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
September 7, 2017

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

## Abstract

EDSAC was the first practical digital stored program computer, and ran its first program in May 1949. It contributed much to the field of computing, and due to its notability is currently being reconstructed at The National Museum of Computing.

EDSAC used delay line memory that was able to store numbers by inserting pulses into a delay medium and feeding the output into the input. In the case of EDSAC the delay medium consisted of very precisely machined steel tubes full of mercury. This is very difficult to reproduce authentically due to constraints of budget and health and safety. The intention of this project has therefore been to use modern technology to emulate this memory in a form and electrical interface indistinguishable from the original.

The project has been successful in creating a digital emulation of a mercury delay line, implemented in hardware using a low cost FPGA. This delay line, coupled with analogue interface circuitry is able to interface with the high voltage used to drive the mercury delay lines. The delay line is also able to phantom power itself from the signal input with the addition of a single passive component to the remainder of the circuitry of EDSAC.

In addition to the delay line itself a comprehensive test harness has been designed that is able to emulate the signals produced by EDSAC and evaluate the performance of the delay line.

The delay line has been integrated in a physical enclosure matching that of the original, and had its performance verified both using the test harness produced by this project as well as integration with the valve circuity of the main reconstruction effort.

The project produces a well documented and tested methodology to implement delay line memory, that performs well enough to be a viable solution to perform the task of implementing the memory of the reconstructed computer. Future work will formalise the construction of the delay line, and work to integrate it as part of the museum display.

# Contents

# 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.

**FTDI** Future Technology Devices International.

**GBP** gain-bandwidth product.

**GPIO** general purpose input/output.

**GUI** graphical user interface.

**HDL** hardware description language.

**HF** high frequency.

**HV** high voltage.

**IC** integrated circuit.

**LC** logic cell.

**LDO** low-dropout.

**LUT** look-up table.

**LVCMOS** low voltage complementary metal oxide semiconductor.

**PC** personal computer.

**PCB** printed circuit board.

**PERT** project evaluation and review technique.

**PLD** programmable logic device.

**PLL** phase locked loop.

**RAM** random access memory.

**RC** resistor-capacitor.

**RCBO** residual-current circuit breaker with overcurrent protection.

**RMS** root mean square.

**SPI** serial peripheral interface.

**SPICE** simulation program with integrated circuit emphasis.

**TNMoC** The National Museum of Computing.

**TQFP** thin quad flat package.

**UART** universal asynchronous receiver/transmitter.

**USB** universal serial bus.

**VHDL** VHSIC hardware definition language.

# List of Figures

# List of Tables

# Chapter 1

# INTRODUCTION

The National Museum of Computing (TNMoC) 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 it should be noted that the recreation of its memory is non trivial, and designing a system to perform this task is the focus of this project.

The reconstruction effort has made good progress towards recreating the machine, and is already developing a separate solution to the problem. The history of EDSAC, along with the technical details of how the memory it used functions is summarised in Chapter 2. The final section in this chapter uses the technical background to describe the exact aims of this project, and compare the goals to the memory currently being developed by the reconstruction effort. Utilising the technical details of the review, Chapter 3 derives a specification to establish the required performance of the memory solution in explicit detail.

Working from the specification derived in Chapter 3, Chapter 4 contains the design details of how the solution is powered, whilst Chapter 5 details the design decisions taken in the creation of the memory mechanism itself.

Given that TNMoC is located in Bletchley, and the project is created in Southampton, it has been necessary to produce a method of testing the memory solution which does not rely upon any of the hardware produced by the reconstruction project. For this reason, Chapter 6 details the design of a test harness which is capable of testing the memory by emulating the signals produced by EDSAC.

The memory solution interfaces with valve circuitry that comprises the remainder of EDSAC. This circuitry can be sensitive, and so to ensure that this project would not have any unintended parasitic effects on EDSAC simulation program with integrated circuit emphasis (SPICE) simulations are used to model the impact this project would have on the circuitry of EDSAC. This is detailed in Chapter 7, alongside details of how the digital circuitry of the memory solution created is verified.

Following the design and implementation stages, all of the components of the memory module and test harness have been integrated and tested. This, along with details of problems encountered and how they were rectified are detailed in Chapter 8.

The scope encompassed by this project has demanded careful time planning, and analysis of any risks that could derail the goals of the project. This is detailed in Chapter 9. Final comments comparing the aims of the project to the outcomes, and detailing scope for future work are then presented in Chapter 10.

# Chapter 2

# TECHNOLOGY REVIEW

This chapter presents an overview of EDSAC, and an overview of the relevant literature surrounding its the 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. This reason for this is that despite the notability of EDSAC its architecture was quickly adapted to develop other machines, so aside from the papers written by Wilkes himself, there is little technical documentation from other sources of the machine as it was originally constructed.

## 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].

The goal of the reconstruction project at TNMoC is to reconstruct the computer as faithfully as possible to its state when it ran its first ever program in May 1949 [9].

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

## 2.2 EDSAC Memory Architecture

The completion of EDSAC in 1949 posed a significant design challenge for the memory of the machine. Transistors were not commercially available at all until 1951 [10], 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 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 [11, 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 magneto-restrictive 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) micro-strip trace.

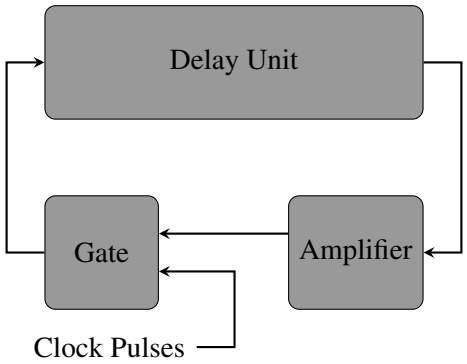



Figure 2.2: Demonstration of the principle of delay line memory, adapted from [2]

EDSAC used acoustic delay lines in the form of steel tubes 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 proportional to the length of the tube. Here a second quartz transducer converts the incoming acoustic impulse into an electronic impulse [11, p.209].

These delay lines are used in two ways [11, p.213]:

1. Short tubes used for the results of calculations.
2. Batteries of longer tubes used as the main memory store, illustrated in Figure 2.3.

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 [11, p.209].

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 [11, p.210].

Figure 2.3: A battery of mercury delay lines in EDSAC [3]

### 2.2.1 Timing

The memory in EDSAC uses a circulating bit rate of 500 kHz. This is made up of a 0.9 µs pulse, and a 1.0 µs gap for each bit, although some literature specifies a pulse of 0.9 µs and a gap of 1.0 µs, implying a 526 kHz bit rate [11, p.209] [12, p.2]. The pulse is a burst of 13.5 MHz carrier frequency if the bit is a logical 1, or it is 0V if the bit is a logical 0 [12, p.2].

It should be noted that the reconstruction will use a bit rate slightly faster or slower than 500 kHz, because 500 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, in order to fully understand this decision, one needs to consider the sensitivity of the mercury originally used to changes in temperature. Mercury was chosen because its acoustic delay does not vary much with temperature[11, p.209]. The variation was still large enough however to cause 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 [13, p.81].

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. This means that the reconstruction project is free to choose a frequency not exactly equal to 500 kHz and still be faithful to the original system.

$$D = \frac{1}{500\,\text{kHz}} \times 576 = 1.15\,\text{ms} \tag{2.1}$$

In the regeneration portion of the circuitry, the pulses are demodulated from the 13.5 MHz carrier and stretched to approximately 1.9 µs long, i.e. just long enough that each pulse fails to overlap its neighbour [11, p.212]. The demodulated pulse then is passed into a logical AND function with the system clock to produce the regenerated pulse. This process is illustrated in Figure 2.4.

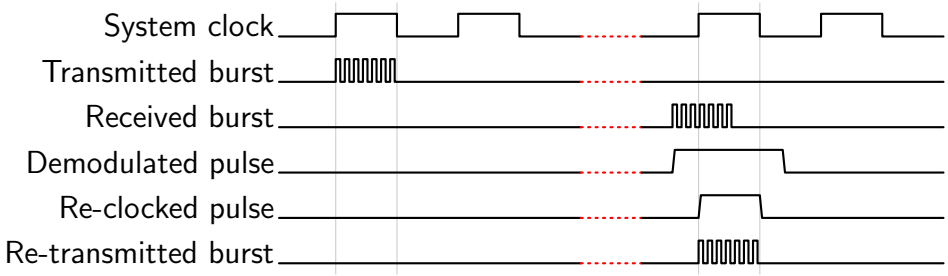



Figure 2.4: EDSAC pulse timing

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 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. Additionally, if the pulse width was not expanded whilst demodulating, any minuscule difference in timing between the system clock and delay line would cause failure very quickly, as each successive regenerated pulse would be narrower than the last.

The regeneration system implies that the maximum acceptable skew of the delay line from its nominal delay is ±500 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 µs, it is unlikely to be consistently at it's 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 ns, seems reasonable.

### 2.2.2 Electrical

Electrically speaking, EDSAC originally drove the delay lines with a nominal voltage of 25 V peak to peak, through a 70 Ω terminated transmission line. The loss in the delay lines was 68 dB, leading to an output voltage of approximately 10 mV [11, p.212].

Despite this, the recreation project has discovered that the regeneration circuitry actually feeds the delay lines with a decreasing voltage as the pulses propagate through the lines. The 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 by the reconstruction project with amplifying the low signal level output by the delay lines at the 10 mV level. Because the current wire delay

line solution has flexibility in its output voltage, currently the delayed signal is output at 100 mV peak.

### 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 [11, p.210]. An illustration of this is shown in Figure 2.3.

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 [11, 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.

## 2.3 Comparison with the current reconstruction effort

Creating a faithful reproduction of the mercury delay line system described in the previous section poses many challenges. These challenges are discussed at length in [2], however a few of the most prevalent are: 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 these challenges, the reconstruction project currently intends to use magneto-restrictive delay lines [14], an example of which is illustrated in Figure 2.5. This technology works by having a length of nickel wire with transducers on each end which can either twist the wire, or detect a change in how the wire is twisted. The input pulses are therefore delayed by momentarily twisting the wire at one end. This impulse then travels down the wire and is detected at the other end after a delay.

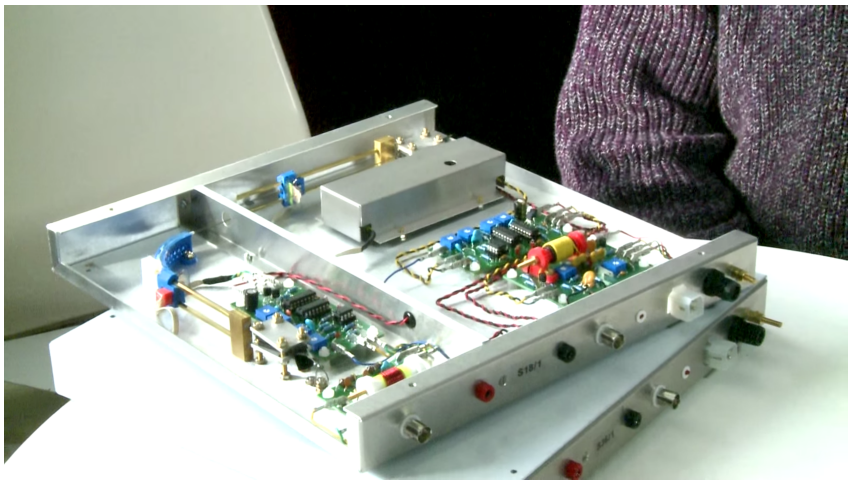


Figure 2.5: A magneto-restrictive delay line used in the reconstruction project [4]

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. Due to these shortcomings, it was decided to investigate the use of modern technology to emulate the original delay lines.

Emulation using modern technology means that it should be possible to create a system indistinguishable from the original in terms of appearance and electrical interface, whilst avoiding the need to the use of hazardous substances or tight machining tolerances. The design of such system is the aim of this project.

# 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 pulse train presented to it's input | This is the primary function of the device. |
| 2 | **Must** be powered from the input signal driven by EDSAC, with only minimal, and 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 connections must be avoided. |
| 3 | **Must** 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 chosen 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 alternating current (AC) coupled bursts of 13.5 MHz carrier, with peak voltages in the range of 25 V to 35 V at the input. | 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 Ω. | 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 significant design effort. 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. This is discussed prior to the design of the remainder of the delay line circuitry, since it is affected by the method of power.

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 seems 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 4.1, and in the best case the line is driven approximately half of the time.

$$P = \frac{V^2}{R} = \frac{10^2}{68} = 1.5\,\mathrm{W} \tag{4.1}$$

The problem with this is approach is that, in the worst case, 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. Attempting to use this pulse to power the delay line would require the turn-on time to be very predictably, so that the delay of the first pulse could be compensated for. Unfortunately the unpredictable delay of the 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 4.2 will discuss the initially proposed solution which turned out to be flawed after further investigation, and Section 4.3 discusses a slightly more complex solution that overcomes the shortcomings of the former.

## 4.1 Power Requirement

The FPGA development board is powered using an LT3030 dual-output linear regulator [15]. This regulator produces a both a 3.3 V, and a 1.2 V rail which powers the general purpose input/output (GPIO) banks, and FPGA core voltage respectively. Initial testing of the FPGA showed that the development board, without any support circuitry connected, showed a current draw of 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 current will be drawn at all voltages. This current is larger than would be expected, as the Lattice power estimation tool estimated that the FPGA would draw approximately 15 mA for the design. Investigation of the other integrated circuits (ICs) on the development board revealed that the remainder of the circuitry on the PCB only accounted for an extra 10 mA – 15 mA. Careful inspection of the evaluation board schematic revealed that both of the power supply rails were loaded with 100 Ω resistors, accounting for 45 mA of current draw, as calculated by Equation 4.2. 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.

$$\begin{aligned} I &= \frac{V_1}{R_1} + \frac{V_2}{R_2} \\ &= \frac{3.3}{100} + \frac{1.2}{100} \\ &= 45\,\text{mA} \end{aligned} \tag{4.2}$$

## 4.2 DC Offset Solution

Figure 4.1 shows the output portion of the store regeneration schematic. The store regeneration unit is the unit responsible for transmitting pulses to the delay line, and amplifying the pulses returned by the delay line. 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 power the delay line would be to add a direct current (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, a solution is illustrated in Figure 4.2.

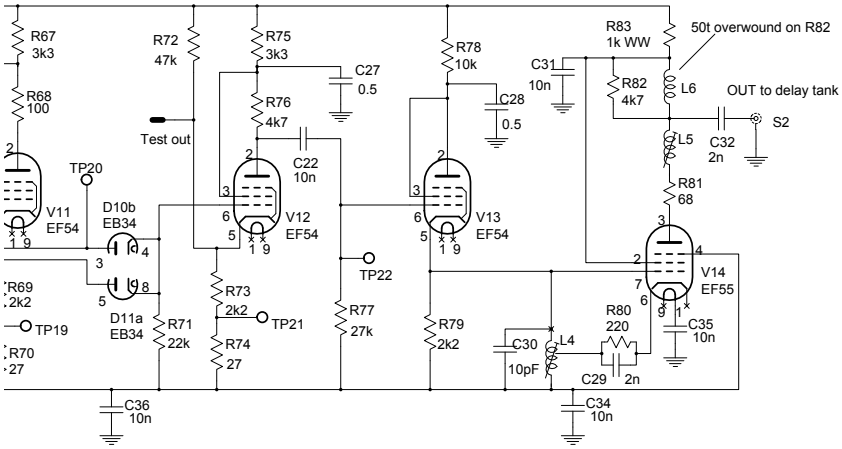



Figure 4.1: Store Regeneration Unit Output [5]

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 high frequency (HF) signal bursts inside the delay line. In addition, since all of the circuitry is powered using DC, the biased input could be used directly, after low pass filtering to remove the HF pulses, 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 high voltage (HV) 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) [16]. This means that any current returned via earth, as opposed to isolated 0 V rail, will cause a fault, and the power supply will be turned off.

