

University of Southampton
Faculty of Physical Sciences and Engineering
School of Electronics and Computer Science

Mercury Delay Line Emulation:
Reconstructing Memory for EDSAC

by

Joshua Tyler
August 30, 2017

A dissertation submitted in partial fulfilment of the degree of
MSc Embedded Systems

Abstract

Contents

Abstract	i
Table Of Contents	ii
List of Acronyms	iv
List of Figures	v
List of Tables	v
1 INTRODUCTION	1
2 TECHNOLOGY REVIEW	2
2.1 EDSAC overview	2
2.2 EDSAC Memory Architecture	2
2.2.1 Timing	3
2.2.2 Electrical	4
2.2.3 Mechanical	5
3 SPECIFICATION	7
4 POWER SYSTEM DESIGN	8
4.1 DC Offset Solution	9
4.2 Valve Power Supply Solution	10
4.2.1 Circuit Design	11
5 DELAY LINE DESIGN	13
5.1 Digital Design	13
5.1.1 Architecture selection	13
5.1.2 HDL Design	17
5.2 Analogue Design	19
5.2.1 Input	19
5.2.2 Output	20
6 TEST HARNESS DESIGN	22
6.1 Digital Design	22
6.2 Analogue Design	22
6.2.1 Input	22
6.2.2 Output	22
6.3 Software Design	22
7 SIMULATION AND VERIFICATION	24
7.1 SPICE Simulation	24
7.1.1 Valve Models	24
7.1.2 Delay Line Input Simulation	24
7.1.3 Delay Line Output Simulation	24

7.1.4	Store Regeneration Input Simulation	24
7.2	HDL Verification	24
7.2.1	Delay Line Verification	24
7.2.2	Test Harness Verification	25
8	SYSTEM INTEGRATION	26
8.1	Mechanical	26
8.2	System Level Testing	26
8.2.1	Issues	26
8.3	Measurements	26
9	PROJECT PLANNING	27
9.1	Time Management	27
9.2	Risk Management	27
9.3	Financial Management	27
10	FINAL COMMENTS	28
Bibliography		29
Appendix A Folder Structure		31
Appendix B HDL Code Overview		33
Appendix C Schematics		34

List of Acronyms

AC alternating current.

ARM advanced RISC machine.

CPLD complex programmable logic device.

DC direct current.

EDSAC electronic delay storage automatic calculator.

ENIAC electronic numerical integrator and computer.

FIFO first in, first out.

FPGA field programmable gate array.

GBP gain-bandwidth product.

GPIO general purpose input/output.

HDL hardware description language.

IC integrated circuit.

LC logic cell.

LDO low-dropout.

LUT look-up table.

LVCMOS low voltage complementary metal oxide semiconductor.

PCB printed circuit board.

PLL phase locked loop.

RAM random access memory.

RC resistor-capacitor.

RCBO residual-current circuit breaker with overcurrent protection.

RMS root mean square.

SPICE simulation program with integrated circuit emphasis.

TQFP thin quad flat package.

VHDL VHSIC hardware definition language.

List of Figures

2.1	W.Renwick and M.Wilkes standing with EDSAC [1]	2
2.2	Demonstration of the principle of delay line memory, adapted from [2]	3
2.3	EDSAC pulse timing	5
2.4	A battery of mercury delay lines in EDSAC [3]	6
4.1	Store Regeneration Unit Output [4]	9
4.2	Proposed DC offset solution	10
4.3	Proposed valve heater supply solution, equivalent circuit	11
4.4	Delay line power supply schematic [replace picture]	12
5.1	The overall architecture of the delay line	13
5.2	Proposed architecture for a microcontroller based system	14
5.3	Proposed architecture for a FPGA or discrete system	15
5.4	Rising Edge Detector State Machine	18
5.5	Comparator State Machine	18
5.6	Delay line input schematic [replace with better schematic]	19
5.7	Delay line output schematic [replace with better schematic]	20
5.8	Effect of having the op-amp output idle at 0 V	21
6.1	Proposed Architecture for Test Harness	22
6.2	Proposed Architecture for Memory Manager	23
7.1	Delay line testbench output	25

List of Tables

3.1	Delay line specification	7
5.1	FPGA Requirements	16

Chapter 1

INTRODUCTION

The National Museum of Computing is currently hosting a project to reconstruct a very early computer: electronic delay storage automatic calculator (EDSAC), the exact nature of the machine will be discussed in Chapter 2, however the recreation of its memory is non trivial and is the aim of this project.

[This segment is out of place] Creating a faithful reproduction of this system poses challenges, namely: the expense of mercury, the health and safety implications of using mercury in a museum environment, and the technical challenges of the precise machining necessary for the steel tubes. As a result of this the reconstruction project currently intends to use magnetostrictive delay lines [5].

As discussed at length in [2], this method is non-ideal since it is: anachronistic for the time, dissimilar in appearance to the original delay lines, and dissimilar in terms of electrical interface to the original. For this reason it was decided to investigate the use of modern technology to emulate the original delay lines. This emulation is required to be indistinguishable from the original in terms of appearance and electrical interface. The design of this system is the goal of this project.

Chapter 2

TECHNOLOGY REVIEW

This Chapter presents an overview of EDSAC, and an overview of the relevant literature surrounding its memory architecture.

The data derived will be used to derive a specification for the recreated memory solution, presented in Chapter 3. It should be noted that much of the literature presented comes from original documentation produced by Maurice Wilkes, the man who led the original EDSAC project.

2.1 EDSAC overview

EDSAC, pictured with two of its creators in Figure 2.1 [6], was the first practical digital stored program computer. This means that it was the first practical computer able to accept a program from the user, store it in memory, and execute it on the fly. In contrast to this earlier computers, such as electronic numerical integrator and computer (ENIAC), hard-coded programs using switches, in the case of ENIAC using 3,600 ten 10-way switches [7]. The only digital stored program computer earlier than EDSAC was the Manchester small-scale experimental machine. This machine was not intended for general purpose computation however, but rather for testing of a new type of memory [8].

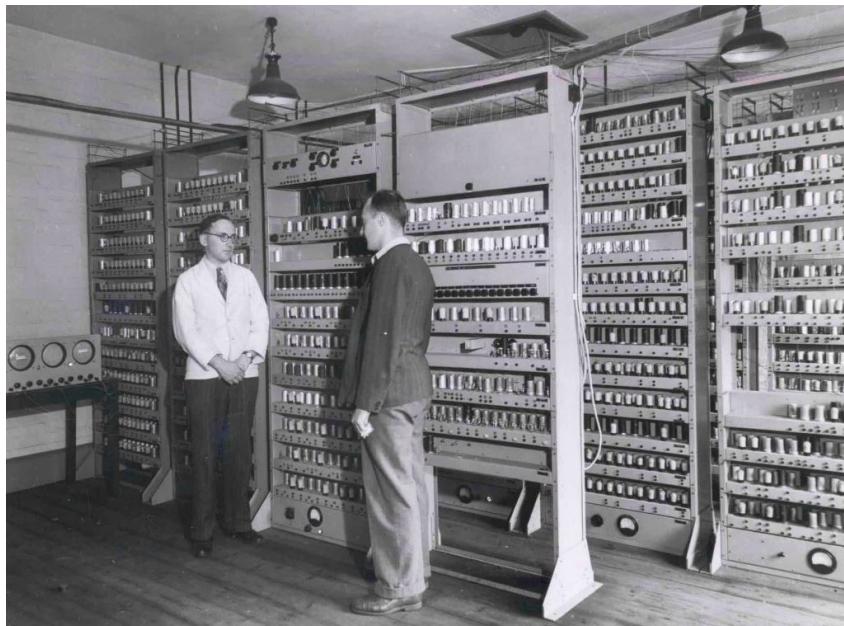


Figure 2.1: W.Renwick and M.Wilkes standing with EDSAC [1]

2.2 EDSAC Memory Architecture

EDSAC ran its first program in May 1949. This posed a significant design challenge for the memory of the machine. Transistors were not commercially available at all until 1951 [9],

and valves, whilst available, were physically large, and were fairly expensive. This meant that creation of even a modest amount of storage would not have been feasible. ENIAC used valves, but it also had very little memory. This was not a problem for its intended application, but would have posed a problem for a general purpose computing platform, such as EDSAC [10, p.208].

The solution chosen for EDSAC's storage problem was delay line memory. This was common with other early computers, and works via a fairly simple mechanism. Given a medium able to delay a pulse train by a certain amount, memory can be created by feeding the output of that delay medium back into the input. If the delay time is tuned to be an integer multiple of the system clock frequency, the system is able to store a sequence of bits proportional in length to the delay time. This principle is illustrated in Figure 2.2.

Delay lines exist in various forms, from magnetoresistive delay lines which function by twisting one end of a coil of wire, then waiting for the stress to propagate to the other end of the wire, to electric delay lines which provide much smaller delays by sending electrical impulses down a length of coaxial wire or a printed circuit board (PCB) microstrip trace.

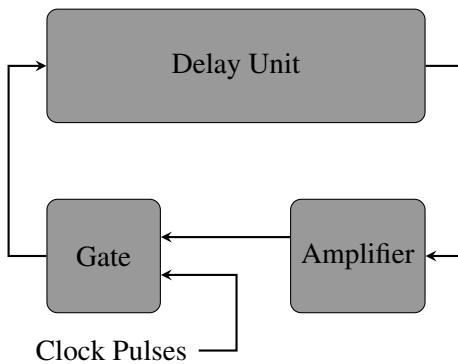


Figure 2.2: Demonstration of the principle of delay line memory, adapted from [2]

EDSAC used acoustic delay lines in the form of steel tubes, approximately 1.5 m long, and full of mercury. Impulses were inserted into one end of the tube via a quartz transducer, and reach the other end of the tube after a delay of approximately 1 ms. Here a second quartz transducer converts the incoming acoustic impulse into an electronic impulse.

