A VGA BASED LOW COST STUDENT OSCILLOSCOPE

By

ROUFURD JULIE

This thesis is submitted
in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Electrical Engineering

University of Cape Town

2004

Declaration

I hereby declare that:

1) this is my own unaided work.

2)Any aid used was referenced.

and that,

This work is being submitted for the award of the BSc(ENG) degree at the University of Cape Town in 2004, and has not been submitted for this degree elsewhere.

Signature of Author

RPM Julie date:

Acknowledgements

I would like to thank the following people for their timely support in the completion of this thesis.

1) Theola – for keeping me from losing my childish enthusiasm for life.

2) Sweeney and Edlauer, for distracting me at critical times. This enabled me to gain different perspectives on the problem I was working on when distracted.

3) The same people, for various other non-academic reasons.

4) My Supervisor, Samuel Ginsberg, for providing me with very clever perspectives on some of the problems experienced during my thesis.

5) Mr Watermeyer, for his once off critical bit of perspective on which was at the time a dead end problem. This enabled me to complete the thesis.

6) God, for his Just In Time (JIT) biasing of the results in my favour

7) and all the other people who I have forgotten to mention.

Thanks. I could not have done it without you.

Abstract

The oscilloscope is the most versatile electronic instrument one can have on ones work table today. It is a very expensive instrument however. I propose a new kind of an oscilloscope. A box of electronics, with a CPLD at its heart and VGA monitor as its visual output, that provides the functionality of an oscilloscope. This instrument should also be a fraction of the cost of conventional oscilloscopes.

This thesis produces an instrument that fulfils this objective – A cheap oscilloscope that uses a VGA monitor as its output. It does not have the complete functionality of conventional scopes, but it can be significantly improved. It can be used in educational laboratories where financial resources do not permit the purchasing of the commercial oscilloscopes.

During the design process, literature was read of the various parts of the prototype system. The UP2 Development Board and QuartusII were the primary tools used in the design.

Experiments, Simulations and Research were the primary methods used to facilitate the design process.

The prototype design consisted of 3 main processes. These processes could however be broken down in many other process. This is an indication of the top-down design process followed.

The thesis was a successful in that it produces a trace of the input signal on the VGA monitor. The input signal was a saw-tooth waveform.

Table of Contents

DECLARATION **i**

ACKNOWLEDGEMENTS **ii**

ABSTRACT **iii**

TABLE OF CONTENTS **iv**

LIST OF FIGURES **vii**

1 INTRODUCTION……1
1.1 Subject of this Thesis 1

1.2 Background to this Thesis 1

1.3 Objectives of this Thesis 2

1.4 Limitations and scope of this Thesis 2

1.5 Plan of Development 2

2 LITERATURE REVIEW ……3
2.1 The Oscilloscope 3
2.1.1 The analogue Oscilloscope 3
2.1.2 The Digital Oscilloscope 11

2.2 VGA monitors 12
2.2.1 VGA technology 12
2.2.2 Controlling the VGA monitor 13

2.3 FPLDs 15

3 METHODOLOGY……17
3.1 Research 17

3.2 Simulations 17

3.3 Experiments 18
3.4 General aspects of my approach 18

4 DEVELOPMENT TOOLS……19
4.1 The QuartusII 4.1 software 19
4.1.1 Design entry of software to be programmed 19
4.1.2 The Compiler 21
4.1.3 The Simulator 21
4.1.4 The Assignment Editor 22
4.1.5 The Programming tool 22
4.1.6 The Mega Wizard Plugin manager 23

4.2 The UP 2 Development board 23
4.2.1 The Peripherals 24

5 MISCELLANEOUS EXPERIMENTS MADE DURING THESIS…..26
5.1 Testing the ADC 1 26
5.1.1 Description of the ADC 26
5.1.2 The Experimental setup and results 26
5.1.3 Motivation for the length of the trigger pulse 27

5.2 Testing the ADC 2 27
5.2.1 The Experimental Setup and Results 28

5.3 Development Board experiment 28
5.3.1 The Experimental Setup and Results 29

5.4 General discussion of Experiments 29

6 SYSTEM VIEW OF DESIGN……..31
6.1 Definition of the problem 31

6.2 Initial Assumptions 32

6.3 Initial Considerations 32

6.4 The Design 33

7 SAMPLER MODULE OF CPLD SYSTEM……..36
7.1 General description of Sampler 36

7.2 ControlADC process 37
7.2.1 Breakdown of controlADC 37

7.3 Data to Memory process (data2mem) 39

7.4 Discussion 39

8 MEMORY MODULE OF CPLD SYSTEM……...…41
8.1 Description of the RAM 41

8.2 General description of the Memory module 41

9 DISPLAYER MODULE OF THE CPLD SYSTEM…..……..42
9.1 Description of VGA monitor 42

9.2 General description of Displayer module 42
9.2.1 A special note on VGA_SYNC 43

9.3 Discussions 44

10 THE CPLD SYSTEM – IN RETROSPECT…….……45
10.1 The CPLD system – a system description 45
10.1.1 The Sampler module 45
10.1.2 The Memory module 46
10.1.3 The Displayer module 46

10.2 Discussions 47

11 THE SUPPORTING ELECTRONICS……48
11.1 The trigger system 48

11.2 The demo circuit 48

12 THE RESULTS OF THE THESIS……49
12.1 The trace 49

13 CONCLUSIONS 50

14 RECOMMENDATIONS 51

15 REFERENCES 52

16 APPENDIX……...53
16.1 Circuit Diagram of demo circuit connected to the ADC 53

16.2 Circuit Diagram of the triggering circuit 54

16.3 Cover page of the ADC0820 55

16.4 The Design software 56

List of Figures

CHAPTER 2 – LITERATURE REVIEW

Figure 2.1 – Simplified diagram of an Analogue Oscilloscope……. 3
Figure 2.2 – Block diagram of the vertical circuitry 4
Figure 2.3 – Block diagram of the horizontal circuitry 6
Figure 2.4 – Triggering illustration 7
Figure 2.5 – Events after triggering pulse 8
Figure 2.6 – Block diagram of a Digital oscilloscope 11
Figure 2.7 – Updating of pixels on a VGA monitor 13
Figure 2.8 – Horizontal refresh on application of HSYNC 14
Figure 2.9 – frame refresh on application of VSYNC 15

CHAPTER 3 – METHODOLOGY
(no figures)

CHAPTER 4 – DEVELOPMENT TOOLS

Figure 4.1 – Typical schematic entry file 20
Figure 4.2 – Picture of the UP2 Development board 25

CHAPTER 5 – MISCELLANEOUS EXPERIMENTS MADE DURING THESIS

Figure 5.1 – Test 2 demonstration 28
Figure 5.2 – Experimental setup for development board experiment 29

CHAPTER 6 – SYSTEMS VIEW OF DESIGN

Figure 6.1 - The problem statement 31
Figure 6.2 – Two process architecture 32
Figure 6.3 – Block diagram of the complete system 33
Figure 6.4 – Block diagram of the internals of the CPLD controller 35

CHAPTER 7 – THE SAMPLER MODULE OF THE CPLD SYSTEM

Figure 7.1 – Input and Output ports of Sampler 36
Figure 7.2 - sampler – 2 process system 36
Figure 7.3 – controlADC's input and output ports 37
Figure 7.4 – Breakdown of controlADC 38
Figure 7.5 – data2mem's input and output ports 39

CHAPTER 8 – THE MEMORY MODULE OF THE CPLD SYSTEM

Figure 8.1 – Design of Memory module 41

CHAPTER 9 – THE DISPLAYER MODULE OF THE CPLD SYSTEM

Figure 9.1 – Block diagram of Displayer 42

CHAPTER 10 – THE CPLD SYSTEM – IN RETROSPECT
(no figures)

CHAPTER 11 – THE SUPPORTING ELECTRONICS

Figure11.1 – The trigger circuit 48

CHAPTER 12 – RESULTS OF THE THESIS

Figure 12.1 – The results 49

CHAPTER 16 APPENDIX

Figure 16.1 – Circuit Diagram of Demo Circuit connected to ADC 53
Figure 16.2 – Circuit Diagram of triggering circuit 54
Figure 16.3 – Cover page of ADC0820 Datasheet 55

1. Introduction

1.1 SUBJECT OF THIS THESIS

This thesis outlines the design of an oscilloscope with a VGA monitor output.

1.2 BACKGROUND TO THE THESIS

Oscilloscopes are one of the most versatile electronic instruments available. It allows one to see a voltage at an arbitrary point in an electronic circuit. It can also be used to observe variables in other phenomena, provided the appropriate sensor is available. Such as perhaps a sound wave, when converted to an electronic voltage by a microphone – the microphone being the sensor. Oscilloscopes are however very expensive, and thus requires substantial monetary investment, which is greater than the average student can afford.

VGA monitors are standard these in days with Desktop PC systems. There is however a surplus of these monitors available, especially since the lifetime of the monitor is much longer than the lifetime of the desktop PC. With the accelerating changes in desktop PCs these days, this surplus is likely to increase at an even faster rate.

This has created an interesting possibility – what if an cheap oscilloscope could be built that has as its display a VGA monitor.

1.3 THE OBJECTIVES OF THE THESIS

The objective of this thesis is thus to investigate this possibility and come up with a prototype design. The prototype design would have the VGA monitor plugged into it, and then proceed to function as an oscilloscope.

1.4 LIMITATIONS AND SCOPE OF THESIS

The thesis tries to implement a low-end oscilloscope with downscaled features from that of commercial oscilloscopes. It does not attempt to build the prototype with great accuracy. It is merely an attempt to see if the idea is possible and if possible, build a prototype system.

1.5 PLAN OF DEVELOPMENT

The thesis starts off by describing the literature that was read to get a deeper understanding of the problem. It then describes the method used to attack the problem, followed by a description of some of the development tools used during the design. Then it describes some of the experiments that were performed.

It then gives an overview of the initial design followed by a description of the design in detail, with pictures of some of the results of the prototype system.

Finally it states some of the conclusions that were made and recommends further action that can be taken to improve the prototype.

2. Literature Review

This chapter is a description of the literature that was read in order to solve the oscilloscope problem. No literature has been found on this specific problem. There is however plenty of literature on the individual parts used to solve this problem. These parts are the Oscilloscope, the VGA monitor and the FPLD. I felt that it would be a good idea to read as much as necessary about these different parts, and in this way have a better idea of how to solve the problem. These parts will thus be described in the sections that follow.

2.1 THE OSCILLOSCOPE

When attempting to build an oscilloscope, one should start by understanding how conventional oscilloscopes work. There are 2 kinds of oscilloscope – the analogue and digital oscilloscopes.

2.1.1 The Analogue Oscilloscope

Shown in figure 2.1 is a simplified block diagram of an analogue oscilloscope.

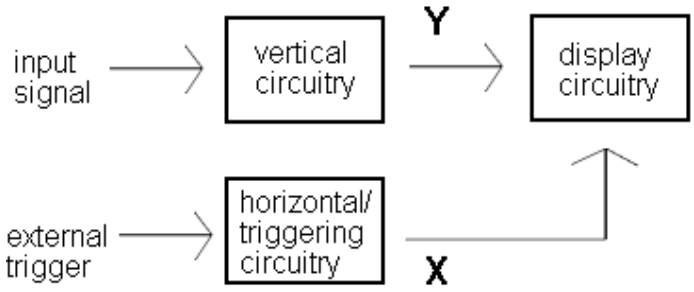


Figure 2.1 – Simplified diagram of an analogue oscilloscope

The vertical circuitry

The *vertical circuitry* conditions the input signal to levels acceptable for the display circuitry. It can be broken down further as shown in figure 2.2 below.