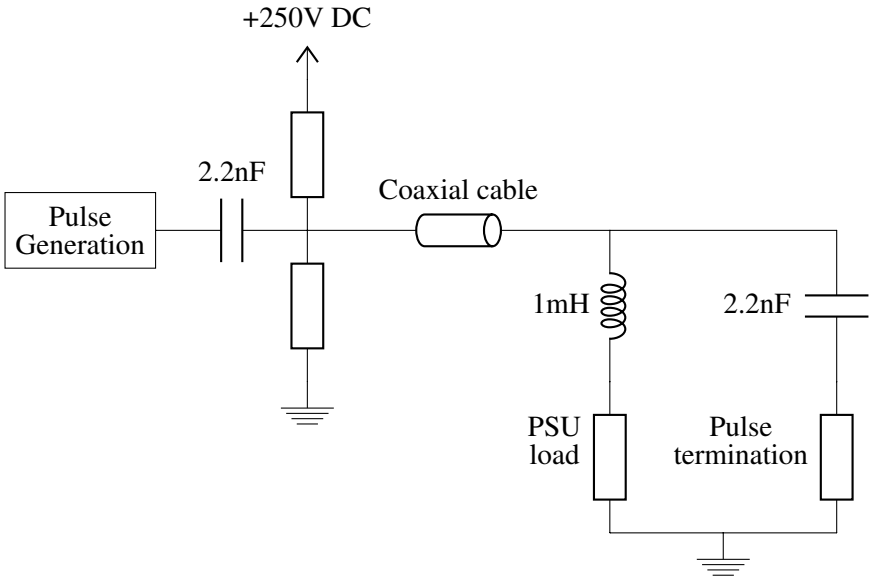



Figure 4.2: Proposed DC offset solution

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 allows the AC termination current to be returned correctly, but prevents any DC current from being drawn. This limitation therefore precludes drawing a DC current from the HV DC valve power supply.

## 4.3 Valve Power Supply Solution

Given that the DC offset solution proposed in the previous section was not 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 [17].

The heater power supply is a 6.3 V root mean square (RMS), 50 Hz, AC supply, generated by mains transformer mounted on each chassis [18, p.1] [16, p.1]. 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 [18, p.1].

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 be easy to filter out with even a first order filter.

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.

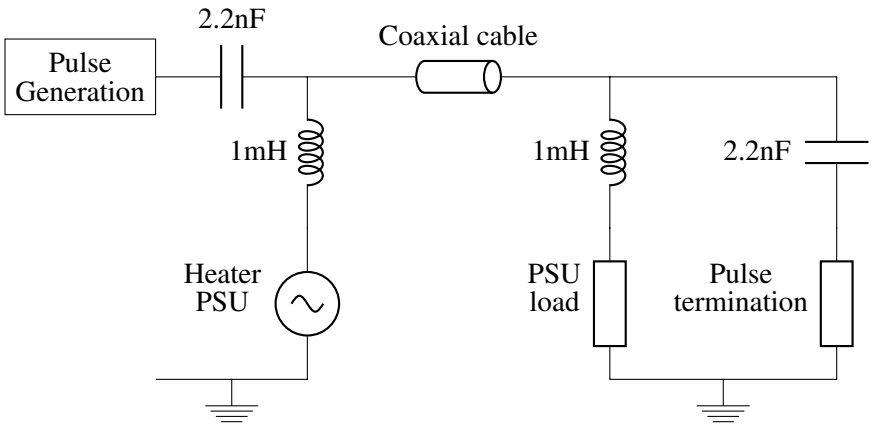



Figure 4.3: Proposed valve heater supply solution, equivalent circuit

$$|Z_{L50\,\mathrm{Hz}}| = 2\pi fL = 2 \times \pi \times 1\,\mathrm{mH} \times 50\,\mathrm{Hz} = 0.31\,\Omega \tag{4.3}$$

$$|Z_{L13.5\,\mathrm{MHz}}| = 2\pi fL = 2 \times \pi \times 1\,\mathrm{mH} \times 13.5\,\mathrm{MHz} = 85\,\mathrm{k}\Omega \tag{4.4}$$

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.

$$|Z_{C50\,\mathrm{Hz}}| = \frac{1}{2\pi fC} = \frac{1}{2 \times \pi \times 2.2\,\mathrm{nF} \times 50\,\mathrm{Hz}} = 1.4\,\mathrm{M}\Omega \tag{4.5}$$

$$|Z_{C13.5\,\mathrm{MHz}}| = \frac{1}{2\pi fC} = \frac{1}{2 \times \pi \times 2.2\,\mathrm{nF} \times 13.5\,\mathrm{MHz}} = 5.4\,\Omega \tag{4.6}$$

The same principle works for the termination and power supply circuitry inside the delay line, which will be discussed in more detail in Section 5.2.1.

### 4.3.1 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.

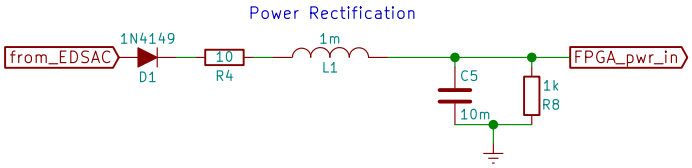



Figure 4.4: Delay line power supply schematic

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 tone burst, preventing unnecessary loading on the input signal, as discussed in Section 4.3.

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 in Section 4.1.
2. Energy will also be stored in the magnetic field of the inductor, as well as the electric field of the capacitor, which is ignored in this approximation.

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

The only additional component is the 1 kΩ 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 with the surrounding analogue circuitry translates between the logic level signal signals of the digital circuit, and the input and output being discussed in Section 5.2

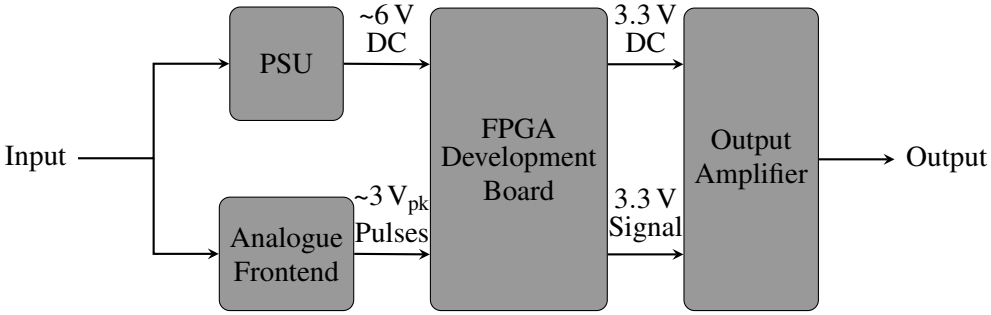



Figure 5.1: The overall architecture of the delay line

## 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 sample the data and clock it into a single bit wide shift register. 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.

$$t_d = \frac{1}{f_s} \times N \tag{5.1}$$

This solution, whilst possible, requires a large number shift registers for even a modest delay. The solution can therefore be simplified by consideration of the fact that the input signal is not arbitrary, the characteristics or the signal are known from the research detailed in Section 2.2.1. It is known that:

- The signal will consist of 0.9 µs pulses of 13.5 MHz tone.
- Each tone burst will be separated by 1.1 µs.
- 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 random access memory (RAM) available than what is required to store the array of times at which pulses arrived.

The principle disadvantage is that microcontrollers, inherently to their architecture, process data sequentially. 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.

As an example the system could be implemented using a typical microcontroller in the advanced RISC machine (ARM) Cortex-M0 family. To achieve the delay, one could utilise the various timers inherent inside the microcontroller architecture. To detect an input pulse a pin change interrupt would interrupt the main execution thread, and branch to an interrupt handler. This interrupt handler would read the value of a free-running timer peripheral that increments unconditionally using microcontroller master clock. The handler would then add a fixed delay value to the counter value and appends it to a queue of timer counts held in RAM. This queue contains the values of the counter, at which the output should be asserted.

The main execution thread of the microcontroller would be responsible for configuring the free running timer 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 [19]). Microcontrollers with timers capable of being configured as described above are readily available also [20].

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 ICs. A block diagram of the proposed architecture is shown in Figure 5.2.

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 is added to the current count and stored in a buffer. A separate hardware module monitors the buffer and triggers a pulse generator at the correct time.

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 first in, first out (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.

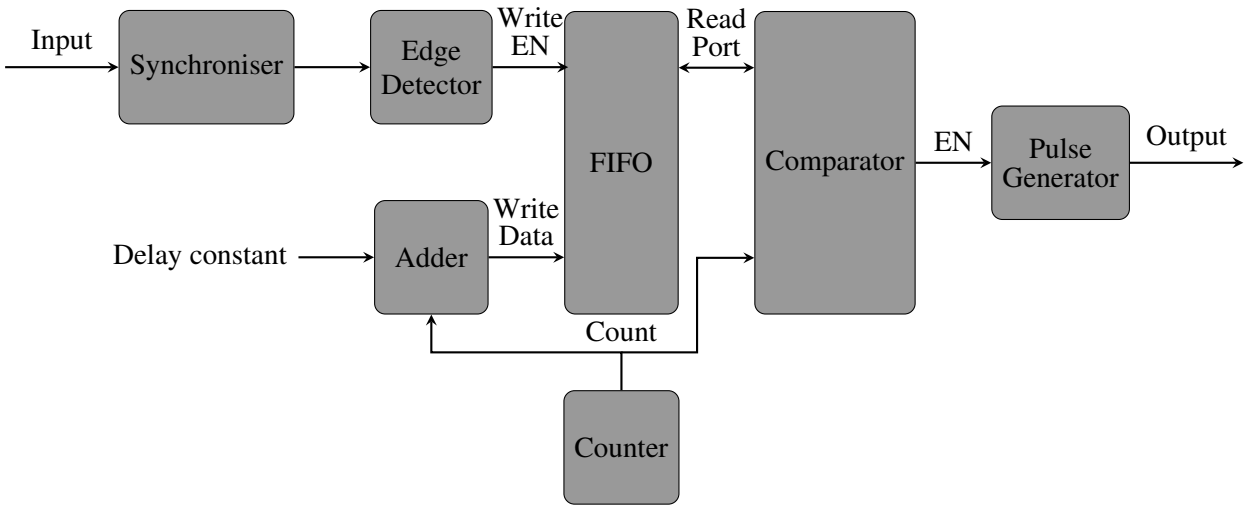



Figure 5.2: Proposed architecture for a FPGA or discrete system

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. This is due to the fact that configuration of many hardware peripherals are required.

In addition, the cost advantage of the microcontroller solution may be smaller than is typical when comparing microcontroller and FPGA based systems. 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 greater than 20 MHz. | System specification point 4 states that the maximum deviation from the chosen delay value is 50 ns. Achieving this requires that the signal is sampled at a rate faster than $\frac{1}{50\,\text{ns}} = 20\,\text{MHz}$. |
| 4 | **Should** have a development board available with a width less than 3.3 cm. | System specification point 7 states that the external diameter of the tube is 4.44 cm. Therefore assuming that the inner diameter is equal to three quarters of the outer diameter, this is the maximum diameter of board which would fit inside the tube. It would be therefore 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 [21]. This is £5.10 in single quantity, and significantly lower in higher quantities, 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 [22, 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.

$$S = 64\,\text{b} \times 576 = 36\,\text{kb} \tag{5.2}$$

It is hard to estimate how fast a design will be able to operate in a FPGA without synthesising it, and running timing analysis. Despite this, we can estimate that the FPGA will easily meet timing at 20 MHz, 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 [15]. The exact dimensions are not provided, but it is barely wider than the 22 mm thin quad flat package (TQFP) of the FPGA itself [15, 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 was 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 [23] which can be used as an alternative to the toolchain provided by Lattice [23]. Project IceStorm synthesises Verilog natively, and Lattice's iCEcube2 toolchain synthesises both Verilog and VHSIC hardware definition language (VHDL) [24, p.10] [23]. 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 5.2, with the rising edge detector, FIFO, comparator, and pulse generator implemented as individual Verilog modules.

#### Rising Edge Detector

The rising edge detector is implemented as the state machine of Figure 5.3. In the wait state, the input is sampled on every clock edge, when it is true, meaning that a rising edge has been detected, the state machine transitions to the assert state, where the output is asserted for exactly one clock cycle, until the state machine unconditionally branches to the time-out state. In the time-out state a counter is incremented until it reaches the required limit, designed to be long enough to guarantee that the pulse has finished. At this point the state machine branches back to the wait state.

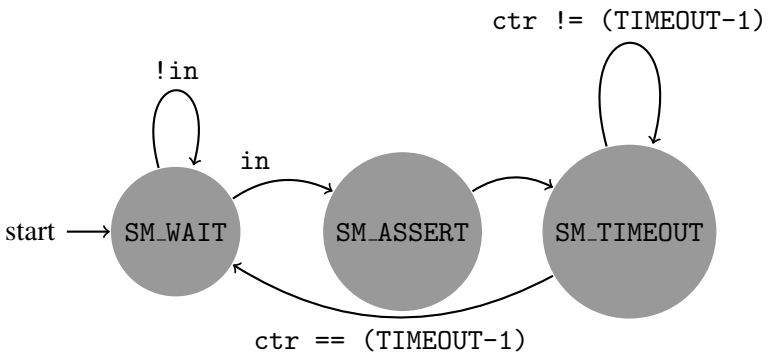



Figure 5.3: Rising Edge Detector State Machine

### FIFO

The FIFO is implemented using a dual port RAM block, around which logic constructs are built to handle addressing of the read and write ports.

The main part of the FIFO is the read and write address generation logic. These addresses act as pointers to the current location of the read/write word in RAM. Each address is incremented on when the FIFO it read from/written to respectively, and the address loops around to the start when it reaches the 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 5.4.

`empty` is the `empty` signal from the FIFO, and the FIFO read request signal is combinationally set to be 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 combinationally set to be true when the state machine is in the `SM_ASSERT` state.

The state machine therefore works be waiting in the `SM_WAIT_FOR_DATA` until the FIFO is not empty, at this point is requests the data from the FIFO and waits in `SM_WAIT` until the correct time to trigger the output, at which point the system transitions to `SM_ASSERT` to trigger the output modulator before resetting.