These delay lines are used in two ways:

1. Short tubes used for the results of calculations.
2. Batteries of longer tubes used as the main memory store. This is analogous to random access memory (RAM) in a modern computer.

EDSAC stores words in units of 36 pulses, which is referred to as a minor cycle. 34 pulses are used to store the magnitude of a number, one stores the sign, and one acts as a space between numbers. This system was chosen to allow storage of ten digit numbers.

The shorter tubes are sized to store only a single minor cycle, but the longer tubes are long enough to store 576 pulses (16 minor cycles). The batteries of each of these tubes each contained 16 tubes, and EDSAC originally had two batteries, allowing for a total storage capacity of $16 \times 16 \times 2 = 512$ numbers.

2.2.1 Timing

The memory in EDSAC uses a circulating bit rate of 500 kHz. This is made up of a $0.9\ \mu\text{s}$ pulse, and a $1.1\ \mu\text{s}$ gap for each bit (although some literature specifies a pulse of $0.9\ \mu\text{s}$ and a

gap of $1.0\ \mu\text{s}$, implying a $526\ \text{kHz}$ bit rate). The pulse is a burst of $13.5\ \text{MHz}$ carrier frequency if the bit is a logical 1, or it is 0V if the bit is a logical 0.

It should be noted that the reconstruction will use a bit rate slightly faster or slower than $500\ \text{kHz}$, because $500\ \text{kHz}$ is an international distress frequency, and given the large wiring looms in EDSAC, it would be quite easy for EDSAC to become an unintentional transmitter of this frequency. This would initially appear to oppose a faithful recreation of EDSAC, however one needs to consider the reaction of the mercury originally used to changes in temperature. Mercury was chosen because its acoustic delay does not vary much with temperature. The variation was still large enough however to cause the system to break over the normal temperature range of a laboratory environment. To combat this, the original clock frequency was adjusted with temperature such that the delay of each delay line was an integer multiple of the clock period. Later this system was deemed unsatisfactory and the coffins were enclosed in a temperature controlled environment.

To allow consistent operation of the machine, it is desirable that the recreated EDSAC does not emulate this variation of delay dependant upon temperature, but the delay of each line should be tunable around the nominal delay given by Equation 2.1.

$$D = \frac{1}{500\ \text{kHz}} \times 576 = 1.15\ \text{ms} \quad (2.1)$$

In the regeneration portion of the circuitry, the pulses are demodulated from the $13.5\ \text{MHz}$ carrier and stretched to approximately $1.9\ \mu\text{s}$ long, i.e. just long enough that each pulse fails to overlap its neighbour [10, p.212].

This is critical because it means that the pulses that have passed through the delay line are effectively resynchronised to the system clock. If the system did not work in this way, then the delay lines in the original EDSAC would never have worked. The propagation time of acoustic waves through mercury varies slightly with temperature, and the cumulative effect of this through all the delay lines of the store would mean that pulses would be unlikely to align with the system clock at the other end.

Despite this resynchronisation after each delay line, the original EDSAC suffered a great deal from the variation of delay with temperature. Originally the system clock was adjusted whenever the temperature of the delay lines drifted out of synchronization, but later on in development the delay lines were put inside a temperature controlled oven.

The regeneration system implies that the maximum acceptable skew of the delay line from its nominal delay is $\pm 500\ \text{ns}$. This does not, however, take into account other factors such as the jitter and longer term drift of the system clock, as well the slew rates of the analogue circuitry. Whilst the demodulating pulse is lengthened to $1.9\ \mu\text{s}$, it is unlikely to be consistently at its peak voltage for this time, and so the output pulse is likely to have a better shape if the delay line produces an output in the middle of this period. Therefore creating a system which is an order of magnitude better than the above calculation, i.e. a maximum deviation from the nominal value of $50\ \text{ns}$, seems reasonable.

2.2.2 Electrical

Electrically speaking, EDSAC originally drove the delay lines with a nominal voltage of $25\ \text{V}$ peak to peak, through a $70\ \Omega$ terminated transmission line. The loss in the delay lines was $69\ \text{dB}$, leading to an output voltage of approximately $10\ \text{mV}$.

Despite this, the recreation project has discovered that the regeneration circuitry actually feed the delay lines with a decreasing voltage as the pulses propagate through the lines. The

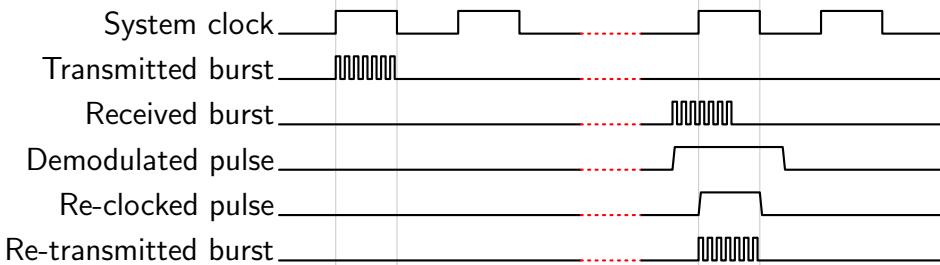


Figure 2.3: EDSAC pulse timing

voltage starts off at approximately 35 V for a signal fed to a delay line at the start of the store, but is reduced to approximately 25 V peak for the tubes at the end of the store.

In addition to this, problems were experienced with amplifying the low signal level output by the delay lines. Because the current wire delay line solution has flexibility in its output voltage, currently the delayed signal is output at 100 mV peak.

The $75\ \Omega$ transmission lines are alternating current (AC) coupled to the main EDSAC chassis, and the shield of the coaxial cable is referenced directly to earth. This poses a small problem for the reconstruction, because modern health and safety regulations dictate that the chassis of EDSAC should be earthed, but the electronics themselves are powered from an isolated 0 V rail with earth leakage protection in place.

This wouldn't be a problem except for the fact that the coaxial shield is earth referenced, meaning the coaxial return current goes to earth, not the isolated 0 V rail. In order to combat this, several capacitors have been added between ground and the 0 V rail, in order to allow the AC current to pass, but still provide direct current (DC) isolation. This means that care will be necessary when implementing the circuitry to power the delay line, as it cannot draw any DC power from the line.

2.2.3 Mechanical

The store delay lines originally consisted of banks of approximately 165.5 cm long steel tubes. The tubes are then held in an array using machined end-plates. An illustration of this is shown in Figure 2.4.

The exact dimensions, aside from the length, of each store tube are not explicitly stated in the available literature. However the dimensions of the short tubes are stated, with each short tube having an outer diameter of 4.44 cm, and an inner diameter of 2.86 cm [10, p. 213]. For the purposes of this project, the long tubes are therefore assumed to have similar dimensions to this.

The main restriction these dimensions place on the project is that the electronics must fit inside a tube of the correct length and outer diameter, so that the reconstructed system can be indistinguishable from the original in terms of form.



Figure 2.4: A battery of mercury delay lines in EDSAC [3]

Chapter 3

SPECIFICATION

The research of Chapter 2 has led to the derivation of a specification for the delay line which will be produced. This specification is detailed in Table 3.1.

Table 3.1: Delay line specification

Item	Specification	Justification
1	Must be capable of producing a delayed copy of the EDSAC pulse train presented to its input	This is the primary function of the device.
2	Must be powered from the input signal driven by EDSAC, with only minimal non-intrusive modifications made to EDSAC.	The goal of the device is to faithfully recreate the appearance and electrical interface of EDSAC, and thus large modifications such as power supply wires must be avoided.
3	Must be able to have a nominal delay of 1.15 ms, adjustable by at least $\pm 10\%$.	1.15 ms is the nominal delay of a long tube, as discussed in Section 2.2.1. An adjustable delay allows synchronisation with the system clock, with may vary.
4	Must have a maximum per burst deviation from the nominal delay of 50 ns.	This ensures that the delay line output will be able to synchronise with the clock of EDSAC. 50 ns is the maximum deviation derived in Section 2.2.1
5	Must be able to interface with input waveforms AC coupled bursts of 13.5 MHz carrier, with peak voltages in the range of 25 V to 35 V.	This is necessary to mimic the performance of the original delay line, 25 V to 35 V is the range derived in Section 2.2.2.
6	Must be able to have an adjustable nominal output voltage in the range of 10 mV to 100 mV (peak to peak), driving into $70\ \Omega$.	An adjustable output voltage in this range allows compatibility with both the original electrical interface, and that used by the reconstruction effort, 10 mV to 100 mV is the range derived in Section 2.2.2.
7	Must be encapsulated in a metal tube of 4.44 cm outer diameter, and 165.5 cm length.	This diameter allows the design to have the same appearance of the main memory store tubes of the original EDSAC design, the width and diameter are discussed in 2.2.3
8	Must be accompanied by a testing device capable of emulating the signals produced by EDSAC.	This allows the delay line to be tested separately to the reconstruction project.

Chapter 4

POWER SYSTEM DESIGN

This Chapter discusses how the delay line is powered, which is something which has required some consideration. As discussed in Specification point 2, the delay line must be powered from the input signal, and this method of providing power via this signal must require minimal modifications to the driving circuitry.

