Enduring the Void and Inferno: Computing for Hostile Environments

Christian Garry supervised by Professor Alton Horsfall

***Abstract*—The aim of this paper was to investigate the use of Silicon Carbide Junction Field Effect Transistors to produce radiation resistant computers that can operate at temperatures exceeding 500 °C. This is the first demonstration of a full 4-bit microprocessor in Silicon Carbide. The architecture was based on the Intel 4004. The device was designed to be as close to the original Intel 4004 in functionality as possible and use the same instruction set. The aim was for programs written for the Intel 4004 to be able to run without changes to the code. The device was simulated using LTSpice along with other supporting software. The simulated device was able to run complex programs such as multiplying two 16-bit floating point numbers. Physical logic gates were built following circuit diagrams from this report and compared to the simulated results with good agreement in overall waveform shape leading to confidence in the simulated results produced by the CPU as a whole.**

***Index Terms*—Silicon Carbide (SiC), Junction Field Effect Transistors (JFETs), Radiation-Resistant Computers, High-Temperature Electronics, Intel 4004 Architecture, Circuit Simulation, Logic Gates**

I. Introduction

THE Intel 4004 microprocessor, recognised as the first general-purpose programmable processor, marked a significant milestone in the development of modern computing. This CPU is characterised by its simplicity: it features sixteen 4-bit working memory locations, known as the Scratch Pad, and two arithmetic logic unit (ALU) registers, the Accumulator and Temporary Registers. The 4004 has a limited instruction set of only 46 instructions [1]. The ALU can only perform addition and bit shifts left or right, subtraction is performed by taking the 2's complement of the temporary register before it is added to the accumulator. The 4004 has a 3-level stack allowing subroutines up to 3 layers deep. The 4004 also has some conditional jump logic allowing for loops and some basic iterative routines. The program counter is 12 bits, the first 4 indicate the read only memory (ROM) chip (Intel 4001) that has been selected and the next 8 bits indicate the address in that specific chip. This means that each 4001 is 256x8 bits [2] and the 4004 can natively address 16 distinct ROMs. The 4004 also has 4 memory control (CM-RAM) pins which are used to address a specific random-access memory (RAM) chip (Intel 4002). Unlike the ROM, each CM-RAM pin is dedicated to one RAM chip as the RAM only has 1 CM-RAM pin [2]. This can be increased by using a 4-to-16 decoder between the 4004 and RAM chips, allowing full use of the address space.

For devices to operate at temperatures greater than 400 °C a larger bandgap is needed compared to room temperature devices. Standard devices use Silicon which has a bandgap of 1.12 eV [3]. Silicon Carbide (SiC) is being investigated as it has a much larger bandgap than Silicon, 6H-SiC has a bandgap of 2.9 eV and 4H-SiC has a bandgap of 3.26 eV [4]. A further strength of Silicon Carbide as a material compared to Silicon is its increased radiation-hardness. SiC has a high threshold energy for atomic displacement (24 eV for C and 66 eV for Si [5], compared to 21 eV in Si [6] and 50 eV for diamond [7]). The charge collection efficiency (CCE) of n-type SiC Schottky barrier diodes has been measured and shows no deviation from linear behavior if irradiated with protons [8], electrons [9], or other light ions [10]. This makes SiC eminently suitable for a variety of applications such as fusion reactors or for off-world environments.

JFETs are being investigated since they use a buried p-n junction and not a Schottky barrier. This is important because a metal/semiconductor interface, as found in metal semiconductor field effect transistors (MESFET), degrades at elevated temperatures making them unsuitable for extreme conditions [11]. SiC JFETs have been shown to operate at above 600 °C [12] with the main degradation being interdiffusion of the metal ohmic contacts. These JFETs use a buried gate structure, however this leads to excess access resistance and gate capacitance which limits the applications in several ways. This means that brand new logic gate designs must be created to utilise these JFETs optimally. Logic gates built from SiC JFET devices have been shown to operate at 600 °C. The limits of these devices currently are not the wafer itself but the ohmic contacts which are subject to degradation [13]. It is difficult to test improvements as the probes used to electrically test the devices are themselves degrading at these temperatures. Complimentary JFET (CJFET) logic gates have been tested at 623 K [14]. The graphs show degradation of the output voltage at higher temperatures and that the output voltage swing is less than the input voltage swing. This has the potential to cascade when used as an input to a further logic gate, until the output voltage is not recognised reliably as a high or low signal. CJFET 4H-SiC logic gates have been simulated [15]. The data from these simulations indicates that the noise threshold increases as temperature increases, however, the propagation delay is much lower at higher temperatures, 50 ns compared to 10 ns at 1.75 V supply voltage for 27 °C and 500 °C respectively. The data also indicates that the maximum switching frequency increases as temperature increases going from 2 MHz to 10 MHz. However, the logic gates in this paper utilise resistor-transistor logic (RTL) rather than CJFET logic. This is because CJFET logic requires both p-type and n-type transistors in order to build gates. Due to only using n-type JFETs, RTL had to be used over the more commonly used complimentary logic. This does however suit the project well

as the original Intel 4004 also used RTL.

This paper describes the most complex digital circuit built from SiC JFETs and is organized as follows. Section II, describes the basic information of the logic primitives and the decomposition of the 4004 into smaller sub-systems. Section III, describes the design method used to build the circuits for these sub-systems and the testing process used to validate them. The results of the simulated programs and physical logic gates built in the lab are presented in Sec. IV. Finally, Sec.V concludes this paper with a discussion of key findings and directions for future research.

II. Theory

A. Logic Primitives

The LTSpice model of the JFET was used to create a logic primitive. This was extended to form more complex Logic Gates. Applying a voltage to the Gate of a JFET causes the depletion region to increase. This narrows the conducting channel increasing the resistance, which decreases the current that can pass through it. If the source of the JFET is connected to a voltage source through a resistor there will then be a change in voltage at the JFET's source when a voltage is applied to the Gate. This does not provide a large enough voltage swing between high and low and has implications for the fan out of the gate. Therefore, a voltage follower was used which works as a voltage level shifter and shifts the output voltage back to input levels of the logic gate (High: -0.8 V, Low: -3.6 V). These values were selected as they kept the output voltage from decaying or oscillating when used as inputs for subsequent gates whilst using standard resistor values.

The design of more complex gates follows on logically from this (Figure 1). To design the NOR gate a second JFET can be connected to the voltage follower in parallel with the first JFET which provides two inputs both with equal function. This means that if either JFET has a voltage applied, the output of the logic gate is the same. The NAND gate was developed using similar reasoning. If both inputs are required to be on to turn the output off, connecting two JFETs in series to the voltage follower will require that both JFETs are sinking current for the output to change. Condensed boolean gates are gates that have an entire Boolean expression compressed into a single gate in order to optimise the number of transistors

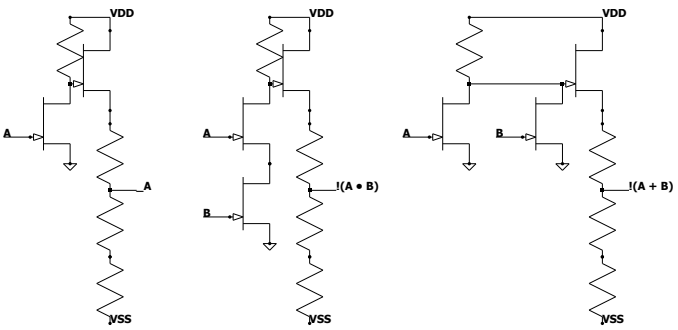


Fig. 1: Circuit diagrams of NOT, NAND, and NOR gates respectively

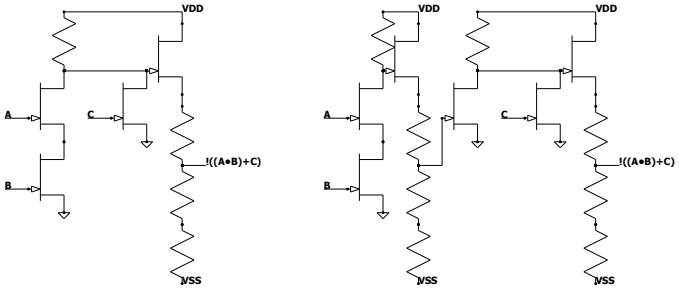


Fig. 2: Circuit diagram of a condensed boolean gate (Left) and a combination of base gates (Right), illustrating the transistor reduction for the boolean equation (A AND B) NOR C

used (Figure 2). For example having a NOR gate but with one of the NOR input JFETs having a second JFET between it and ground provides the same effect as having (A AND B) NOR C whilst saving two JFETs. This was common in integrated circuits in the past [16].

NAND and NOR gates were built in the lab and compared to the simulated devices. They were characterised from 1 Hz up to a maximum of 1 MHz. The results were found to mostly agree with the simulations, showing some differences in voltage levels. These results will be discussed further in Sec. IV.

B. CPU Design

The Intel 4004 can be decomposed into several distinct sub-systems. The arithmetic logic unit (ALU), the instruction register and decoding, the stack and program counter (PC), the Scratch Pad, and the circuits related to the pins and interfacing with other external components. The specific instructions relating to each sub-system were evaluated in detail to precisely match the functionality of the original Intel 4004.

1) ALU: From the architecture diagram (Figure 3) it can be seen that the ALU (Figure 4) has two 4-bit registers connected to it, the Accumulator (Acc) and the Temporary (Temp) register. From the instruction set it can be seen that the ALU consists only of a 4-bit adder. The contents of the accumulator can also be shifted to the left or right by one bit, however it also includes the carry flag in this shift operation. For example, shifting right would place the least significant bit of the accumulator into the carry and what was in the carry into the most significant bit of the accumulator. Based on the instruction set it can be deduced that there is no inverter on the accumulator and that for subtraction and the other inversion instructions, for example complement accumulator (CMA), there must be an inverter between the temporary register and its connections to the bus and adder. CMA is then accomplished by transferring the value from the Accumulator to the Temp register, inverting the Temp register and transferring this inverted value back to the Accumulator. The carry flag also needs to be able to be inverted. There is a separate incrementer circuit which adds one to the temporary register. This is because the instruction Increment (INC), increments the value stored in a specific Scratch Pad register without effecting the accumulator or the carry flag of the

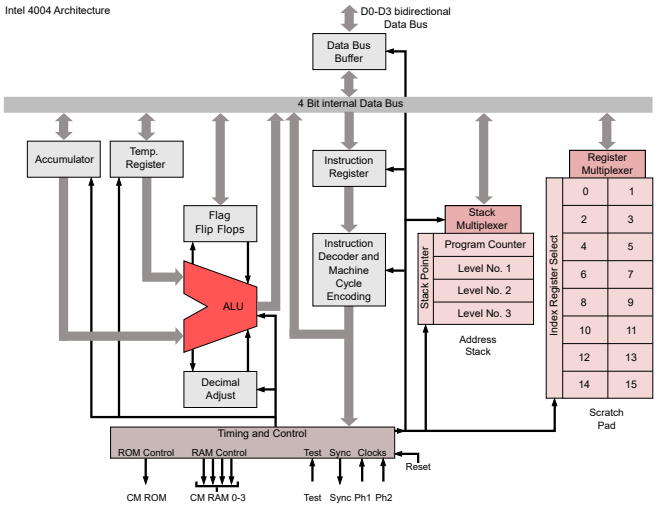


Fig. 3: Architecture of the Intel 4004 [17] showing the layout of registers and connections between CPU sub-systems

ALU. There is not enough time in the execute portion of the instruction cycle for the accumulator value to be moved to some intermediate location to be moved back after the increment instruction is completed. This means that there has to be a separate circuit for incrementing Scratch Pad registers.

There was a race condition caused by the carry and accumulator not saving their new values at the same time. This lead to a scenario where the sum was saved to the accumulator, passed through the adder again and updated the carry before the signal to save to the carry went low. This resulted in uncertainty as to whether or not the value saved in the carry register was that of the original sum operation or the carry from the sum output added to the temporary register again. This issue also occurred if the carry was saved first leading to the sum output being unreliable. The solution to this was to have two stages of carry flag, the first flip-flop would be loaded to store the new carry value without effecting the value seen on the input of the adder. Then the sum could be stored to the accumulator and the second stage of the carry flag could be loaded from the first stage.