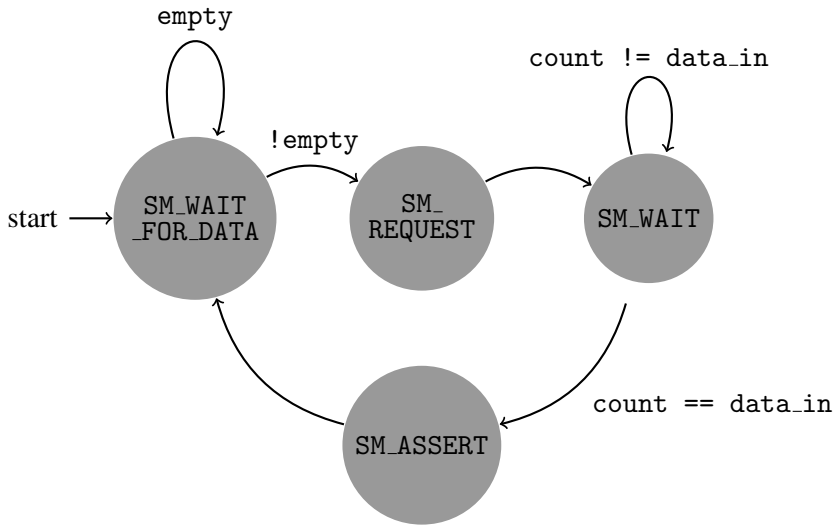



Figure 5.4: Comparator State Machine

### Pulse Generator

The pulse generator is implemented as another simple state machine, illustrated by Figure 5.5. The system remains in the `WAIT` state until the generator is enabled once this is true it alternates between the `HIGH` and `LOW` states transitioning when an internal counter reaches a set limit (i.e. `period_done` is true). Once a second counter, which counts the correct number of pulses to output, reaches its limit, the system transitions back to the `WAIT` state.

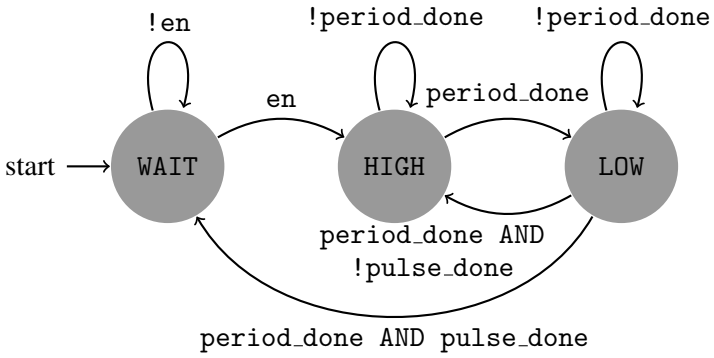



Figure 5.5: Pulse Generator State Machine

## 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. This section discusses the implementation of this circuitry

### 5.2.1 Input

The input voltage needs to be terminated with 68 Ω, the closest E12 preferred resistor value to 70 Ω and the value which is used by the remainder of the reconstruction project to terminate the transmission lines [5]. In addition to this, the power system imposes a requirement to remove the 50 Hz power signal discussed in Chapter 4.

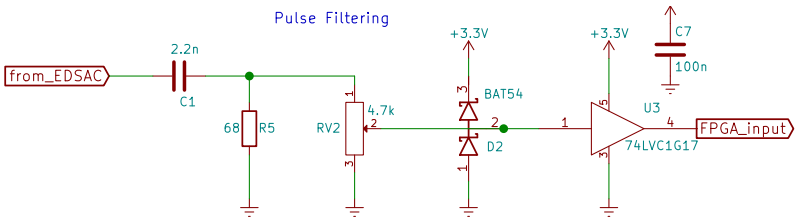



Figure 5.6: Delay line input schematic

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

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

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

The next stage of the input circuitry is a 4.7 kΩ 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 Ω 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 [25, p.2]), and the Schmitt buffer input leakage current (5 µA max [26, 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. The are suitable for the task due to their fast recovery time, 5 ns, and low forward voltage, 0.4 V at 10 mA [25, p.2]. Additionally their continuous forward current rating, 200 mA max is far greater than what can be sourced through the 4.7 kΩ 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, using a standard LVCMOS output, the output pin of the FPGA will be 0 V when no pulse is being transmitted, and 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 5.7.

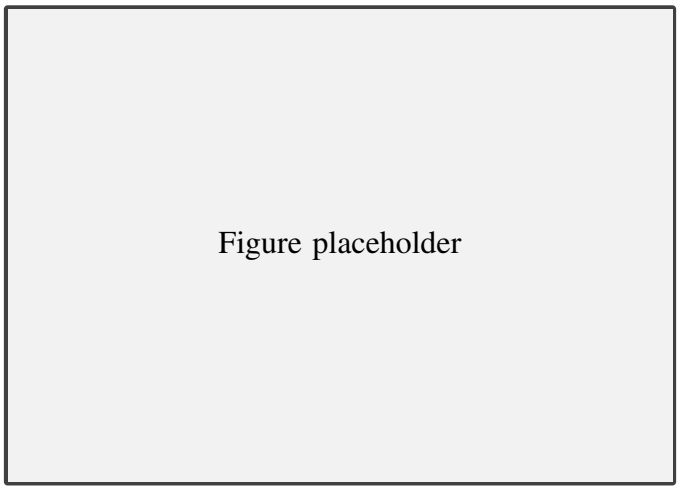



Figure 5.7: Effect of having the op-amp output idle at 0 V

In order to solve this problem, 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.

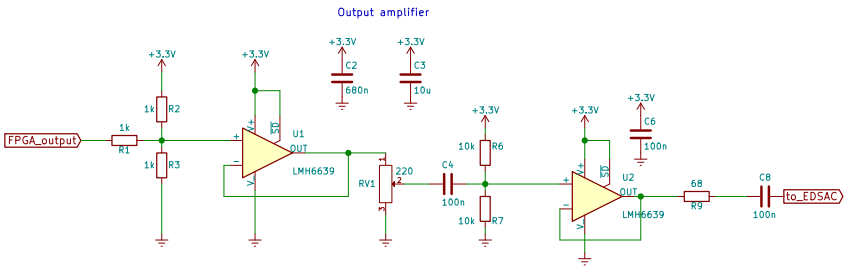



Figure 5.8: Delay line output schematic

For both amplifiers in the circuit LMH6639 amplifiers are used [27]. This amplifier is selected 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 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 Ω.

# Chapter 6

# TEST HARNESS DESIGN

The test harness is intended to serve two purposes. The first is to output signals to the delay line amplified to the correct output voltage level, and the second is to amplify the received pulse from the delay line and verify that it matches what it expected from the delay line.

The test harness is able to control the following parameters about the modulated signal:

1. Output pulse on and off period
2. Number of words in store
3. Demodulator output pulse width
4. Modulator frequency

In addition to setting the parameters, the test harness is able to store any number in any one of the memory locations.

In order to facilitate control of the test harness, a piece of personal computer (PC) software able to interact with the test harness has been created.

Section 6.1 discusses the communication format used between the graphical user interface (GUI) application and the FPGA, Section 6.2 discusses the hardware description language (HDL) design of the test harness, Section 6.3 the design of the software interface, and Section 6.4 the design of the input and output amplifiers.

## 6.1 Communication Format

The PC and test harness communicate over a universal asynchronous receiver/transmitter (UART) link. The reason for this is the ubiquity of serial interfaces inside embedded electronics, leading to readily available universal serial bus (USB) to UART converters with good driver support. The Lattice FPGA development boards contain such a converter already on their iCE40 development board, removing the need for external hardware. In addition to this, a UART protocol does not have a bus master, instead allowing both parties to transmit messages asynchronously across separate data channels. This allows the FPGA development board to inform the PC of any changes in state without the PC having to poll it.

The UART packet format used is eight data bits, one stop bit, and no parity bits transmitted at 9600 baud. This was chosen because it is very simple to implement in the FPGA, and is also very commonly used, and widely supported.

Since the messages communicated between the PC and FPGA are greater than eight bits in length, messages will be transmitted as eight packets, each containing eight bits of data. The first packet transmitted is a header packet indicating the type of the message being sent, and the following seven packets are the payload. The message is transmitted using little-endian format. The contents of the payload can vary across messages, and can contain multiple binary numbers. These numbers can be extracted using bit masking. Table 6.1 shows the valid test harness message types. It should be noted that the message types which echo back a received message, such as the acknowledge message, and some of the error messages necessarily omit the last byte from the incoming message, so as to obey the eight byte message size. This is not a problem, however, since the last byte of all the messages that do not echo an incoming message is guaranteed to be all zeros.

Table 6.1: UART message types

| Message Type | Purpose | Payload Contents |
|---|---|---|
| `received wrong num` | A message sent by the FPGA to the PC to indicate that an incorrect number was received. | `addr` – The address where the incorrect number was received.<br>`data` – the erroneous data received. |
| `replace num` | A message sent by the PC to the FPGA to request replacement of a number in the EDSAC store. | `addr` – The address of the number to replace.<br>`data` – the new data. |
| `replace num done` | A message sent by the FPGA to the PC to inform the PC that a number has been received correctly. | `addr` – The address of the number replaced. `data` – the old data. |
| `mod params` | A message sent by the PC to the FPGA to change the parameters of the modulator. | `cycles per half period` – The number of clock cycles per half period of the modulated signal. |
| `demod params` | A message sent by the PC to the FPGA to change the parameters of the demodulator. | `pulse width` – The width that each incoming pulse should be demodulated to. |
| `sys status` | A message sent by the PC to the FPGA to change the status of the system. | `run` – A boolean value that determines whether the system should run or not. |
| `mem params` | A message sent by the PC to the FPGA to change the parameters of the memory manager. | `no nums` – The number of EDSAC numbers in the store.<br>`test mode` – Tells the system if it should output numbers as usual, or output a continuous stream of pulses for testing.<br>`pulse width` – The width of the output pulse burst.<br>`pulse gap` – The width of the gap between output pulse bursts. |
| `err fifo full` | A message sent by the FPGA to the PC to indicate that its UART input FIFO is full. | None. |

Table 6.1: UART message types

| Message Type | Purpose | Payload Contents |
|---|---|---|
| `err mem overrun` | A message sent by the FPGA to the PC to indicate it missed a message from the memory manager because dealing with a UART request took too long. | None. |
| `err update whilst run` | A message sent by the FPGA to the PC to indicate that the PC attempted to send an update parameters request whilst the system was running. | `payload` – the message which caused the error (less the final byte). |
| `err invalid msg` | A message sent by the FPGA to the PC to indicate that the PC sent the FPGA a message of invalid type. | `payload` – the message which caused the error (less the final byte). |
| `ack` | A message sent by the FPGA to the PC to acknowledge that a message has been received and acted upon. | `payload` – the message which is being acknowledged (less the final byte). |

Since a large number of messages exist, manually writing header files for the software and HDL that encode and decode the messages would be tedious and error prone. In addition it would be likely that as the formats of the messages need to be changed in future, allowing discrepancies between the software and hardware designs would be introduced. In order to eliminate this a Python script was written to automatically generate the Verilog and C++ header files necessary for the software and hardware implementation. This means that changes to the message format only need to happen in one place, and both systems can be updated automatically by regenerating the files.

## 6.2 Digital Design

The digital part of the test harness is implemented as a FPGA design. There are several reasons for this choice. Firstly, the design is largely based around on parametrised generation of output signals, and sampling of input signals. This is the type of application that lends itself to a hardware design rather than a software design. Secondly, the part of the design which would lend itself to a software application, namely interfacing with the external controller and sending / receiving messages is designed to be simple enough that a hardware implementation is not too cumbersome. Finally choosing a FPGA design, targeted towards a Lattice iCE40 FPGA means that the same IceStorm toolchain as was used for the delay line can be reused for the test harness.

A block diagram of the digital test harness architecture is shown in Figure 6.1. At the centre

of the diagram is the controller. This maintains system state, and acts on received messages and events as they occur, by polling both the memory manager and received message FIFO.

The two signal paths above the controller are the transmit and receive data paths. Both have a FIFO so that messages can be enqueued for transmission without having to wait for the UART to finish transmitting the current message, and messages can be received faster than the controller is able to instantaneously handle them. The message assembler block waits to receive eight UART packets, and then enqueues the assembled 64 b word in the FIFO, and the message disassembler does the opposite taking a 64 b word from the FIFO and transmitting it as eight separate packets using the UART transmitter.

The demodulator block samples its input and when it receives a rising edge it outputs a pulse of the length determined by its parameters. This mimics the analogue demodulation which occurs in EDSAC.

The modulator does the opposite, it contains a free running oscillator whose frequency is set by its parameters. The output of the system is then the logical AND of the memory manager output and the oscillator output.

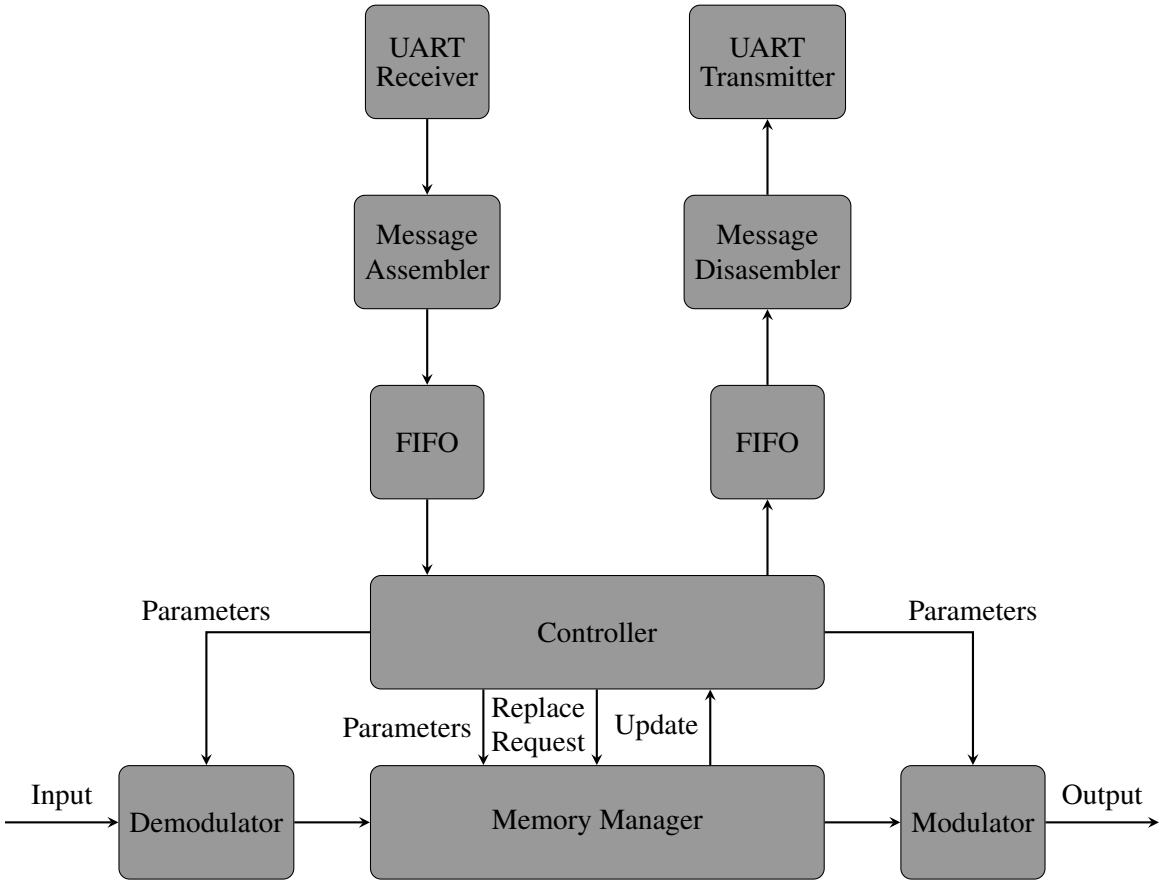



Figure 6.1: Proposed Architecture for Test Harness

The most complex module in the design is the memory manager, a block diagram of which is shown in Figure 6.2. This module is responsible for transmitting pulses to the delay line, and ensuring that a pulse is received back at the correct time, or that no pulse is received when none is expected.