The field programmable gate array (FPGA) development board is powered using an LT3030 dual-output linear regulator [11]. This regulator produces both a 3.3 V, and a 1.2 V rail which powers the FPGA core voltage, and general purpose input/output (GPIO) banks respectively. Initial testing of the FPGA showed that the development board, without any support circuitry drew 50 mA from a 5 V power supply. The exact voltage of the power supply does not matter however, since the use of a linear power regulator means that approximately the same amount of power will be drawn at all voltages. This current larger than expected, as the Lattice power estimation tool estimated that the FPGA would draw approximately 15 mA for my design, and the remainder of the circuitry on the PCB only accounted for an extra 10 mA – 15 mA. Careful investigation of the evaluation board schematic revealed that both of the power supply rails were loaded with 100Ω resistors, accounting for 45 mA of current draw, as calculated by Equation 4.1. Inspection of the LT3030 datasheet shows that the load regulation is specified for a minimum current draw of 1 mA. The current draw of the resistors is therefore excessive, and the current draw of the remaining components on the PCB are sufficient to achieve satisfactory load regulation. The resistors were therefore removed and both the 3.3 V and 1.2 V rails were stable and at the correct voltage during normal operation.

The total current draw of the final design, including all of the analogue interface circuitry was measured at 28 mA.

$$\begin{aligned} I &= \frac{V_1}{R_1} + \frac{V_2}{R_2} \\ &= \frac{3.3}{100} + \frac{1.2}{100} \\ &= 45 \text{ mA} \end{aligned} \tag{4.1}$$

A naive solution to power the delay line would be to attempt to harvest the power of the input pulses themselves, and store power from each pulse to power the circuitry when the line is inactive. This is plausible because a 20 V peak-to-peak input signal, delivers 1.5 W of power into the delay line for the time it is active, as calculated by Equation 4.2.

$$P = \frac{V^2}{R} = \frac{10^2}{68} = 1.5 \text{ W} \tag{4.2}$$

The problem with this approach is that one cannot guarantee that pulses will be delivered to the delay line with any frequency. When the store is storing all zeros, no pulses will be delivered to the delay line at all. This means that the delay line would have to power on with a predictable latency such that the first pulse could be delayed by the correct amount. Unfortunately the unpredictable delay of the supply rails reaching the correct level, coupled with the hard to predict delay of the FPGA configuring from flash memory would break the timing budget.

Clearly a different solution to this problem is required, and this is what will be discussed in this Chapter. Section 4.1 will discuss the initially proposed solution which turned out to be flawed after further investigation, and Section 4.2 discussed a slightly more complex solution that overcomes the shortcomings of the former.

4.1 DC Offset Solution

Figure 4.1 shows the output portion of the store regeneration schematic. Inspection of the output port that drives the delay line, S2 in the schematic, it can be seen that it is AC coupled, by nature of capacitor C32. A simple solution to interface with EDSAC is to add a DC offset to the output. This could be achieved as simply as adding a potential divider to the output of the store regeneration unit, stiff enough to provide a stable voltage for the delay line. This solution is illustrated in Figure 4.2.

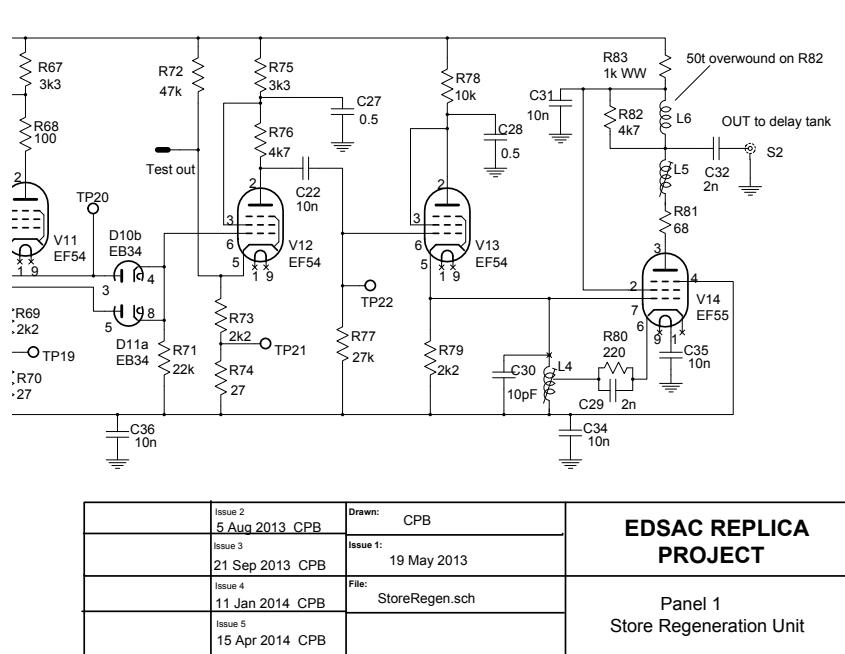


Figure 4.1: Store Regeneration Unit Output [4]

This is an ideal solution in theory, since the DC bias is trivial to add to the AC coupled output, and can easily be separated from the signal delay line. In addition, since all of the circuitry is powered using DC, the biased input could be used directly to power the low-dropout (LDO) regulator on the FPGA development board.

Unfortunately there is a problem with this solution. In order to meet modern health and safety requirements the DC supply the store regeneration unit, which drives the delay line, has a 0 V rail which is isolated from ground, and is protected by a residual-current circuit breaker with overcurrent protection (RCBO) [12]. This means that any current returned to earth, as opposed to the neutral of the supply will cause a fault, and the power supply will be turned off.

Despite this, the shields of the coaxial cables used for the delay lines are referenced directly to earth. In order to prevent earth leakage in this situation, since the termination of the pulse driving the return line would return a significant current to ground, substantial decoupling capacitors are added between earth and the isolated 0 V rail on each regeneration chassis. This

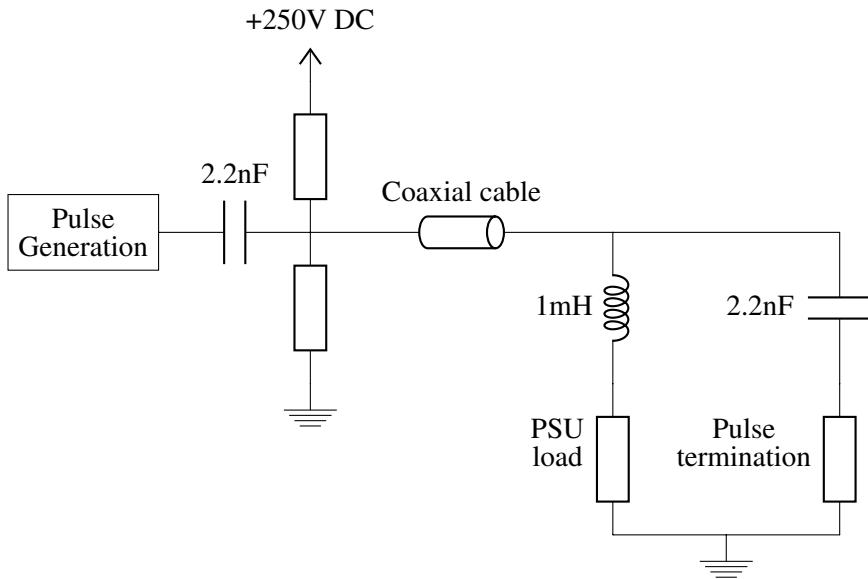


Figure 4.2: Proposed DC offset solution

allows the AC termination current to be returned correctly, but prevents any DC current from being drawn. This means that the DC offset solution isn't viable in reality.

4.2 Valve Power Supply Solution

Given that the DC offset solution proposed in the previous section wasn't feasible, an alternative solution was devised.

Inspection of the assembly instructions for the store regeneration chassis shows that the heater power supply is referenced directly to ground [13].

The heater power supply is a 6.3 V root mean square (RMS) 50 Hz AC supply. In the regeneration chassis it is used to power the heaters for each valve. A typical valve used in the design draws 1 A, so the 28 mA load from the delay line will add a negligible load onto this supply.

The supply is generated by mains transformer mounted on each chassis.

Since the supply is a 50 Hz supply, this is a frequency sufficiently far from the 13.5 MHz pulse carrier frequency, that it is easy to filter out.

In order to couple the heater supply onto the store regeneration output, a single inductor is sufficient. The addition of this inductor is the only modification necessary to the EDSAC chassis circuitry. A diagram showing the equivalent circuit of the modified circuitry is shown in Figure 4.3.

The reason that this circuitry works is that 1 mH inductor in series with the heater power supply is a very low impedance to the 50 Hz heater supply. This allows the heater supply to be passed to the delay line with minimal series impedance, whilst providing a very high impedance path to the memory pulses, preventing unnecessary loading of them, as calculated by Equations 4.3 and 4.4.

$$|Z_{L50\text{Hz}}| = 2\pi f L = 2 \times \pi \times 1 \text{ mH} \times 50 \text{ Hz} = 0.31 \Omega \quad (4.3)$$

$$|Z_{L13.5\text{MHz}}| = 2\pi f L = 2 \times \pi \times 1 \text{ mH} \times 13.5 \text{ MHz} = 85 \text{ k}\Omega \quad (4.4)$$

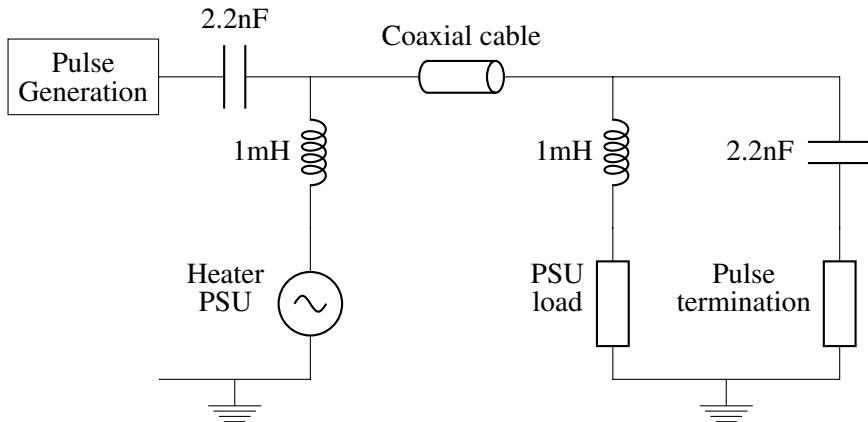


Figure 4.3: Proposed valve heater supply solution, equivalent circuit

In a similar manner the 2.2 nF capacitor that is used to couple the output of the pulse generation circuitry presents a low impedance to the pulses from the pulse generation circuitry, but is sufficiently small to prevent a high impedance to the 50 Hz heater supply signal, which prevents it from interfering with the pulse generation circuitry, as demonstrated by Equations 4.5 and 4.6.

$$|Z_{C50\text{Hz}}| = \frac{1}{2\pi f C} = \frac{1}{2 \times \pi \times 2.2\text{nF} \times 50\text{Hz}} = 1.4\text{M}\Omega \quad (4.5)$$

$$|Z_{C13.5\text{MHz}}| = \frac{1}{2\pi f C} = \frac{1}{2 \times \pi \times 2.2\text{nF} \times 13.5\text{MHz}} = 5.4\Omega \quad (4.6)$$

The same principle works for the termination and power supply circuitry inside the delay line, which will be discussed in more detail in Section [xx].