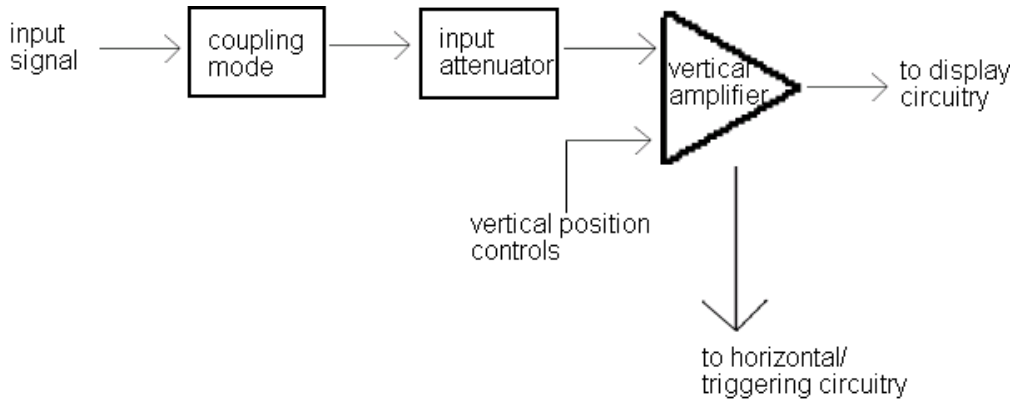


Figure 2.2 – Block diagram of the Vertical circuitry

In the ***coupling module***, the user selects the appropriate mode of coupling of the input signal to the attenuator module. This can be AC, DC, or GND. In AC coupling, the DC (non-changing) component of the input signal is removed, and the scope then only displays the AC (or changing) component of the signal. In DC coupling, the scope shows both the AC and the DC components of the signal. While in GND coupling, the input to the next module (*input attenuator*) is simply connected to ground. This is done so that the ground level of the trace can be established.

The ***input attenuator*** is responsible for attenuating a wide range of input signals to the input sensitivity of the vertical amplifier. The sensitivity of the vertical amplifier is typically 5mV. A network of resistor dividers performs the division, and the volts per division (v/div) setting, chooses the division factor.

In order to provide a calibrated trace on the display, *input attenuators* must have very accurate resistive networks that have a flat response over the whole oscilloscope measuring bandwidth. This means that an oscilloscope with a 20MHz bandwidth, should divide input signals down by (the user chosen) a factor from DC up to 20MHz. If the oscilloscope is set to an input sensitivity that is greater (e.g. 1mV) than the sensitivity of the vertical amplifier (normally 5mV), then the input signal is amplified as opposed to attenuated.

The ***vertical amplifier*** is responsible for amplifying the scaled input signal to the levels required for the vertical plates of the Cathode Ray Tube (CRT). The CRT forms part of the *display circuitry.*

An oscilloscope has different display modes – Single or Dual trace, Chopped or Alternate display – a discussion of which is beyond the scope of this thesis. The vertical amplifier is responsible for providing the switching signals to facilitate these modes. This is configured with the *vertical position controls*.

In addition, a DC bias input is also provided, as part of the *vertical position controls.* This allows the vertical movement of a trace on the scope.

As shown in figure 2.2, there is a signal tap from the vertical amplifier, to the *horizontal / triggering circuitry.* This will be discussed next.

The horizontal / triggering circuitry

The *horizontal / triggering* circuitry handles the horizontal aspects of the trace: how long (in seconds) should the sampling window be; on which portion of the input waveform should the trace begin. The *horizontal circuitry* module can be broken down into 3 blocks. This is illustrated in figure 2.3 below.

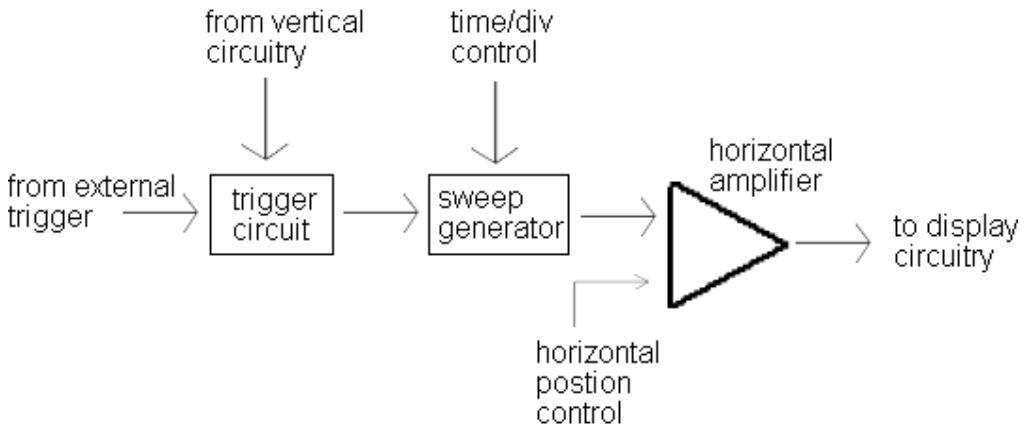


Figure 2.3 – Block diagram of Horizontal circuitry

In the ***trigger circuit*** block, the start of the sampling window, or trace, is selected. This is important else the trace will start at different times, and this will make the display unreadable.

The triggering conditions are the *slope* and the *voltage level* of the signal. These are user adjustable parameters, and it chooses a particular point on the triggering signal. This is illustrated in figure 2.4.

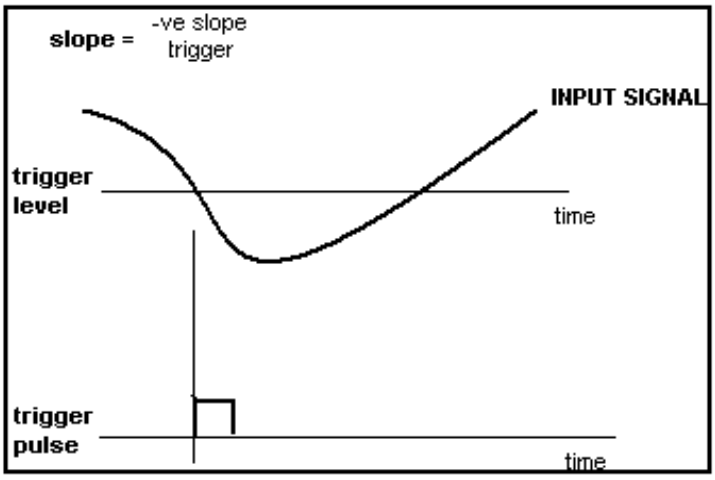


Figure 2.4 – Triggering illustration

The triggering circuitry also allows different sources to trigger the trace. These modes are: the INPUT signal, LINE and EXTERNAL triggering. The source of the triggering circuit will from now on be called the *triggering signal*.

LINE causes the slope to trigger on the AC power line frequency, while EXTERNAL allows the user to connect an external trigger signal.

Another user settable parameter is the way the triggering signal is coupled to the triggering circuit. Here the user has 4 choices: DC, LF reject, HF reject and AC.

When DC coupled, the “raw” triggering signal is used. The triggering signal retains both its AC and DC components.

When in the Low Frequency reject (LF Reject) coupling mode, the circuit only triggers on the higher frequency components of the triggering signal.

On the other hand, when in High Frequency reject (HF reject) coupling mode, the circuit only triggers on the lower frequency components of the input signal.

Finally, when in AC coupling mode, the circuit nullifies the DC components of the triggering signal, and only triggers on the AC components.

The output of the triggering circuit is a pulse. This pulse is sent to the ***Sweep Generator***. On the reception of the triggering pulse, this module produces a linear ramping voltage. This voltage is an input to the *Horizontal Amplifier*.

Figure 2.5 illustrates the various events that take place after a triggering pulse and during the subsequent ramping cycle.

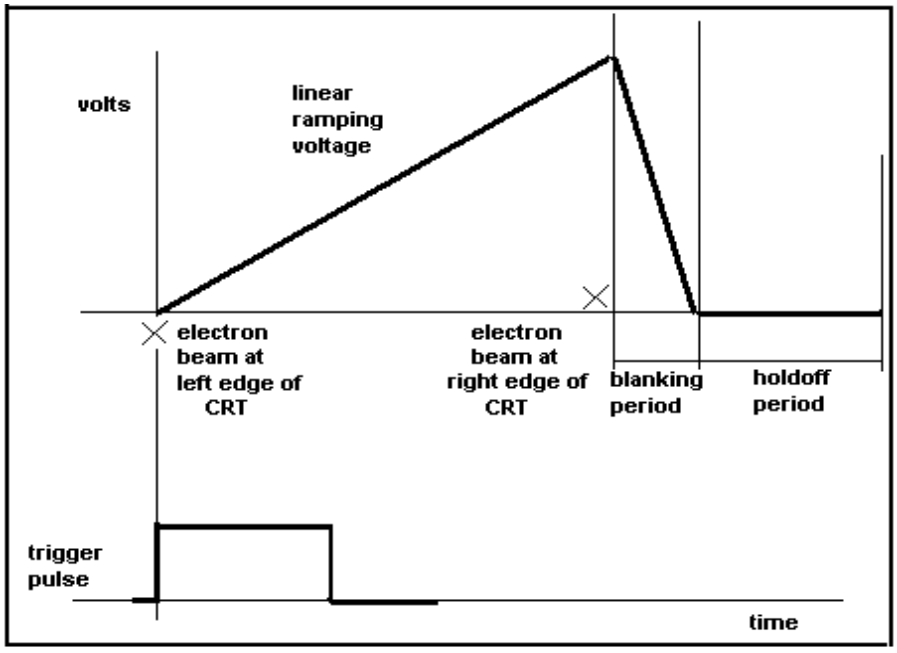


Figure 2.5 – Events after triggering pulse

The start of the ramping voltage coincides with the electron beam being at the left end of the CRT. It then progresses until the ramping voltage reaches a maximum, with the electron beam reaching the right end of the CRT. The electron beam will be explained when describing the CRT in a later section. The levels of the ramping voltage correspond with the input sensitivity of the *horizontal amplifier*.

When the ramping voltage reaches a maximum, this voltage is reduced back to zero. During this time the electron beam is turned off – this is also called the retrace or blanking period. A complete ramp is called a sweep. The rate at which this occurs is called the *sweep rate*.

The holdoff period varies with the sweep rate, but ensures that there is complete stabilization and retrace before the next trigger occurs.

Time is a desirable measure of the input signal. This means that the sweep rate should be both linear and calibrated. A convenient measuring tool is the time/div, which is user settable. The CRT's horizontal display has a graticule that divides the display into equal divisions, which facilitates this measure.

The ***horizontal amplifier*** simply amplifies the ramp voltage to the levels required to drive the CRT horizontal deflection plates.

The *sweep speeds* can range from milliseconds to nanoseconds, thus it is important that the *horizontal amplifier* has a wide bandwidth.

The horizontal position of the trace can also be adjusted by the application of a DC bias to the input of this amplifier. This is configured through the horizontal position control.

The Display circuitry

This is the part of the oscilloscope that displays the input signal as a trace. By this stage, most of the work has already been done by the vertical and horizontal circuitry stages. The display circuitry is essentially the Cathode Ray Tube (CRT), and the graticule, which is the calibration marking.