The transmitted pulse is regenerated by sending it directly to the output, or is replaced if a different number is being transmitted rather than recirculating the existing number.

The first stage of the memory manager is the clock generation blocks. The output clock

generator uses a counter to divide the system clock and generate a 500 kHz clock used for the output, it additionally generates strobe signals for the rising and falling edge of the output clock. The bit counter increments using the rising edge strobe of the output clock, and keeps track of the current bit being transmitted. The word counter performs a similar purpose, but keeps track of the current word in the store which is being transmitted.

The number memory is an array that stores the expected value of each number, as well as a flag telling the system whether this is the current number, or if the store should be updated with this number. If it is a replacement number, the number is loaded into the replacement number shift register at the correct time, and the multiplexer outputs the least significant bit of the replacement number shift register, which is then shifted by the output clock, this is as opposed to normal operation where the output is simply connected to the input to regenerate the number.

The input is clocked into the received number shift register. When an entire word has been received, the contents of the shift register is compared to the corresponding number in the number memory. If the two do not agree than an error is signalled to the controller.

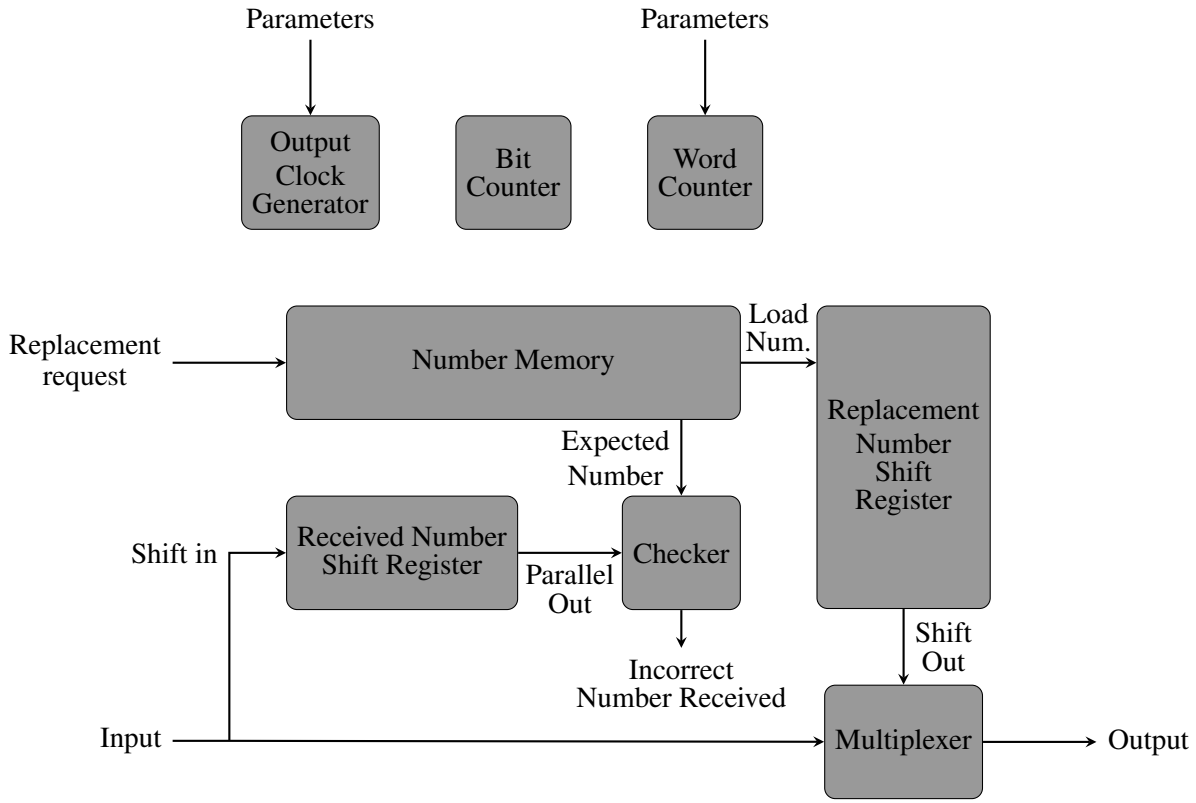



Figure 6.2: Proposed Architecture for Memory Manager

## 6.3 Software Design

The PC software to interact with the test harness must be capable of keeping track of the system state, sending messages to the delay line, and interpreting the results, and presenting that information to the user. Initially, a command line tool was considered for this but, in order to provide a simpler user interface, it was decided to create a GUI application, shown in Figure

6.3. Despite this, it would be fairly simple to replace the GUI with a command line frontend, due to the object orientated nature of the design.

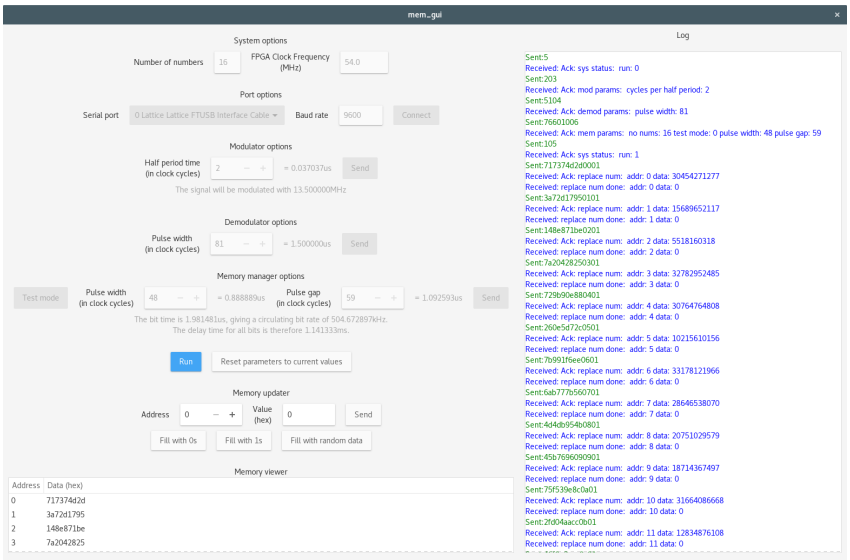



Figure 6.3: Screenshot of the Software which interacts with the delay line

The software is designed to be cross-platform and due to this it uses open-source, cross platform libraries to implement the interface to the outside world, namely the GUI framework and USB to UART converter. Additionally, the CMake build system is used to allow compilation on different platforms [28]. The GUI is implemented using gtkmm, which is a C++ wrapper for the popular GTK+ GUI framework [29]. The library used for serial communication is libftdi, which is an open-source alternative to the proprietary drivers provided by Future Technology Devices International (FTDI) for their USB to UART bridge ICs [30].

In order to allow for a modular design, the system is comprised of a series of classes. The central class is called `MainGui`. This class is responsible for creating the GUI and responding to human input. `main()` passes control to this class once some initialisation has been performed. This class then relies on a series of small classes which are responsible for the GUI elements of each sub-section of the display. These small classes all inherit from a virtual parent class, that provides a common interface for `MainGui` to communicate with the subclasses. Allowing operations such as message passing between the classes in the `MainGui` processing loop to occur simply via iterating over a list of references to each GUI element.

Aside from the classes relating to the operation of the GUI, there are a number of other classes which implement lower level operations of the software. `FtdiWrapper` is a class written to create a C++ wrapper for the, C, libftdi library. This is the class which handles setting up the serial port and the transmitting and receiving of messages. One layer of abstraction above this is the `UartManager` class. This class maintains queues of messages to be sent to, or which have been received from, the FPGA. It is also responsible for ensuring that the FPGA acknowledges each message. The messages stored in these message queues are of the type `UartMsg`. This is a base class which encapsulates a UART message and provides access to the underlying data, as well as the header and payload bits. In addition to this for each message type there is a derived class which provides a print function to print the various data words inside the message, as well as getter and setter functions for each field. The `UartMsg` header is automatically generated by the Python software described in Section 6.1.

The other classes which manage lower level operations are the `StatusManager` and `MemoryManger` classes. The `StatusManager` class keeps track of whether the system is running or not, as well

as the parameters for the various FPGA modules. It also generates objects of type `UartMsg` to change the system status or parameters. The `MemoryManger` class manages the status of a single word of EDSAC memory. It keeps the current value, sends messages to the test harness to update the value and ensures that the corresponding update messages are received. The system uses an array of these objects to keep track of the entire store.

## 6.4 Analogue Design

Unfortunately, as discussed in Section 2.2.2, interfacing with the delay lines poses difficulty since the delay line needs to be driven with a very high voltage, 25 V to 35 V, and the incoming pulse is very low amplitude, 10 mV to 100 mV. This section discusses the amplification circuitry required to do this.

### 6.4.1 Input

The input to the delay line needs to be able to amplify the signal from the attenuated level output by the delay line, and present the FPGA with a 0 V to 3.3 V logic level signal

The circuitry to do this is relatively simple, and is illustrated in Figure 6.4. A two stage non inverting amplifier is sufficient to amplify the incoming pulse to a voltage level suitable for the Schmitt trigger. The gain of each op-amp is adjustable in order to accommodate variation in the output voltage of the delay line.

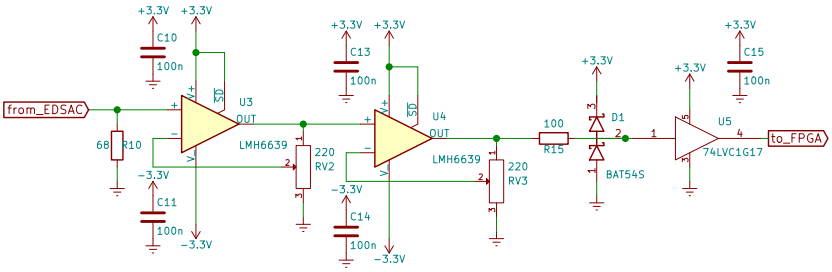



Figure 6.4: Test harness input schematic

The LMH6639 op-amps selected for the delay line output, discussed in Section 5.2.2 are also suitable for this purpose. The 190 MHz GBP allows for a amplification of fourteen times per stage before appreciably attenuating the 13.5 MHz carrier frequency. The limited slew rate of 172 $V\mu s^{-1}$ is also not a concern due to the relatively low amplitude of both the input and output signals.

It should be noted that a gain of fourteen times per stage, meaning a total gain of 196, would amplify a 10 mV peak to peak signal to 1.96 V peak to peak. The positive portion of this would only be 0.98 V peak, which is not high enough for the Schmitt trigger to detect. This is not a problem, however, since the delay line can be tested using the higher 100 mV peak to peak signal amplitude used by the reconstruction project. If it is desirable in the future to test the lower amplitude signal then a further amplification stage can be added.

The Schmitt buffer provides a clean square wave to the FPGA for detection of the input pulses, and provides protection against incorrect detection of noise on the line. The diodes on the input of the Schmitt buffer are to protect the Schmitt buffer input from the negative pulses output by the op-amp by clamping the input to just below 0 V, in a similar manner to the circuitry discussed in Section 5.2.1.

### 6.4.2 Output

The output circuitry of the test harness is similar in principle to the input circuitry, but is more complex in the execution. In contrast to the amplification of the input signal, the output signal is very large in amplitude, 25 V to 35 V peak to peak, and needs to be able to drive this into a 68 Ω load. This means that, when a tone burst is being transmitted, 4.5 W is delivered to the load, as shown by Equation 6.1. In this equation the voltage is 17.5 V, this is because the maximum output from EDSAC is a 35 V square wave symmetric about 0 V, this is equivalent to delivering a constant voltage of 17.5 V to the load.

$$P = \frac{V^2}{R} = \frac{17.5^2}{68} = 4.5\,\mathrm{W} \tag{6.1}$$

In order to simplify implementation of the output stage, one restriction was added. It was decided that the amplifier selected would be limited to voltage rails of ±15 V. This widely improves the range of op-amps available to purchase as the vast majority are not specified for voltages above 30 V. This design choice does, however, mean that maximum peak to peak output voltage will be a little under 30 V. This is lower than the maximum voltage driven by EDSAC, but is large enough to demonstrate the principle of the delay line correctly receiving a large amplitude pulse. It is simple to reason, given the topology of the delay line input stage discussed in Section 5.2.1, that if the delay line works well with lower amplitude pulses, 35 V peak to peak pulses will not present a problem.

To implement the amplification circuit, there were several requirements for the amplifier. First the amplifier must have the ability to be powered from ±15 V rails, as previously discussed. In addition, the amplifier must have a high GBP, at least several times greater than the 13.5 MHz carrier frequency to allow modest amplification.

Another specification required that the amplifier have a slew rate greater than $810\,\mathrm{V\,\mu s^{-1}}$, as calculated by Equation 6.2. This equation calculates the minimum slew rate in order for the amplifier to transition from the lower output rail to the upper output rail in half a period of the modulation frequency, meaning at this slew rate a triangle wave would be output.

$$\begin{aligned} SR &\geq \frac{V_{\text{pk-pk}}}{\frac{1}{2} \times \frac{1}{f_m}} \\ &\geq 2 \times V_{\text{pk-pk}} \times f_m \\ &\geq 2 \times 30\,\mathrm{V} \times 13.5\,\mathrm{MHz} \\ &\geq 810\,\mathrm{V\,\mu s^{-1}} \end{aligned} \tag{6.2}$$

Finally the op-amp must be able to deliver a current of at least 220 mA, as calculated by Equation 6.3. This equation calculates the minimum output current that the op-amp must be able to deliver assuming that two op-amps are being used in parallel to deliver a tone burst of 50% duty cycle to the load.

$$\begin{aligned} I &\geq \frac{1}{2} \times \frac{1}{2} \times \frac{V}{R} \\ &\geq \frac{1}{2} \times \frac{1}{2} \times \frac{15}{68} \\ &\geq 110\,\mathrm{mA} \end{aligned} \tag{6.3}$$

Surveying the op-amps available for purchase, the LM7372 was selected [31]. This op-amp is specified for $\pm 15\,\mathrm{V}$ operation, has a very high slew rate of $3000\,\mathrm{V\,s^{-1}}$, and a GBP of 120 MHz. In addition to this the amplifiers have a maximum output current of 150 mA per amplifier, with two amplifiers present per package.

Having selected the amplifier, the circuit of Figure 6.5 was designed to drive the delay line. This amplifier is in three stages. The first stage is a passive biasing stage that presents the input to the first amplifier with 0 V when nothing is being transmitted, and $\pm 1.65\,\mathrm{V}$ when the tone burst is being transmitted. This first stage amplifier buffers the input signal and amplifies it with gain tunable by potentiometer.

The second stage is an output buffer which uses two parallel op-amps to buffer the signal and drive the load. The op-amps outputs are connected in parallel with a modest current sharing resistance. Ideally the series resistance on the output of each op-amp should be twice the characteristic impedance of the transmission line, as discussed in [32]. This was not possible however because a series resistance of $2 \times 68\,\Omega = 136\,\Omega$ would limit the peak output voltage to slightly less than $15\,\mathrm{V} \times \frac{68\,\Omega}{68\,\Omega + 136\,\Omega} = 5\,\mathrm{V}$, as the output op-amps saturate close to the voltage rails. Fortunately reflected pulses are not an issue in this case since the delay line only needs to detect the carrier, integrity of the pulses is not critical.

