important is clock speed to generate the VGA signal, however indications are that a 25 Mhz clock will fit the requirement thus allowing the output logic of the VGA controller to operate at exactly half the clock speed of the Nexys2 board.

4 Improvements

Given time once the features specified above have been implemented, there are a number of additional features which can be considered (in order of interest):

ECMA-48 graphics mode Expected to be trivial, a small additional font set and a translation in software.

Telnet Networking support using an Ethernet controller and a IP stack

Storage Allow for storing output on a SD card and read input to enable replay functionality

Wireless Either via XBee (or similar wireless serial) or (unencrypted) WLAN

5 Time plan

The plan assumes that there will be 4 weeks of effective work time. The first two weeks to get started, then four weeks to work and the last week to write the final report and demo. The work division between team members can be seen in table 1.

Week 1

- Basic VGA output
- PS/2 controller
- ECMA decoder in Java/C

Week 2

- VGA shared ram
- ECMA on MicroBlaze

Week 3

- VGA & ECMA completion
- Audio

Week 4

- Testing and bugfixes

Task	Björn	Mattias	Andreas
Project proposal	X	X	X
Project presentation	X	X	X
PS/2	X		
ECMA-48 in java/c			X
Basic VGA controller		X	
ECMA-48 on microblaze			X
VGA on shared RAM	X		X
VGA font		X	
PmodAMP	X	X	
Testing and bugfixes	X	X	X
Final report	X	X	X

Table 1: Individual tasks division.