4.2.1 Circuit Design

Whilst Figure 4.3 models the delay line power supply as a resistor, the reality is slightly more complex since the AC supply needs to be rectified in order to power the DC circuitry. The circuit to achieve this is shown in Figure 4.4.

This circuit is a half wave rectifier. This design was chosen in preference to a full-wave rectifier because it removes the need for an isolating transformer inside of the delay line tube.

The combination of the diode, 10 Ω resistor, and 10 mF capacitor act as a traditional half wave rectifier, with the diode only allowing a low impedance path for the positive half of the sine wave, the capacitor storing the voltage to provide continuous power to the load, and the resistor limiting the peak current drawn by the circuit. The addition of the inductor presents a high impedance to the 13.5 MHz memory pulse, preventing unnecessary loading on the input signal, as discussed in Section 4.2.

The value of the capacitor was designed to keep the ripple of the supply to approximately 0.1 Ω, as illustrated by the linear approximation of Equation 4.7. 10 mF, is the closest readily available value to the calculated 12 mF. In reality the ripple will be much smaller than the calculated value, for two reasons:

1. The current is approximately half of the originally anticipated 60 mA as discussed at the start of this chapter.
2. Energy will also be stored in the magnetic field of the inductor, which is ignored in this approximation.

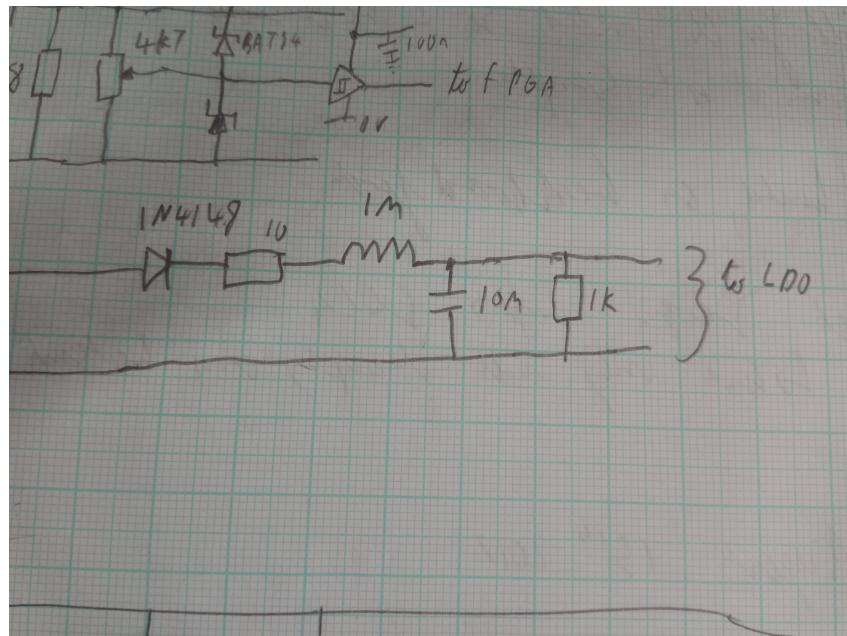


Figure 4.4: Delay line power supply schematic [replace picture]

$$\begin{aligned}
 I &= C \frac{dV}{dt} \\
 &\approx C \frac{\Delta V}{\Delta t} \\
 \therefore C &\approx \frac{I \Delta t}{\Delta V} \\
 &\approx \frac{60 \text{ mA} \times \frac{1}{50 \text{ Hz}}}{0.1 \text{ V}} \\
 &\approx 12 \text{ mF}
 \end{aligned} \tag{4.7}$$

The only additional component is the resistor. This is included in order to drain the voltage across the capacitor when the circuit is powered off, whilst only providing a small current drain of $I \approx \frac{6.3}{1\text{k}\Omega} \approx 6.3 \text{ mA}$ in normal operation.

Chapter 5

DELAY LINE DESIGN

This Chapter describes the development of the delay line itself, covering the architectural choices made, as well as the detailed design, development and testing. The overall block diagram of the delay line is shown in Figure 5.1.

The delay itself is implemented in the digital domain, with the goal of accurately emulating the analogue domain of the original design. This development is detailed in Section 5.1. The surrounding analogue circuitry translates between the logic level signal signals of the digital circuit, and the input and output.

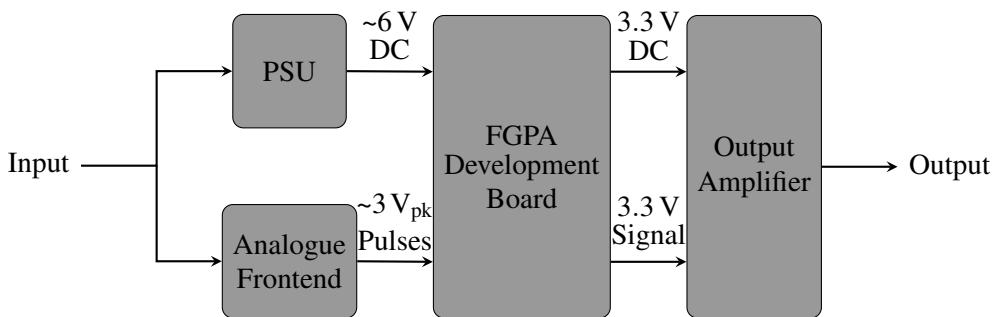


Figure 5.1: The overall architecture of the delay line

5.1 Digital Design

The goal of the digital design is to delay the signal on its input by a certain amount. If the input signal were arbitrary, then the optimal solution would be to clock a 1 b wide first in, first out (FIFO) buffer with a sampling clock. The delay would then be given by Equation 5.1, where t_d represents the delay time, f_s sampling clock frequency, and N the depth of the buffer.

$$t_d = \frac{1}{f_s} \times N \quad (5.1)$$

This solution can be simplified by consideration of the fact that the input signal is not arbitrary, the characteristics of the signal are known from the research detailed in Section 2.2.1. It is known that:

- The signal will consist of $0.9\text{ }\mu\text{s}$ pulses of 13.5 MHz tone.
- Each tone burst will be separated by $1.1\text{ }\mu\text{s}$.
- Each delay line can store a maximum of 576 pulses.

Using these characteristics it can be seen that a digital delay line only needs to sample store the time at which the rising edge of an incoming pulse is received. Since the packet length and modulation frequency is fixed, this can be asserted on the line a fixed delay later.

5.1.1 Architecture selection

Various architectures could be used to implement the system described in the previous section. The three most obvious methods of implementation being

- A microcontroller design.
- A discrete logic design
- A FPGA design

Microcontrollers have the advantage of being comparatively cheap and readily available, in addition they typically have more than enough RAM available to implement the memory to store the array of times at which pulses arrived.

The principle disadvantage is that microcontrollers inherently to their architecture process a single thread of data at once. This means that even a tight processing loop which samples the input would add a much larger amount of jitter to the input compared to a hardware solution with the same clock rate. Fortunately however modern microcontroller architectures, typically have a large number of peripherals embedded in the silicon, which can remove computation from the core. This combined with interrupts can provide an architecture with very predictable latency.

An example implementation is detailed in Figure 5.2. This details a system, based upon a typical microcontroller in the advanced RISC machine (ARM) Cortex-M0 family. A pin change interrupt interrupts the main execution thread, and branches to an interrupt handler. This interrupt handler reads the value of a free-running timer peripheral that counts up using the microcontroller master clock, adds a fixed delay value to it, and appends it to a queue of timer counts held in RAM. This queue contains the values of the counter which the output should be asserted at.

The main execution thread of the microcontroller configures the timer peripheral to trigger a second timer peripheral once its count equals the value on the top of the queue. This second counter is connected directly to an output GPIO pin, and is clocked such that the output toggles at 13.5 MHz.

This system can therefore meet the requirements of the delay line, with a worst case jitter of one system clock period (since the Cortex-M0 family allows for deterministic latency interrupts [cite]). Microcontrollers with timers capable of being configured as described above are readily available also [\[cite STM32 reference manual here\]](#).