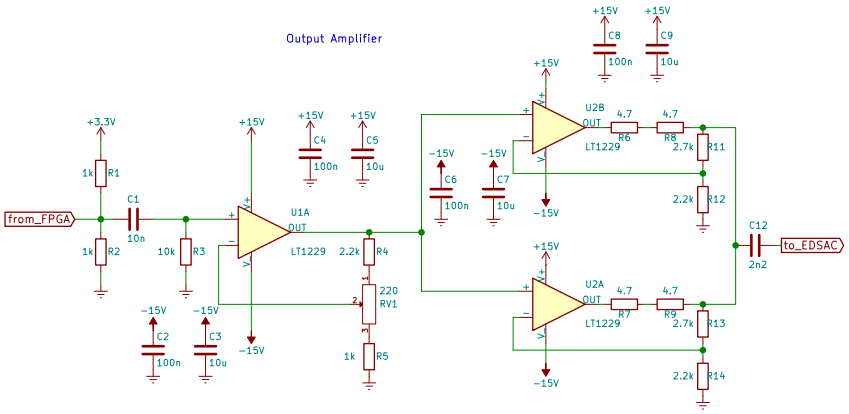



Figure 6.5: Test harness output schematic

It should be noted that despite the design and implementation containing the twin amplifier output stage, the second amplifier is disconnected in the final matrix board implementation. This is due to issues encountered with the stability of the op-amps when driving the load in parallel. This does not matter since consultation with the datasheet shows that the 150 mA previously stated was the short circuit current at $\pm 5\,\mathrm{V}$, when using a $\pm 15\,\mathrm{V}$ supply a single amplifier can source 260 mA, and sink 250 mA, which is greater than double the 110 mA of Equation 6.3. Therefore, with adequate heat sinking, a single amplifier can drive the load. This was achieved in practice by 'dead-bug' soldering the amplifier IC upside down to expose the heatsink pad, then attaching a high thermal mass heatsink and further cooling with a fan, illustrated in Figure 6.6

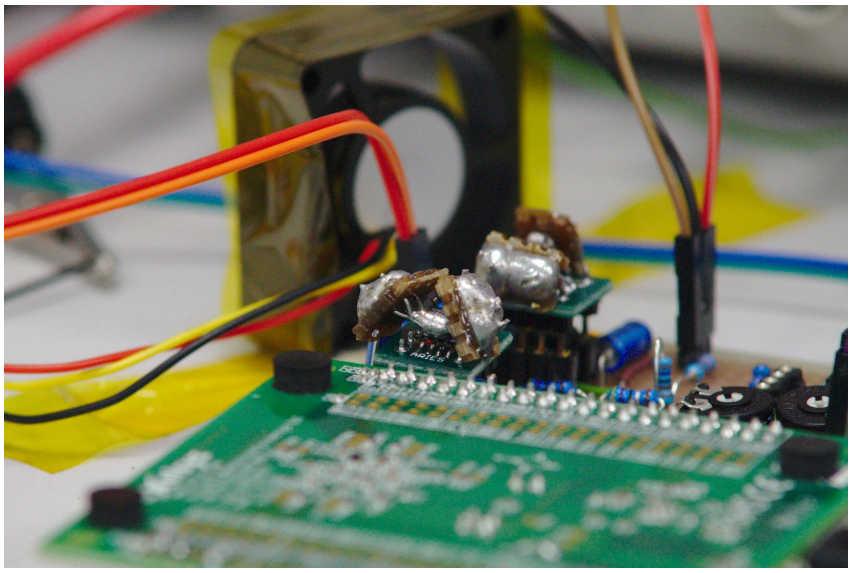


Figure 6.6: Test harness cooling solution

# 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 [33]. 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. It was therefore 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 [34]. This 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 SPICE netlists which use the models from the DesignSpark [35] schematics which the reconstruction project uses [34]. Instead of using this workflow, 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

The main SPICE simulation is the simulation of EDSAC output, and delay line input. This simulation is important because it models several of the concerns for integration with the rest of EDSAC, namely:

- Checking that the method of phantom powering does not effect the store regeneration output
- Checking that the phantom power circuitry works correctly
- Checking that the pulses are able to be detected and output to the FPGA

In order to run the simulation, first a model had to be created of the output of the store regeneration unit. To achieve this an LTspice schematic was created containing the store regeneration circuitry from test point 22 to the output. This section was chosen because the expected signal at TP22 is known from the commissioning documentation of EDSAC [36, p.7], and so

simulation from this point onwards is sufficient to model the output of the store regeneration unit.

This model was then tested and works correctly when terminated with 68 Ω.

The valve heater voltage was then modelled as a 50 Hz AC source, and connected to the store regeneration output with the correct inductance.

Following this, both the power generation frontend and the pulse detection frontend of the delay line were added to the schematic. The pulse regeneration frontend is modelled as far as the Schmitt buffer which connects to the FPGA, and the power frontend is modelled as far as the outputs of the LDO regulator on the FPGA development board, which is loaded with resistors that approximate the anticipated load.

Figure 7.1 shows the simulation of the voltage rails start-up from 0 V across the capacitor, gradually charging to just under 6 V. The LDO regulator is able to use this to generate stable 3.3 V and 1.2 V rails within 2 s from starting the 50 Hz generator. The green signal shows the input to the delay line, the blue signal shows the voltage input to the FPGA LDO regulator, the red signal is the 3.3 V rail, and the blue signal the 1.2 V rail. It should be noted that in this simulation the store regeneration output is disconnected. This is done for simulation speed as simulating the store regeneration output also would cause the simulation to take an unmanageably long time. A simulation was run of one 50 Hz cycle to ensure that the store regeneration output did not pass to the power supply rail, and this did not occur.

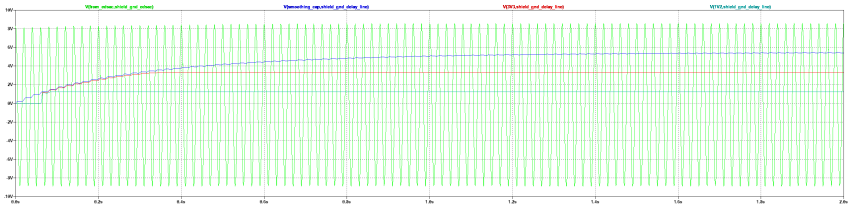


Figure 7.1: Voltage Rails Initialisation Simulation

Figure 7.2 shows a simulation of a pulse burst from the store regeneration unit output, coupled onto 50 Hz valve heater signal. In this simulation the green signal is the output of the store regeneration unit. The red signal is the output of the store regeneration unit with the 50 Hz signal filtered out, and the blue signal is the output of the Schmitt trigger which is an input to the FPGA. It can be seen that the signal input to the FPGA exactly follows the positive going pulses of the modulated packet.

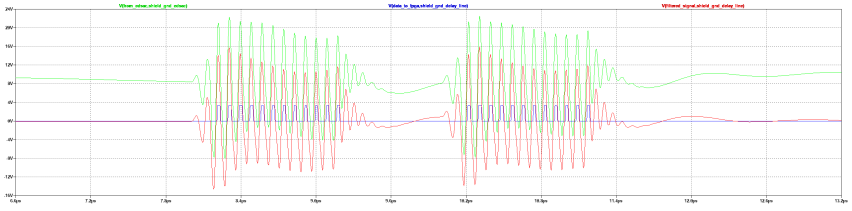


Figure 7.2: FPGA Input Simulation

### 7.1.3 Store Regeneration Input Simulation

The purpose of simulating the input of the store regeneration unit is to ensure that biasing the store regeneration output onto the heater power supply does not allow any of the modulated carrier signal to couple into the sensitive analogue input section. The theory of why this would

happen is illustrated in Figure 7.3. Pentodes typically have a non-insignificant parasitic capacitance between their heater pins, and cathode. This would not ordinarily cause issues 50 Hz AC heater supply, as even a moderately large parasitic capacitance would present a very high impedance to 50 Hz. If, however, there is a significant 13.5 MHz signal present on the heater power supply, then this could cause a problem as the frontend of the store regeneration unit is very sensitive. On the original pulses of as little as 10 mV peak to peak could be detected, as discussed in Section 2.2.2.

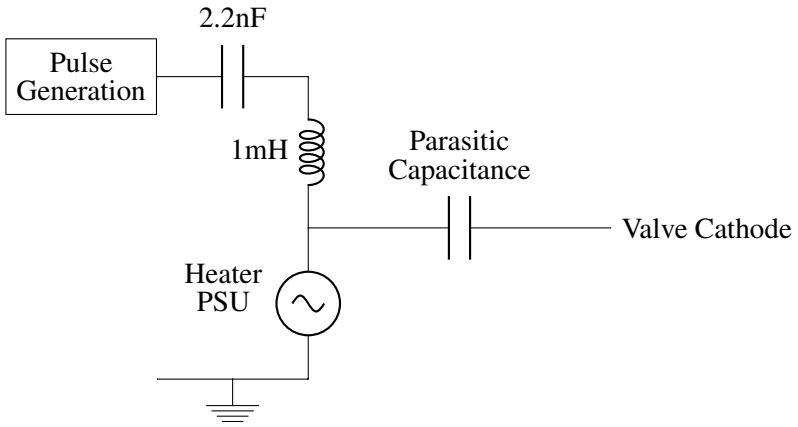



Figure 7.3: Valve heater supply, path to couple onto heater input

Figure 7.4 shows the input portion of the store regeneration circuitry. The pentodes used here are EF54 pentodes, and the capacitance of concern is the parasitic capacitance between pins one and five of each pentode.

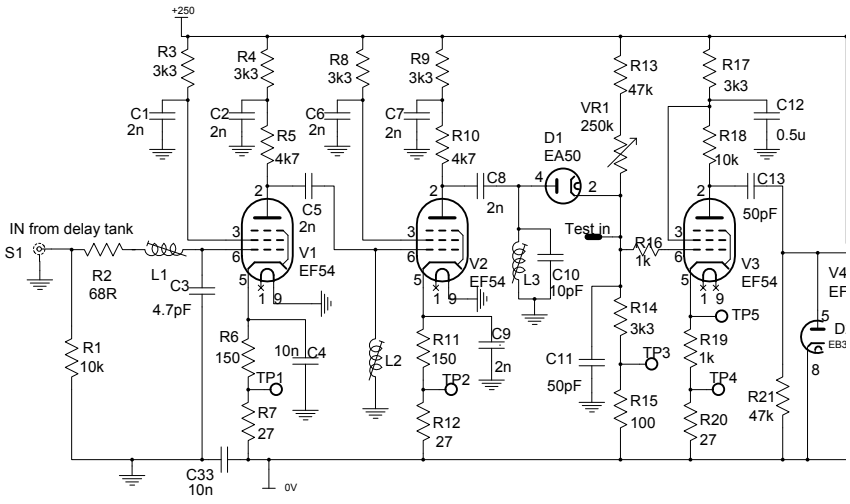



Figure 7.4: Store Regeneration Unit Input [5]

The first stage of simulation was to create a simulation model of the store regeneration frontend, without making any modification to the valve model. This was performed, and the valve model was created up to test point five, since the expected waveform at this point was known [36, p.3]. This model was created in an LTspice schematic and the input was fed with the expected output waveform from the delay line, simulation showed that the observed waveform matched what was expected.

The next stage was to look at the EF54 model netlist and check if any capacitance was

present between pins five and one of the model. No capacitance was present, and so the parasitic capacitance was not modelled.

The datasheet was then consulted, and it was found that the heater-cathode capacitance was not specified, however several other capacitances were, namely the input capacitance, output capacitance, anode to gate one capacitance, and gate one to gate two capacitance [37]. The largest of these capacitances is the input capacitance, 6.2 pF, and the smallest is the anode to gate 1 capacitance at 0.02 pF. Since the capacitance was not stated in the datasheet, it seems reasonable to assume that it is not largely more significant than those which are stated, and thus 10 pF was chosen as the value to model.

In order to simulate the effect of the output coupling onto the heater voltage, a 13.5 MHz, 30 V peak-to-peak generator was connected to the heater voltage via a 1 mH inductor, mimicking the proposed output circuit.

Figure 7.5a shows the result of the simulation with the 13.5 MHz tone generator disconnected. This is the waveform which would normally be observed at test point five, and is what the modified topology should aim to match. Figure 7.5b shows the result with the added 1 mH inductor. Figure 7.5b matches Figure 7.5a almost exactly, and so the proposed circuit works correctly.

As a test, the 1 mH inductor used to connect the 13.5 MHz tone generator to the output was replaced with a 10 nH inductor. This presents a much lower impedance to the 13.5 MHz tone, and thus one would expect the waveform at test point five to be disrupted. Figure 7.5c shows this result, and the effect of the tone generator can clearly be seen, this demonstrates that the modified valve model works as anticipated.

## 7.2 HDL Verification

Due to the lengthy iteration cycle required to synthesise a HDL design and test it in hardware, as well as the necessity of treating a synthesised HDL design as a black-box model, rather than being able to inspect internal signals, verification via simulation is critical.

In order to verify the designs, in the first instance a SystemVerilog testbench was written for each module, and each module was simulated using Mentor ModelSim [38]. These unit-level 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 system level testing, rather than extensive unit based testing. The reason for this choice is that many of the modules which make up a design are fairly simple and their functionality is trivial to verify. The majority of the problems occur during integration of the blocks at a higher level, as well as unexpected problems interfacing two modules.

SystemVerilog has been chosen as the language for verification, rather than plain Verilog which the synthesizable HDL is implemented using. There are several reasons for this. Firstly ModelSim will be used for all of the verification in this project, so maintaining compatibility with multiple toolchains is not a concern. Secondly, SystemVerilog is a superset of Verilog, so therefore integrates well with the Verilog code. The final, and most important reason, however is that SystemVerilog includes a number of advanced simulation constructs, such as complex assertions, properties, and a number of abstract data types, that allow for simpler and more comprehensive verification.

The remainder of this section presents the verification strategies used for the delay line, and test harness. It should be noted however that the simulation waveforms are best presented in

landscape format, so are presented in Appendix D.

### 7.2.1 Delay Line Verification

In addition to the delay line HDL design testbenches were written for several of the subsystems including: the edge detector, pulse generator and FIFO. For brevity, however, these will not be discussed in this report, and the focus will be on the behaviour of the top level module.

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, and each transmitted pulse burst is added to a queue with the simulation time when the pulse is expected to arrive on the delay line output. 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 D.1 shows the output of this testbench. Figure D.1a shows the full simulated output. The top signal is the generated by the testbench and input to the delay line, and the second signal is the output of the delay line. The output is in a high impedance state (blue) when no pulse is output, and is black when the signal is being driven. The pulses in the output signal relate exactly to a delayed version of the input pulses. This may not immediately be obvious from the simulation output image due to the image aliasing effects of having so many pulses displayed. The correct performance of the testbench is verified by the textual simulation output which displays `# Simulation completed successfully` after the simulation run, indicating that none of the assertions failed, and each output pulse was asserted at the correct time. The lower two signals are the internal FIFO full and empty signals. It can be seen that the FIFO never entirely fills up which is the expected behaviour, and the empty signal is asserted once there are no more delayed pulses being stored in the FIFO.