2) Instructions: The instruction register of the 4004 is 16-bits (Figure 5). Instructions can be either one or two words, with each word being 8-bits. The first 4-bits of any instruction are the opcode which specifies the instruction that is being performed. Some opcodes require the second 4-bits to distinguish them from other opcodes. For example the difference between jump indirect (JIN) and fetch indirect (FIN) is that the least significant bit of the lower first word is a 0 for FIN and 1 for JIN (FIN: 0011 RRR0, JIN: 0011 RRR1, where R represents the address of a Scratch Pad register pair). The rest of the instruction is an operand which specifies either the data or the location for the instruction to act upon. The most significant 8-bits must therefore be de-multiplexed to create a control signal telling the CPU which instruction is being executed. The CPU executes instructions by activating sub-system control lines in a specific order. This means the CPU needs to know both what step it is on in the instruction and what control lines need to be activated for each step. The list of control lines that need to be activated on each clock cycle are known as micro-instructions (Figure 6). This is done using a micro-instruction step counter (Figure 7) and an AND gate for each step of each instruction. These will only have a logical high output when the CPU is executing that particular instruction and is on that specific step, allowing the CPU to execute the micro-instructions in the correct sequence to correctly perform the functionality required by the instruction. The micro-instruction steps are broken down into distinct phases. A_1, A_2, and A_3 are the first step of the fetch portion of the fetch, decode, execute cycle. This is when the address is sent from the CPU onto the external bus for any chips connected to it to read. The order in which the address is sent out is important as the ROM and RAM will expect a certain format for the address. The middle order 4-bits are sent out at time A_1, the low order 4-bits are sent out at time A_2 and the high order 4-bits are sent out at A_3. The next step of the fetch portion is M_1 and M_2 where the CPU receives back 8-bits of data from the ROM.

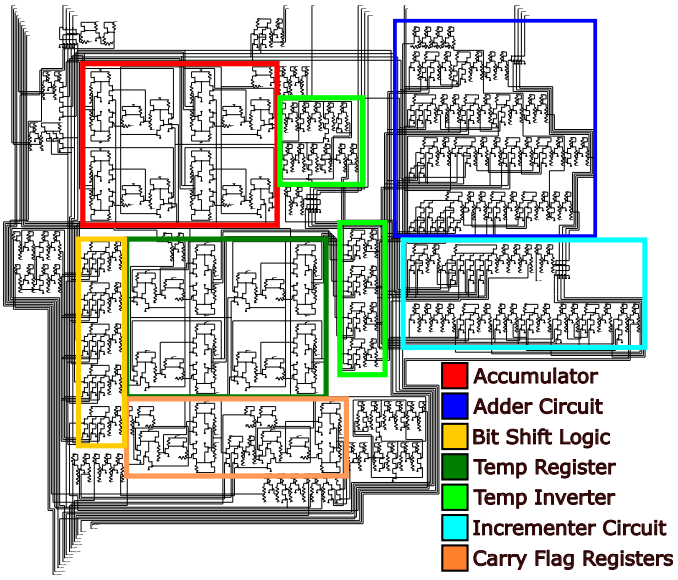


Fig. 4: Circuit diagram for the Arithmetic and Logic Unit with colour-coded sub-circuits

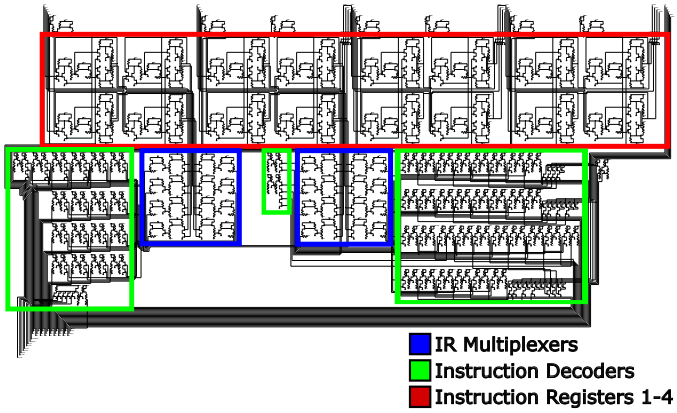


Fig. 5: Circuit diagram for the Instruction Register with colour-coded sub-circuits

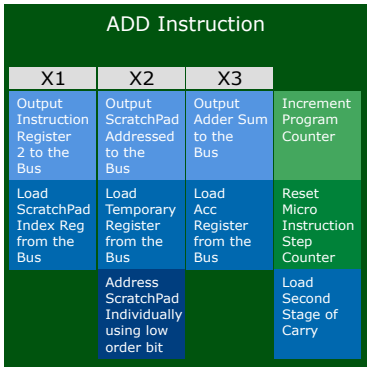


Fig. 6: Diagram showing the control lines that need to be activated per step to execute the ADD instruction

The Intel 4001, in order to decrease complexity and therefore transistor count, has no way to store 16-bit instructions in one ROM address. After the first 8-bits are received from the ROM, if an instruction is detected that requires a further 8-bits of data there is a flip-flop on the CPU that is toggled and the program counter is incremented. The flip-flop being toggled means that when the CPU goes through the next fetch cycle ($A_{1,2,3}, M_{1,2}$) it will place the new values into the lower word of the instruction register. The CPU then has all 16-bits stored in the instruction register ready to execute. The execute portion is denoted as X_1, X_2, and X_3. This is where the instruction is executed by manipulating the control lines of the CPU.

3) Program Counter and Stack: The Program Counter (PC) (Figure 8) consists of a 12-bit register and an incrementing circuit. The PC also needs to be able to write values to and from the Bus. The conditional jump (JCN) instruction indicates that each 4-bits in the PC need to be individually controlled as the high order 4-bits are not changed when JCN is performed. The limit of three micro-instruction steps for executing instructions means that the stack (Figure 9) has to have a direct connection to the PC as all 12-bits need to be written, along with additional steps related to the stack level control, within three clock pulses. This is not possible with the 4-bit bus unless the PC and Stack have a direct connection. The stack is three layers deep and needs to keep track of which level it is on. This is done using a shift register which is able to be shifted up and down to select each level. The output of each register of the stack is fed into an AND gate along with the level control line that corresponds to it so that the register only outputs when its level is selected and the "Output Stack" control line is enabled.

4) Scratch Pad: The 4004 has sixteen 4-bit registers it can use as general purpose memory. This is so that it is able to perform basic functions without the need for any external

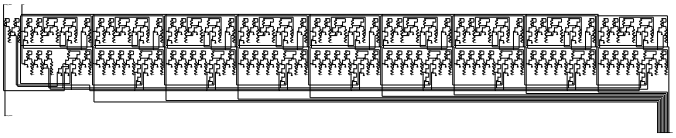


Fig. 7: Circuit diagram for the Micro-Instruction step counter, composed of a series of 1-bit registers

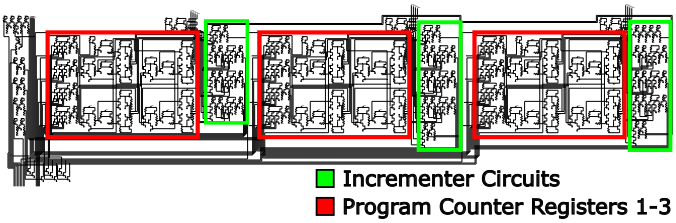


Fig. 8: Circuit diagram for the Program Counter with colour-coded sub-circuits

RAM chips. It is known as the Scratch Pad (Figure 10). Each Scratch Pad location can be addressed individually or as a pair of registers for storing 8-bit values. The Scratch Pad has its own address register. When the Scratch Pad is in pair-mode the highest order 3-bits of the address are used to select a register pair. The instruction then places the value of each register of the pair onto the bus sequentially. When the Scratch Pad is being individually addressed the full address is looked at and the control signal for the correct register is generated. This is done with a separate "Scratch Pad Register Select" control line which is fed into an AND gate with the least significant bit from the Scratch Pad address register. This then generates a "Word Select" line which in turn activates either the first or the second word of the specified pair of registers.

5) Bus: The Bus may seem trivial, however, the normal method for bus connections is to use tri-state logic. This is where there are three states: logic high, logic low and high impedance (disconnected). This was difficult to implement and the approach that was used was to treat each input to the bus as if it were passing through a multi-input OR gate. If any sub-system places a 1 onto the bus the bus will reflect a 1 otherwise it will stay 0. Due to the design of the voltage level shifters, if this multi-input OR gate approach was not used the resulting values would be some intermediate value between high and low. The precise value depending on the number of connections to that specific bus. The connection from the

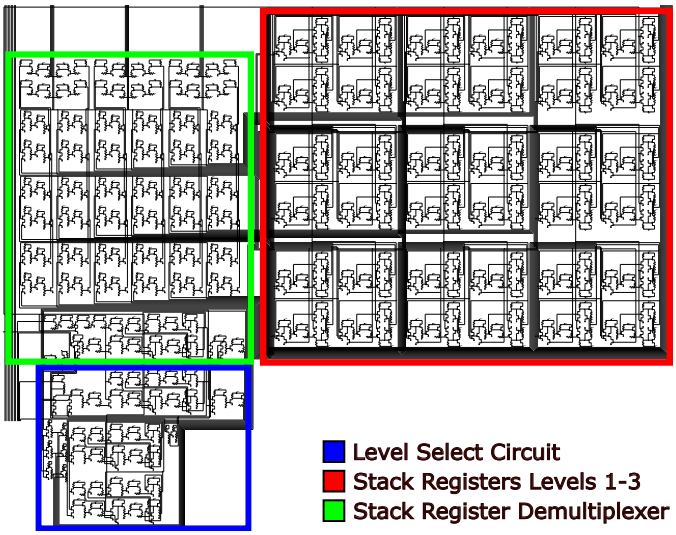


Fig. 9: Circuit diagram for the Stack with colour-coded sub-circuits

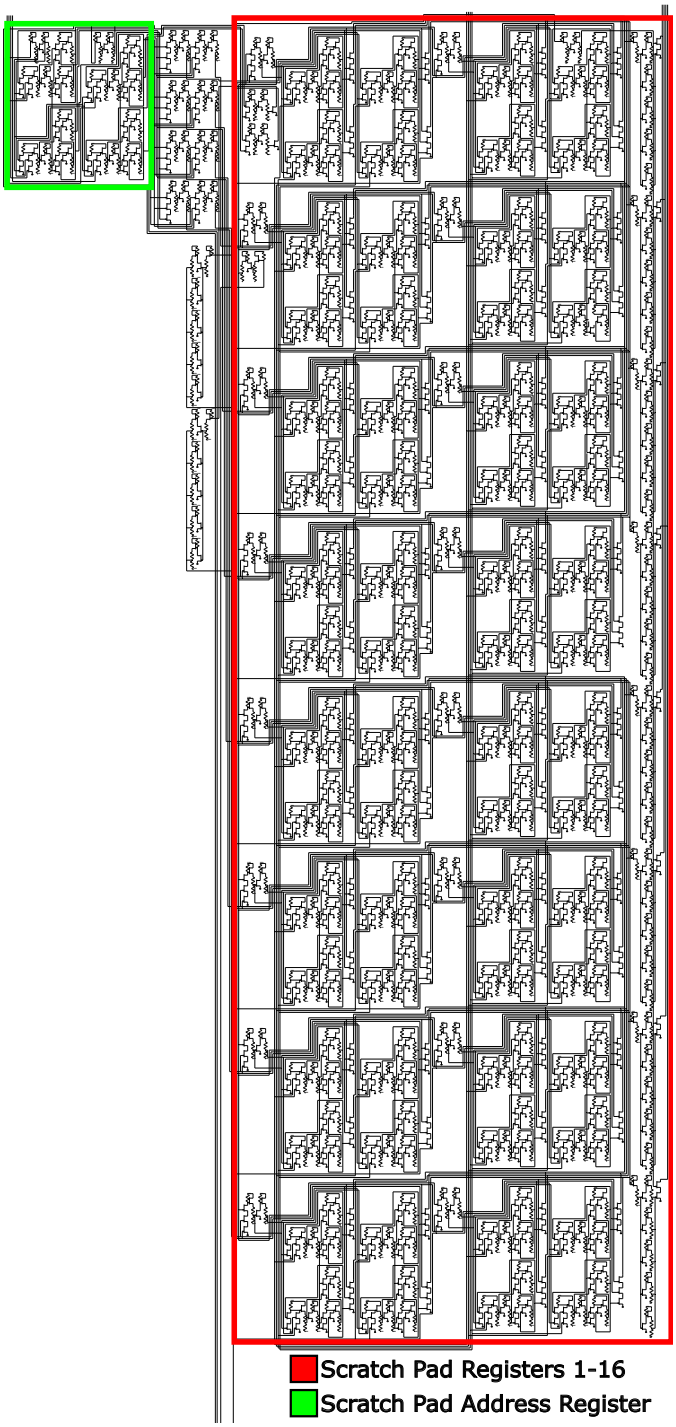


Fig. 10: Circuit diagram for the Scratch Pad with colour-coded sub-circuits

Bus to the input of each circuit works as a normal bus would because these connections are always connected to the Gate of a JFET which means the voltage combining described above does not occur.

