Implementing the ABC Computer with Modern Technology (Phase 2)

Design Document

sdmay26-12
Dr. Alexander Stoytchev
Vance Arnold - Team Facilitator, Parts & Purchasing
Gavin Bunker - Technical Manual Lead
Kyle Koch - VHDL Lead
Ryan McFadden - Client Liaison, Recorder
Samuel Schroeder - Team Leader
Bryan York - PCB Lead
sdmay26-12@iastate.edu
https://sdmay26-12.sd.ece.iastate.edu

Executive Summary

The Atanasoff-Berry Computer (ABC) was the first electronic digital computer, built at Iowa State University in the late 1930s and early 1940s by John Atanasoff and Clifford Berry. The original machine was dismantled in 1948, and a 1990s replica built at Iowa State was eventually sent to the Computer History Museum in California, leaving Iowa State without an important piece of its history. In addition to the historical significance, the ABC also demonstrates foundational concepts in digital logic and computer architecture like binary arithmetic, electronic switching, and capacitive memory, which are still relevant to engineering education today. The lack of an on-campus replica means that students, instructors, and visitors do not have a way to interact with these concepts in this important piece of Iowa State history.

Our design requirements had four main categories. Functionally, the system must accurately recreate the ABC's original logic and solve systems of up to 29 linear equations in 29 unknowns using the same algorithm as the original. Resource constraints required the use of modern integrated circuits, printed circuit boards, and through-hole components for ease of repair and debugging. Physically, the system must be portable, modular, and able to fit on a desk or table for demonstration. From a user experience perspective, the system must have LED indicators for all major signals to clearly show the internal state and operation of the system.

To meet these requirements, our team designed a modular system of three main subsystems. The control subsystem coordinates the timing and control signals of all the other systems through the Control System module, Clock & Program Counter, and Control Panel. The input/output subsystem replaces the ABC's physical punch cards with ESP32-based digital card readers and writers that communicate with Android interfaces. The compute subsystem contains six identical Memory Drums each with five Add/Subtract modules to perform the core functionality of the replica. All of these designs were first verified in VHDL or breadboard prototypes and then implemented into PCBs. Testing from the unit to system level was used to confirm that the individual modules and critical paths functioned as intended, and a full VHDL system simulation validated was used to perform more extensive system design testing.

The primary next steps for this project are a second hardware revision to address remaining defects and polish the physical product, expansion to the full-scale six memory drums, and development of automated control software.

Learning Summary

Development Standards & Practices Used

In developing the modern recreation of the Atanasoff-Berry Computer, our team relied on a combination of established circuit, hardware, and software practices. Early prototypes were constructed on breadboards to validate the logic of individual modules before committing to permanent designs. To simplify debugging and repair, through-hole and socketed components were chosen, ensuring that faulty chips could be easily replaced. As the designs progressed, PCBs were created following IPC 2221 guidelines, which define trace widths, clearances, and via specifications to guarantee safe and reliable operation. IEEE 1149.1-2013, which provides boundary-scan techniques for verifying interconnections between integrated circuits across multiple PCBs, guided our testing practices as well as our PCB design to ensure that our final system will be easily testable.

For the software side, our team adopted professional standards to maintain clarity and consistency. The simulation and verification of modules were performed in VHDL, adhering to IEEE/IEC 61691-1-1-2023, which specifies the syntax and functionality of the language. This allowed the team to formally describe and test the behavior of each module before moving to hardware implementation. Standard communication protocols between wireless devices like Bluetooth (IEEE 802.15.1) were used to communicate between out software applications.

Beyond technical standards, the project follows standard engineering practices to manage our design process. A waterfall project management approach was adopted, as it suited the sequential nature of hardware development and minimized delays caused by PCB manufacturing cycles. GitLab was used for version control and collaborative review. Finally, testing was incorporated into the design process at multiple stages to ensure that our project is tested early and often. Thus, each module and subsystem could be validated before full assembly.

Summary of Requirements

- The system should accurately recreate the original functionality and overall design of ABC, including the same modules as the original.
- The system should be able to solve systems of linear equations, up to 29 equations in 29 unknowns, using the same algorithm as the ABC.
- The system should allow users to input data, observe the computation process, and retrieve the output data of each step.