Figure D.1b shows two individual output pulses from the FIFO. The signal is high impedance when no pulse is being driven, then is a 13.5 MHz tone burst which is the desired behaviour as discussed in Section 5.2.2.

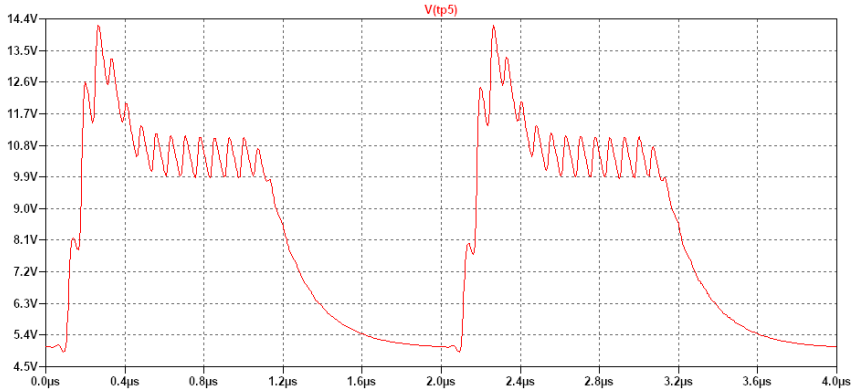
### 7.2.2 Test Harness Verification

To verify the test bench at a system level, a testbench was created that emulates a UART and transmits messages to set up the test harness and transmit numbers into the delay line, which is represented by a Verilog delay in the simulated model. It can be observed whether the test harness returns the expected results.

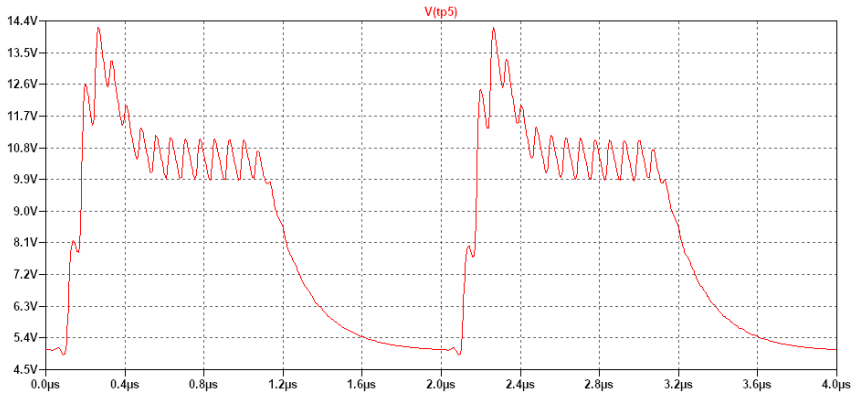
Figure **??** show the results of the test harness simulation. This simulation tests configuration of the modulator, demodulator, and memory manager, starting the system, requesting that the number at address one is replaced with data three, and then requesting that the number at address one is reset to zero again.

The top signal shows the UART messages sent from the test harness to the HDL design, and the signal below it shows the messages returned by the FPGA, it should be noted that the UART signal from the test harness to the design is superimposed with random noise in order to test the immunity of the UART receiver to noise, this is the reason why there are many more transitions than would be expected shown in the simulation waveform. The two signals below this are the output to the delay line from the test harness, and delayed signal returned to the test harness. It can be seen from these two signals that a pulse is transmitted into the delay line, received, and regenerated several times before being removed, as intended.

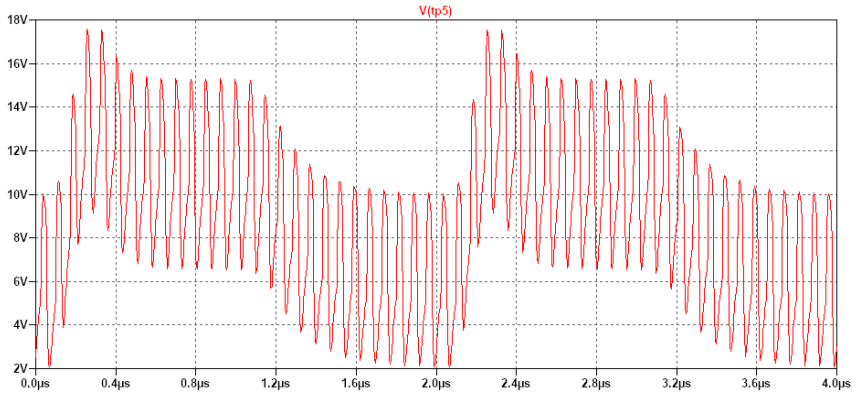
The messages to and from the test harness are logged in the test bench output and reproduced in Figure 7.6.




(a) 13.5 MHz output disconnected from heater




(b) 13.5 MHz output connected to heater with 1 mH inductor




(c) 13.5 MHz output connected to heater with 10 nH inductor

Figure 7.5: Store Regeneration Unit Input Simulation Results

```
# Sending 0000000000000005 at time          1000000000
# Sending 000000006c621006 at time         10333333360
# Sending 0000000000000203 at time         19666666720
# Received 000000000000050b at time         23489018149
# Sending 0000000000005104 at time         29000000080
# Sending 0000000000000005 at time         38333333440
# Received 0000006c6210060b at time         38488598149
# Sending 0000000000000105 at time         47666666800
# Received 000000000002030b at time         53488178149
# Sending 0000000000030101 at time         57000000160
# Sending 0000000000000101 at time         66333333520
# Received 000000000051040b at time         68487758149
# Received 000000000000050b at time         83487338149
# Received 000000000001050b at time         98486918149
# Received 000000000301010b at time        113486498149
# Received 0000000000000102 at time        128486078149
# Received 000000000001010b at time        143485658149
# Received 0000000000010102 at time        158485238149
```

Figure 7.6: Logged output of the simulation

# Chapter 8

# System Integration

Having developed and tested the delay line and test harness subsystems described in Chapters 5 and 6, it was necessary to integrate the subsystems together and test the system as a whole. This chapter details how the subsystems were integrated mechanically, and tested at a system level. The results of this testing is presented, alongside the issues encountered in this process.

## 8.1 Mechanical

In order to integrate the delay line subsystems together, a metal tube of correct outer dimensions was sourced, meeting Specification Point 7. In order for the tube to function as a standalone unit, it was necessary to deviate slightly from the original design. This is because the original tube did not have end caps, instead the delay lines were terminated with a metal plate that held all of the tubes in the 'coffin', and the coaxial connectors were on the top of the 'coffin'. This can be seen in Figure 2.3. As a compromise end caps were machined for the delay line that cover the ends of the tube and mount the coaxial connectors perpendicular to each end cap. A photograph of the assembled delay line, whilst being tested is reproduced in Figure 8.1.

Figure 8.1: Testing the assembled delay line with the test harness

In order to mount the delay line design inside of the metal tube, the circuits described in Figure 5.1 were assembled on matrix board and connected to the end caps using coaxial cable.

The final result of this assembly is shown in Figure 8.2. In this figure, the input port is the coaxial socket at the bottom of the photo. This feeds through a short section of coaxial cable to the first matrix board. This board implements the input circuitry described in Section 5.2.1, apart from the protection diodes and Schmitt buffer, as well as the power circuitry described in Section 4.3.1.

From this input board two coaxial cables, one carrying the DC power supply for the remaining circuitry, and one carrying the attenuated input signal connect to the collection of boards visible in the middle of the image. This is a stack of three boards. The lower board in the stack contains the Schmitt buffer and diode protection for the input. The middle board is the Lattice iCEstick FPGA development board, with the USB connector, and GPIO header removed, and the top board is the analogue output circuitry. The attenuated signal is then carried from the analogue output circuitry to the output coaxial connector via coaxial cable.

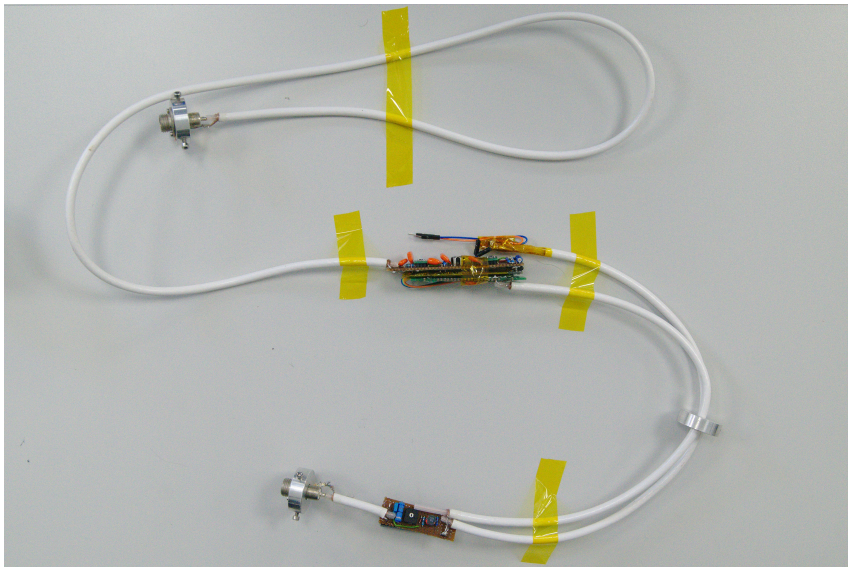


Figure 8.2: The delay line assembly

## 8.2 Issues

During the integration stages, several issues were encountered that had not been anticipated during development. This section describes these issues, as well as how they were resolved.

### 8.2.1 FPGA Reset Issue

The first issue which was encountered was that when the input cable was connected the FPGA would not always configure correctly, and the output would remain in a high impedance state, irrespective of the input waveform.

The reason for this was determined to be the fact that the input power rail is brought up slowly, as the smoothing capacitor on the input was charged. Reading the programming and configuration guide for the FPGA used, it can be seen that the FPGA checks all of it's supply voltages are within the correct range before it configures from the serial peripheral interface (SPI) flash [39, p.4].

Despite this check, the FPGA considers its GPIO power rail to be stable once it is somewhere between 0.7 V and 1.59 V [22, p.3-2]. This means that as the supply voltage slowly

ramps up, the 1.2 V core voltage will be stable, and the 3.3 V GPIO supply voltage will be within the range the FPGA considers stable long before it is close to 3.3 V. Therefore the FPGA will start configuration long before the 3.3 V is high enough to power the SPI flash IC correctly.

Further consultation with the programming and configuration guide reveals that the FPGA has a pin, named CRESET_B which can be held low to inhibit the configuration of the FPGA, and the FPGA will not configure itself until the signal is asserted [39, p.4]. As supplied this pin is held high on the development board with a 10 kΩ resistor.

Inspection of the datasheet of the LDO regulator which provides the 3.3 V and 1.2 V rails reveals that it provides two pins, named PWRGD1 and PWRGD2 to indicate when the output is within 10% of the nominal output voltage [40, p.11]. This is an open collector output which is high impedance when the power is below 90% of the nominal value and sinks current when the power output is above this. The output can sink a maximum current of 100 µA, and is pulled up with a 1 MΩ resistor on the development board.

The obvious solution is therefore to remove the pull-down from CRESET_B, and connect the pin to PWRGD1, the PWRGD output corresponding with the 3.3 V output, but there is a subtle problem with this approach. Lattice specify that CRESET_B must be either pulled with a maximum resistance of 10 kΩ, or actively driven [22, p4-1]. Using a 10 kΩ pull-up resistor would however require PWRGD1 to sink $I = \frac{3.3\,\text{V}}{10\,\text{k}\Omega} = 330\,\mu\text{A}$, far more than it is capable of sinking.

The devised solution is that of Figure 8.3. In this schematic the 1 MΩ pull-up resistor on PWRGD1 is replaced with a 50 kΩ resistor to provide a strong enough current for the Schmitt buffer, whilst not exceeding the 100 µA sinking limit of the LDO regulator. The output of this Schmitt buffer then drives CRESET_B, which is pulled down with a 10 kΩ resistor in case the FPGA detects that its power input are at the correct levels before the buffer IC has powered up.

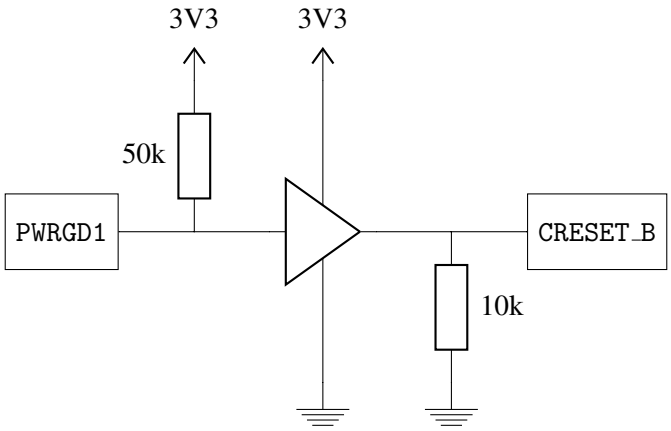



Figure 8.3: FPGA reset issue solution

### 8.2.2 Signal Breakthrough

Breakthrough of the input signal to the output has been the most difficult problem encountered during integration of the subsystems. Figure 8.4 demonstrates the problem and is a capture of the output of the system as initially integrated. The yellow trace in the oscilloscope capture is the input to the delay line, and the green trace is the output. The larger right hand pulse on the output trace represents the correct output of the delay line, however the left hand pulse should not be present, and is a result of the input coupling to the output. It is approximately 200 mV in amplitude peak to peak. This capture was taken with the oscilloscope connected directly to the

output of the amplifier, but the same behaviour occurred with the circuit inside the tube, and the coaxial output was probed.

One point of note is that for all of the traces captured in this section, the standard oscilloscope probe hood and ground clip were removed and a small spring clip was added to provide the probe with a local ground reference. The reason for this probing setup is that a standard length ground lead was able to pick up as much as 500 mV of the input signal, by acting as an antenna when not connected to the circuit.


Figure 8.4: Initial input coupling at FPGA output

The initial step to attempt to solve this issue was to implement the input circuitry, which was originally part of the central board assembly, separately and add the metal ring visible on the right hand side of Figure 8.2. When assembled, the ring will be connected to the, earthed, metal tube and shield the output circuitry from the input.

These modifications improved the performance, approximately halving the amplitude of noise which was coupled onto the output, but the problem still persisted.

The eventual solution was to remove the output amplifier and instead use a passive attenuation circuit on the FPGA output. Figure 8.5 shows the schematic for this circuit. The 4.7 kΩ potentiometer and 68 Ω resistor act as a potential divider to attenuate the signal, whilst the 10 nF capacitor removes DC bias from the signal to centre it about 0 V, this circuit was built on a small piece of matrix board mounted at the output coaxial connector, as far as possible from the rest of the circuitry.

This passive attenuation circuit performs well, and a screenshot of its performance is presented in Figure 8.6. In this figure the test harness has been modified to not regenerate pulses, and to output a pulse with a fixed different to that of the delay line. This makes it easy to see where the input couples onto the output. As is evident from the screenshot, any coupling of the input to the output is below the noise floor of the oscilloscope. It should be noted that the second channel of the oscilloscope is 50 Ω terminated because in this instance the delay line was assembled, and the output coaxial connector was directly probing the output instead of the output connecting to the amplifier circuit. 50 Ω termination was therefore added to the oscilloscope to approximately match the 70 Ω characteristic impedance of the transmission line.