6) Pins and Communication: The 4004 has a number of pins used to interface with external components (Figure 11). There are four data pins which are used to transfer opcodes and data between the CPU and the external bus ($D_{0,1,2,3}$). There are four CM-RAM pins which are used to address

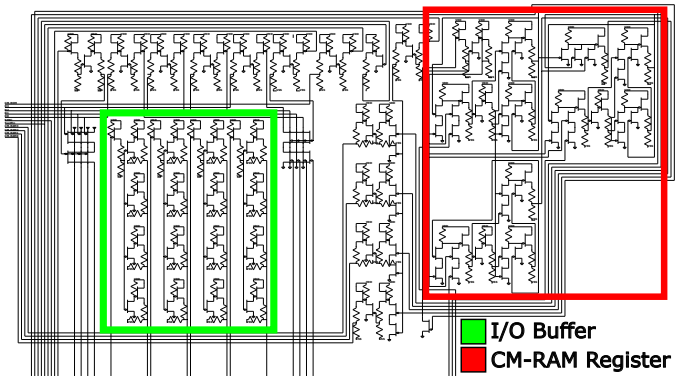


Fig. 11: Circuit diagram for the pins and associated logic with colour-coded sub-circuits

the RAM chips ($CM-RAM0,1,2,3$). When the designate command line (DCL) instruction is executed the value that is in the accumulator is loaded into the CM-RAM register. These pins are "1 Hot" due to the fact that each RAM chip only has one CM-RAM input pin. However, the user is able to put an external de-multiplexer in between the CM-RAM pins and the input of the RAM chip to allow for the full address space to be used rather than just the maximum of four chips as default. The CM-ROM pin is used to communicate with the ROM and the RAM. This pin will go high at specific points in the instruction cycle to indicate various things such as a ROM/RAM instruction. Finally the SYNCH pin which is used to synchronise the operations of all chips in a larger circuit.

III. Method

The first step was to validate simple logic circuits built in LTSpice using the logic primitives. These could then be connected to form the larger sub-circuits and blocks of the CPU. The first of these simple logic circuits was the 1-bit synchronous D-type flip-flop. The synchronous part of this simply means that it only latches the D value in on the clock pulse. First the flip-flop was built using discrete logic gates, however, it was noticed that some gates could be reduced using condensed boolean gates as described in the logic primitives section (Section II-A). This was done until the circuit could not be condensed any further due to race conditions and the path for individual signals becoming unbalanced compared to other paths through the circuit. This unbalancing occurred due to the layout of the logic circuit for a D-type flip-flop. The output is taken from a pair of cross-connected gates meaning that if one gate was reached by the signal significantly before the other gate, the output would not latch correctly (Figure 12). D Flip-Flops were used for storage in this CPU as 6 Transistor (6T) Static RAM (SRAM) and 1 Transistor (1T) Dynamic RAM (DRAM) proved challenging to implement. There are 146 bits of storage on the 4004 which, using the current D-Flip-Flop storage, requires 2190 transistors. Development of 1-T DRAM would reduce the total transistor count by 2044 excluding the RAM refresh circuit. Storage technology should be the highest priority for further development in this field. Each sub-system of the CPU was built separately at first in

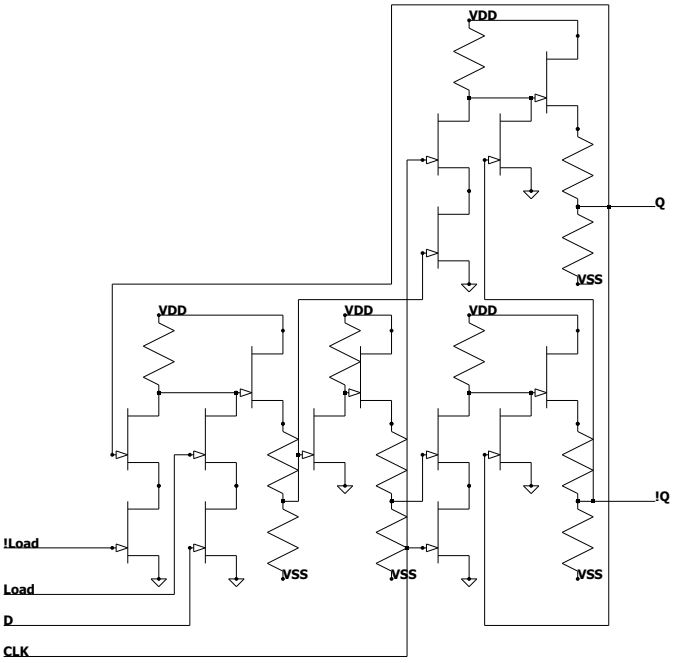


Fig. 12: Circuit diagram for synchronous D-type flip-flop with transistor reductions using combined boolean logic gates

order to decrease simulation times. The system was tested using the following method. Each control line was connected to a voltage source utilising a Piecewise linear (PWL) file. A PWL file is a type of file used by LTSpice which specifies the voltage at specific time points. Each control line was listed in a spreadsheet along with register values if appropriate. The necessary voltage level for each control line was then plotted on a timeline to "program" the circuit to perform operations. The PWL files were then generated using Matlab from the list of times the control line needed to be high. The final state of the sub-system could then be compared to the expected state calculated from the spreadsheet. Once each sub-system had been verified as correctly functioning they were assembled into the full CPU by means of a Python script which collated each sub-system and renamed any duplicated component names. This was then tested by connecting the Bus to voltage sources and emulating the ROM by feeding it data when it expected to receive data from the ROM. This allowed the majority of the testing to be completed but did not allow for testing more complex programs or anything that required a "decision" to be made by the CPU as every bit of data received by the CPU had to be hard-coded beforehand.

A. ROM

In order to fully test the capabilities of this CPU it was decided that a ROM needed to be designed and built (Figure 13). The 4001 functions in a similar way to the 4004. It uses a micro-instruction counter to see where it is in the instruction cycle and also has an instruction register which is used for some ROM specific instructions. The ROM receives an address from the CPU at times A_1 and A_2. It then receives a chip select address at time A_3, which determines which of the 4001 chips is being addressed, along with CM-ROM going high.

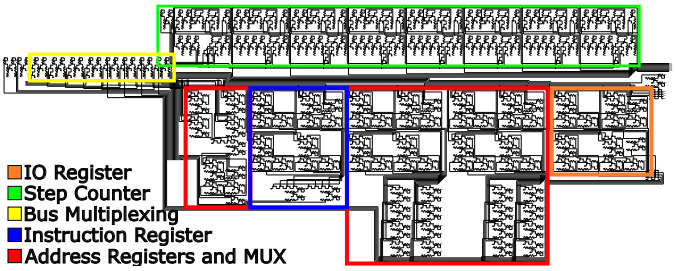


Fig. 13: Circuit diagram for computational part of Intel 4001 with colour-coded sub-systems

Each 4001 has a specific identity which is hard-wired during production. If the value that is wired into the 4001 matches the address received at A_3 the 4001 then outputs the values stored at the address received from $A_{1,2}$ during M_1 and M_2 otherwise it is deactivated until the next fetch cycle. If a ROM/RAM instruction is detected by the ROM, the instruction register is loaded from the external bus during X_1. The ROM has two instructions as implemented in this paper, write ROM port (WRR) and read ROM port (RDR) which are used to read and write to the I/O pins on the ROM chip. This is performed during X_2 unless no instruction was received during X_1 in which case the execution cycle is reset ready for the next address to be received. Storage in the ROM was implemented using a voltage source linked to a PWL file for each bit of every address. When the ROM is addressed the value defined by the PWL file for that address is sent out onto the bus. This was done so that each bit would have a specific PWL file which could be written to, allowing for the ROM to be programmed.

B. Assembler

An assembler converts the mnemonics of the instructions from text to binary and converts the operands from hexadecimal or decimal to binary. So FIM 1 0 11 would be converted to 0010 0010 0000 1011. An assembler also handles labels which are used for jumps. When the code comes across text that does not correspond to a known instruction it adds it to the dictionary along with the line that the assembler was on when the unknown text was encountered. Then, once the entire program has been read for a first time, the code makes a second pass. During this second pass every instance of a label is replaced with the binary value of its location in the program, essentially acting as pointers to a specific line in the program. The assembled program is saved as a .bin file and a .hex file. The .bin file is formatted so that each line has 8-bits, which corresponds to the 8-bits that the ROM stores per address. Each line then refers to a specific address in the ROM. The assembler was written in Python.

C. PWL File Generation

The .bin file is used by another python script to generate the PWL files that populate the voltage sources of the ROM. The first bit in the .bin file is read, if it is a 1 the PWL file will have a logical high voltage, if it is a 0 it will have a

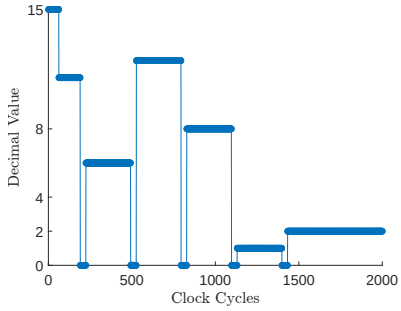
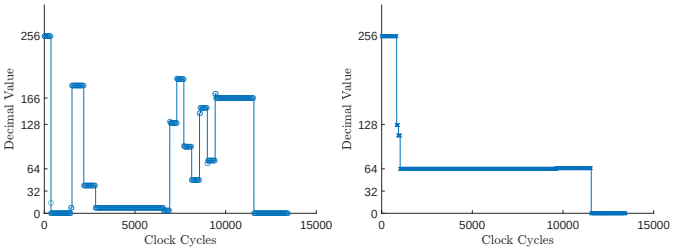


Fig. 14: Figure showing the decimal value in Scratch Pad Register 0 for the operation 11 & 6. This illustrates the first input (11) being shifted left 4 times to compare the highest order bit to the second input (6) with the final result of 2 being loaded in just before 1500 clock cycles



(a) Significand which is calculated to be 166 in decimal

(b) Exponent which is calculated to be 65 in decimal

Fig. 15: The waveforms of the registers during the computation of the product of 1.4375×3.60937. Plots 15a and 15b show the values of the significand and exponent respectively during the operation

logical low output. The first bit will be saved as 0.txt, bit two will be 1.txt etc. The format of these files is as follows, for logical high: "0u -800m", and for logical low: "0u -3.6". This sets the voltage source to a value of either -0.8 V or -3.6 V permanently unless the text file is changed.

D. Testing and Benchmarking

With a full 4001 and 4004, more complex programs could be run and the outputs of these programs compared between the LTSpice simulated CPU and the CPU that was emulated using JavaScript. One of the programs built was a floating point emulator which allowed the multiplication of a custom implementation of 16-bit floats (Listing 3). This was chosen to allow for astronomical calculations which require a large range of values. Another program that was run was pulled from a contemporary Intel 4004 programming manual [1] and was used for the purpose of performing a logical AND operation. This was able to run without any modifications which is a good indication that the high-level functionality has parity with the original CPU (Figure 14). The results of these computations are discussed further in Section IV.

1) Waveform Value Interpreter: In order to streamline the process of examining the waveform data of the LTSpice simulations on lengthy computations, software was written which extracts the voltage values and processes them. Matlab was used to read through the entire waveform file and search for specific nodes. These nodes were then converted from analogue to digital based on if the voltage was above or below -2.6 V. This was done looking at the average voltage in 5 μs time steps as this is the length of time that the clock is high at 100 kHz. These nodes were then combined into 4-bit numbers, if appropriate, giving each register a 4-bit binary number for each time step.

2) Emulator: An online emulator [18] was referenced during the development of the CPU where quirks such as the incrementer not effecting the accumulator were spotted. However, this emulator had some oversights which were corrected and a version of the emulator which more closely matched the internal architecture of the CPU built in LTSpice was created using HTML and JavaScript. Most of the issues were related to the decoding of instructions and specific jump conditions that were improperly implemented. A second version of this site was made where the JavaScript for the emulation was removed and the interface was filled with values from the Matlab Waveform Value Interpreter script. This allowed the Emulator and the LTspice model to be compared at every register, step by step for long computations to ensure operational parity.