The ***CRT*** section consists of an electron gun, vertical and horizontal deflection plates, and the coating that produce the visual colours.

The CRT's electron gun produces a stream of electrons. These are accelerated to the front end of the tube and on the way focused into a narrow electron beam. The vertical and horizontal plates, mentioned earlier, deflect this narrow beam of electrons to a particular orientation. This forms the shape of the input signal. The display end of the tube has a coating which when struck by this electron beam emits a particular colour (usually green). This is the trace, which appears on the display end of the tube.

The **graticule**, is that part of the display, which allows the user to measure the characteristics of the trace. It divides the display into equally spaced horizontal and vertical lines, which correspond to the time/div and volts/div setting respectively. Provision is also made for rise and fall time measurements of pulses, by the 0, 10, 90 and 100 percentage markings.

2.1.2 The Digital Oscilloscope (DSO)

The simplified block diagram shown in figure 2.6 summarizes the operation of the digital oscilloscope.

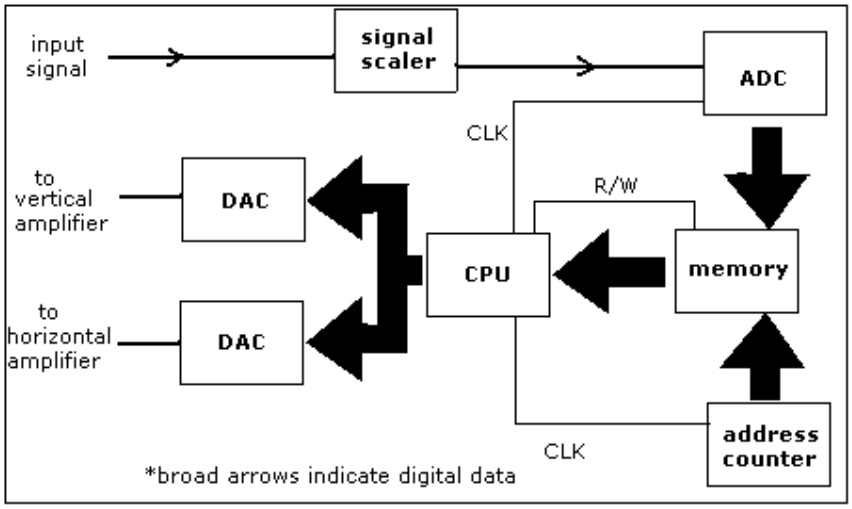


Figure 2.6 – Block diagram of a Digital oscilloscope

The ***signal scaler*** is similar to the *input attenuator* of the analogue oscilloscope discussed above. The ***vertical*** and ***horizontal amplifiers*** are also similar to those of the analogue scope.

The ***ADC*** digitizes the signal output from the signal scaler, and stores this in ***memory.*** This however, only happens on a signal from the CPU. The CPU thus completely controls the sampling, the writing to and reading from the memory, as well as the transferring of the correct data to the vertical and horizontal amplifiers, via the ***DACs***.

The ***address counter*** holds the current sample address and this is produced to **memory's** address lines.

Because the DSO operates primarily with digital data, new capabilities are created. Mathematical and non-periodic signal analysis of signals is now possible. More sophisticated triggering is also now possible, including pre triggering. is also now possible. These topics are however beyond the scope of this thesis.

2.2 VGA MONITORS

Since a VGA monitor is the output device in this thesis, an understanding of how it works is necessary. This section describes the basics of the VGA technology, and how one goes about controlling it.

A VGA monitor is a complete video displaying system that provides a simple controlling interface, for the displaying of video images. This interface is defined in a VGA standard. Video images can be defined as images that consist of the 3 primary colours in various intensities; this creates a 2 dimensional image that is lifelike in appearance. The VGA monitor interface consists of 5 controlling signals. These are the Red, Green and Blue colour signals (RGB), the horizontal and vertical sync signals.

2.2.1 VGA technology

At the heart of the VGA monitor is the CRT described earlier. This time however, there are 3 electrons guns for each of the primary colours. The electron beam is scanned across the display section, row by row, starting from the top row. On the displaying end of the tube, there are 3 different phosphors, for each of the colours.

The VGA monitor also contains the electronics that drive the horizontal and vertical deflection plates to control this scanning process. The scanning process takes place at a constant rate, which is defined in the VGA standard.

The viewing part of the monitor contains 480 x 640 picture elements, or pixels. The voltage levels on the RGB control signals determine the colour of each pixel, when the electron beam scans across this pixel. In order to provide motion in the image, the image needs to be refreshed at least 60 times per second as the human eye can spot flicker at refresh rates less than about 30 times per seconds. For 480 x 640 pixels VGA monitor, at a 60Hz refresh rate, approximately 40ns is needed per pixel. A 25Mhz clock has a 40ns period.

2.2.2 Controlling the VGA monitor

Pixels are updated in a serial fashion. The pixel clock runs at 25MHz. The electron beam starts at (0,0) and goes horizontally through the first row, up to the last row, and last pixel at (479,639). This illustrated in figure 2.7 below.

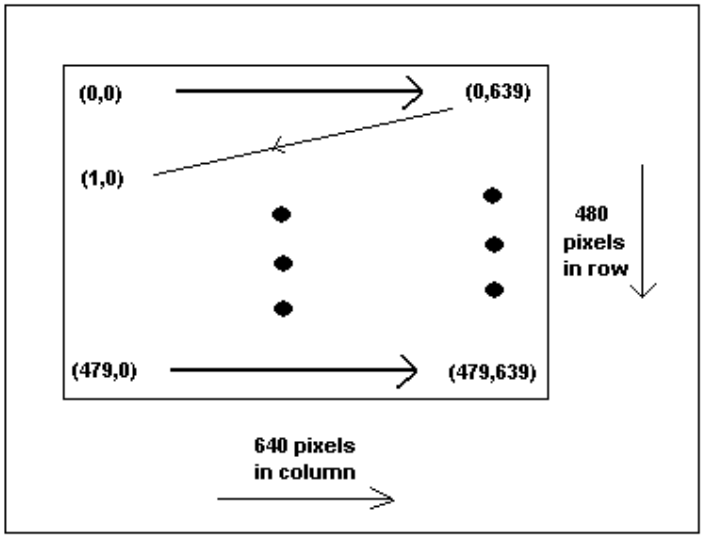


Figure 2.7 – Updating of pixels on a VGA monitor

This process is controlled by vertical sync (VSYNC) and horizontal sync (HSYNC) control signals. The colour data for the monitor is sent out in the first 640 clock cycles. The colour signals are then forced to zero in a blanking stage. During this blanking stage, a pulse is put on HSYNC. This indicates to the monitor that it should move on to the next row. The process then repeats. This can be visualized in figure 2.8 below.

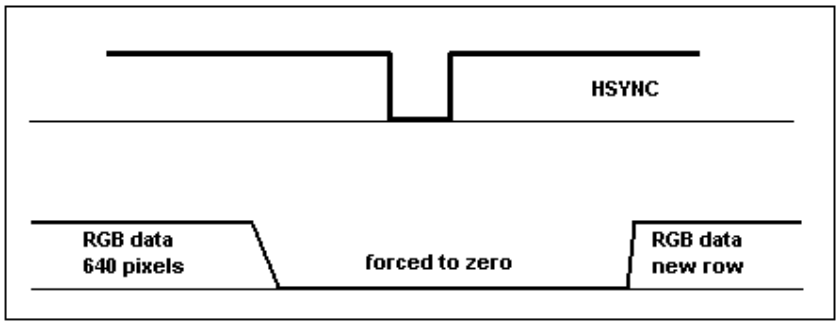


Figure 2.8 – Horizontal refresh on application of HSYNC

There are constraints placed on the maximum and minimum length of the blanking interval and the HSYNC pulse.

480 rows of pixels constitute a frame. After a frame has been displayed the colour signal are forced to zero. During this time, a pulse is put on VSYNC. This indicates to the monitor that it should move back to the row zero, and start a new frame. This is illustrated in figure 2.9.

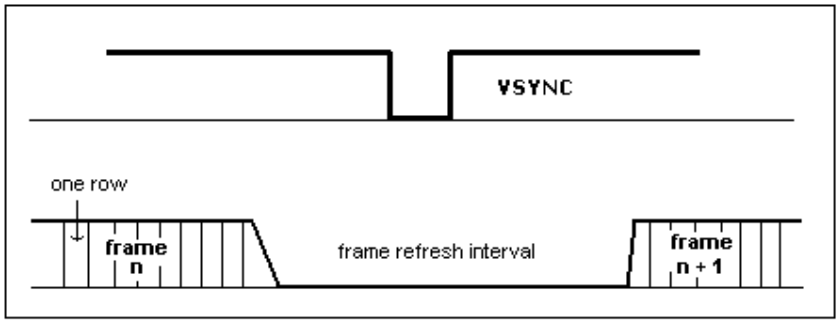


Figure 2.9 – Frame refresh on application of VSYNC

Again, there are constraints on the size of the VSYNC pulse and the frame refresh interval.

2.3 FPLDs

This section on FPLDs is included, as a CPLD forms the heart of the solution to the oscilloscope problem. I did not have much knowledge of FPLDs, and it was thus necessary to familiarize myself with them, and in this way learn their capabilities and shortcomings. The descriptions that follow are very simple as it is not necessary to understand FPLDs in any more detail, for the purposes of this thesis.

Field **P**rogrammable **L**ogic **D**evices (FPLDs) are those devices that consist of multiple copies of a basic logic element (LE). Essentially a LE consists of some configurable logic gates and flipflops. These LE are connected to one another by a programmable interconnect array (PIA). Configuring a LE and interconnecting it with other LEs can create complex systems. By configuring is meant loading a map of the desired arrangement and interconnections of the LEs and PIA into the FPLD.

A FPLD contains thousands of gates. Some new generation FPLDs contain a million gates. They are thus much too complicated to configure on a gate level. Special programming languages have been develop to facilitate this process. These languages are called Hardware Description Languages (HDL). There are 2 standard HDLs available today: Verilog and VHDL.

3. Methodology

This chapter describes my method and approach used to solve the problem

3.1 Research

A large component of my thesis had been the reading of a wide range of literature. The oscilloscope, and VGA interfacing and controlling readings, had been essential to understanding how these components worked. These were fairly straightforward readings however.

A greater part of my reading however, was on FPLDs and their configuration. I had no experience with these devices, and thus had to read a wide range of books, papers and articles, in order to understand, what exactly a FPLD was and how to apply them. I did not use most of the content I read on FPLDs, however, the reading was essential in order to develop an initial “feel” for the capabilities and limitations of FPLDs. This reading was done throughout my thesis. These readings did of course gain greater direction during the progression of the thesis, when certain concepts needed to be understood on a deeper level.

3.2 Simulations

Most of my software testing was done in computer simulation. This facility is described in the section describing the QuartusII software. I would program a process and then simulate it. I did this for all the processes I have used in the solution to the thesis. I would put processes together to form systems, and simulate the system again.

Software simulations have been a fundamental part of the development process in this thesis.

3.3 Experiments

There was also a time for experimenting. I did some experiments in order to confirm my understanding of certain process.

For example, when wanting to confirm my understanding of the timing diagram of the ADC, I wired up the ADC and used a pulse generator to trigger it, and checked for correct results.