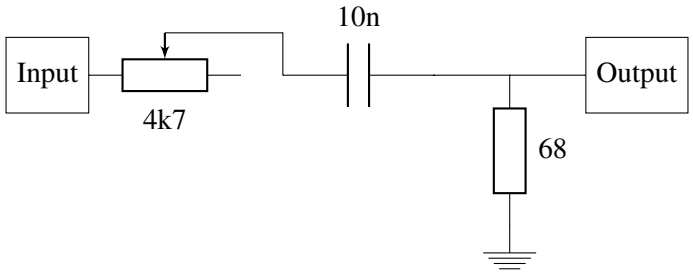



Figure 8.5: The passive attenuation circuit used

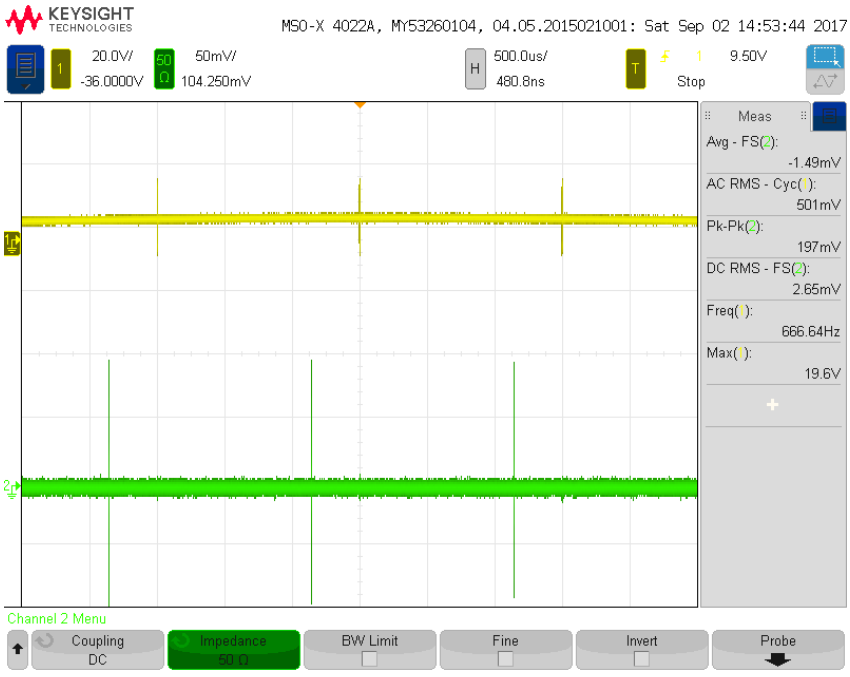



Figure 8.6: Results of passive attenuation circuit

# Chapter 9

# PROJECT PLANNING

Good planning was essential for the conduction of this project, due to the tight time frame, as well as the scale of the project. This chapter briefly details how the project was planned in terms of both the time allocated to each task, as well as the analysis of risks relating to the project.

## 9.1 Time Management

Time management was critical in this project due to the fact that the project is quite broad, involving the implementation of two separate systems, the delay line itself and the test harness, which involve work across several different disciplines including: software design, analogue electronics design, and programmable logic device (PLD) design.

Appendix B contains the charts used to plan how time would be spent on various activities during the project. The first chart included is Figure B.1 which is a project evaluation and review technique (PERT) chart produced for the project plan [41, p.3]. In contrast to this the Gantt chart of Figure B.2 is a record of how time was actually spent on the project.

There are two reasons why the charts are difficult to compare. Firstly the PERT chart assumed that the design and testing of each sub-system would be followed by a PCB implementation of the system, whereas in reality the final system was implemented using matrix board as discussed in Chapter 8. In addition, the PERT chart schedules time at the start of the project for a literature review. In reality, much of the background research was conducted during the writing of the research review, negating some of the need for the literature review [2]. It was therefore easier to begin planning the implementation of each module, and research any of the finer points of the original design as necessary. It should also be noted that the plan breaks the project into three modules: the delay line itself; the test harness, which is used to mean the digital part of the test harness; and an amplifier module, which is used to refer to the analogue parts of the test harness. For simplicity the final design combines the test harness and amplifier modules.

In order to compare the two plans it is therefore simplest to break the project into two phases. The first phase is the initial design phase, which includes design of each module, research of any of the finer points of the original design as may be necessary, and testing of the design. The second phase is the final implementation of the design, as well as creation of the formal documentation.

Inspection of the PERT chart shows that it schedules a total of six weeks for this phase. Comparison with the Gantt chart shows that the corresponding stages, all of the work up until the end of the group labelled ‘Initial Analogue Design’, took six weeks and one day. This shows that the project was largely on track up until this point. Unfortunately work could not begin on the second stage at this point, because it was here understood that the phantom power would require further work and investigation than was originally anticipated, as discussed in Chapter 4, and so a further sixteen days were used to solve this problem.

For phase two, the PERT chart schedules a total of three weeks for the PCB construction and formal documentation to be written. In reality, the corresponding Gantt chart groups, ‘Building final delay line circuity’, and ‘Documentation’ took five weeks. This has taken longer than

expected for several reasons. Firstly the project plan only allocates time for a standalone PCB implementation. It was not anticipated at the planning stage that the scope of the project would include integration into an actual metal tube. This has required additional construction time, as well as time to debug the issues relating to the signal breakthrough discussed in Section 8.2.2.

Considering that the unallocated time in the original PERT chart was intended for changes in scope and unexpected problems, and given that the project was completed successfully achieving all of its goals, it is considered that the time planning in this project has been successful.

## 9.2 Risk Management

In order to ensure that the project was able to deliver on its intentions, risk analysis was used. The aim of this exercise is to identify the main risks that would impact on the ability of the project to deliver against all of its goals. For each risk, the chance of occurrence, and impact are assessed as well as any mitigation efforts as necessary. Table 9.1 shows the risk matrix used by this project. A severe impact is an impact that would cause the project to not deliver some or all of its goals. A moderate impact is an impact that would cause the project to not deliver one goal, or cause undesirable performance despite achieving goals. A minor impact is an impact which would cause undesirable performance of the project whilst meeting all goals, and a negligible impact has no measurable impact upon the project.

Table 9.1: Risk Matrix

| | | Impact | | | |
|---|---|---|---|---|---|
| | | Negligible | Minor | Moderate | Severe |
| Chance of Occurrence | Very Unlikely | | | | |
| | Unlikely | | | | |
| | Likely | | | | |
| | Very Likely | | | | |

Table 9.2: Project Risks

| Risk | Estimated Likelihood and Impact | Risk Rating | Risk Controls | Treated Risk Rating |
|---|---|---|---|---|
| Modules will work independently, but not together | Unlikely, Severe | | 1. Comprehensive test harness GUI to allow easy debug.<br>2. Integration both before and after analogue circuit design | |
| The design will work in development, but the final design will not have the correct performance | Likely, Moderate | | 1. The final design is be implemented on matrix board, not using a custom PCB for greater reconfigurability.<br>2. Slack time was available at the end of the project to debug issues with the final design | |
| The design will work using the testbench, but will require further work to integrate with EDSAC | Very Likely, Moderate | | 1. The testbench is comprehensive to identify problems before integration.<br>2. SPICE simulation was performed with valve models to ensure the analogue part of the design will work with EDSAC | |

# Chapter 10

# FINAL COMMENTS

The aim of this project was to create a delay line capable of forming the store of EDSAC, without requiring excessive modification to the rest of EDSAC, and without deviating from the physical appearance of the original. This has been achieved in full, with this report presenting methodology to both achieve the delay, and emulate the driving signals of EDSAC. The circuitry required for the delay unit fits inside of the footprint of the original storage tubes and only requires the addition of a single 1 mH inductor, per delay line installed, in order to power the store.

In addition to the delay line itself, a comprehensive test harness has been created which is capable of driving each delay line, receiving and regenerating the returned pulses, and detecting any errors in the output of the delay line. This is suitable for testing the performance of each delay line without integration with EDSAC.

Due to difficulties with testing the design with the physical EDSAC reconstruction machine, notably the physical machine being located far from Southampton and the incomplete nature of the machine, extensive SPICE simulations were performed to ensure that the delay line would integrate well with the relevant components of EDSAC. These simulations were successful and showed that the delay line was able to integrate well with the store regeneration chassis in all respects.

Following the successful construction and testing of the delay line, it was taken to TNMoC to integrate with the rest of the machine. The integration effort was successful and the delay line was powered from the valve chassis, accepted the pulses sent to it, and transmitted pulses back to the chassis after a small delay. These pulses were correctly amplified and demodulated. The only problem encountered was an issue with the valve chassis not correctly regenerating the pulses. It was ambiguous as to whether this is a problem with the delay line or the chassis itself however, as there had not previously been any method to test this particular chassis with a delay medium.

There is some scope for further development on this project, notably to continue integration with the rest of the system, and there is also scope to produce the design more formally using a PCB now that the concept is proven. This is work which is intended to take place, with the intention that the delay line will be able to form some part of the final EDSAC exhibit and help to educate the general public about the history of computing.

# 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] P. Linnington. (2016, apr) Delay lines. National Museum of Computing. [Online]. Available: https://www.youtube.com/watch?v=TjDOo-Mwfl0

[5] C. Burton, "Panel 1 store regeneration unit schematic," apr 2014.

[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] National Museum of Computing. EDSAC replica project aims. [Online]. Available: http://www.tnmoc.org/special-projects/edsac/edsac-replica-project-aims

[10] 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

[11] M. V. Wilkes and W. Renwick, "An ultrasonic memory unit for the EDSAC," *Electronic Engineering*, vol. 20, pp. 208–213, jul 1948.

[12] C. U. M. Laboratory, "The EDSAC," may 1948.

[13] M. V. Wilkes, "The EDSAC computer," in *Review of Electronic Digital Computers (AIEE/IRE Conference 1951)*. AIEE, 1952, pp. 79–83.

[14] 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

[15] *iCEstick Evaluation Kit*, 1st ed., Lattice, aug 2013.

[16] A. Passmore, "HN57: Power supply operation and maintenance," jan 2015.

[17] C. Burton, "HN29: Chassis 01 assembly," nov 2014.

[18] Mullard, *Video Frequency Pentode EF55*, 1st ed.

[19] J. Yiu. (2016, apr) A beginners guide on interrupt latency - and interrupt latency of the Arm Cortex-M processors. ARM. [Online]. Available: https://community.arm.com/processors/b/blog/posts/beginner-guide-on-interrupt-latency-and-interrupt-latency-of-the-arm-cortex-m-processors

[20] *Kinetis KL46 Sub-Family*, 5th ed., Freescale Semiconductor Inc., aug 2014.

[21] 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

[22] *iCE40™LP/HX Family Data Sheet*, 3rd ed., Lattice, mar 2017.

[23] C. Wolf and M. Lasser, "Project icestorm," http://www.clifford.at/icestorm/.

[24] *iCEcube2 User Guide*, Lattice, feb 2017.

[25] Vishay, *Small Signal Schottky Diodes, Single and Dual*, 1st ed., feb 2013.

[26] Diodes Incorporated, *Single Schmitt-Trigger Buffer*, 4th ed., mar 2014.

[27] Texas Instruments, *LMH6639 190MHz Rail-to-Rail Output Amplifier with Disable*, mar 2013.

[28] CMake, "CMake." [Online]. Available: https://cmake.org/

[29] M. Cumming and D. Elstner, "gtkmm." [Online]. Available: https://www.gtkmm.org/

[30] intra2net, "libFTDI - FTDI USB driver with bitbang mode." [Online]. Available: https://www.intra2net.com/en/developer/libftdi/

[31] Texas Instruments, *LM7372 High Speed, High Output Current, Dual Operational Amplifier*, sep 2014.

[32] Intersil, *AN1111.1: Doubling the Output Current to a Load with a Dual Op Amp*, 1st ed., may 2005.

[33] Linear Technology, "Ltspice." [Online]. Available: http://www.linear.com/designtools/software/#LTspice

[34] P. Linnington, "HN68: Spiced valves," dec 2015.

[35] RS Components, "Designspark pcb." [Online]. Available: https://www.rs-online.com/designspark/our-software

[36] P. Linnington, "HN49: Spiced valves," may 2016.

[37] Mullard, *Video Frequency Pentode EF54*, 1st ed.

[38] Mentor, "Modelsim." [Online]. Available: https://www.mentor.com/products/fv/modelsim/

[39] *TN1248: iCE40 Programming and Configuration*, Lattice, may 2016.

[40] *LT3030 Dual 750mA/250mA Low Dropout, Low Noise, Micropower Linear Regulator*, A ed., Linear Technology, jun 2013.

[41] J. Tyler, "ELEC6211 project preparation  project plan: Mercury delay line emulation: Reconstructing memory for edsac," may 2017, this document is an unpublished project plan.

# Appendix A

# Folder Structure

The project is structured as a single repository using the git version control system. For the reader 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 Figure A.1 (in alphabetical order).

Note that unlabelled folders contain identical types of files to the same name of folder elsewhere in the hierarchy.

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

Figure A.1: Repository Directory Structure

# Appendix B

# Project Planning Charts

This Gantt Chart shown in Figure B.2 shows how time on the project was actually spent, and this is in comparison to the PERT chart of Figure B.1 which shows how time was initially planned to be spent at the start of the project.

These charts are discussed more fully in Chapter 9.