- The system should be composed of modern components, including integrated circuits, printed circuit boards, and semiconductor memory devices.
- The system should be easily carried by one person and should not exceed the dimensions of a typical desk or table.
- The complete system should be modular to allow for easy assembly and disassembly as needed for repairs or long-distance travel.
- The system should have LEDs to indicate the state of the inputs and outputs of each module and be visually appealing to the average user.
- The system should be intuitive to operate, with clear instructions, so that the average user can easily navigate and use it.

Applicable Courses from Iowa State University Curriculum

- CPRE 2810: Digital Logic
- CPRE 3810: Computer Organization and Assembly Level Programming
- EE 2300: Electric Circuits II
- EE 3330: Electronic Systems Design
- SE 3090: Software Development Practices
- SE 3290: Software Project Management

New Skills/Knowledge acquired that was not taught in courses

- PCB Design Tools: Learning KiCad for schematic capture and PCB layout
- Soldering and Assembly: Hands-on soldering of through-hole components, socketed ICs, and debugging assembled boards
- Hardware Debugging Techniques: Using continuity testing and modular replacement strategies
- Timing analysis in complex digital logic circuits

Table of Contents

1. Introduction **11**
1.1. Problem Statement 11
1.2. Intended Users 11
2. Requirements, Constraints, and Standards **13**
2.1. Requirements & Constraints 13
2.1.1. Functional requirements 13
2.1.2. Resource Requirements 13
2.1.3. Physical Requirements 13
2.1.4. User Experience Requirements 13
2.2. Engineering Standards 13
2.2.1. IEEE 1149.1-2013 14
2.2.2. IEEE 802.15.1 14
2.2.3. IEEE/IEC 61691-1-1-2023 14
2.2.4. IPC 2221 14
3. Project Plan **16**
3.1. Project Management/Tracking Procedures 16
3.2. Task Decomposition 16
3.3. Project Proposed Milestones, Metrics, and Evaluation Criteria 17
3.4. Project Timeline/Schedule 19
3.5. Risks and Risk Management/Mitigation 20
3.6. Personnel Effort Requirements 22
3.7. Other Resource Requirements 24
4. Design **26**
4.1. Design Context 26
4.1.1 Broader Context 26
4.1.2 Prior Work/Solutions 26
4.1.3 Technical Complexity 27
4.2. Design Exploration 28
4.2.1 Design Decisions 28
4.2.2 Ideation 28
4.2.3 Decision-Making and Trade-Offs 29
4.3. Final Design 31
4.3.1 Overview 31
4.3.2 Detailed Design and Visual(s) 33
Control Panel 33
Purpose 33
Input and Output Signals 33
Input Signals 33
Output Signals 34
Design 34

Full Control System..... 36
Unit E: Add-Subtract Control..... 37
Purpose..... 37
Input and Output Signals..... 37
Input Signals..... 37
Output Signals..... 37
Description of Operation..... 37
Generating the Add-Subtract Signals..... 37
Shifting the Decimal Card Reader..... 38
Decimal Output Control..... 38
Design..... 39
Timing Diagrams..... 39
Unit F: Negative Number Indicator..... 40
Purpose..... 40
Input and Output Signals..... 40
Input Signals..... 40
Output Signals..... 41
Description of Operation..... 41
Design..... 41
Timing Diagram..... 41
Unit G: Shift Control..... 42
Purpose..... 42
Input and Output Signals..... 42
Input Signals..... 42
Output Signals..... 42
Description of Operation..... 42
Design..... 42
Timing Diagram..... 43
Unit H: One-Cycle Relays..... 43
Purpose..... 43
Input and Output Signals..... 43
Input Signals..... 43
Output Signals..... 43
Description of Operation..... 44
Design..... 45
Timing Diagram..... 46
Unit I: Division Control..... 46
Purpose..... 46
Input and Output Signals..... 46
Input Signals..... 46
Output Signals..... 46