E. Experimental Validation of Simulation Models

The first two logic gates built using physical JFETs were the NAND and the NOR gate. These were built using a JFET integrated circuit (IC) and a copper circuit board with through-hole components. These were connected to a power supply, a function generator, and an oscilloscope. They were then tested at a range of frequencies from 1 Hz to 1 MHz. The data from this experiment was then used to characterise the gates. The inputs from the function generator were taken and a Matlab script was written to convert these into a PWL file which could then be used as the inputs for the simulated gates to get an exact comparison between experimental and theoretical.

IV. Results and Discussion

A. Program Testing

1) AND Subroutine: From the MCS4 programming manual an AND sub-routine (Listing 4) was run on the CPU and its results compared to the emulator. Figure 14 shows the functioning of the AND sub-routine. The input A, 1011 (11), is loaded into Scratch Pad register 0 and subsequently shifted left to extract the highest order bit into the carry. This is done four times, from 1011 (11) to 0110 (6) to 1100 (12) to 1000 (8) and the final output value, 0010 (2), is loaded in after roughly 1400 clock cycles. The value starts at 15 as each register defaults to being high when the CPU is first initialised. From the plots we can see that 6 (0110) AND 11 (1011) gives an output of 2 (0010) which is correct.

2) Floating Point Multiplication Subroutine: The floating point system implemented takes the following form.

$$-1 \times \text{signbit} \times 1.sssssss \times 2^{eeeeeee-63} \tag{1}$$

Where s is a bit of the significand and e is a bit of the exponent. The power of the CPU is demonstrated in the following plots showing the computation $1.4375 \times 3.609375 = 5.18847$.

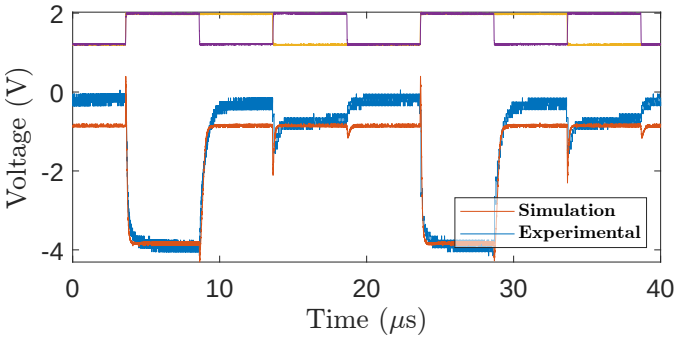


Fig. 16: Experimental and simulation output voltages of a NAND gate at 100 kHz. This is the operational frequency of the CPU as simulated in this paper. The logic gate input voltages have been scaled and offset to the top of the diagram.

Due to limitations in the accuracy of the floating point representation of the numbers they are rounded slightly from their exact values. This is shown in Table I, where the value for the product is rounded from 5.18847 to 5.1875. The majority of the rounding in the multiplication comes from the floating point implementation itself as effort was put into retaining as much accuracy as possible in the algorithm. Explanations of the algorithm can be found in Appendix B. The final values in the registers in Figures 15a and 15b being zero is due to an LTSpice simulation error which occurred after the program had finished running.

B. Experimental Data

The logic gates were tested at a range of input frequencies up to 800 kHz and 1 MHz for the NOR and NAND gate respectively. The key results are at 100 kHz (Figures 16 and 17) and the maximum tested frequency (Figures 21 and 20). These results are important as the CPU was tested at 100 kHz allowing confidence in results obtained from the simulations. The data from the operation at higher frequencies allows predictions about the theoretical top speed of the CPU. The limiting factor is the NOR gate, after 800 kHz the swing of

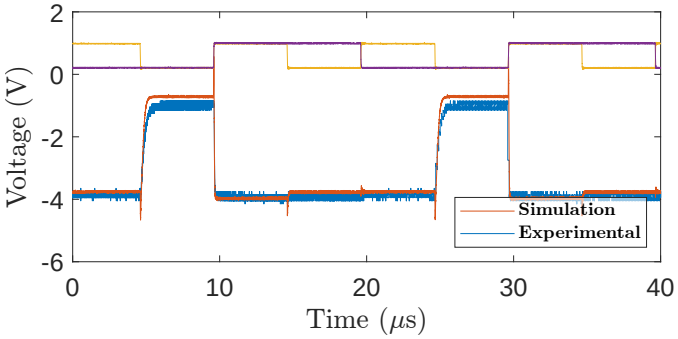


Fig. 17: Experimental and simulation output voltages of a NOR gate at 100 kHz. This is the operational frequency of the CPU as simulated in this paper. The logic gate input voltages have been scaled and offset to the top of the diagram.

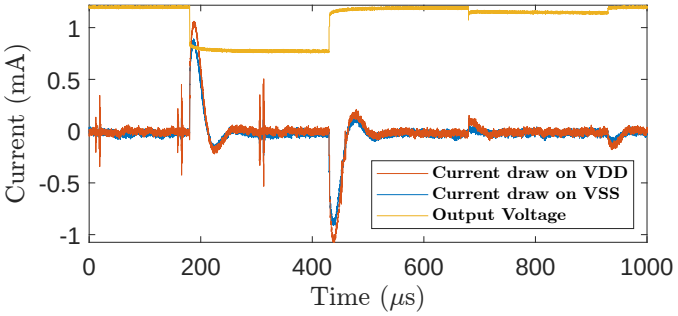


Fig. 18: Plot showing current draw for NAND gate at 2 kHz, the highest of the sampled frequencies that exhibits zero current draw unless the output state is transitioning. The gate output voltage has been scaled and offset to the top of the diagram.

TABLE I: Floating Point Number Values Used in Subroutine

Binary	Decimal	Exponential Form	Value
10111000 00111111	184 63	1.4375×2^0	1.4375
11100111 01000000	231 64	1.8046875×2^1	3.609375
10100110 01000001	166 65	1.296875×2^2	5.1875

TABLE II: Register values for Figures 15a and 15b

Clock Cycle	Significand	Exponent	Decimal
0	255	255	1.9921875×-2^{64}
370	0	255	0
828	0	127	0
924	0	112	0
1020	0	64	0
1527	184	64	2.875
2178	40	64	0.625
2830	8	64	0.125
6580	4	64	0.0625
6968	130	64	2.03125
7356	193	64	3.015625
7744	96	64	1.5
8128	48	64	0.75
8608	152	64	2.375
9039	76	64	1.1875
9472	166	64	2.59375
9640	166	65	5.1875

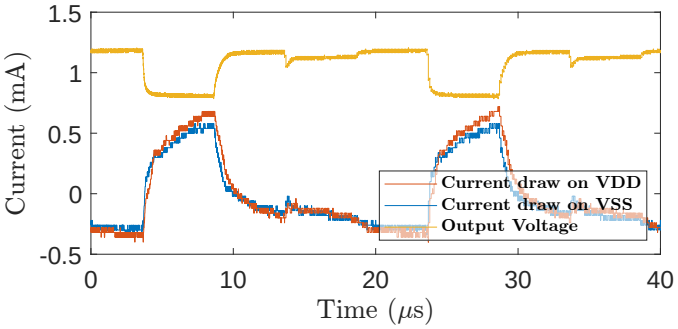


Fig. 19: Plot showing current draw for NAND gate at 100 kHz. This is the operational frequency of the CPU as simulated in this paper and shows that the device will always draw current regardless of input states at these frequencies. The gate output voltage has been scaled and offset to the top of the diagram.

the output voltage becomes too slow to react to the changing inputs. This means that a theoretical maximum operating frequency of the CPU can be set at 800 kHz. This is an 8% increase compared to the original 4004 (some sources have the clock speed of the original at 400 kHz which would make this CPU two times faster).

The oscilloscope probes introduced scaling by a factor of two, necessitating post-experimental adjustments. There was a 1.5 - 2 V baseline shift in the data compared to what was expected from the simulations, the origins of which are currently under investigation. Some of the difference was due to resistor inaccuracies, however, this cannot account for the entire 2 V shift. Further, it can be seen that as the frequency increases this offset increases, starting at approximately 1.5 V at 2 Hz going up to approximately 2 V at 100 kHz. This indicates that it is some function of the JFETs themselves. This change in output voltage levels as a function of frequency was only observed in the NAND gate. The NOR gate retained its 1.5 V shift for the entire range of examined frequencies. This could be due to differences in the JFETs used between the NAND and NOR gates rather than due to gate topology. Further investigation into the nature of these discrepancies is necessary to determine their source, be it manufacturing variability or the LTSpice model itself. The voltage levels and resistor values used should be explored in greater depth to determine optimum performance for specific applications.

The current draw demonstrated by the NAND gate at lower frequencies displays CMOS like qualities where it is only actively drawing current on the transition between output states as shown in Figure 18. This behaviour was not apparent at higher frequencies as shown in Figure 19. This is an unexpected result and should be investigated further. The current draw on each voltage source (positive supply voltage: 24 V, negative supply voltage: -20 V) was about 0.5 mA per logic gate which consisted of 3 transistors. Extrapolating from this, the peak current draw for the full CPU of 5300 transistors would be about 900 mA. Due to the CMOS like nature of the current draw at lower frequencies, if entire systems of the CPU such as the lower levels of the stack or some scratchpad registers are not being used, the current draw would be lower.

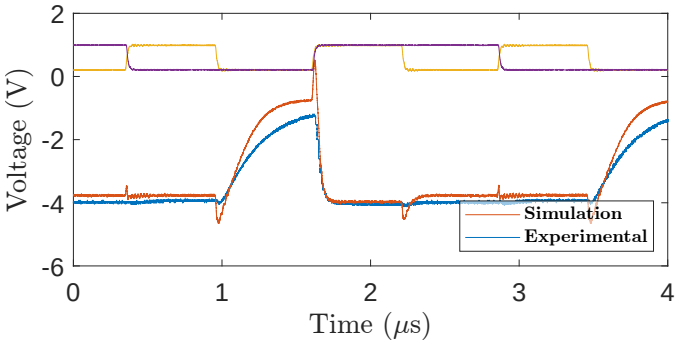


Fig. 20: Experimental and simulation output voltages of a NOR gate at 800 kHz, the maximum frequency that the NOR gate produced adequate output peaks. The gate input voltages have been scaled and offset to the top of the diagram.

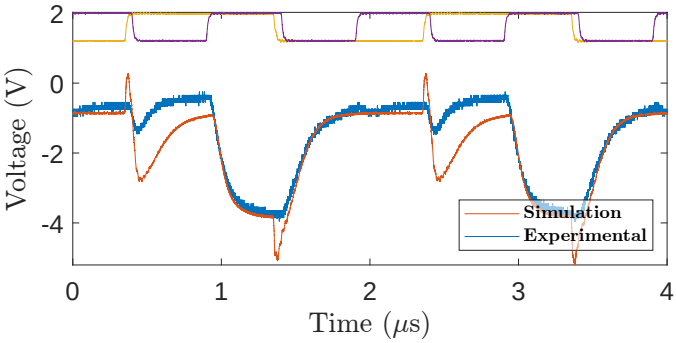


Fig. 21: Experimental and simulation output voltages of a NAND gate at 1 MHz, the maximum frequency that the NAND gate was tested at. The gate input voltages have been scaled and offset to the top of the diagram.

V. Conclusion

In this paper, the design and implementation of a Silicon Carbide (SiC) Junction Field Effect Transistor (JFET) based computing platform has been successfully demonstrated. The design process was comprehensive, encompassing the design of individual logic gates, the construction of a complete SiC JFET CPU and ROM, and the development of supporting software such as an assembler and waveform analyser. These tools were instrumental in validating the functionality of the SiC CPU against the Intel 4004. Thereby marking a significant step forward in the development of electronics capable of enduring the harsh conditions found in space exploration, nuclear reactors, and other extreme environments.

This CPU consists of 5328 transistors compared to 2300 in the original 4004 and achieves full operational parity at a frequency of 100 kHz. The transistor count increase is due to the original 4004 utilising DRAM which saves considerable transistors compared to the D Flip-Flop storage technology used in this paper. As CPU complexity increases, the number of individual registers also increases. Therefore, developing transistor optimised storage technology should be a high priority for any further research. Development of 1-T DRAM would reduce the total transistor count to 3284 excluding the RAM refresh circuit.