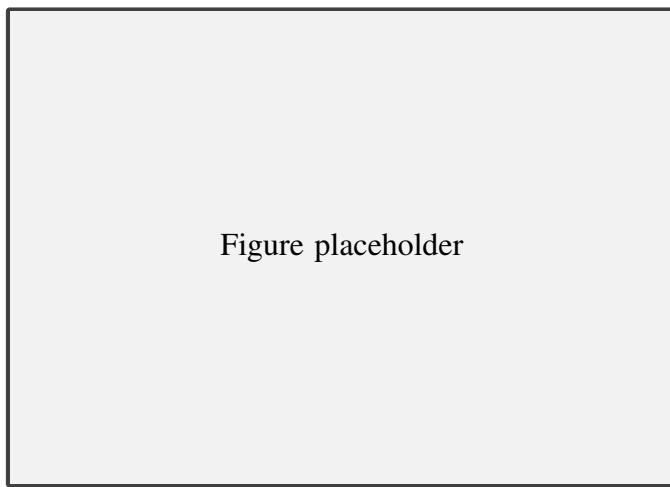


Figure 5.2: Proposed architecture for a microcontroller based system

The other two alternatives, a discrete logic system, and a FPGA based system would be similar in architecture, but differ in implementation, with the FPGA based system implementing the function using the programmable fabric of a FPGA, whereas the discrete implementation

would use individual integrated circuits (ICs). A block diagram of the proposed architecture is shown in Figure 5.3.

The concept of this architecture is similar to that of the microcontroller based system. There is a free running counter inside the FPGA, when a pulse is detected on the input, the value of the counter, added to a fixed delay is saved into a FIFO. A second hardware module outputs a pulse train whenever the value of the counter matches the value on the top of the FIFO.

The difference between the hardware implementation and the microcontroller implementation is that there isn't a processor core to set up the hardware blocks, instead each hardware block is designed to perform the correct function, and interacts with the other modules using logic signals. In addition there is no need for an external modulator, as a hardware block can be created to output the modulated 13.5 MHz signal.

The FIFO is the centre of this system. It is set to be the width of the free-running counter. At its input is the current value of the counter, added to a fixed constant that represents the required delay. This value is saved into the FIFO when the rising edge detector produces a pulse on its output.

The rising edge detector is a fairly simple hardware block that outputs a pulse for a single clock cycle when it detects a rising edge on its input. It then times out for a fixed interval, to avoid triggering on the remaining pulses in the same packet, and resets.

At the output of the FIFO feeds into a comparator. This comparator compares the value on the top of the FIFO with the current value of the counter, and triggers the pulse generator when the two values are equal.

The pulse generator outputs a fixed length burst of 13.5 MHz tone whenever it is triggered. This could be implemented simply by choosing a clock frequency that is 2^n times greater than 13.5 MHz. Thus the carrier frequency can be generated by a counter clocked from the input clock.

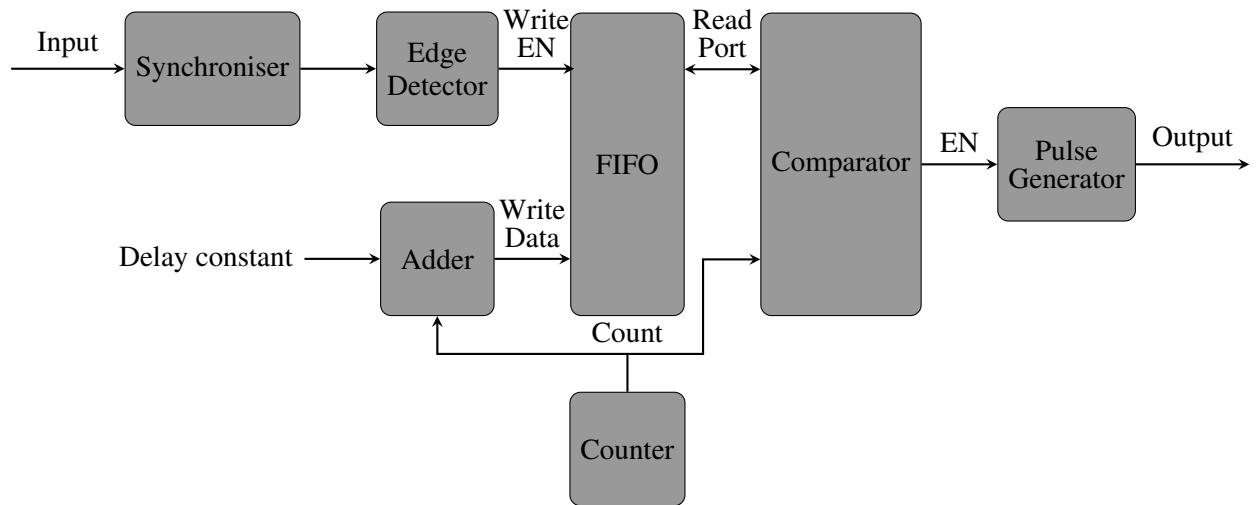


Figure 5.3: Proposed architecture for a FPGA or discrete system

[Could add power in here]

The advantage of the microcontroller system over a FPGA or discrete solution is twofold. Firstly the simplicity of the physical circuit necessary. A microcontroller based design would require a IC for the microcontroller itself, and an external crystal oscillator (although some microcontrollers do have a reduced precision internal resistor-capacitor (RC) oscillator, removing the need even for this), and are generally powered from a single 3.3 V supply. This is in contrast

to FPGAs which typically require more than one supply rail, with extensive decoupling, and a discrete solution which would require many ICs.

Secondly a microcontroller design would likely be slightly cheaper than a discrete logic design, as low end microcontrollers are typically only slightly more expensive than individual logic ICs, and are much less costly than typical FPGAs.

Despite these advantages, a hardware solution does have the advantage of elegance. While it has been demonstrated above that a microcontroller solution could achieve single cycle jitter – the same as a hardware solution. It is a lot more difficult to implement and verify, given the fact that configuration of many hardware peripherals are required.

In addition, the cost advantage of the microcontroller solution may be smaller than anticipated. This is because the simplicity of the hardware solution would only require a very small FPGA, which may be competitive in price to a microcontroller. Alternatively a complex programmable logic device (CPLD) with an external RAM IC could be used. A FPGA solution would be preferred to a discrete implementation, due to the ease of testing, and reconfiguring the hardware if the requirements change.

Therefore on balance it has been decided to proceed with a FPGA based solution.

FPGA Selection

The primary requirements for the FPGA are as described in Table 5.1.

Table 5.1: FPGA Requirements

Number	Requirement	Justification
1	Must have a moderately low cost	Using the IC for every delay line in the store must not be cost prohibitive.
2	Must have enough logic elements and block RAM to implement a single long delay line	This is the proposed function of the delay line
3	Must have fabric capable of being clocked fast enough to implement the design with a clock rate of [xx]	It was decided at [xx] that this is the required sampling speed.
4	Should have a development board available with a width less than [xx]	As described in Section [xx] , [xx] is the internal diameter of the delay line tube, it would be ideal if the development board could fit inside the tube, to save a custom PCB being designed.
5	Should have a phase locked loop (PLL) to enable to fabric clock to be generated from a crystal oscillator	An internal PLL is ideal, but an external PLL could also be used.

At the time of writing the least expensive FPGA available from component suppliers Farnell, is the a 1280 logic cell variant of the iCE40 FPGA family, produced by Lattice [14]. This is £5.10 in single quantity, which is low cost enough that it could reasonably be used to replace all of the delay lines, thus meeting specification point 1.

This FPGA has 1280 logic cells (LCs), which should be plenty to implement the simple design, given that each LC in this architecture consists of a four input look-up table (LUT), and

a flip-flop [15, p.2-2]. The FPGA also has 64 kb of block RAM. This meets specification point 2, when one considers that is enough to store a clock sample for each of the 576 possible pulses, even if a 64 b clock width was used, as demonstrated by Equation 5.2, where S represents the size of memory required. In addition to this, there is one PLL available on the FPGA die, meeting specification point 5.

$$S = 64 \text{ b} \times 576 = 36 \text{ kb} \quad (5.2)$$

It is hard to estimate how fast a design will be able to operate in a FPGA without synthesising it, and running timing analysis. However, despite this we can estimate that the FPGA will easily meet timing at [xx], thus meeting specification point 3, given that the register-to-register performance of the fabric is as good as 403 MHz for a dual-port RAM, 305 MHz for a 16:1 multiplexer, and 105 MHz for a 64 b counter.

In addition to this, a development board is available which is very narrow [11]. The exact dimensions are not provided, but it is barely wider than the 22 mm thin quad flat package (TQFP) of the FPGA itself [11, p.2], meaning it is highly likely to fit in the [xx] diameter tube, thus meeting specification point 4.

Based upon the fact that it meets all of the specification points, it has been decided to implement the design using the Lattice iCEstick evaluation board, with the possibility of moving to a custom PCB if many instances of the design were required.

5.1.2 HDL Design

Uniquely, the Lattice iCE40 family of FPGAs has an open-source toolchain available, named Project IceStorm which can be used as an alternative to the toolchain provided by Lattice [16]. Project IceStorm synthesises Verilog natively, and Lattice's iCEcube2 toolchain synthesises both Verilog and VHSIC hardware definition language (VHDL) [17, p.10]. Therefore in order to maintain compatibility with both toolchains, the design will be written using Verilog. Both toolchains were trialled, and the codebase is compatible with both, however Project IceStorm was used in the end as it is the more user-friendly toolchain.

The design was implemented using the structure of Figure 5.3, with the rising edge detector, FIFO, comparator, and pulse generator implemented as individual Verilog modules.

[Could talk about how the parameters are determined here]

Rising Edge Detector

The rising edge detector is implemented as the state machine of Figure 5.4. In the wait state, the input is sampled on every clock edge, when it is true, the state machine transitions to the assert state, where the output is asserted, until the state machine unconditionally branches to the timeout state. In the timeout state a counter is incremented until it reaches the required limit. At this point the state machine branches back to the wait state.

FIFO

The FIFO is implemented by building logic constructs around a dual port RAM to handle addressing of the read and write ports.