Description of Operation....46
Starting Computation....46
Detecting Zero....47
Stopping Computation....47
Design....47
Timing Diagram....48
Figure 23. Division Control Timing....48
Timing Drum....48
Purpose....48
Input and Output Signals....48
Input Signals....48
Output Signals....48
Description of Operation....49
Design....49
Input Selector....49
Purpose....49
Input and Output Signals....49
Input Signals....49
Output Signals....49
Description of Operation....49
Design....50
Clock & Program Counter....50
Purpose....50
Input and Output Signals....50
Input Signals....50
Output Signals....50
Description of Operation....51
Design....51
Card Readers/Writers....51
Purpose....51
Input and Output Signals....52
Input Signals....52
Output Signals....52
Description of Operation....52
Decimal Card Reader....52
Binary Card Reader....53
Binary Card Writer....53
Design....53
Base Conversion Module....54
Purpose....54
Input and Output Signals....54

Input Signals....54
Output Signals....54
Description of Operation....54
Design....55
Decimal Output Module....56
Purpose....56
Input Signals....56
Input Signals....56
Description of Operation....56
Design....57
Add/Subtract Module....57
Purpose....57
Input and Output Signals....58
Input Signals....58
Output Signals....58
Design....58
Memory Drum....58
Purpose....58
Input and Output Signals....58
Input Signals....58
Output Signals....59
Description of Operation....59
Memory....59
Shifting....59
Input Selector....60
Data Flow....60
Design....61
4.3.3 Functionality....62
4.3.4 Areas of Challenge....63
4.4. Technology Considerations....63
5. Testing....64
5.1. Unit Testing....64
5.2. Interface Testing....65
5.3. Integration Testing....66
5.4. System Testing....67
5.5. Regression Testing....68
5.6. Acceptance Testing....68
5.7. User Testing....68
5.8. Testing Results....69
6. Implementation....71
6.1. Design Analysis....77

7. Ethics and Professional Responsibility..... 79
7.1. Areas of Professional Responsibility/Codes of Ethics.....79
7.2. Four Principles.....81
7.3. Virtues..... 82
7.3.1 Group Virtues.....82
7.3.2 Individual Virtues..... 82
8. Conclusions..... 87
8.1. Summary of Progress..... 87
8.2. Value Provided.....87
8.3. Next Steps..... 88
9. References..... 89
[10] Appendices..... 90
[a] Appendix 1 – Operation Manual.....90
[b] Appendix 2 – Alternative/initial version of design.....92
[c] Appendix 3 – Other considerations..... 93
[d] Appendix 4 – Code and Materials.....93
[e] Appendix 5 – Team Contract.....94

List of Figures

1. Figure 1. Milestones and Metrics Table
2. Figure 2. Full Gantt Chart
3. Figure 3. Risk Management Table
4. Figure 4. Expected Effort Table
5. Figure 5. Actual Effort Table
6. Figure 6. Bit-Width Cost Analysis
7. Figure 7. System-Level Diagram
8. Figure 8. High-Level Block Diagram
9. Figure 9. Detailed Block Diagram
10. Figure 10. Original ABC Control Panel
11. Figure 11. Control System Module
12. Figure 12. Add-Subtract Control Design
13. Figure 13. Add-Subtract Control Timing 1
14. Figure 14. Add-Subtract Control Timing 2
15. Figure 15. Add-Subtract Control Timing 3
16. Figure 16. Negative Number Indicator Design
17. Figure 17. Negative Number Indicator Timing
18. Figure 18. Shift Control Design
19. Figure 19. Shift Control Timing
20. Figure 20. One-Cycle Relays Design
21. Figure 21. One-Cycle Relays Timing
22. Figure 22. Division Control Design
23. Figure 23. Division Control Timing
24. Figure 24. Input Selector Control Design
25. Figure 25. Clock & Program Counter Design
26. Figure 26. Card Readers/Writers Design
27. Figure 27. Base Conversion Table Visualization
28. Figure 28. Base Conversion Table Slice
29. Figure 29. Base Conversion EEPROM Address Format
30. Figure 30. Base Conversion Module
31. Figure 31. Decimal Output Module Design
32. Figure 32. Add/Subtract Module Design
33. Figure 33. Memory Drum Timing Logic
34. Figure 34. Memory Drum Timing Diagram
35. Figure 35. Memory Address Space
36. Figure 36. Control Panel Unit Tester
37. Figure 37. Clock - Control System Interface Tester
38. Figure 38. Partial Data Input Integration Tester
39. Figure 39. Testing Results Defect Table
40. Figure 40. Control System Implementation
41. Figure 41. Clock & Program Counter Implementation
42. Figure 42. Control Panel Implementation
43. Figure 43. Decimal Output Implementation
44. Figure 44. Base Conversion Implementation
45. Figure 45. Decimal Card Reader Interface