I also experimented with the development board, in order to verify my understanding of its interface.

3.4 General Aspects of my approach

It will become apparent as you read through my thesis, that a top-down design approach was used to come up with the prototype system. This is not completely true however, for as I reached the bottom end of the system hierarchy, it became necessary to sometimes use a bottom–up approach to the design. For at these levels hardware properties started to become significant – for e.g. propagation delays of signals. Thus the lower levels of the design sometimes guided the higher-level decisions.

4. Development Tools

This section describes the development tools used in designing the oscilloscope.

4.1 The QuartusII 4.1 programming software

I used the QuartusII 4.1 Web Edition software. This software is freely available from Altera, and I found it at www.altera.com during July 2004. There is a licensing procedure that needs to be undertaken before use.

The software has many features and tools to aid the designer. I will however only be mentioning those tools that I had used.

Before I mention these tools, I first need to describe how the organization of a design is implemented in QuartusII. Designs are based on project files. Project files contain information about the specific FPLD used, the IO pin allocations, gate layout, and lots of other information; i.e. every aspect of the design and device.

A project file need to be created before any simulation, compilation, or other functions are exercised on the design, as these functions are dependant on the information contained in a project file.

4.1.1 Design entry of the software to be programmed onto the FPLD

There were a numerous methods to enter the design. I used 2 methods however: Schematic entry, and HDL entry.

Schematic entry is exactly what it says. You visually place various parts, and then join them using a wire or signal. You can place a logic gate, which is on the lowest level, to a complete system at the highest level. Figure 4.1 show a typical design entered using the schematic entry method.

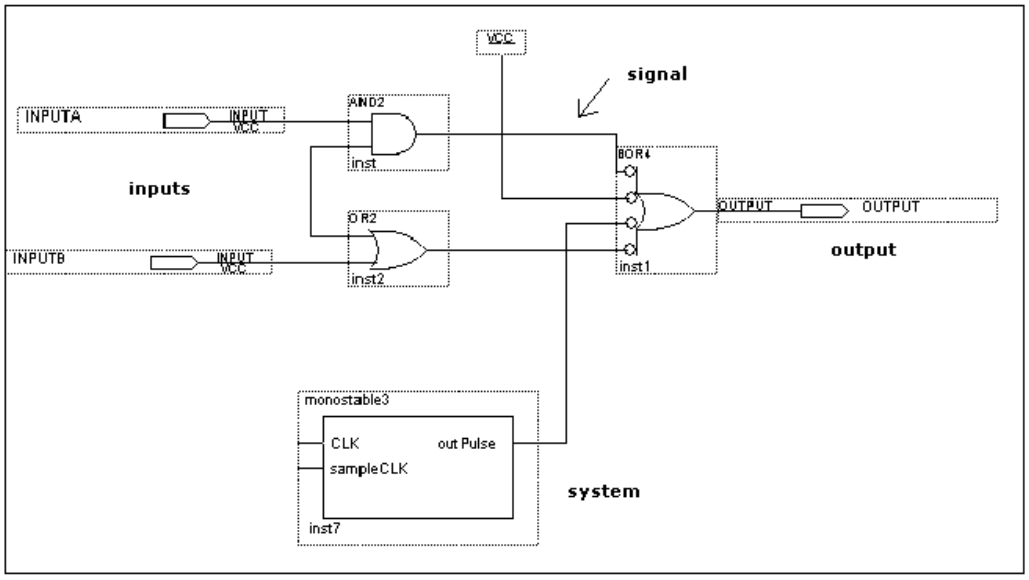


Figure 4.1 – Typical schematic entry file

Since systems can be entered into the schematic, this method lends itself to the engineering principle of *Abstraction*. Where a design is entered at different levels of implementation, such that the designer is abstracted away from the details of the implementation. This aids the designer's thinking in complex designs.

HDL entry is the programming of the software using a **H**ardware **D**escription Language (HDL). The 2 standard languages, Verilog and VHDL, are both supported by the QuartusII software. I made use of **VHDL.** An HDL editor is provided as part of the QuartusII package.

A HDL design can be used in a schematic file as well. This is accomplished by first creating a symbol file from the HDL design. There is a Quartus utility to do this. This symbol file is a visual representation of the HDL design. Symbol files can be made of schematic files as well, which facilitates Abstraction.

4.1.2 The Compiler

Compiling the design, involves the conversion of the design into FPLD resources, as well as the placement and routing of these resources.

Other processes also take place during compilation, such as Optimization, enforcement of certain timing and size constraints, and much more. These however are not necessary to understand for the purposes of this thesis.

4.1.3 The Simulator

The *Simulator,* is a very useful tool, as it allows the user to verify the functionality of the design, before configuring of the FPLD. There are 2 kinds of simulations: functional and timing simulations. Both have a similar user interface, but differ in emphasis.

Timing simulations take into consideration the actual propagation delays of signal, when passing through gates, and across chip lengths. The compilation process, as well as libraries that contain the characteristics of the actual FPLD used, generates this information. Thus this simulation provides a very close approximation to the actual behaviour of the FPLD after it is programmed.

Functional simulations, on the other hand, assume that the signal has zero propagation delay – i.e. passes through the system instantaneously. This is useful, as the design algorithm can be checked for functional correctness; especially during the initial design phase, when simply ideas are being tested. A lesser advantage (some would say greater) is that the compilation process is not needed to do such a simulation. This saves significant amounts of time, as the compilation process takes a long time (2 minutes, on a 3GHz Pentium, for my design) on bigger systems.

The simulator works, by producing a set of outputs from a set of inputs. This particular set of outputs and inputs, is defined in a **V**ector **W**aveform **F**ile (VWF). In this VWF you firstly place your desired input and output ports on a graphic table, then you change the values (e.g.1 or 0, if type bit) of the inputs. You then run the simulator. After a few seconds, there will be values on the outputs.

4.1.4 The Assignment tool

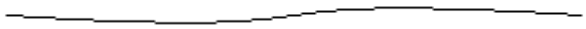
The *Assignment tool* simply takes care of your pin assignments. This means that you define on which specific pin of your FPLD (or device, in the Quartus terminology), you want your inputs and outputs. You also choose the FPLD, which is the target for your design, here.

4.1.5 The Programming tool

The Programmer, programs/configures the FPLD. Before you program, you first have to setup which download cable you are using and the particular computer port this download cable is attached to.

4.1.6 The Mega Wizard Plugin Manager

The Mega Wizard Plugin Manager is a useful tool for creating Megafunctions. Megafunctions are essentially, predefined functional blocks of logic, that are efficiently implemented in specific FPLDs. Examples of these blocks include RAM, multiplexers and other digital modules.



As mentioned previously, there are many more tools that are included in the Quartus software package. The ones mentioned above, were enough to complete this thesis. The reader is referred to Altera's website at www.altera.com, for a more comprehensive overview of the software.

There is one more feature of the QuartusII software that deserves special mention. After the design is compiled, the Quartus software indicates how much resources have been used by the design. The measure is the amount of gates, and it is also given as a percentage of the total amount of gates. This is extremely useful, as it allows the designer to choose an appropriately sized device for the design, at the corresponding price as well.

4.2 The UP2 Development Board

When prototyping designs it is often very useful to have a development board available. This is a general board, with indicators, LEDs, access to the input/output (IO) ports of a programmable device, and a host of other peripherals, that are designed to provide for most of the testing needs that a designer might have. Normally these

boards contain a particular type of programmable device that is very close to the target device that a designer has in mind for his design.

The University Program 2 (UP2) development board is one such board. This board have 2 programmable devices – the EPF10K70 and the EPM7128S CPLDs.

The EPF10K70 is based on the well-known FLEX10K family. It has 70 000 gates and 9 embedded array blocks (EAB). Each EAB provides 2048 bits of memory.

The EPM7128S is based on the MAX7000S family. It has 2500 gates.

On this board, each CPLD has its own share of peripherals. The FLEX10K was used in this thesis so only its peripherals will be described. It is suffice to say that the MAX7000 chip has similar support.

4.2.1 The peripherals

The ***FLEX_EXPANSION (FE)*** slots, is a very useful peripheral. They provide access to the IO pins of the FLEX10K. There are 3 FEs: FLEX_EXPANSION_A, FLEX_EXPANSION_B, and FLEX_EXPANSION_C. Each slot has a numbering system with a range of 60 down to 0. These numbers correspond to specific pin numbers on the FLEX10K chip. Another useful feature of the expansion slots is the provision of a regulated and unregulated voltage outputs. This facilitates connecting other systems to the board if necessary.

The next peripheral is the ***VGA female connector***. This allows a VGA monitor to be plugged into and controlled by the board. Some output pins of the FLEX10K are hardwired to this VGA connector.

A ***seven-segment LED display*** is the next peripheral. Some output pins of the FLEX10K are hardwired to this LED display. It should be remembered that the segments of the LED display have an active LOW input. This means that the LED segment should be driven with 0V to switch it on.

There is also a ***mouse connector***. A mouse can thus be connected and provide input to the FLEX10K. Again, dedicated pins exist for this purpose.

There are some ***LEDs***, ***pushbuttons*** and **DIP switches** also available on the board. Figure 4.2 shows a picture of the development board.

Figure 4.2 – Picture of the UP2 development board

5 Miscellaneous Experiments made during thesis

This section describes the set of experiments that was undertaken during the thesis. These experiments were done to verify my understanding of certain concepts, and also to test the limitations of certain ideas that I had. I am condensing the experimental setup and results in one subsection, for briefness.

5.1 TESTING THE ADC – TEST 1

5.1.1 Description of the ADC

The ADC used is the 8bit Flash ADC0820, from National Semiconductor. I studied the timing diagrams of the ADC. I then wire up the ADC to see that it functions as I expected. First a brief description of how the ADC works.

The easiest mode of operation, for controlling purposes, is the stand-alone mode. This mode is configured by tying the ***notRD*** **(pin 8)** input low. In stand-alone mode, a conversion is started by putting a very narrow (<50μs), active-low pulse on the ***notWE*** **(pin 6)** input. After a maximum time of 1.5μs the conversion is completed and the ***notINT*** **(pin 9)** pin then goes low. When the conversion is completed the data on the output pins are valid. The notINT pin stays low, until the next notWE pulse. (note that pin numbers are quoted for the dual-in-line package).

5.1.2 The experimental setup and results

I wired up the ADC to function in stand-alone mode. I then connected a pulse generator – which was a programmable Hewlett Packard HP signal generator – to the notWE input of the chip. The pulse length was programmed to be 1μs. The sampled

voltage was from a simple 10kΩ pot voltage divider. The input impedance allowed of the ADC allowed me to do this.

I then pulsed the ADC, and thereafter measured the voltage on the ADC data outputs with a multimeter. The outputs changed as I changed the input voltage and pulsed the circuit. This was an indication that the ADC worked as I expected, and that concluded my test.

5.1.3 Motivation for the choice of the length of the triggering pulse

The triggering pulse was chosen to be 1µs, as a first guess. Ideally I would have liked to choose the shortest possible pulse, which was 100ns. I did not know how the breadboard would deal with such short pulses however, considering the stray capacitances and inductances. Too long a pulse (>2µs) would start limiting the maximum sample rate (667kHz) that could be achieved, and cause the ADC sampling rate capabilities to be underutilized. 1µs was thus chosen. This would allow the ADC's sampling rate potential to be fully utilized, as well as, minimize the "second order" effects (ringing and overshoot) on the input pulse. The 1µs pulses worked, thus justifying my choice.