The functionality of the CPU was demonstrated by running code that was written for the Intel 4004. The specific code run was a floating point multiplication and a bitwise AND. The outputs of which were all validated against an emulator and found to be correct. The experimental data for the SiC JFET logic gates that were built, delivered promising experimental validation of the simulated models up to frequencies of 800 kHz for NOR gates and 1 MHz for NAND gates. This demonstrates that the LTSpice logic gate models are reliable and shows that the CPU should be able to be run at frequencies of at least up to 800 kHz which is higher than the original (400-740 kHz). This work however, also uncovered challenges, notably the baseline voltage shifts observed between experimental and simulated results for the logic gates. The cause of this needs further investigation as each logic gate was only tested using one JFET IC. This means manufacturing variability could play some role in the differences observed.

As the boundaries of where and how computing can be applied continue to be pushed, the lessons learned and the groundwork laid by this project will undoubtedly contribute to the development of more resilient, efficient, and powerful computing platforms for use in the most challenging environments imaginable.

VI. Acknowledgements

I would like to thank Professor Alton Horsfall for encouraging and inspiring me when I felt like I had been making no progress. I have appreciated his invaluable guidance and constructive feedback. Without him this report would be a lot shorter.

References

[1] Intel Corporation, *Mcs-4 micro computer set user's manual*, 3065 Bowers Avenue, Santa Clara, California 95051: Intel Corporation, Feb. 1973. [Online]. Available: https://archive.org/details/manualzilla-id-7026262 (visited on 08/01/2023).

[2] Intel Corporation, *Intel 4004 family components specifications*. [Online]. Available: https : / / deramp . com / downloads / mfe _ archive / 050 - Component % 20Specifications / Intel / Microprocessors % 20and % 20Support/4004%20Family/ (visited on 09/01/2023).

[3] J. Low, M. Kreider, D. Pulsifer, A. Jones, and T. Gilani, "Band gap energy in silicon," *American Journal of Undergraduate Research*, vol. 7, Jun. 2008. DOI: https://doi.org/10.33697/ajur.2008.010.

[4] X. Wang, J. Zhao, Z. Xu, *et al.*, "Density functional theory calculation of the properties of carbon vacancy defects in silicon carbide," *Nanotechnology and Precision Engineering*, vol. 3, Dec. 2020. DOI: https://doi.org/10.1016/j.npe.2020.11.002.

[5] W. Li, L. Wang, L. Bian, *et al.*, "Threshold displacement energies and displacement cascades in 4H-SiC: Molecular dynamic simulations," *AIP Advances*, vol. 9, no. 5, p. 055 007, May 2019, ISSN: 2158-3226. DOI: https://doi.org/10.1063/1.5093576.

[6] S. Messenger, E. Burke, M. Xapsos, G. Summers, and R. Walters, "The simulation of damage tracks in silicon," *IEEE Transactions on Nuclear Science*, vol. 51, no. 5, pp. 2846–2850, 2004. DOI: https://doi.org/10.1109/TNS.2004.835094.

[7] S. Ditalia Tchernij, N. Skukan, F. Picollo, *et al.*, "Electrical characterization of a graphite-diamond-graphite junction fabricated by mev carbon implantation," *Diamond and Related Materials*, vol. 74, pp. 125–131, 2017, ISSN: 0925-9635. DOI: https://doi.org/10.1016/j.diamond.2017.02.019.

[8] F. Nava, E. Vittone, P. Vanni, *et al.*, "Radiation tolerance of epitaxial silicon carbide detectors for electrons, protons and gamma-rays," *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 505, no. 3, pp. 645–655, 2003, Cited by: 105; All Open Access, Green Open Access. DOI: 10.1016/S0168-9002(02)01558-9.

[9] N. Iwamoto, B. Johnson, N. Hoshino, *et al.*, "Defect-induced performance degradation of 4h-sic schottky barrier diode particle detectors," *Journal of Applied Physics*, vol. 113, no. 14, 2013, Cited by: 35; All Open Access, Bronze Open Access. DOI: https://doi.org/10.1063/1.4801797.

[10] K. Lee, T. Ohshima, A. Saint, T. Kamiya, D. Jamieson, and H. Itoh, "A comparative study of the radiation hardness of silicon carbide using light ions," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 210, pp. 489–494, 2003, 8th International Conference of Nuclear Microprobe Technology and Applications, ISSN: 0168-583X. DOI: https : / / doi . org / 10 . 1016 / S0168 - 583X(03)01096-6.

[11] J. Zolper, "A review of junction field effect transistors for high-temperature and high-power electronics," *Solid-State Electronics*, vol. 42, no. 12, pp. 2153–2156, 1998, ISSN: 0038-1101. DOI: https://doi.org/10.1016/S0038-1101(98)00210-X.

[12] P. G. Neudeck, D. J. Spry, M. J. Krasowski, N. F. Prokop, and L. Chen, "Demonstration of 4h-sic jfet digital ics across 1000 °c temperature range without change to input voltages," in *European Conference on Silicon Carbide and Related Materials (ECSCRM 2018)*, Document ID: 20190027358, NASA Glenn Research Center, University of Warwick, Birmingham, United Kingdom, Jul. 2018. [Online]. Available: https://ntrs.nasa.gov/citations/20190027358.

[13] P. G. Neudeck, G. M. Beheim, and C. S. Salupo, "600 °c logic gates using silicon carbide jfet's," NASA Glenn Research Center, Cleveland, Ohio, NASA Technical Memorandum NASA/TM–2000-209928, 2000. [Online]. Available: https://ntrs.nasa.gov/api/citations/20000033844/downloads/20000033844.pdf.

[14] M. Kaneko, M. Nakajima, Q. Jin, and T. Kimoto, "Sic complementary junction field-effect transistor logic gate operation at 623 k," *IEEE Electron Device Letters*, vol. 43, no. 7, pp. 997–1000, 2022. DOI: https://doi.org/10.1109/LED.2022.3179129.

[15] H. Habib, "Complementary jfet logic in silicon carbide," PhD Thesis, Ph.D. dissertation, Newcastle University, Newcastle upon Tyne, UK, 2013. [Online]. Available: http://hdl.handle.net/10443/2233.

[16] K. Shirriff, *Reading silicon: How to reverse engineer integrated circuits*. [Online]. Available: https://www.youtube.com/watch?v=aHx-XUA6f9g (visited on 08/13/2023).

[17] Appaloosa, *Intel 4004 architecture*, Aug. 21, 2007. [Online]. Available: https://commons.wikimedia.org/wiki/File:4004_arch.svg (visited on 06/01/2023).

[18] M. Szyc. "Intel 4004 emulator." (2007), [Online]. Available: http://e4004.szyc.org/ (visited on 10/01/2023).

APPENDIX A

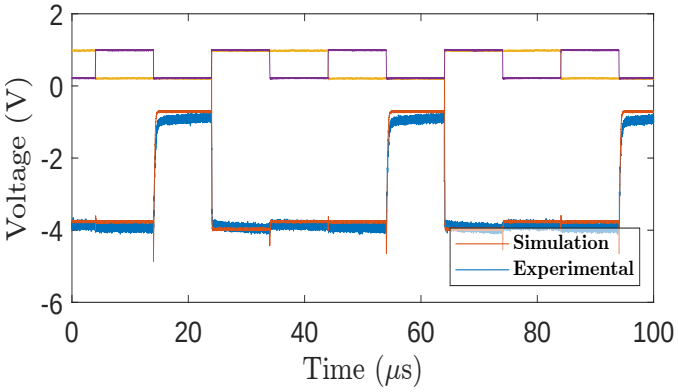


Fig. 22: Experimental and simulated NOR gate output voltage at 50 kHz Shifted by 1.5 V. The gate input voltages have been scaled and offset to the top of the diagram.

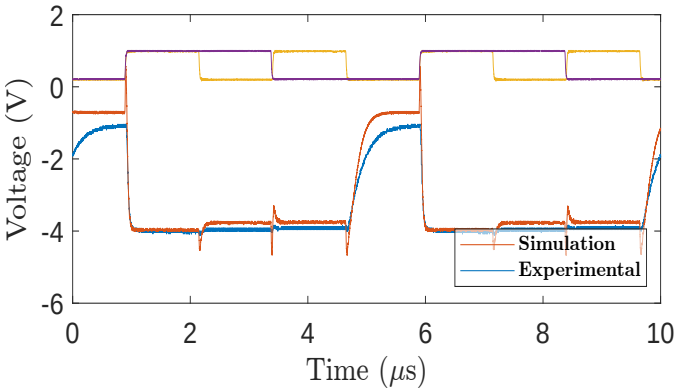


Fig. 23: Experimental and simulation NOR gate output voltage at 400 kHz Shifted by 1.5 V. The gate input voltages have been scaled and offset to the top of the diagram.

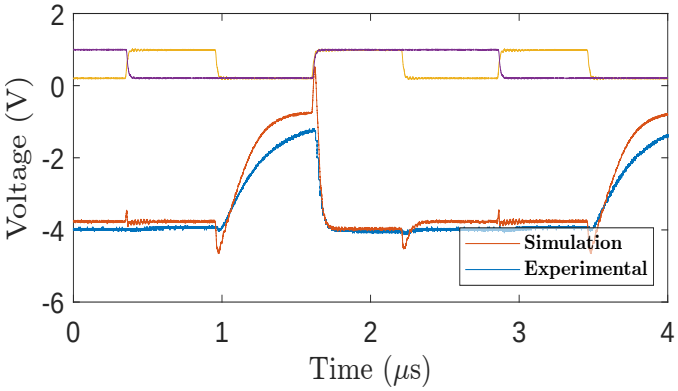


Fig. 24: Experimental and simulation NOR gate output voltage at 800 kHz Shifted by 1.5 V. The gate input voltages have been scaled and offset to the top of the diagram.

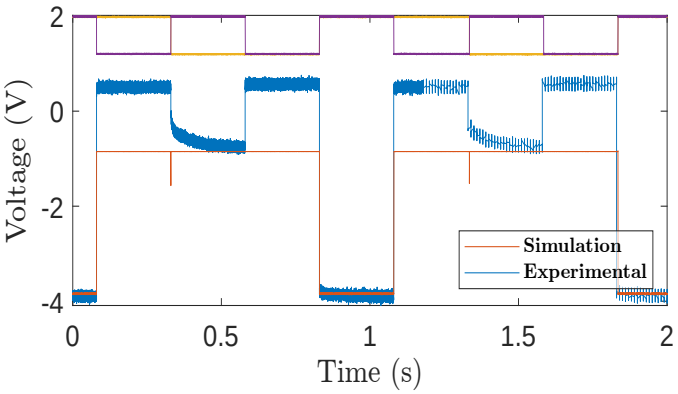


Fig. 25: Experimental and simulation NAND gate output voltage at 2 Hz Shifted by 1.5 V. The gate input voltages have been scaled and offset to the top of the diagram.

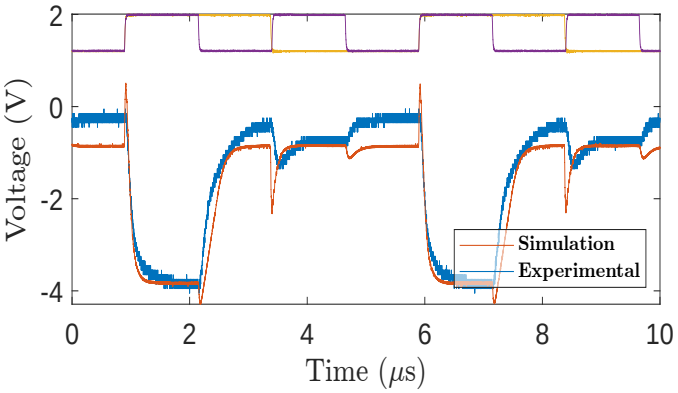


Fig. 26: Experimental and simulation NAND gate output voltage at 400 kHz Shifted by 2 V. The gate input voltages have been scaled and offset to the top of the diagram.