46. Figure 46. Binary Card Reader Interface
47. Figure 47. Binary Card Writer Interface
48. Figure 48. Add/Subtract Module Implementation
49. Figure 49. Memory Drum Implementation
50. Figure 50. Assembled Memory Drum Module Implementation
51. Figure 51. Motherboard Implementation
52. Figure 52. Assembled Motherboard Implementation
53. Figure 53. Memory Drum Module Side Profile
54. Figure 54. Code of Ethics Table
55. Figure 55. Ethical Principles Table
56. Figure 56. Alternative Decimal Output Design
57. Figure 57. Skills Table

1. Introduction

1.1. Problem Statement

The Atanasoff-Berry Computer (ABC) was the first electronic digital computer, built at Iowa State University in the late 1930s and early 1940s. Unfortunately, the original ABC was dismantled in 1948. In the 1990s, a team at Iowa State built a replica of the ABC, which was eventually moved to the Computing History Museum in California. The ABC is an important piece of electrical and computer engineering history, which is missing from Iowa State's campus. The goal of this project is to recreate the ABC at the logical level using modern technology. By examining the foundational concepts of the ABC and applying them from a contemporary perspective, we aim to create an accurate and visually engaging recreation of the ABC. This resource will be useful for teaching a diverse audience about computer architecture, design, and history, as well as the legacy of innovation at Iowa State.

1.2. Intended Users

Computing Teachers

1. This user is someone who is teaching computer history, digital logic, or computer architecture. They have expertise in one or more of these topics and are looking for ways to better teach their students, who may not have any previous experience with these concepts, through a more hands-on approach.
2. Computing teachers need a way to accurately demonstrate the logic of the ABC to their students, so that they can help their students better understand the theoretical concepts taught in class. They also need to be able to easily operate this demonstration in a classroom environment and visually show the internal state and operation of the machine.

Computing Students

1. This user is someone who has enrolled in a course about computer history, digital logic, or computer architecture. They are interested in these topics and want to learn more about them in hands-on ways. They also want to better understand and remember what they are taught in class.
2. Computing Students need to see a practical application of what they are learning to better understand the course content. They need to be shown a demonstration that is engaging and visually communicates what the computer is doing, so that they can see how the course concepts are being used.

Computing Enthusiasts

1. This user is someone who has an interest in and some familiarity with computing history, digital logic, or computer architecture, and wants to learn more about the

ABC's design, function, or historical role for their personal enjoyment. They may even want to build a replica of their own as a hobby.
2. Computing Enthusiasts need a way to read about the ABC's history and design, so that they can teach themselves about the ABC and its underlying theoretical principles. They need a way to learn how to replicate ABC modules without doing their own deep research through archives into the original design.

Iowa State Visitor

1. This user is someone visiting Iowa State for a university-hosted event, a prospective student tour, or to visit a student or faculty member at Iowa State. They are interested in learning about what makes Iowa State unique and the history of innovation at the university. This user is not familiar with the ABC or the technical issues surrounding its design and implementation.
2. Iowa State Visitors need a way to easily understand what the ABC was and why it is significant, without having to understand the technical details of the machine.

2. Requirements, Constraints, and Standards

2.1. Requirements & Constraints

2.1.1. Functional requirements

1. The system should accurately recreate the original functionality and overall design of ABC, including the same modules as the original.
2. The system should be able to solve systems of linear equations, up to 29 equations in 29 unknowns, using the same algorithm as the ABC.
3. The system should allow users to input data, observe the computation process, and retrieve the output data of each step.