5.2 – TESTING THE ADC – TEST 2

TEST2 is a similar to TEST1, except that in TEST2, I create my own pulse generator. This was done to support the next experiment, which is described in 5.3 below.

5.2.1 The experimental setup and results

In this design the same setup is used as in TEST1, except that now the notWE pulse comes from a positive edge triggered monostable, with a 1µs time constant. This means that on a transition from low-to-high on the trigger input of the monostable, a 1µs pulse will be output. This pulse will be output to the ADC and a conversion will result. This can be summarized in figure 5.1 below.

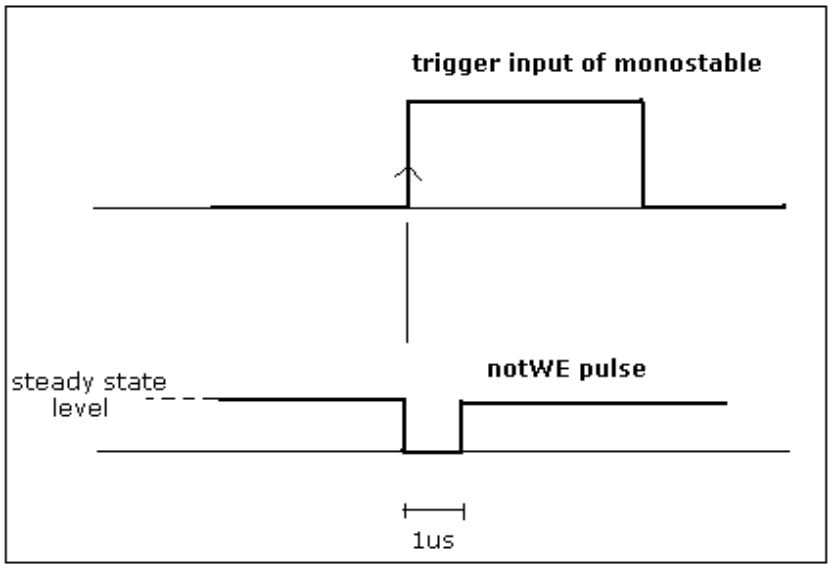


Figure 5.1 – TEST2 demonstration

The trigger was then fed with a periodic CLK. This produced periodic conversions as expected.

5.3 DEVELOPMENT BOARD EXPERIMENT

A few other smaller experiments were done. They were all integrated in the following bigger experiment. The goal of this experiment is to simulate a small section of the data2mem module. The data2mem module will be explained in section 7.3. For now it is enough to know that it waits for the notINT output of ADC to go low, indicating

a completed conversion, then it transfers the data on the ADC data lines to memory inside the CPLD. This experiment will however transfer the ADC data to the 7-segment LED display on the development board.

5.3.1 Experimental setup and results

This is experiment is setup as shown in figure 5.2 below.

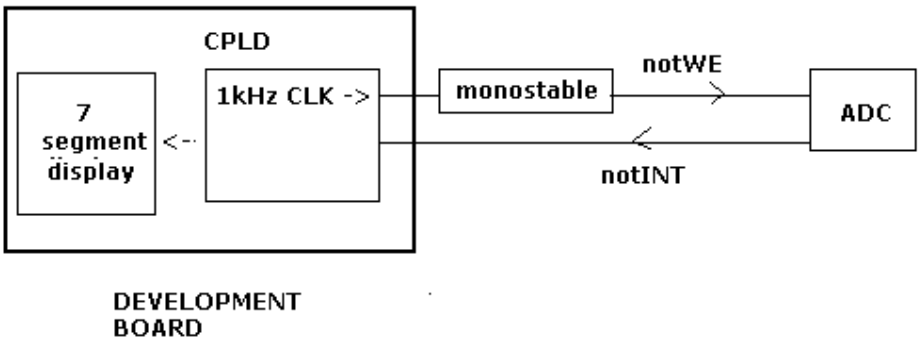


Figure 5.2 – Experimental setup for Development board experiment

The CPLD generates a 1kHz CLK. This causes the ADC to sample its input signal at 1kHz. The input signal is the variable voltage derived from the pot as explained earlier. The CPLD then on reception of the notINT transition to LOW, routes the data through to the 7-segment display. The experiment performed as expected. The output was in real time as perceived by a human, in that as the pot was adjusted the 7-segment display changed.

5.4 GENERAL DISCUSSION OF EXPERIMENTS

In all of the experiments, the aims of the experiments were never achieved on the first try. There was always some form of error, in either some of the circuitry, or a logical

error in the CPLD programming. In the end however, the objectives of the experiments were reached and the new lessons learned.

6. Systems View of Design

This section describes my design path, starting from the definition of the problem to be solved, assumptions made at the start of project, through to the final design that was implemented. This final description is on a systems level, with the later chapters filling in the details.

6.1 Definition of Problem

The problem to be solved can be shown in the high-level system diagram show in figure 6.1 below.

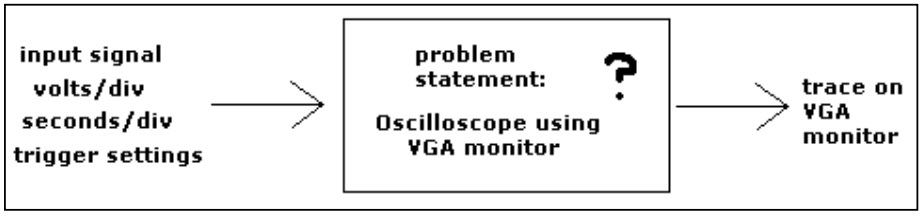


Figure 6.1 – The problem statement

This can be seen as the proverbial black box, the problem statement, system to be engineered, etc. We have a set of inputs and a corresponding set of outputs.

In this case the inputs are an input signal, amplitude/division, time/division and trigger settings. The outputs are the VGA monitor control and colour signals to produce the trace.

We can add more body and definiteness to the problem by describing a set of specifications that our black box should adhere to.

The initial specifications are:

i) 1MHz input signal bandwidth;

ii) 36V peak-to-peak maximum input signal range;

iii) DC coupled, level and slope triggering modes

6.2 Initial assumptions

Here I will describe some of the initial assumptions I made of the design. This was done to give some direction to a solution of the problem statement.

1) I felt that at least 2 processes would be required. One process would sample and control the ADC, while the other, would handle the controlling and displaying of data on the VGA monitor. This is shown in figure 6.2 below.

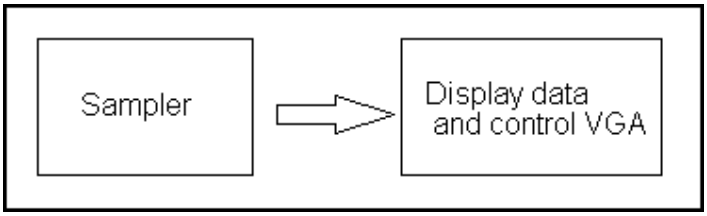


Figure 6.2 – Two-process architecture

The *sampler* process would sample the input signal at the appropriate times. The *display data* process, would display the data. This display process runs continuously.

2) Some form of memory might be needed. The sampling process will store the digitised samples. The display process would read from this memory at the appropriate time.

6.3 Initial Considerations

As described earlier a VGA monitor has a 25MHz clock. Pixel data are sent serially and at this rate. This calls for fast updates and thus a fast processor.

The first option might be to design around a microprocessor that can handle the fast processing. This was considered, but would have resulted in complex software coding. As at least 2 processes, sampling and displaying of data, were to run concurrently.

Another alternative is to implement this control in hardware, which is relatively fast, and inherently parallel. Perhaps using a PLD. I felt at this early stage that the design might be more complex than a PLD could handle. I decided to design around a FPLD, and after the hardware had been designed, I could then see if a less complex device would suffice. Especially since, the design software I was using shows the amount of resources being used. This facilitates the choosing of an appropriate programmable device.

The development board I had access to, had a FLEX10K CPLD on it. It also had an expansion facility that allowed easy access to the IO pins on the chip. I decided to base my design around this board.

6.4 The Design

The first design of the entire system is shown in figure 6.3 below.

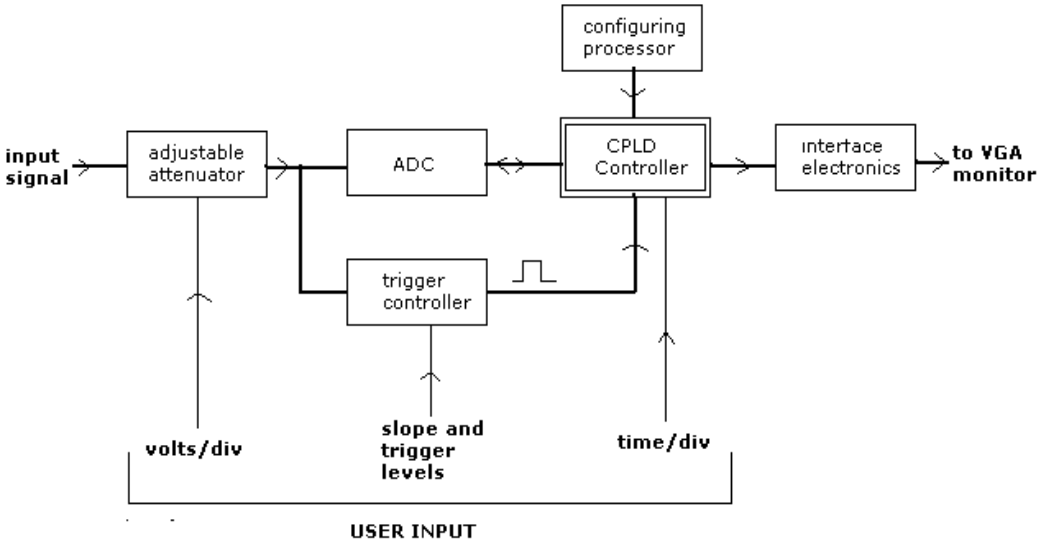


Figure 6.3 – Block diagram of the complete system

Starting from the left we have an *adjustable attenuator*. This module scales the input signal, down or up, to the input sensitivity of the ADC. The user selects the scaling

factor. This block should have a high impedance in order to minimize the loading of the circuit from which the input signal is being measured. This input impedance should also be compatible with the electrical characteristics of standard scope probes.

Next we have the *ADC.* This module is responsible for the conversion of the analogue input signal to its digital form, for further processing. The conversion rate should be as fast as possible as it has a direct bearing on the bandwidth of the scope. A faster conversion rate implies a faster sampling time, and thus a higher range of frequencies that can be measured.

The *trigger controller* is responsible for the selection of the triggering modes of the oscilloscope. The user sets the trigger mode and level. This should be designed with high impedance to minimize loading. The output of this module is a fixed length pulse which initiates a sampling frame, as will be described later.

The *CPLD controller* is the brain of the whole system. Essentially it implements the 2-process architecture mentioned earlier (section 6.2). The software that describes this controller will be described in detail in the later chapters. For now it is enough to know that this module is responsible for:

i) The controlling of the ADC;

ii) Controlling the sampling rate.

iii) Starting a sampling frame in response to a trigger signal

iv) The transformation of input digital data from ADC, to a form that can be displayed on a VGA monitor;

v) The controlling of the VGA monitor.

The *configuring processor* configures the CPLD at start up. This could be a configuring ROM, especially made for this function, or a simple microprocessor.