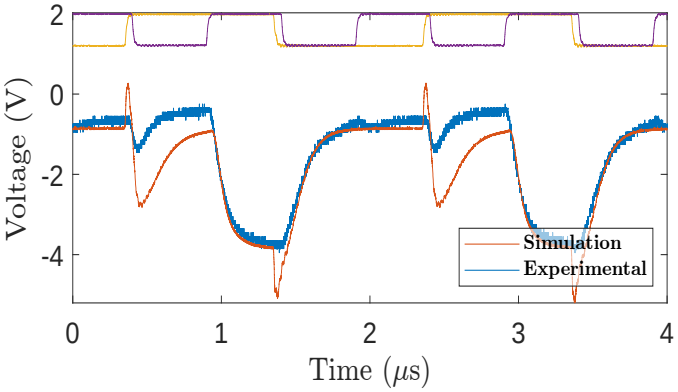


Fig. 27: Experimental and simulation NAND gate output voltage at 1 MHz Shifted by 2 V. The gate input voltages have been scaled and offset to the top of the diagram.

APPENDIX B

```
SIGNBITS CLC
LD 6
RAL
TCC
XCH 2
LD 6
RAL
CLC
RAR
XCH 6
CLC
LD 10
RAL
TCC
XCH 3
LD 10
RAL
CLC
RAR
XCH 10
BBL 0
```

Listing 1: Assembly Code for SIGNBITS function

```
START LDM 15
XCH 4
LDM 4
XCH 5
LDM 4
XCH 6
LDM 7
XCH 7
LDM 10
XCH 8
LDM 0
XCH 9
LDM 4
XCH 10
LDM 11
XCH 11
JMS MULT
NOP
```

Listing 2: Assembly Code for initialising registers for multiplication function

The way the multiplication algorithm (Listing 3) works is two separate computations. First the sign bit is extracted from the exponent using the SIGNBITS Subroutine (Listing 1). Then the two exponents are added together, however this means that the offset of 63 has been added twice and needs to be subtracted to get the final value. Next the significands need to be multiplied. This is achieved by checking the lowest order bit of the multiplier. If it is high, the multiplicand is added to the output register, the multiplicand is shifted to the left and the multiplier is shifted to the right. This then repeats until the entire multiplier has been checked and the final answer can be read. There are some tricks that have been done to preserve accuracy of the final answer such as using 4 registers to store the multiplicand and the product allowing for 16 bits of accuracy. The final step shifts the entire product until it detects that there are no 1s in the two registers used for the highest order bits. If the product is shifted more than 8 times it indicates that the product required normalising, shifting so that there is a 1 in the highest order bit. If this occurs, 1 needs to be added to the exponent for each time the product is shifted past 8 times.

```
MULT FIM 6 0 0
JMS SIGNBITS
LD 7
ADD 11
XCH 15
LD 6
ADD 10
XCH 14
CLC
LDM 15
XCH 0
LD 15
SUB 0
XCH 15
CMC
LDM 3
XCH 0
LD 14
SUB 0
XCH 14
CLC
LD 2
ADD 3
RAL
RAL
RAL
CLC
ADD 14
XCH 14
FIM 0 0 0
FIM 1 0 0
BCHECK LD 8
JCN 12 IF1
LD 9
JCN 12 IF1
JUN RETURN
IF1 CLC
LD 8
RAR
XCH 8
LD 9
RAR
XCH 9
JCN 10 IF2
CLC
LD 13
ADD 5
XCH 13
LD 12
ADD 4
XCH 12
LD 3
ADD 1
XCH 3
LD 2
ADD 0
XCH 2
IF2 CLC
LD 5
RAL
XCH 5
LD 4
RAL
XCH 4
RAL
LD 1
RAL
XCH 1
LD 0
RAL
XCH 0
JUN BCHECK
RETURN LD 2
JCN 12 IF3
LD 3
JCN 12 IF3
JUN MULTEND
IF3 LD 2
RAR
XCH 2
LD 3
RAR
XCH 3
LD 12
RAR
XCH 12
LD 13
RAR
XCH 13
CLC
JUN RETURN
MULTEND BBL 0
```

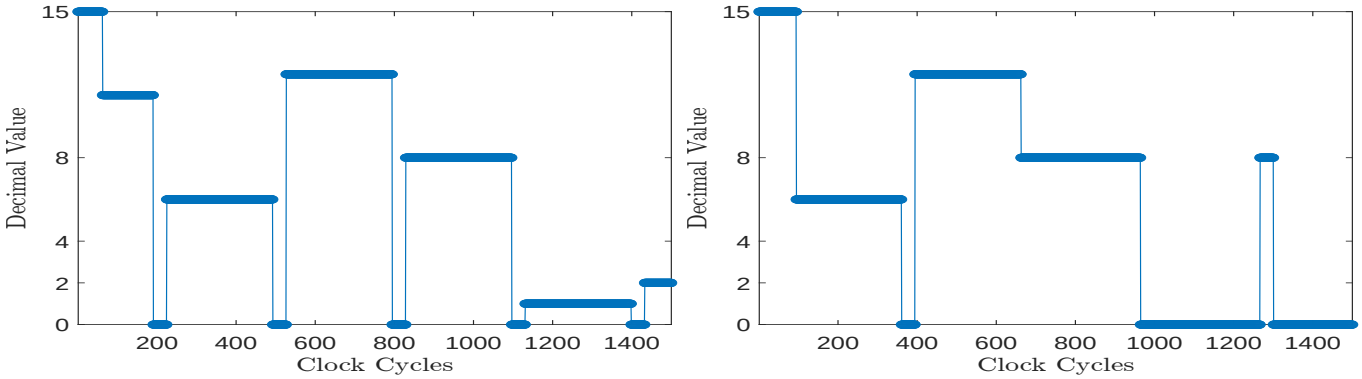
Listing 3: Assembly Code for floating point multiplication subroutine

APPENDIX C

```
JUN START
AND FIM 1 0 11
Loop LDM 0
XCH 0
RAL
XCH 0
INC 3
XCH 3
JCN 4 Done
XCH 3
RAR
XCH 2
XCH 1
RAL
XCH 1
RAR
ADD 2
JUN Loop
Done BBL 0
START LDM 11
XCH 0
LDM 6
XCH 1
JMS AND
NOP
```

Listing 4: Assembly Code for AND Subroutine

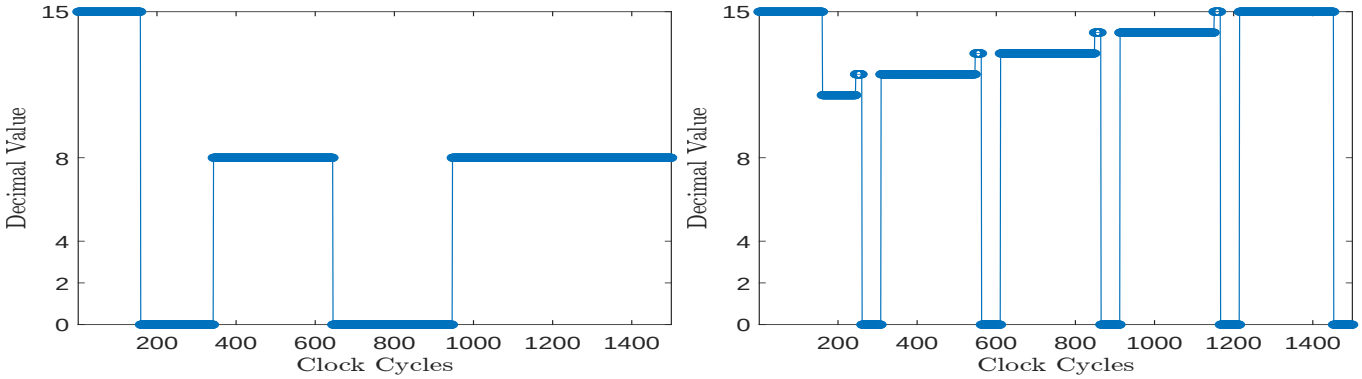
The AND subroutine produces the AND of two input words by placing the bits in the leftmost position of the accumulator 30a and register 2 29a, respectively, and zeroing the rightmost three bits of the accumulator and register 2. Register 2 is then added to the accumulator, and the resulting carry is equal to the AND of the two bits. Scratch Pad register 3 (Figure 29b) is used as the counter, counting up from 11. When this overflows from 15 to 0 the program breaks and returns.



(a) Scratchpad 0 value, containing the first input word which is shifted left four times. This is the register the function returns the output value to. The shifts are as follows, 1011 to 0110 to 1100 to 1000. The returned value is then placed in and shifted once, 0001 to 0010.

(b) Scratchpad 1 value, containing the second input word which is shifted left three times. The shifts are as follows, 0110 to 1100 to 1000 to 0000.

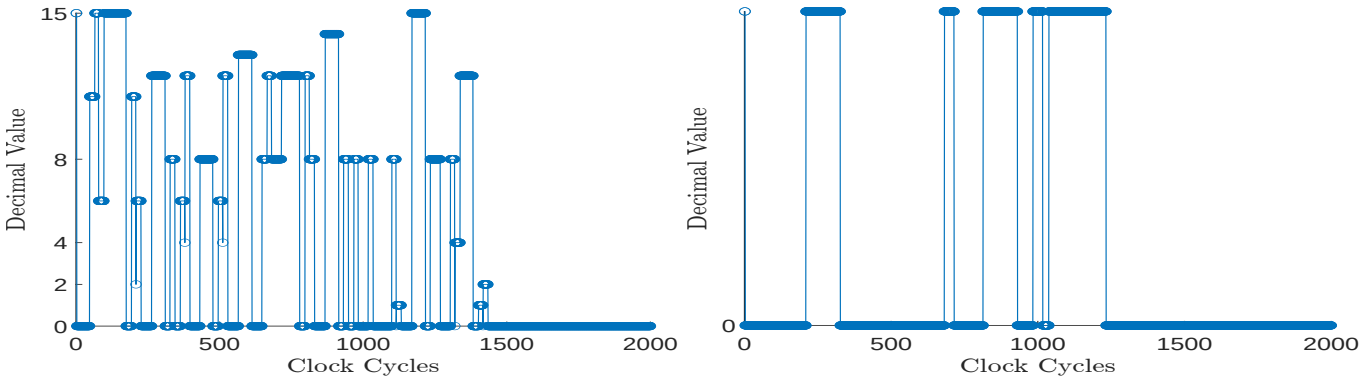
Fig. 28: Scratchpad Pair 0, the input and output registers.



(a) Scratchpad 2 value, this is working memory for the program. The highest order bit from one input is put into here and added to the accumulator which contains the high order bit from the other input.

(b) Scratchpad 3 value, this is the counter for counting the number of times register 0 has been shifted. This starts at 11 and the program breaks when the value overflows from 15 to 0.

Fig. 29: Scratchpad Pair 1



(a) Accumulator register value

(b) Carry Flag register value, when the highest bit of word one and word two are added together the carry is equal to the AND of the two bits

Fig. 30: Accumulator and Carry Flag register values

APPENDIX D

Full page circuit diagrams of CPU sub-systems as described in II-B.