2.1.2. Resource Requirements

1. The system should be composed of modern components, including integrated circuits, printed circuit boards, and semiconductor memory devices (constraint).
2. Only Through-Hole Socketed microchips should be used, along with through-hole components to make debugging and repair easier (constraint).

2.1.3. Physical Requirements

1. The system should be a tangible, physical object that users can interact with.
2. The system should be easily carried by one person (constraint).
3. The system should not exceed the dimensions of a typical desk or table, but fit within a space suitable for interactive demonstrations (constraint).
4. The complete system should be modular to allow for easy assembly and disassembly as needed for repairs or long-distance travel.

2.1.4. User Experience Requirements

1. The system should have LEDs to indicate the state of the inputs and outputs of each module.
2. The system should be intuitive to operate, with clear instructions, so that the average user can easily navigate and use it.
3. The system should be visually appealing to the average user.

2.2. Engineering Standards

Engineering standards are an important part of ensuring consistency, safety, and interoperability across the industry. They allow engineers to design systems that work reliably with components from other vendors, based on shared specifications of interaction. By following established safety and testing standards, engineers can ensure that products operate safely and correctly. Standards also reduce errors and streamline the development process, since engineers can focus on improving systems rather than reinventing basic protocols. In this way, standards also promote innovation by providing a shared technical foundation. In education settings like senior design, it teaches students how to think professionally and design with real-world constraints in mind.

2.2.1. IEEE 1149.1-2013

- Defines a method for testing connections between integrated circuits on PCBs and the integrated circuits themselves using boundary-scan techniques. This standard is intended to enhance the testing process of interconnected circuits, improving the safety and reliability of various electronic devices [5].
- This standard is relevant to our project because we created many interconnected PCB modules, each with several integrated circuit chips, which required testing.
- To incorporate this standard into our project, we used a modular design that enabled the easy insertion of test values and the reading of outputs from individual PCB components. This design helped improve the testing process to ensure the correct and reliable functioning of each component.

2.2.2. IEEE 802.15.1

- Defines the Bluetooth and Bluetooth Low Energy (BLE) technologies. The goal of this standard is to ensure wireless data transfer with nearby devices [6].
- This standard is relevant to our project, since we used both classic Bluetooth and BLE technologies to transfer data between different components and interfaces.
- To incorporate this standard into our project, we used the protocols defined in this standard to transfer input and output data between Android tablet interfaces and ESP32 receivers on the replica.

2.2.3. IEEE/IEC 61691-1-1-2023

- Defines VHISC Hardware Description Language (VHDL) syntax and functionality. VHDL is a formal notation designed to aid in all stages of creating electronic systems, from design to testing. This standard is also intended to help engineers develop tools that support the language [7].
- This standard is directly relevant to our project, since we created electronic components to recreate modules of the ABC, which is exactly the process that VHDL is intended to support.
- To incorporate this standard into our project, we used VHDL, rather than other hardware description languages like Verilog, to develop and verify our component designs.

2.2.4. IPC 2221

- Establishes general design requirements for printed circuit boards (PCBs), including parameters like trace width, clearances, and via design. The goal of this standard is to provide a foundation for designing and manufacturing safe and reliable PCBs [8].
- This standard is directly relevant to our project, since the final product of our project is a recreation of the ABC implemented using PCBs. Much of our work involved converting component designs or breadboard prototypes into PCB designs for manufacturing.

- To incorporate this standard into our project, we ensured that our PCB design software (KiCad) was configured with design settings that comply with this standard. We also created a checklist of standards to review prior to submitting our PCBs for manufacturing.

3. Project Plan

3.1. Project Management/Tracking Procedures

For this project, we adopted the waterfall project management style. The waterfall approach, with its sequential structure, is a much more suitable approach given the hardware-oriented nature of our project. Taking an iterative approach, such as agile, would not be cost- or time-effective, as each iteration would require purchasing a batch of PCBs and waiting for them to be delivered before proceeding. With the waterfall approach, we went through the full process of gathering requirements, designing, and simulating our designs before manufacturing anything. To minimize waiting time, we had multiple "waterfalls", one for each module. With this strategy, we were able to start the waterfall process on the next module while we waited for the previous module to be manufactured.