The main part of the FIFO is the read and write addresses. These addresses act as pointers to the current location of the read/write word in RAM. Each address is incremented on when the FIFO is read from/written to, and the address loops around to the start when it reaches the

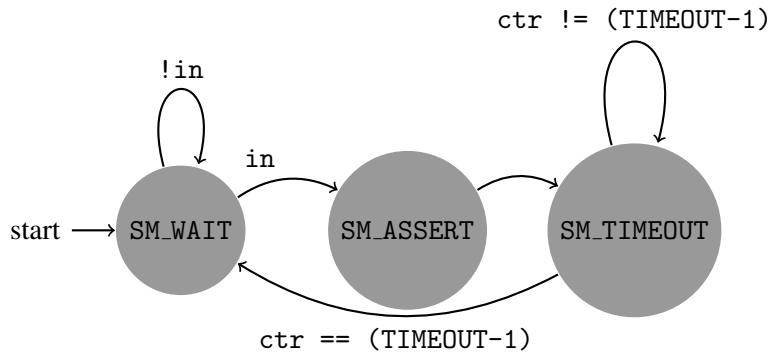


Figure 5.4: Rising Edge Detector State Machine

end of the RAM buffer. This ‘round robin’ approach means that the FIFO can be read from and written to an unlimited amount of times, so long as the total number of words stored is not greater than the depth of the buffer.

In addition to the read/write address pointers, there is a counter which keeps track of the total number of words stored in the buffer. This counter is decremented if a read is requested, and incremented if a write is requested (its value does not change if both a read and a write is requested at the same time, or neither a read or write is requested). This counter is used to generate empty and full signals so that the surrounding logic knows the state of the counter.

Comparator

The comparator is implemented as a state machine that requests data from the FIFO and asserts the output when the counter matches the data word, as illustrated by the state transition diagram of Figure 5.5.

`empty` is the empty signal from the FIFO, and the FIFO read request signal is true when the state machine is in the request state.

`count` is the value of the system counter, and `data_in` is the data read from the FIFO. The output trigger is true when the state machine is in the `SM_ASSERT` state.

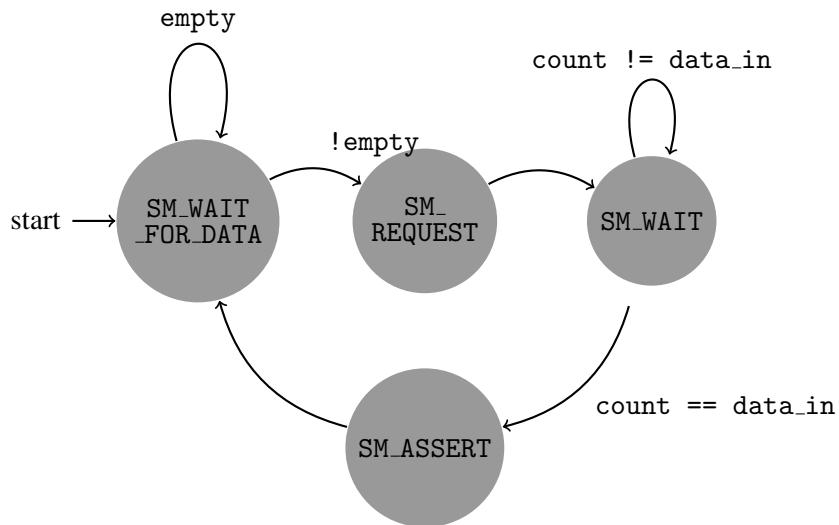


Figure 5.5: Comparator State Machine

5.2 Analogue Design

The FPGA communicates using 3.3 V low voltage complementary metal oxide semiconductor (LVCMOS) inputs and outputs. This is a great difference from the 25 V to 35 V input, and 10 mV to 100 mV output range. In addition to this, the input and outputs are AC coupled and so require a voltage symmetrical about 0 V.

5.2.1 Input

The input voltage needs to be terminated with $68\ \Omega$, the closest E12 preferred resistor value to $70\ \Omega$ and the value which is used by the remainder of the reconstruction project to terminate the transmission lines, in addition to this the power system imposes a requirement to remove the 50 Hz power signal discussed in Chapter 4.

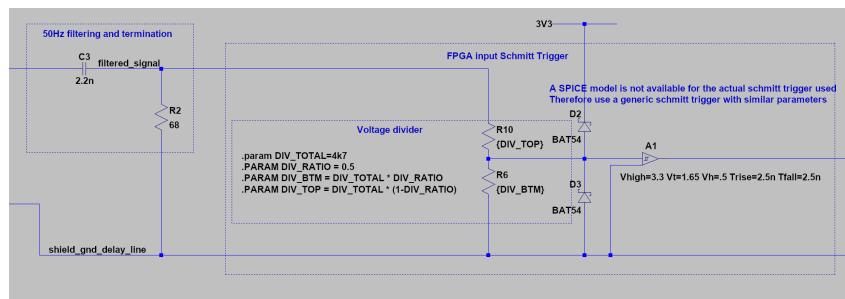


Figure 5.6: Delay line input schematic [replace with better schematic]

The schematic designed to interface the input signal to the FPGA is shown in Figure 5.6.

The 2.2 nF capacitor and $68\ \Omega$ resistor act as a low pass filter with a break frequency of 1.1 MHz, as derived in Equation 5.3. This correctly terminate the incoming 13.5 MHz bursts, as well as presenting a high impedance to the 50 Hz power supply signal.

$$f_0 = \frac{1}{2\pi RC} = \frac{1}{2\pi \times 68\ \Omega \times 2.2\text{nF}} = 1.1\text{MHz} \quad (5.3)$$

The next stage of the input circuitry is a 4.7 k Ω potentiometer. This attenuates the input signal by an adjustable amount to provide a logic level input signal for the Schmitt trigger buffer.

The value of the potentiometer is designed such that it provides negligible loading on the input signal compared to the $68\ \Omega$ termination resistor. Despite this the current through it for a 10 V input signal is $\frac{10\text{V}}{4.7\text{k}\Omega} = 2.1\text{mA}$, an order of magnitude greater than both the BAT54 diode leakage current (2 μA max [18, p.2]), and the Schmitt buffer input leakage current (5 μA max [19, p.4]).

The Schmitt trigger removes small glitches which may occur on the input so that they do not reach the FPGA input. It also helps to provide a defined LVCMOS output signal from the unclean input signal.

The BAT54 protection diodes clamp the input signal to the Schmitt buffer close to its supply rails. They are suitable for the task due to their fast recovery time, 5 ns and low forward voltage, 0.4 V at 10 mA [18, p.2]. In addition their continuous forward current rating, 200 mA max is far greater than what can be sourced through the 4.7 k Ω potentiometer.

5.2.2 Output

The output section of the delay line is required to drive the line with 0 V when no pulse is present, and a 10 mV to 100 mV peak-to-peak signal, centered about 0 V when no signal is present.

The biasing about 0 V is achieved by AC coupling the output of the amplification circuit. One problem which can occur from this is that the output of the FPGA will be 0 V when no signal is being output, but will oscillate between 0 V and 3.3 V when a pulse is being transmitted. This is a problem for the circuit because to correctly AC couple the signal, the voltage at the capacitor should be half way between the two extreme values, i.e. 1.65 V. This is illustrated in Figure 5.8.

In order to achieve this effect, the internal tri-state buffer capability of the FPGA is used. This is the purpose of the resistor network shown on the output of the FPGA. When the FPGA output is disabled, the two resistors between power rails pull the op-amp non-inverting input to $3.3\text{ V} \times \frac{1\text{k}\Omega}{2\text{k}\Omega} = 1.65\text{ V}$. When the FPGA drives its output to 3.3 V, the voltage at the op-amp is equal to $3.3\text{ V} \times \frac{1\text{k}\Omega}{1.5\text{k}\Omega} = 2.2\text{ V}$, and when the FPGA drives 0 V, the voltage at the op-amp will be $3.3\text{ V} \times \frac{0.5\text{k}\Omega}{1.5\text{k}\Omega} = 1.1\text{ V}$. This arrangement is used, rather than having the FPGA directly drive the non-inverting input of the op-amp in order to avoid driving the op-amp into saturation, as it is powered from the same 0 V to 3.3 V rails as the FPGA GPIO pins.

The first stage op amp is a unity gain buffer amplifier used to drive the output into a 220 Ω potentiometer. This potentiometer is used to attenuate the signal to the required level from the 1 V peak to peak amplitude driven by the op-amp to the required level. This also attenuates the bias point away from half way between the voltage rails. For this reason, the AC component of the signal is re-biased onto 1.65 V before buffering by the second amplifier.

The second amplifier is a second unity gain buffer, that critically has an output impedance of 68 Ω and AC couples the output signal in order to remove the DC bias.