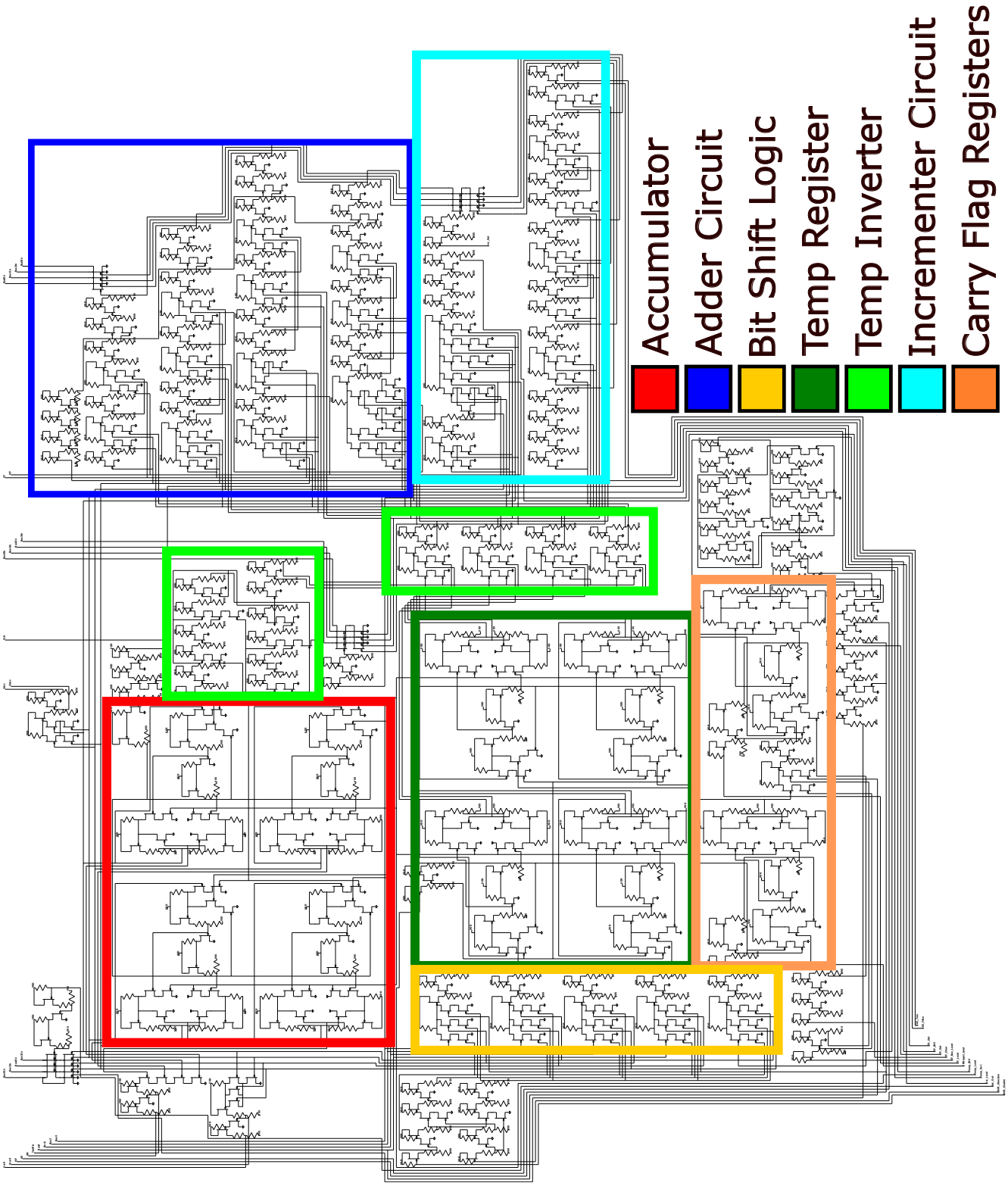


Fig. 31: Circuit diagram of the arithmetic logic unit

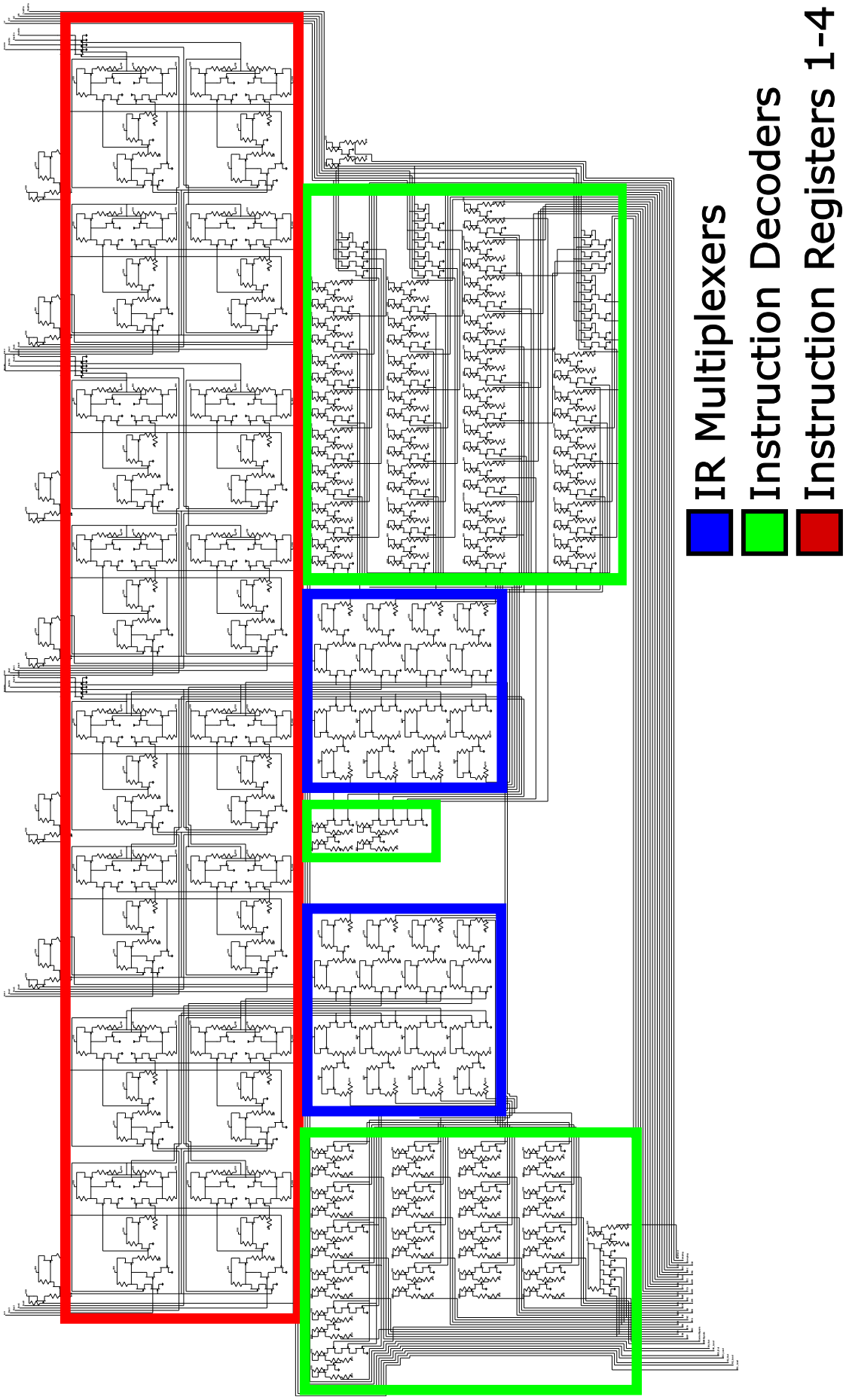


Fig. 32: Circuit diagram of the Instruction Register

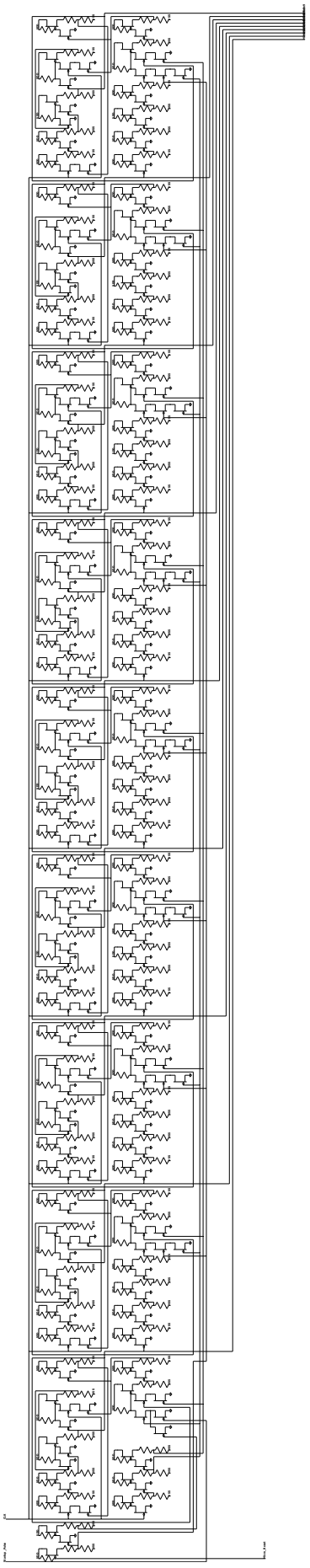


Fig. 33: Circuit diagram of the Step Counter

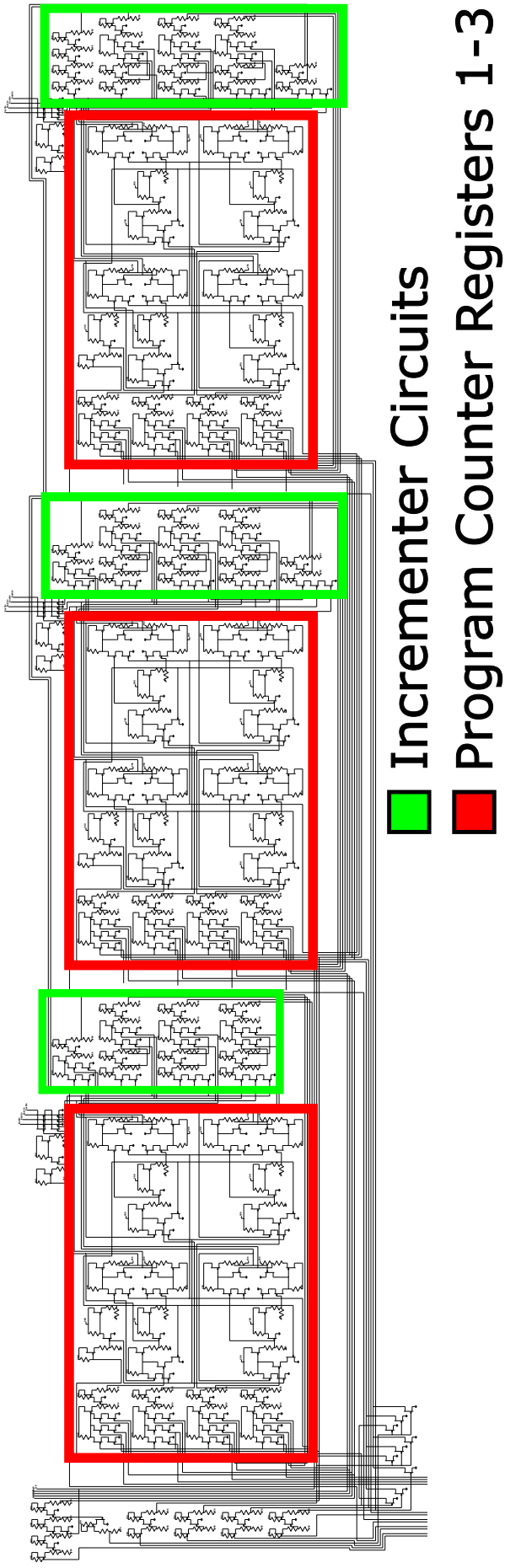


Fig. 34: Circuit diagram of the Program Counter