We primarily used GitLab to track and manage our progress. Using the version control system and merge request process of Git, we tracked the progress and reviewed our designs as well as provided feedback to the designer.

3.2. Task Decomposition

1. General Module Design Procedure
    a. Create the logical design
        i. Research how the original module worked and how the previous team implemented the module
        ii. Create a logical circuit diagram of the module
        iii. Prototype the design on breadboards (if needed)
    b. Simulate the design in VHDL or build a breadboard prototype
        i. Write a VHDL representation of the logical design
        ii. Write and run a VHDL test bench to verify the correctness of the logical design
        iii. OR Create a breadboard prototype and verify the correctness of the logical design
    c. Design the PCB
        i. Implement the logical design as a KiCad schematic with any other needed electrical components
        ii. Convert the KiCad schematic into a PCB layout
    d. Manufacture the PCB
        i. Submit the PCB design to ETG for purchasing
        ii. Solder any necessary components onto the PCB
    e. Unit testing to verify the correct operation of each PCB
2. Perform the General Module Design Procedure for the following modules
    a. Control Panel
    b. Control System
    c. Clock & Program Counter

    d. Card Readers/Writers
    e. Base Conversion Module
    f. Decimal Output Module
    g. Add/Subtract Module
    h. Memory Drum
3. Testing
    a. Interface Testing
        i. Test the functionality between each module and its directly related modules
    b. Integration Testing
        i. Test the compute system
        ii. Test the I/O system
        iii. Test the control system
    c. System testing
4. Top-Level Assembly
    a. Design the final assembly to connect all of the modules together into a single product
    b. Create the assembly
    c. Assemble the final product

3.3. Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestone	**Metric**
Logical Design	
Completion of historical research and prior implementation review	Summary document with references and design notes for each module
Logical circuit diagram finalized	Diagram reviewed and approved by the team and the client
Structural VHDL representation completed	Code compiles without errors and matches the logical diagram

VHDL test bench verifies correctness	All expected outputs match actual outputs for test inputs
PCB Design	
KiCad schematic completed	Schematic includes all required components and passes ERC (Electrical Rule Check) with zero critical errors
PCB layout finalized	DRC (Design Rule Check) passes with zero critical errors
PCB submitted to ETG	Submission confirmation received and logged
PCB manufactured and received	Physical board matches design specs and passes visual inspection
Components soldered	All components are correctly placed, and solder joints pass a continuity test
Unit testing of PCB	≥ 95% of functional tests pass
Integration and Testing	
Interface testing between modules	Each module communicates correctly with adjacent modules; signal integrity verified
Subsystem integration testing	Subsystems pass ≥ 90% of integration tests

Full system testing	ABC replica solves a system of equations correctly and visibly demonstrates each step
Final assembly completed	All modules are physically connected, and the system operates as a unified unit