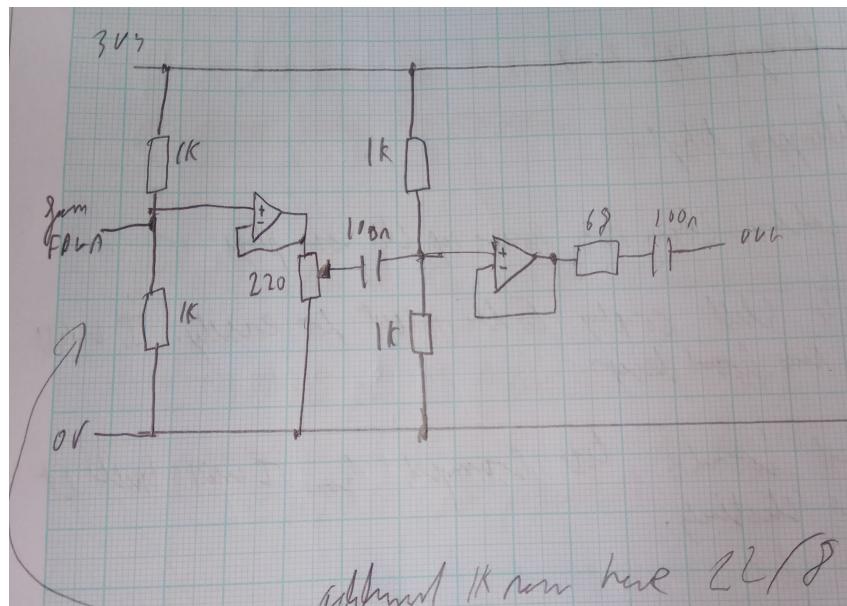


Figure 5.7: Delay line output schematic [replace with better schematic]

For both amplifiers in the circuit LMH6639 amplifiers are used [20]. This amplifier is chosen because of several desirable properties. Firstly it has a gain-bandwidth product (GBP) of 190 MHz, more than what is necessary to buffer the 13.5 MHz signal with unity gain. In addition to this, unlike many high GBP op-amps, the LMH6639 is a voltage feedback architecture

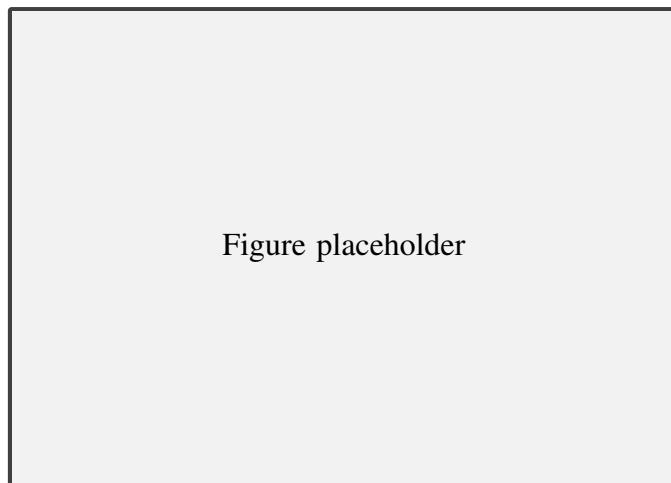


Figure 5.8: Effect of having the op-amp output idle at 0 V

simplifying design. Crucially for this application the amplifier is stable at a gain of +1, and has almost rail-to-rail performance at a single supply voltage of 3 V (typical output range is 75 mV to 2.93 V driving $150\ \Omega$.

Chapter 6

TEST HARNESS DESIGN

6.1 Digital Design

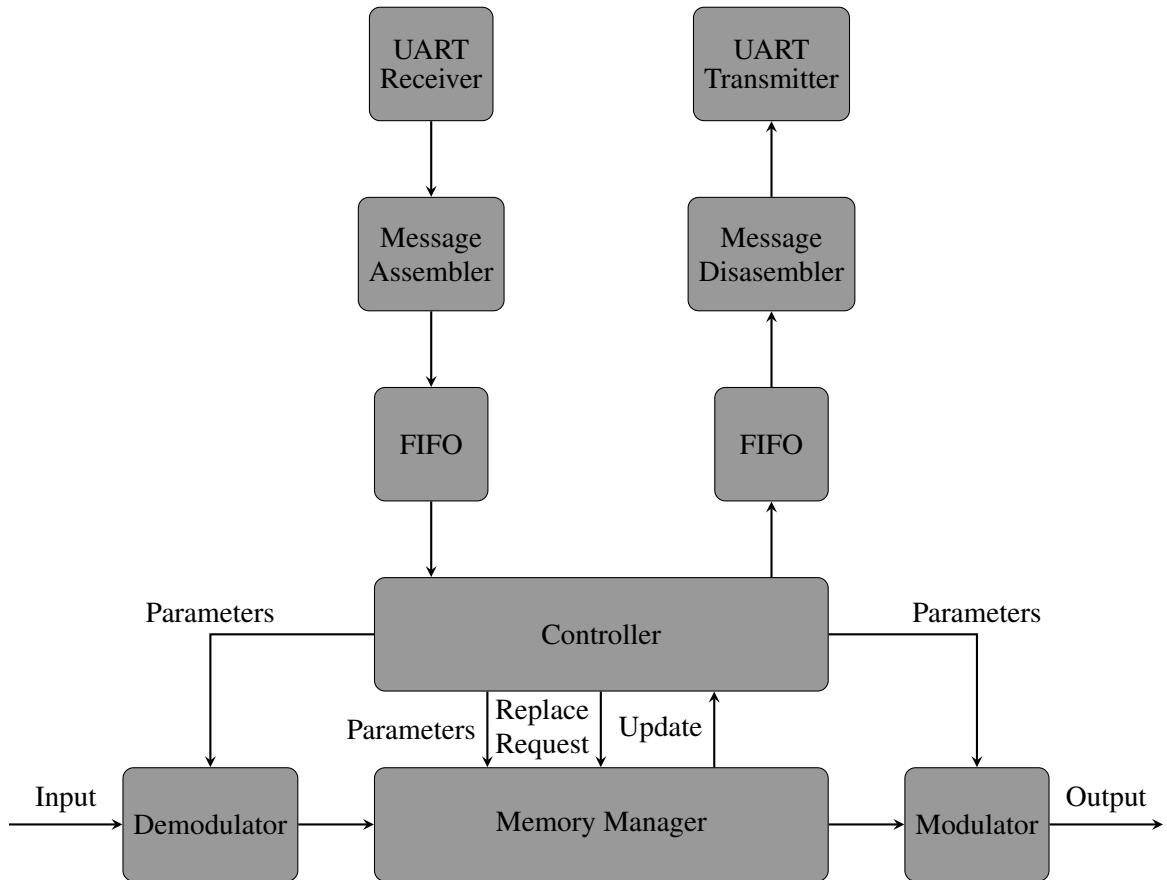


Figure 6.1: Proposed Architecture for Test Harness

6.2 Analogue Design

6.2.1 Input

6.2.2 Output

6.3 Software Design

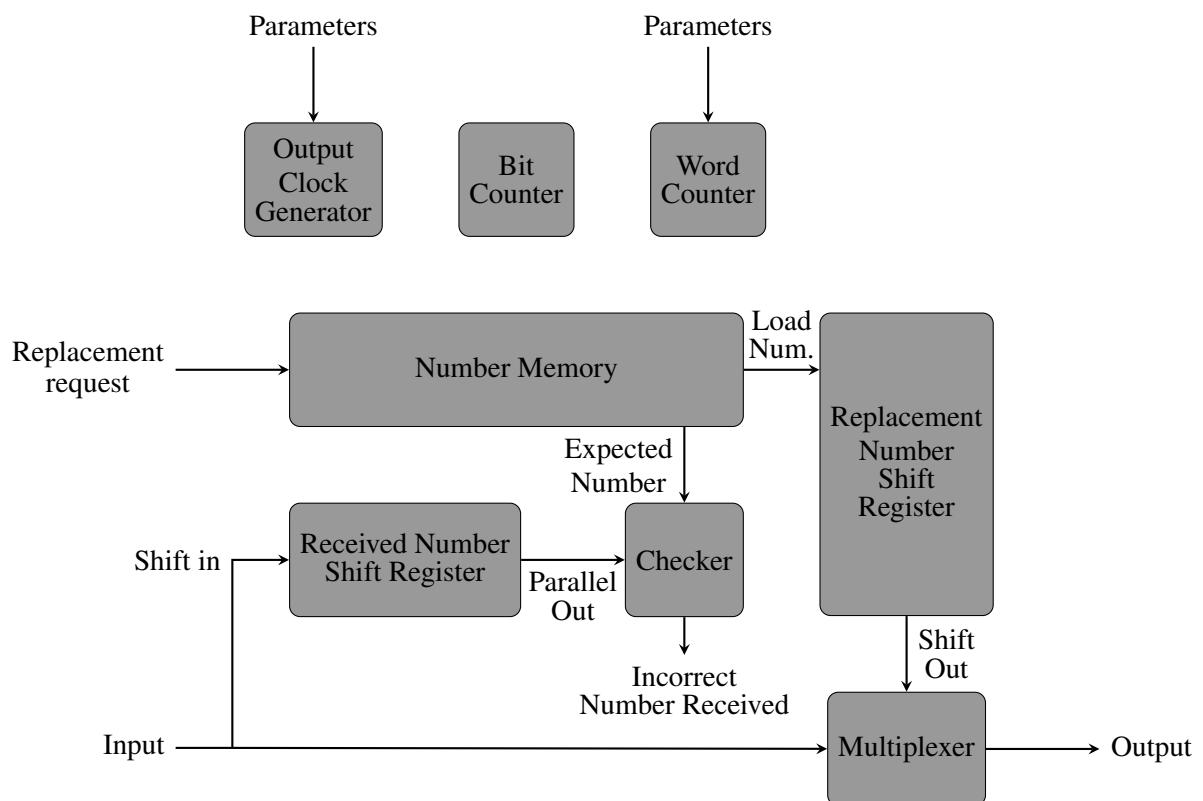


Figure 6.2: Proposed Architecture for Memory Manager

Chapter 7

SIMULATION AND VERIFICATION

7.1 SPICE Simulation

As well as practically building and testing the circuits, it was desirable to simulate the performance of the circuits using LTspice [21]. There were two main reasons for this. Firstly, building the circuits and testing the circuits practically takes a large amount of time, and involves expense of purchasing components. Therefore it was desirable to limit the number of iterations of this process by first testing performance in simulation.

In addition to this, interfacing with the EDSAC circuitry is difficult since it requires travel to Bletchley, and consideration of the practicalities of interfacing with high voltage valve circuitry.

This chapter presents the results of the various simulations which were run. Section 7.1.1 discusses the modelling of the valve circuitry used in EDSAC and the remaining sections in the chapter discuss various simulations which took place.

7.1.1 Valve Models

In order to model the store regeneration circuitry, it was necessary to have models for the valves used in the design. Fortunately, work has been done previously to create models for the purposes of simulating EDSAC circuitry [22]. This previous work has created models for the valves based both on the datasheet characteristics, as well as practical testing with the valves used in EDSAC. The models produced had been tested with simulations of some parts of the store regeneration circuitry and was found to match practical measurements well.