Finally, *the interface electronics*, simply provides the appropriate signal levels for the VGA monitor control lines.

This design covers the whole system – the black box described earlier. Most of the modules discussed above are fairly straightforward to understand. The CPLD controller however, is a system by itself. A block diagram is shown in figure 6.4.

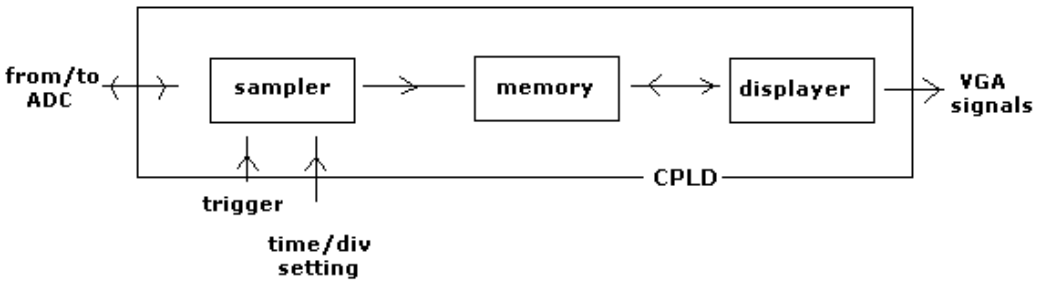


Figure 6.4 – Block diagram of the Internals of the CPLD controller

Sampler controls the ADC, and also receives the digitised signal from the ADC. It is brought into action, when it receives a trigger pulse. It is also responsible for adjusting the sampling rate in order to provide a time-calibrated sequence of input data to memory. The time/div setting controls this. ***Memory*** passively stores and retrieves data when asked to do so by Sampler and Displayer while, ***Displayer*** reads, transforms and displays the data in memory continuously.

These different blocks of the CPLD controller will be described in more detail in the following chapters.

7. The Sampler module of the CPLD system

This module is responsible for the sampling aspects of the CPLD controller.

7.1 General Description of Sampler

Figure 7.1 shows the input and output ports of sampler.

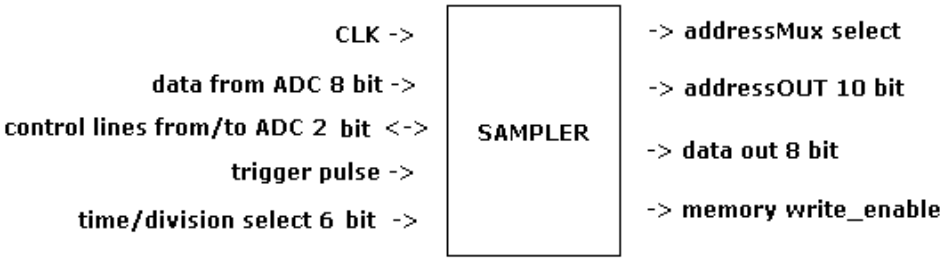


Figure 7.1 – Input and Output ports of Sampler

Sampler consists of 2 processes, which run concurrently. Figure 7.2 illustrates.

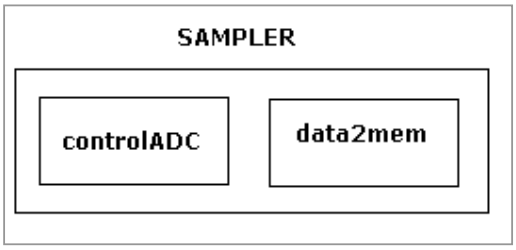


Figure 7.2 – Sampler – 2 process system

7.2 ControlADC process

Figure 7.3 illustrates this process.

Figure 7.3 – controlADC's input and output ports

This process controls the ADC. This ADC was described in section 5.1. On the reception of a trigger pulse, it sends out 640 1μs pulses to the **notWE** pin of the ADC, at a certain rate – i.e. 640 samples are taken of the input signal at this rate. This rate is adjusted to provide a calibrated sampled sequence that corresponds to the **time/division** setting. This time/division setting is set by the user and is input to this module

It also has an index counter (**sampleNo**) that keeps track of the current sample. This is important for addressing purposes, as we will see next.

7.2.1 – Break down of ControlADC

ControlADC is in fact a system on its own. The block diagram in figure 7.4 describes this breakdown.

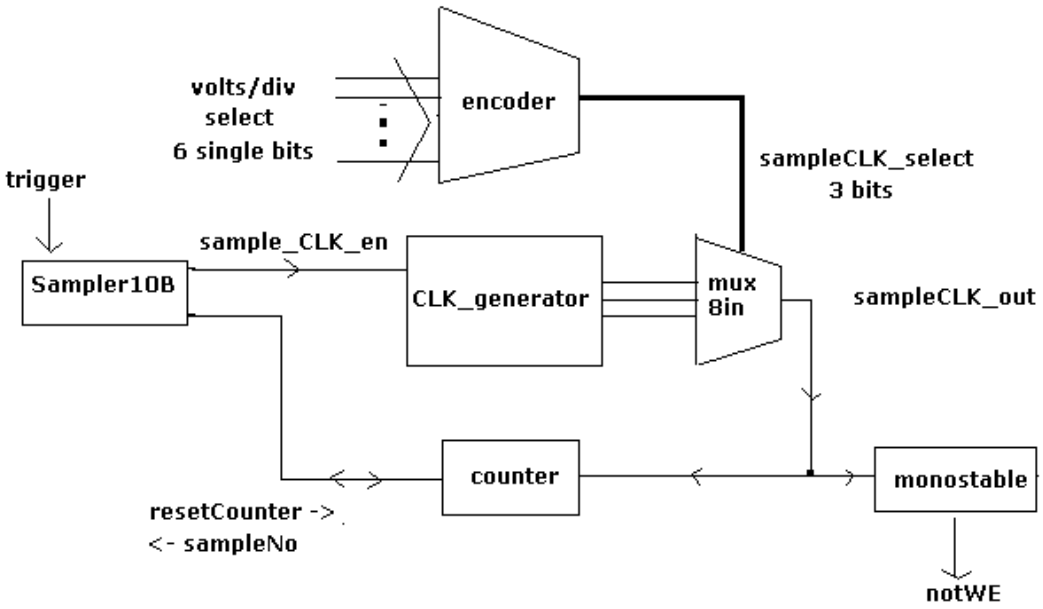


Figure 7.4 – Breakdown of ControlADC

Most of the intelligence of controlADC lies in **sampler10B**. This process **enables** the **CLK_generator** to send out an array of CLKs. These CLKs are put into a multiplexor (**mux8in**), the select bits of which, comes from an **encoded** version of the volts/div bit selected. The output of this MUX is the sampling CLK at the calibrated rate. This goes into a **monostable that** converts the sampling CLK to 1µs pulses. This sampling CLK also goes into the sampleNo **counter** which increments on the reception on a +ve edge on sampling CLK. The counter is reset after 640 samples by sampler10B.

7.3 Data to Memory (data2mem) process

Figure 7.5 illustrates this process.

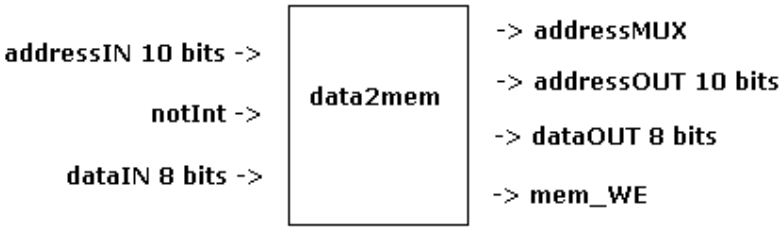


Figure 7.5 – Data2Mem's input and output ports

The notINT output pin of the ADC initiates this process. As described earlier, this HIGH-to-LOW transition indicates that the ADC has completed a conversion. On being initiated, the process gathers the data from the ADC (**dataIN**), the current sample number (**addressIN**) from the controlADC process (**sampleNo**), and makes the **addressMux** bit high. This is in preparation to write to **memory.** The following clock cycle, memory_write_enable (**mem_WE**) goes high, which stores the **dataOUT** bits in **memory**. The next clock cycle, addressMux and mem_WE goes low. This reason for this will be described next.

7.4 Discussion

The two processes of Sampler are independent, in that they are initiated from external events, yet I put them in the bigger process. **Sampler.** This was done to aid in the understanding of the design. Both of these processes deal with the sampling and subsequent shuffling of the data to memory. They thus have something in common, hence the grouping.

Another question might be why a memory write takes place in 3 clock cycles. This is done to so that the write to memory can be stable. The FLEX10K datasheets suggest that the address and data busses be stable when the mem_WE input is made high, for effective writing. Hence the 3-clock cycle write.

640 samples are taken per sampling frame, as the VGA monitor have 640 horizontal pixels. More light will be shed on this later.

Finally, the reason for the **addressMux** will be explained later, when the **memory** and **displayer** modules have been described.

8. The Memory module of the CPLD system

This chapter describes the functionality of the Memory module

8.1 Description of the RAM

The FLEX10K have memory modules embedded inside of it. This can be efficiently utilized as a RAM, by using a megafunction that is specific to the FLEX10K. A 1024 x 8 bit array of memory cells was implemented in this design. When writing to this memory, the address and data line should be stable before the write mode is enabled.

8.2 General description of Memory module

Memory is a passive system, in that it is completely controlled by processes external to it. It is also a relatively simple module. Figure 8.1 shows a complete breakdown of Memory.

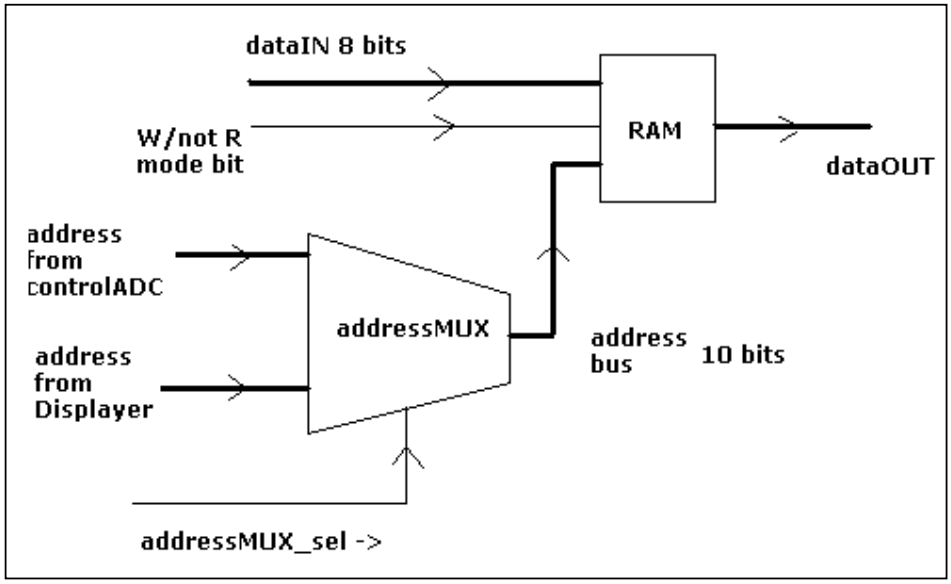


Figure 8.1 – Design of Memory module

This completely describes Memory module. In a later chapter some of the design issues – such as the size of memory, and its organization – will be discussed

9 The Displayer module of the CPLD system

This module describes the functionality of the Displayer module. This module takes care of the visuals of the oscilloscope.