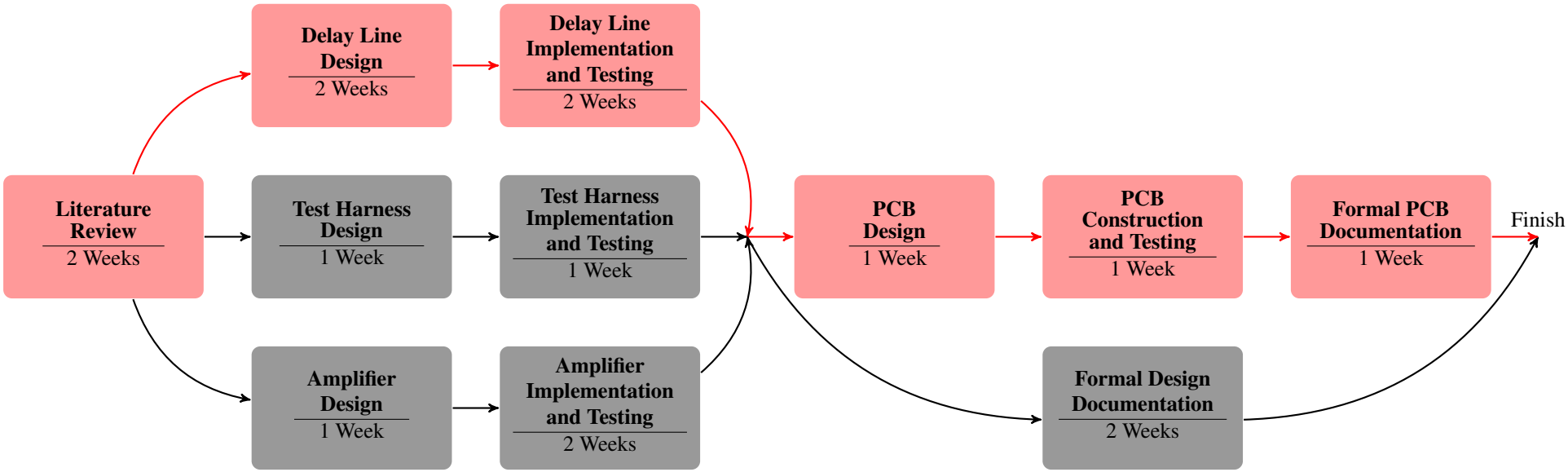



Figure B.1: PERT Chart used to plan the project

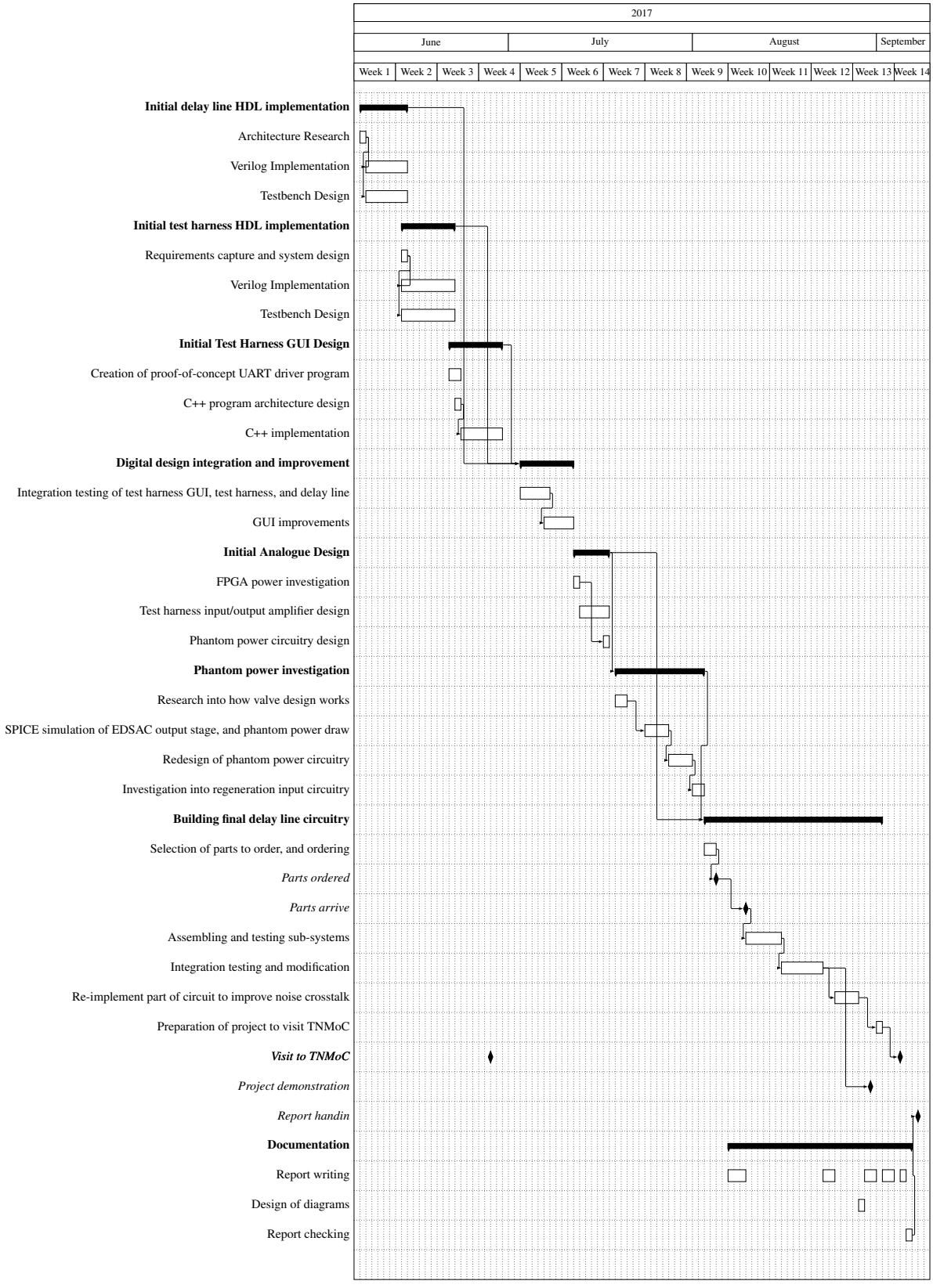
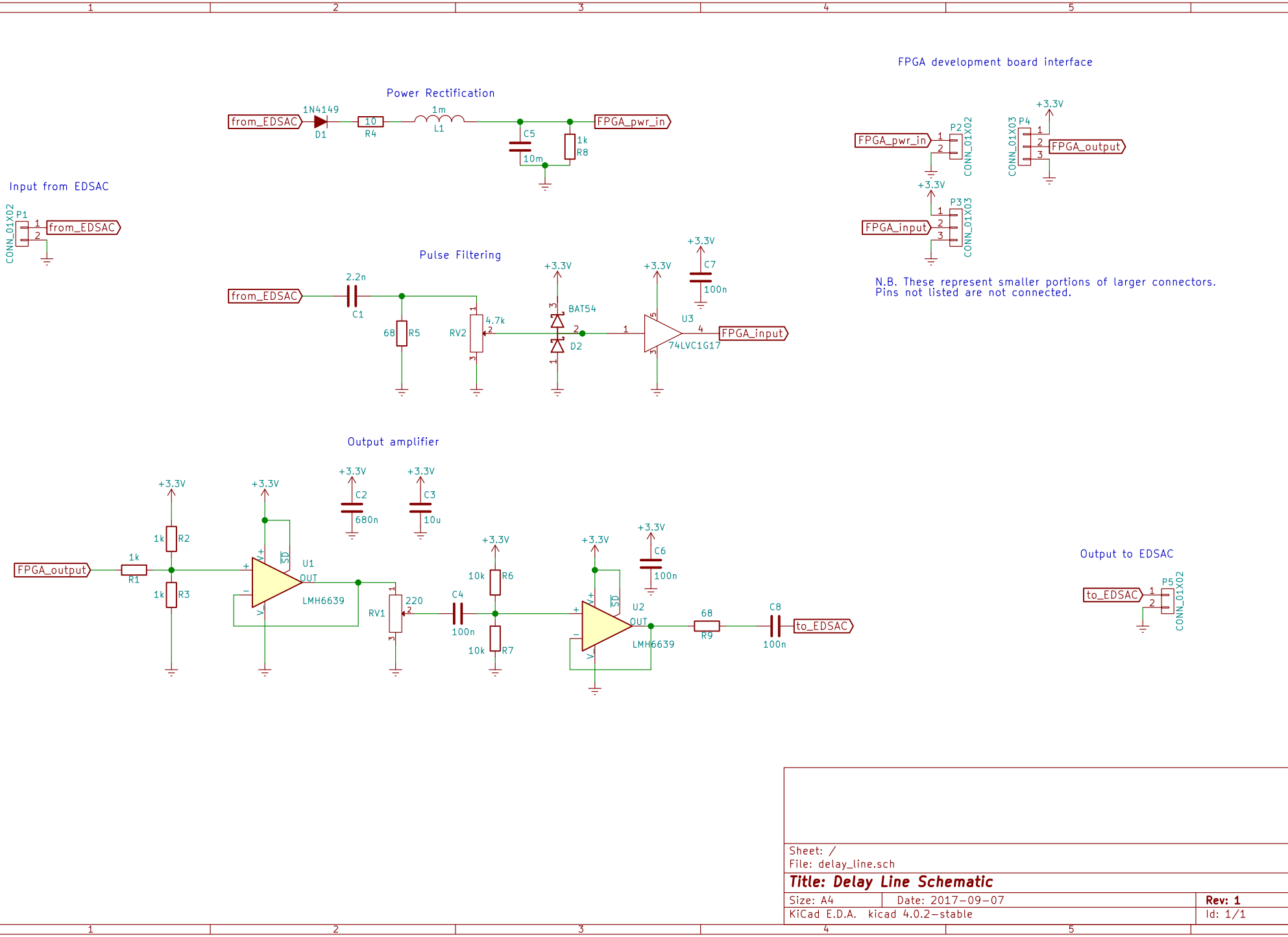
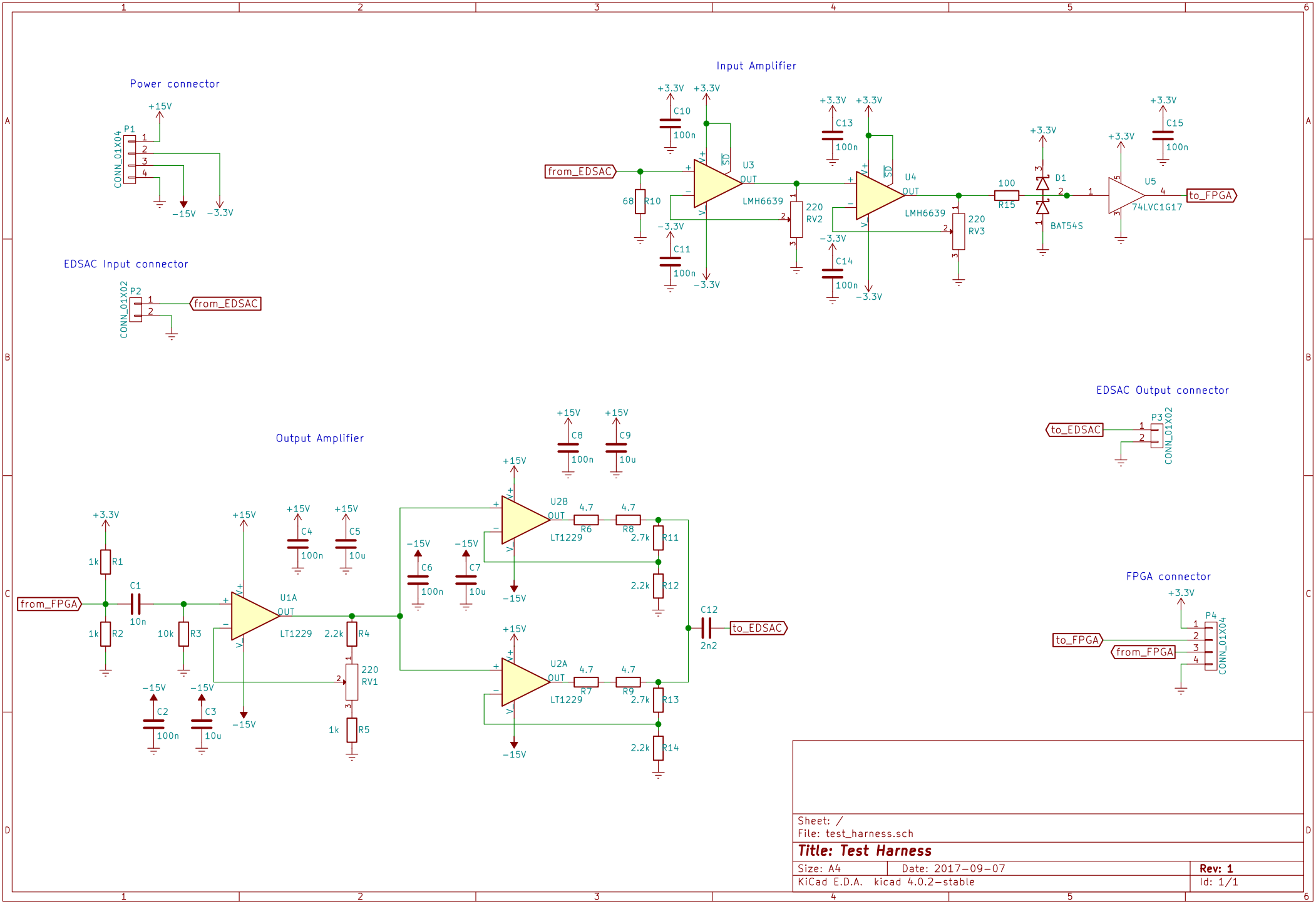



Figure B.2: Gantt Chart showing how time was allocated in the project

# Appendix C
# Schematics

This appendix includes the schematics for the delay line and test harness.


Power Rectification
from_EDSAC
1N4149
D1
10
R4
1m
L1
C5
10m
1k
R8
FPGA_pwr_in
FPGA development board interface
FPGA_pwr_in
P2
CONN_01X02
+3.3V
P4
CONN_01X03
FPGA_output
+3.3V
FPGA_input
P3
CONN_01X03
Input from EDSAC
P1
CONN_01X02
from_EDSAC
Pulse Filtering
from_EDSAC
2.2n
C1
68
R5
RV2
4.7k
+3.3V
BAT54
D2
+3.3V
C7
100n
U3
74LVC1G17
FPGA_input
N.B. These represent smaller portions of larger connectors.
Pins not listed are not connected.
Output amplifier
FPGA_output
1k
R1
1k
R2
1k
R3
+3.3V
U1
LMH6639
C2
680n
C3
10u
RV1
220
C4
100n
10k
R6
10k
R7
U2
LMH6639
C6
100n
68
R9
C8
100n
to_EDSAC
Output to EDSAC
to_EDSAC
P5
CONN_01X02
Sheet: /
File: delay_line.sch
Title: Delay Line Schematic
Size: A4
Date: 2017-09-07
Rev: 1
KiCad E.D.A. kicad 4.0.2-stable
Id: 1/1




| | | |
|---|---|---|
| Sheet: / | | |
| File: test_harness.sch | | |
| **Title: Test Harness** | | |
| Size: A4 | Date: 2017-09-07 | **Rev: 1** |
| KiCad E.D.A. kicad 4.0.2-stable | | Id: 1/1 |

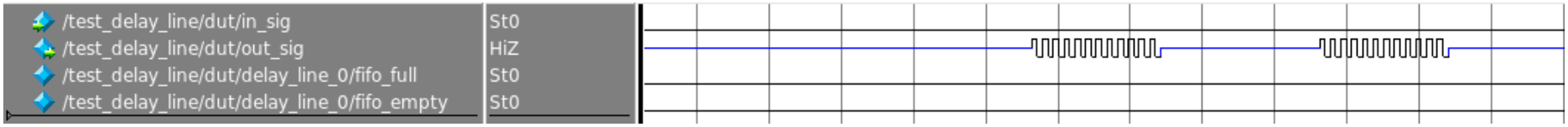
# Appendix D

# HDL verification results

This appendix includes the the simulation waveforms resulting from verification.

The waveforms are included here since they are more clearly presented in landscape format, however they are discussed in Section 7.2.


(a) Full Simulation




(b) Simulated output of a single pulse

Figure D.1: Delay Line Simulation Results

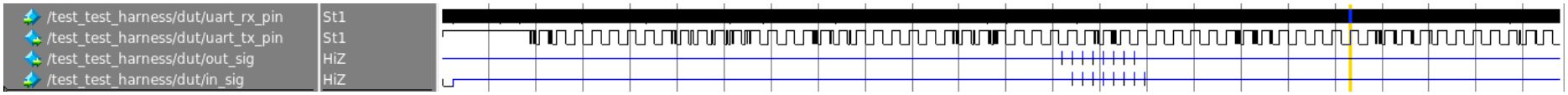



Figure D.2: Full Simulation