An existing workflow was provided to produce simulation program with integrated circuit emphasis (SPICE) netlists which use the models from the DesignSpark [23] schematics which the reconstruction project uses. Instead of using this workflow however it was decided to create an LTspice symbol for each valve model and reconstruct the relevant parts of each schematic using LTspice. This decision was made in order to allow better integration with the other simulation models, and to allow for faster iteration of the design.

7.1.2 Delay Line Input Simulation

7.1.3 Delay Line Output Simulation

7.1.4 Store Regeneration Input Simulation

7.2 HDL Verification

7.2.1 Delay Line Verification

In order to verify the hardware description language (HDL) design, a SystemVerilog testbench was written for each module, and these modules were then simulated using Mentor Graphics ModelSim. These testbenches provide stimulus to verify that each block works as anticipated before integration with the system as a whole.

The focus of the verification is, however, on the system level testing, which tests the delay line as a whole. This testbench generates pulse trains representing random numbers in the same format as EDSAC. The number is then transmitted as a series of modulated pulses in the same way as EDSAC, and each pulse burst is added to a queue with the simulation time when the pulse is expected to arrive. The testbench then checks the queue on each time step to verify that the system produces output bursts of the correct frequency at the correct time. Figure 7.1 shows the output of this testbench.

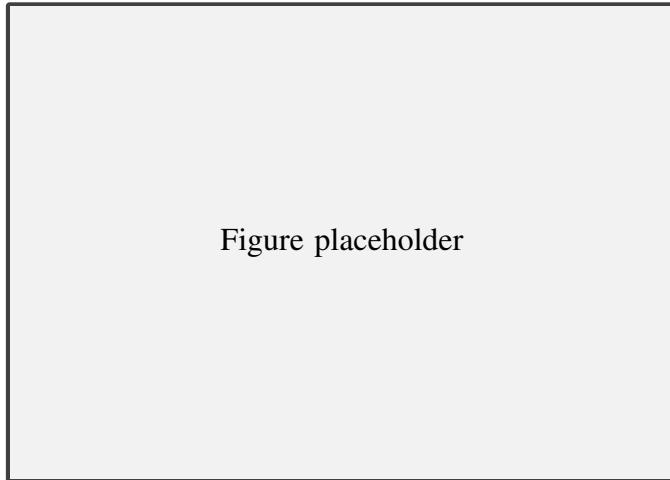


Figure 7.1: Delay line testbench output

7.2.2 Test Harness Verification

Chapter 8

SYSTEM INTEGRATION

8.1 Mechanical

8.2 System Level Testing

8.2.1 Issues

FPGA Reset Issue

Signal Breakthrough

8.3 Measurements

Chapter 9

PROJECT PLANNING

9.1 Time Management

9.2 Risk Management

9.3 Financial Management

[I'm not sure that I actually need this section – it's not mentioned in the mark scheme]

Chapter 10

FINAL COMMENTS

Bibliography

- [1] University of Cambridge. (2011, feb) EDSAC I, W.Renwick, M.Wilkes. Copyright Computer Laboratory, University of Cambridge. Reproduced by permission. Cropped from original photo. [Online]. Available: http://www.cl.cam.ac.uk/relics/jpegs/edsac_wilkes.jpg
- [2] J. Tyler, “Mercury delay line emultion: Reconstructing memory for edsac,” may 2017, this document is an unpublished research review written for the project.
- [3] University of Cambridge. (2011, feb) M.V.Wilkes, mercury delay line (2). Copyright Computer Laboratory, University of Cambridge. Reproduced by permission. [Online]. Available: http://www.cl.cam.ac.uk/relics/jpegs/delay_lines.jpg
- [4] C. Burton, “Panel 1 store regeneration unit schematic,” apr 2014.
- [5] M. Ward. (2011, jan) Pioneering edsac computer to be built at bletchley park. BBC News. [Online]. Available: <http://www.bbc.co.uk/news/technology-12181153>
- [6] University of Cambridge, “Pioneer computer to be rebuilt,” *CAM*, vol. 62, p. 5, mar 2011.
- [7] F. da Cruz. (2013, nov) Programming the ENIAC. Columbia University. [Online]. Available: <http://www.columbia.edu/cu/computinghistory/eniac.html>
- [8] K. S. Jones. (2001, dec) A brief informal history of the computer laboratory. University of Cambridge. [Online]. Available: <http://www.cl.cam.ac.uk/events/EDSAC99/history.html>
- [9] J. Bonne. (2007, march) Transistor transition began in Allentown. *The Morning Call*. [Online]. Available: http://articles.mcall.com/2007-03-04/features/3712894_1_transistor-production-line-bell-labs-allentown-plant
- [10] M. V. Wilkes and W. Renwick, “An ultrasonic memory unit for the EDSAC,” *Electronic Engineering*, vol. 20, pp. 208–213, jul 1948.
- [11] *iCEstick Evaluation Kit*, 1st ed., Lattice, aug 2013.
- [12] A. Passmore, “HN57: Power supply operation and maintenance,” jan 2015.
- [13] C. Burton, “HN29: Chassis 01 assembly,” nov 2014.
- [14] Farnell. (2017, aug) ICE40HX1K-TQ144 -FPGA, iCE40, PLL, 95 I/O’s, 133 MHz, 1.14 V to 1.26 V, TQFP-144. [Online]. Available: <http://uk.farnell.com/lattice-semiconductor/ice40hx1k-tq144/fpga-1280-luts-1-2v-hx-144tqfp/dp/2362849>
- [15] *iCE40™LP/HX Family Data Sheet*, 3rd ed., Lattice, mar 2017.
- [16] C. Wolf and M. Lasser, “Project icestorm,” <http://www.clifford.at/icestorm/>.
- [17] *iCEcube2 User Guide*, Lattice, feb 2017.
- [18] Vishay, *Small Signal Schottky Diodes, Single and Dual*, 1st ed., feb 2013.
- [19] Diodes Incorporated, *Single Schmitt-Trigger Buffer*, 4th ed., mar 2014.

- [20] Texas Instruments, *LMH6639 190MHz Rail-to-Rail Output Amplifier with Disable*, mar 2013.
- [21] Linear Technology, “Ltpice.” [Online]. Available: <http://www.linear.com/designtools/software/#LTspice>
- [22] P. Linnington, “HN68: Spiced valves,” dec 2015.
- [23] RS Components, “Designspark pcb.” [Online]. Available: <https://www.rs-online.com/designspark/our-software>

Appendix A

Folder Structure

The project is structured as a single repository using the git version control system. For those unfamiliar with git, the project folder may be treated as an ordinary computer folder, however there exists inside the root of the directory a hidden folder, `.git`, which contains the entire history of the project. This history is not of use to a user only interested in the latest state of the project, but is useful to find files or parts of files which may have been deleted. An example of such files are the project files for the Lattice and Xilinx toolchains, which were later replaced in the project by the open-source Project IceStorm toolchain.

The philosophy of the project repository is that it should not contain any:

- Files not directly related to the project itself (e.g. reference material)
- Auxillary files automatically generated by tools
- Binary files that may be easily regenerated by tools

but should contain:

- Design files
- Project files/configuration files used by tools
- Documentation files, even if they may be re-generated (e.g. `LATEX`generated `.pdf` files), provided they are not excessively bulky.

This philosophy is enforced inside the repository using `.gitignore` files.

The structure of each sub-folder in the project repository is shown in the diagram below (in alphabetical order).

<code>admin</code>	Administration files. E.g. purchase orders
<code>demo</code>	<code>LATEX</code> files used to generate the project demonstration slides
└ <code>figs</code>	Figures used in the demonstration slides
<code>hdl</code>	HDL design files/project files, and simulation testbenches
└ <code>common</code>	Common Verilog constructs used across multiple designs
└ <code>sim</code>	SystemVerilog Simulation testbenches
└ <code>sim_modelsim</code>	ModelSim project file for simulation
└ <code>src</code>	Synthesisable Verilog code
<code>delay_line</code>	Files related to the delay line design
└ <code>icestorm</code>	Makefile for the Project IceStorm toolchain
└ <code>sim</code>		
└ <code>sim_modelsim</code>		
└ <code>src</code>		
<code>delay_line</code>	Files related to the delay line design
└ <code>icestorm</code>		
└ <code>sim</code>		
└ <code>sim_modelsim</code>		
└ <code>src</code>		
<code>report</code>	<code>LATEX</code> files used to generate the project report (this document)
└ <code>docs</code>	<code>LATEX</code> source files that are not the master project file
└ <code>figs</code>		
└ <code>refs</code>	Files for <code>BIBTeX</code> referencing
<code>software</code>	Supporting software intended to run on a PC
└ <code>mem_gui</code>	Source related to the GUI software that talks to the test harness

```
└── .idea ..... Project files for CLion
    ├── ftdi_wrapper ..... Source for wrapping the FTDI driver
    ├── generic_classes ..... Source for classes used in various parts of the project
    ├── main_gui ..... Source to implement GUI itself
    ├── memory_manager ..... Source to monitor each unit of EDSAC memory
    ├── status_manager ..... Source to control the state of the system
    ├── uart_manager ..... Source to control the transmission and receipt of messages
    └── uart_msg ..... Source relating to UART message formats
        └── uart_msg_const_gen .... A script to generate header files containing UART message
            constants
            ├── cpp_out ..... Output folder for C++ header(s)
            └── verilog_out ..... Output folder for Verilog header(s)
    └── spice ..... LTspice simulation schematics
        └── component_models ..... Models for various components used in the design
```

Appendix B

HDL Code Overview

[Is this section really necessary]

Appendix C

Schematics