9.1 Description of VGA monitor

The VGA monitor was explained earlier in more detailed. It is now enough to say that it consists of a 480 x 640 matrix of pixels; RGB data is sent serially to the monitor; and that control is accomplished by setting the VSYNC and HSYNC signals at the appropriate times.

9.2 General description of Displayer module

Displayer is responsible for conditioning the value of the data sample stored in memory, in order for it to be displayed on a VGA monitor. It then goes on to display the trace stored in memory. Displayer runs continuously. Figure 9.1 summarizes the Displayer module.

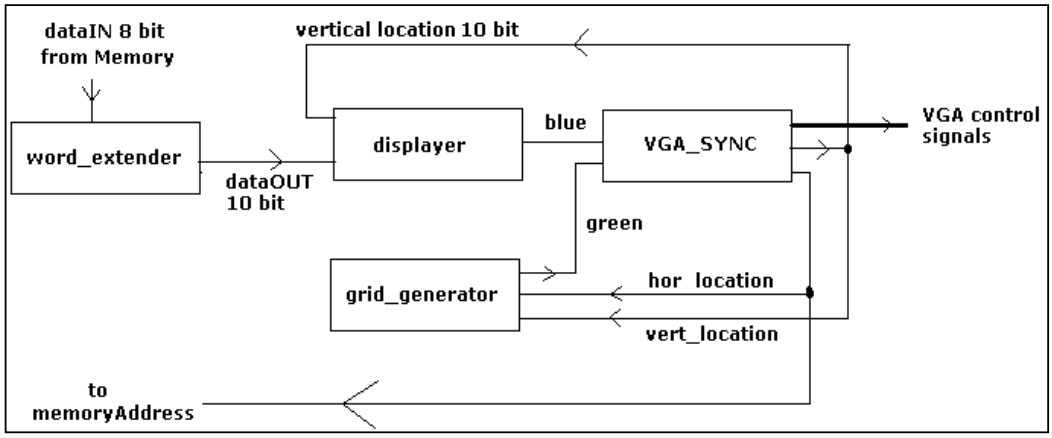


Figure 9.1 – Block Diagram of Displayer

VGA_SYNC outputs the horizontal location of the current pixel. This is sent to Memory as an address. Memory produces the data at this address, on the **dataIN** port, when it is in **READ** mode. **Word_extender** takes in this data and extends it to a 10 bit data word, by adding a leading and a trailing bit. The leading bit (MSB) is made 0, and the trailing bit (LSB) is made 1. This 10 bit word is produced to **displayer** (not the big system – Displayer). This process compares the current vertical (row) location to an inverted version of the 10 bit data word. If they are equal, the current BLUE pixel is set 1, else it is set to 0.

The Displayer also takes in the addressMux_select signal (not shown) from the sampler process. This is seen as a **busy** signal. If it is high, the **displayer** block (not the big Displayer) outputs a 0 to the **BLUE** colour signal, regardless of the data on **DataIN**.

Grid_generator simply takes in the horizontal and vertical locations of the current pixel, and sets **GREEN** to 1 if they are equal to the grid locations. The grid locations are defined by logic.

9.2.1 A special note on VGA_SYNC

VGA_SYNC is the actual interface to the VGA control signals. It simply consists of horizontal and vertical counters that keep track of the current pixel location on the VGA monitor. It takes whatever signals are on the RGB inputs at a current time, and sends them to the VGA RGB signal. It also sends out the HSYNC and VSYNC signals at the appropriate times.

VGA_SYNC is a module that was taken directly from a book[2].

9.3 Discussions

The word_extender process was included, as the **data input** (8 bits) was compared to the **row_address,** which was 10 bits wide. There are programming facilities for VHDL, whereby a comparison could be made with different data words. By using only parts of the greater word and comparing it to the smaller. However, the dataIN had to also be inverted and this made using those programming facilities difficult.

The dataIN was inverted as the VGA monitor starts counting from the top, whereas my dataIN word would have its least value at the bottom. This orientation had to be corrected by the inversion process.

The dataIN was not inverted per se. There was also a maximum value of 480 placed on this data after it had been inverted. This was an arbitrary design decision, just done to show that the input signal was saturating the display for testing purposes – an indication to increase the volts/division setting for the uninitiated. This could just have been left unchecked. Then nothing will have been shown at these saturation sections, which is what conventional oscilloscopes do.

Finally, the fact that the LSB of the extended word was set to 1, meant that the trace had a pixel resolution of 2 pixels. This is too small to be spotted by the eye however.

10. The CPLD system – in retrospect

This chapters unifies the entire CPLD system. It also discusses the major design decision that were made at the system level.

10.1 The CPLD system – a system description

We saw the individual parts, as well as the broad overview of the CPLD system. Now I will describe the operation of the whole system.

10.1.1 The Sampler module

The **Sampler** module samples the input signal 640 times at a sampling rate set by the time/div setting. The maximum sampling rate is 500kHz. This data is written to the **Memory** module, with the address being the current sample number. The Sampler module controls the memory module, including whether the memory module is in WRITE/not READ mode. When a sample is received from the ADC, Sampler uses 3 clock cycles to write this value to memory. In the first CLK cycle the following happens:

1. the address lines **output** from the sampler module and **input** to one of the address lines of the addressMux in memory, are set to the current sample number;
2. the data lines **output** from the sampler, and **input** to the memory, are set to the ADC data lines;
3. the a**ddressMUX select bit,** output from **sampler,** and input to the **addressMUX** select bit in **memory** is set to 1, which selects the **sampler** address as the address bus in **memory;**

4. This addressMUX select bit also sets the input BUSY signal of the **Displayer** module.

During the **second** CLK cycle the mode of memory is set to WRITE, which causes the data word to be written. In the **third** and final CLK cycle, the AddressMUX bit of memory is again set to route the Display-address through to the memory address bus and the mode bit is again reset to READ.

As can be inferred from the above description, the memory is normally in the READ mode – i.e. the display address is always routed to be on the memory address bus via the addressMux of memory.

10.1.2 The Memory module

This module is self explanatory, and is described while describing the other modules

10.1.3 The Displayer module

The Displayer module continually reads and displays data from memory. If the sampler module is performing a write, reading from memory is disabled, and the 3 pixels that are missed during this time are just set to 0. The address to memory is the current location of the horizontal pixel.

There are 640 horizontal pixels and thus 640 memory addresses for each of the pixels. The value of the data determines its vertical height on the monitor.

10.2 Discussions

The sampler module controls the memory. This happens because relative to the rate at which the display module does reads from memory, writes to memory occur rarely. Reads from memory occur every 40ns during the displaying of 1 row of pixels. Writes to memory however, occur at a maximum rate of 500kHz (2μs).

Also, a write to memory is more critical than a read from memory. If the display process misses a few pixels in a single display frame, the observer of the monitor will not see a difference; whereas if the write process does not write a few data samples, that data is lost. The pixels are missing from the display.

There are other reasons, such as the stability required by a write, and more. But the point has been made.

11. The Supporting Electronics

This section describes the supporting electronics that implemented the prototype system. Only the trigger circuit deserves special mention, as the ADC was basically just connected to the power supply and the IO ports of the development board. The Demonstration circuit is included just for completeness. The actual circuit diagrams can be found in the appendix.

11.1 The Trigger System

The trigger system implemented positive slope, level triggering. Figure 11.1 shows the simple system, and its description.

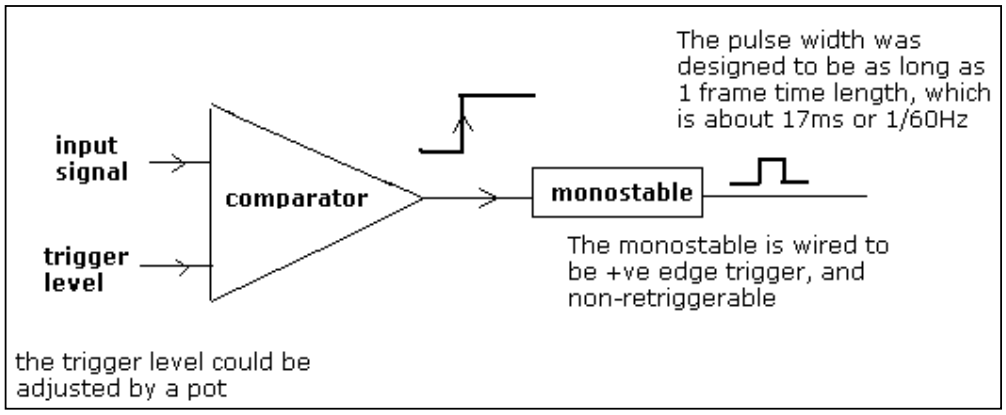


Figure 11.1 – the trigger circuit

11.2 The Demonstration circuit

The demo circuit was just the popular 555 triangular wave generator.

11. Results of the Thesis

This section displays the results of the previous chapters of work.

11.1 The Trace

The figures below are pictures of the trace. It displays the images obtained from the prototype system, with the demo 555 triangular wave generator as the input signal.

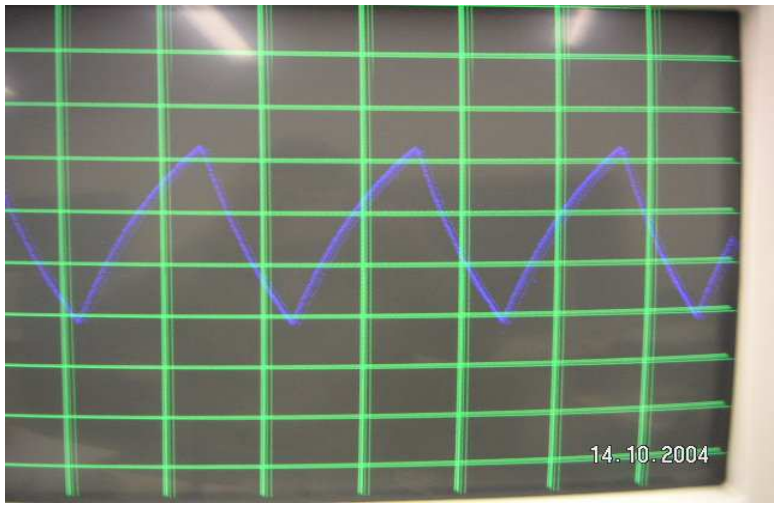
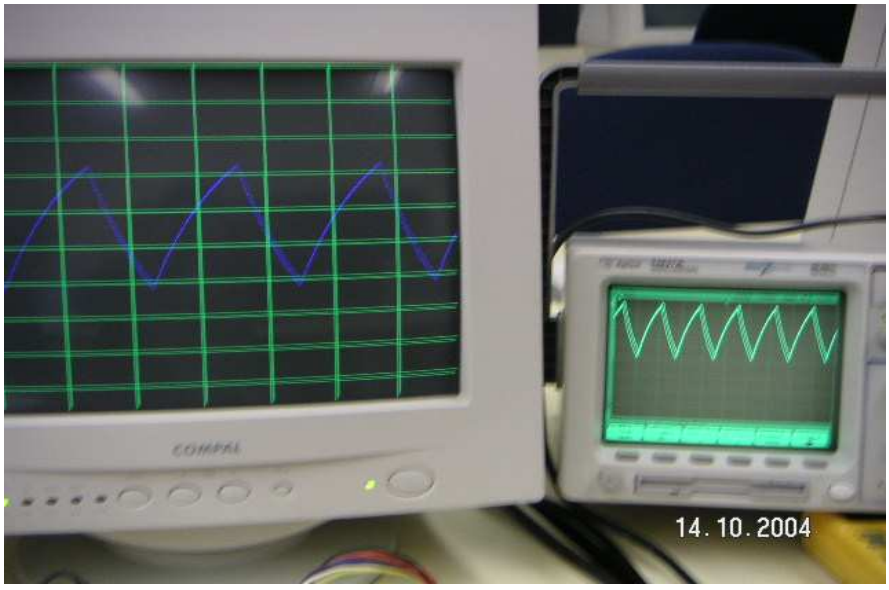


Figure 11.1 – the results



13. Conclusions

You have seen the results. It works! It is however just a partial solution to the problem statement described in the introduction.