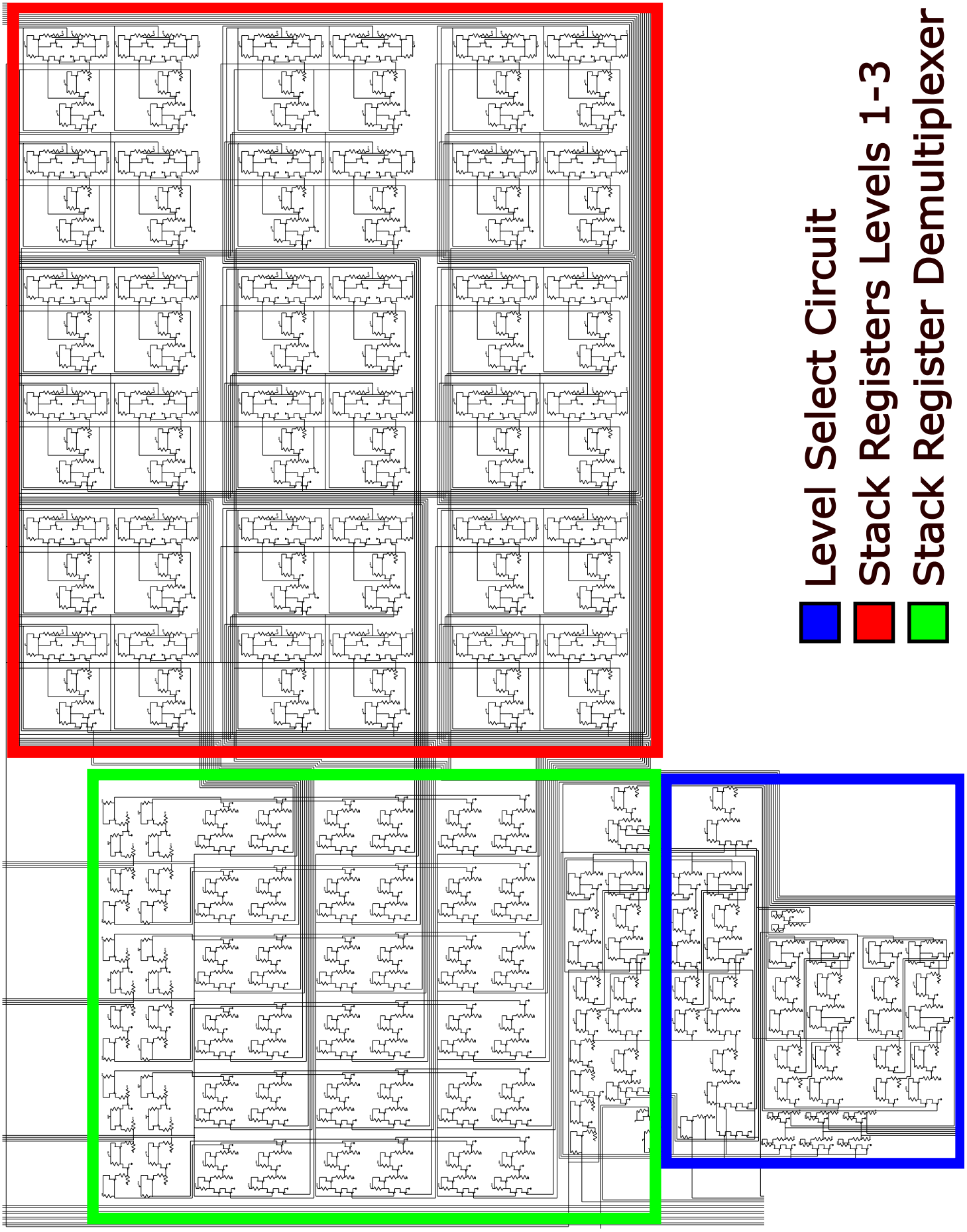


Fig. 35: Circuit diagram of the Stack

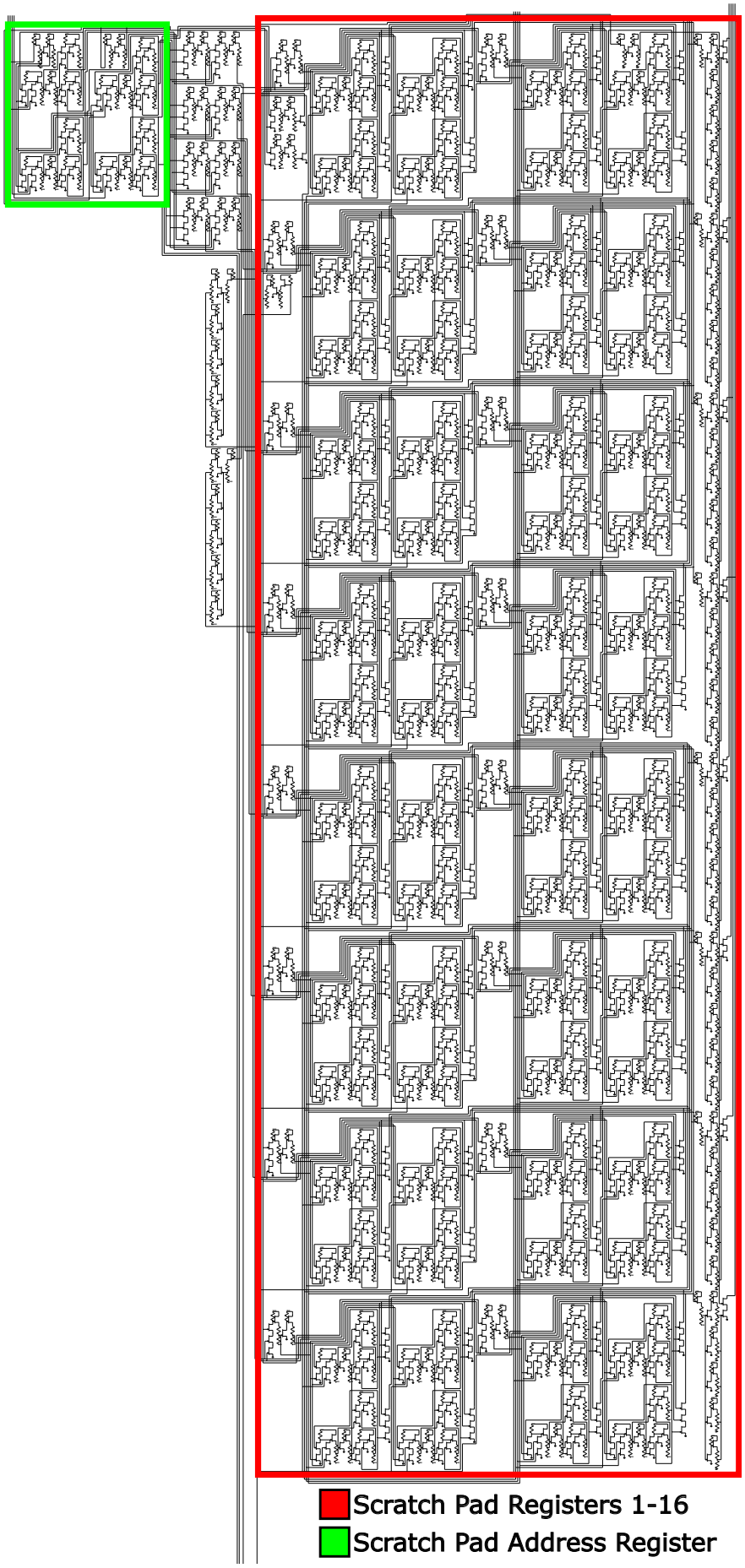


Fig. 36: Circuit diagram of the Scratch Pad

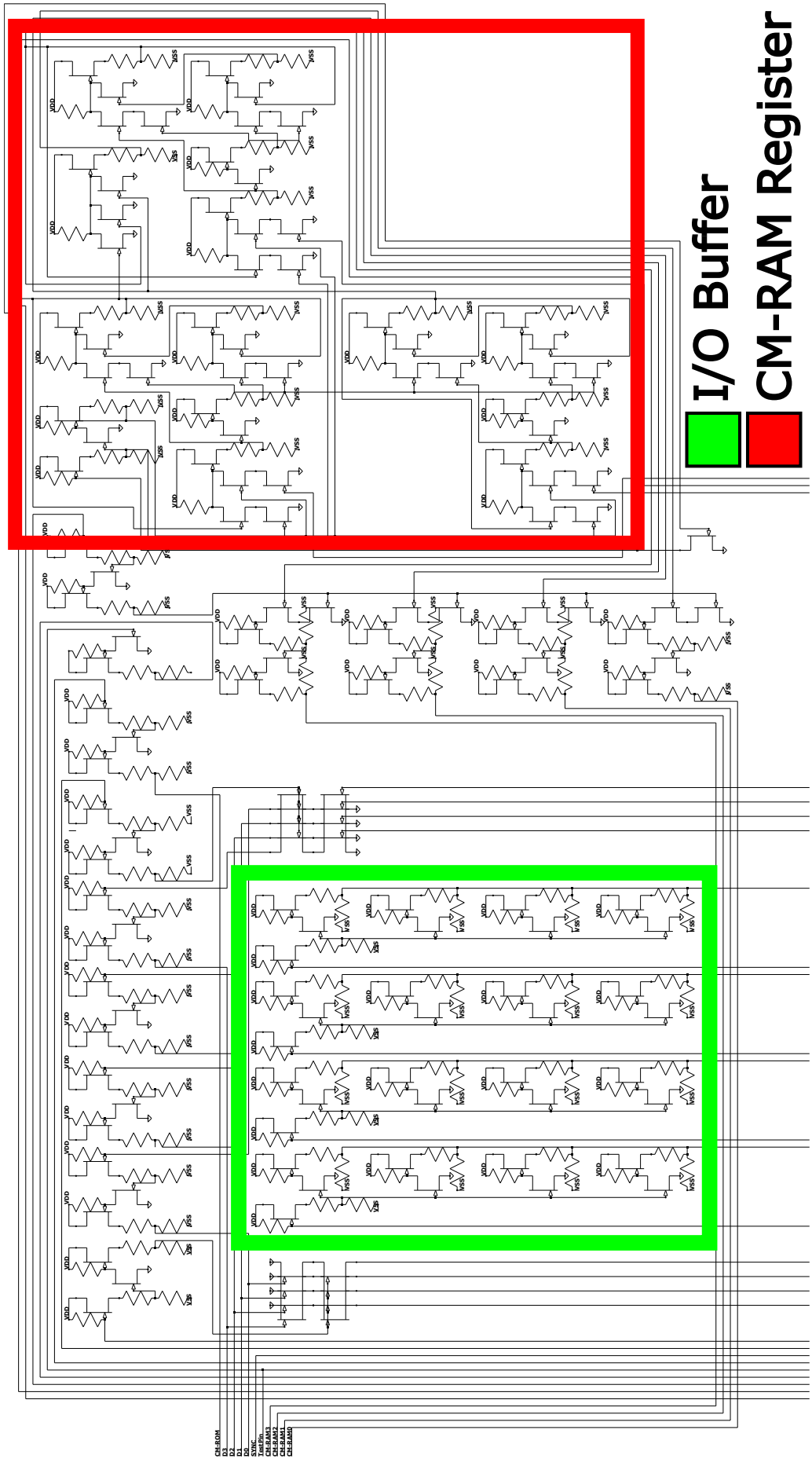


Fig. 37: Circuit diagram of the Pins

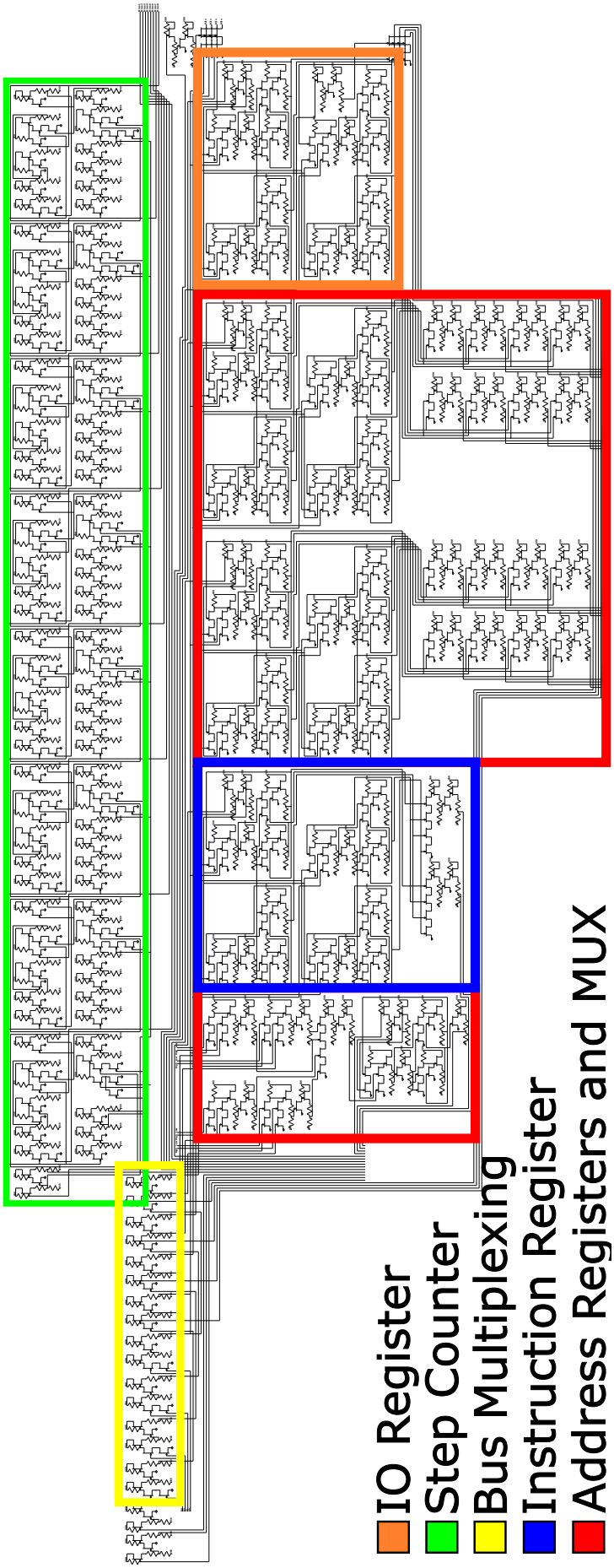


Fig. 38: Circuit diagram of the computational part of the ROM