13.1 Shortcomings to the design

The prototype design has the following shortcomings:

1) *It does not have volts/division scaling*. This is however an electronics problem. It is very easy to add amplitude scaling.
2) *It does not have a 1MHz input signal bandwidth*. The maximum sampling rate is 500kHz. The initial limit to the sampling time is the ADC's conversion rate. It is rate at 1.5μs – which implies a maximum sampling rate of 667kHz.

The oscilloscope does however comply with the other requirements of the design.

13.2 Potential of Design

The partial solution of getting a trace on a VGA monitor, has proven that these kind of oscilloscopes are possible. This has enormous potential for educational institutions who often do not have the laboratory resources to provide a high degree of user interaction. Old VGA monitors can now be plugged into this oscilloscope board. With this prototype as it is, I do not see any complication to adapting it to have a higher input signal bandwidth.

14. Recommendations

The person following on from this project is advised to:

14.1 Increase the main CLK rate.

The CLK rate should be increase to at least 75MHz to provide the capability of a higher ADC. Increasing the main CLK rate, is however coupled to the next recommendation.

14.2 Build the design on a PCB

This should be done to support the higher CLK rates required on the board.

14.3 User a faster ADC

A faster ADC should be used to support a greater input signal bandwidth.

14.4 Add more coupling modes, for the trigger and input signal

This is a very simple electronic addition to the prototype design.

14.5 Add a bolded horizontal and vertical centerline on the display.

This does not require any extra resources. It simply requires a modification of the grid code.

15. References

1) Various web site were visited and information was obtained from these websites They are too numerous to enumerate however. The primary web sources of help are outlined below. These were visited during the second half of 2004.

The main web sites are:
i) www.altera.com
ii) www.fpga4fun.com
iii) www.vhdl-online.de/~vhdl/tutorial/
iv) http://members.tripod.com/michaelgellis/scope.html
v) http://www.icd.com.au/vhdl.html

2) Hamblen J.O. & Furman M.D, 2001, Kluwer Academic Publishers, **Rapid prototyping of Digital systems, 2nd edition**

3) Horowitz P.,Hill W.,1989,Cambridge University Press, **The art of Electronics, 2nd Edition.**

4) Skahill K., 1996, Addison-Westley, **VHDL for Programmable Logic**

5) Mano M.M, Kime C.R., 2001, Prentice Hall**, Logic and Computer Design Fundamentals, 2nd edition**

6) **Datasheets** of the various manufacturers' electronic parts I have used.

7) **UP2 Development board User Manual**

16 Appendices

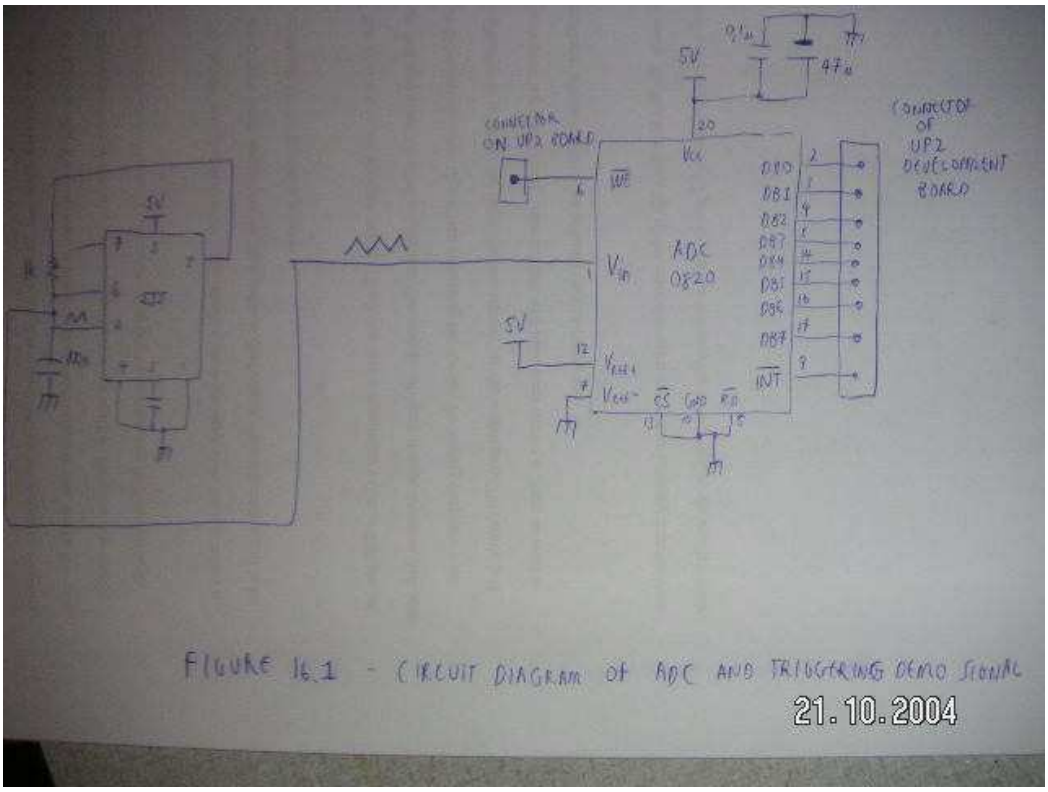


FIGURE 16.1 - CIRCUIT DIAGRAM OF ADC AND TRIGGERING DEMO SIGNAL

16.2

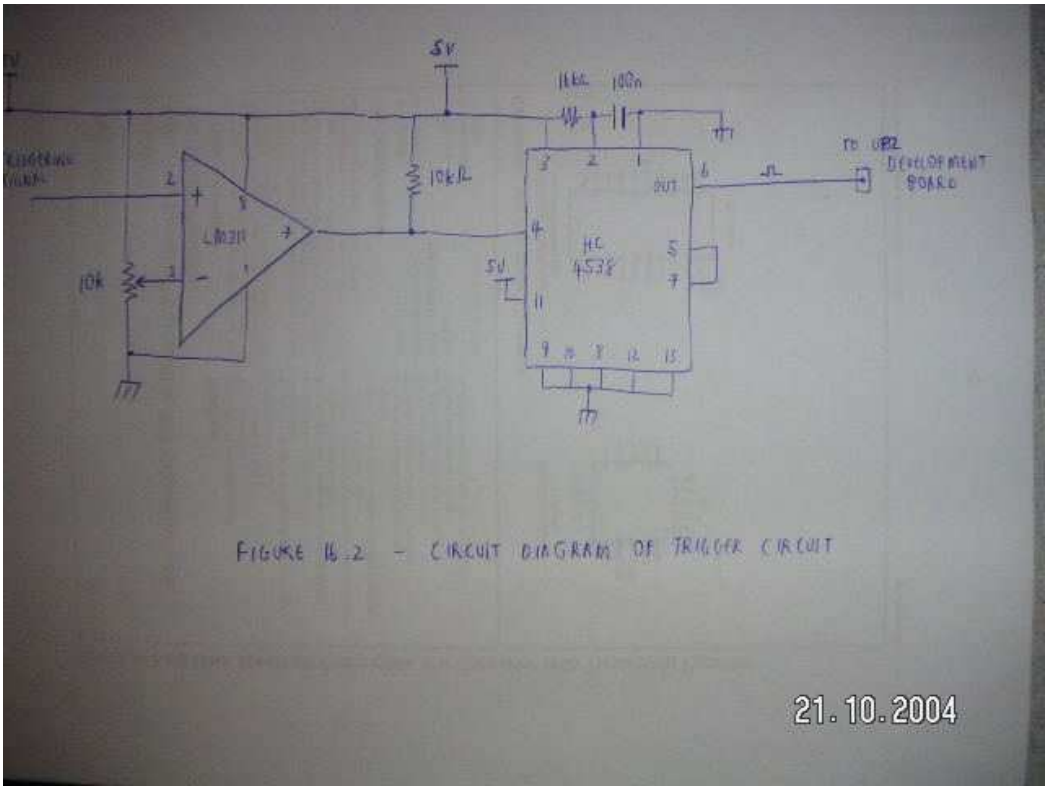


FIGURE 16.2 - CIRCUIT DIAGRAM OF TRIGGER CIRCUIT

National Semiconductor

March 2004

ADC0820
8-Bit High Speed µP Compatible A/D Converter with Track/Hold Function

General Description

By using a half-flash conversion technique, the 8-bit ADC0820 CMOS A/D offers a 1.5 µs conversion time and dissipates only 75 mW of power. The half-flash technique consists of 32 comparators, a most significant 4-bit ADC and a least significant 4-bit ADC.

The input to the ADC0820 is tracked and held by the input sampling circuitry eliminating the need for an external sample-and-hold for signals moving at less than 100 mV/µs.

For ease of interface to microprocessors, the ADC0820 has been designed to appear as a memory location or I/O port without the need for external interfacing logic.

Key Specifications

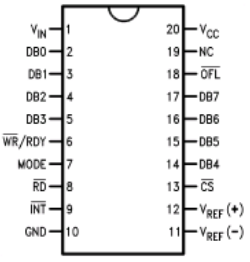
■ Resolution	8 Bits
■ Conversion Time	2.5 µs Max (RD Mode) 1.5 µs Max (WR-RD Mode)
■ Low Power	75 mW Max
■ Total Unadjusted Error	±½ LSB and ± 1 LSB

Features

- Built-in track-and-hold function
- No missing codes
- No external clocking
- Single supply—5 V_{DC}
- Easy interface to all microprocessors, or operates stand-alone
- Latched TRI-STATE output
- Logic inputs and outputs meet both MOS and T^2L voltage level specifications
- Operates ratiometrically or with any reference value equal to or less than V_{CC}
- 0V to 5V analog input voltage range with single 5V supply
- No zero or full-scale adjust required
- Overflow output available for cascading
- 0.3" standard width 20-pin DIP
- 20-pin molded chip carrier package
- 20-pin small outline package
- 20-pin shrink small outline package (SSOP)

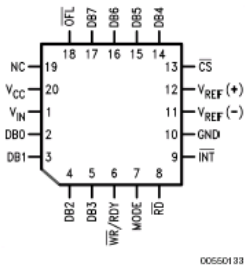
Connection and Functional Diagrams

Dual-In-Line, Small Outline and SSOP Packages



Top View

Molded Chip Carrier Package



16.4 – THE PROGRAMMING FILES

All the software can be found on the included CDROM. The software folder is the projects folder with all the design files. Folders within this folder, contains the design of the individual parts in a logical hierarchy. They are not needed for the compilation process however.

The CDROM is structured as follows:

1. The University Program 2 USER MANUAL
2. The Software

   -> FINAL_SYSTEM is the top level schematic