Figure 1. Milestones and Metrics Table

3.4. Project Timeline/Schedule

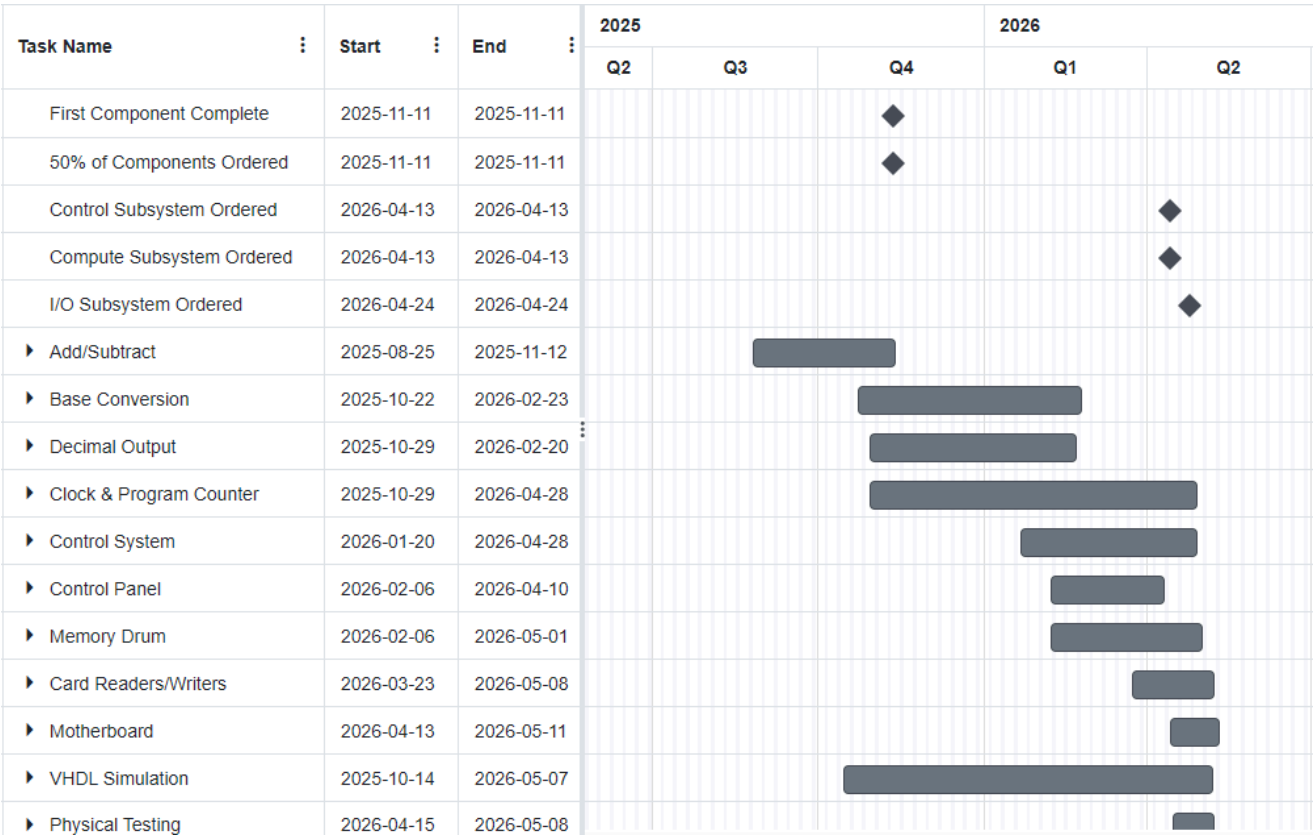


Figure 2. Full Gantt Chart

The first module that we will worked on is the Add/Subtract module. Since the previous team was able to fully implement this module, we each individually designed the module to gain familiarity with all of the tools we would use and the design process. Once this module was completed, we split into groups, with each group working on a single module. With this method, we could schedule tasks for modules simultaneously, as shown in

Figure 2. For example, the base conversion, decimal output, and clock modules were all worked on simultaneously. To minimize time lost waiting for PCB manufacturing, each group will started work on the next module after the order for their previous module was submitted. Once most modules are completed, we proceeded to various levels of integration and system testing and assemble the final product. See the detailed gantt chart here.

3.5. Risks and Risk Management/Mitigation

Task	Potential Risk	Probability	Impact	Mitigation Strategy
Logical Design & Simulation	Incorrect logic implementation leading to system malfunction	Medium	High	Perform extensive VHDL simulation before hardware implementation, review logic diagrams against the original ABC documentation, and conduct peer design reviews.
PCB Design & Fabrication	PCB errors (incorrect routing, footprint mismatch, or design rule violations)	Medium	High	Use KiCad's ERC/DRC checks, perform design reviews before fabrication, prototype with small test boards, and order extra PCBs for rework if necessary.
Component Availability	TTL or discrete logic components may be difficult to source	Low	Medium	Identify alternative or pin-compatible components early; maintain multiple supplier options.
Hardware Integration	Timing or signal integrity issues between modules	Medium	High	Use oscilloscope and logic analyzer testing to validate interconnections, ensure consistent power and ground planes, and document signal interface specifications clearly.

Power Distribution	Excessive current draw or unstable voltage regulation	Low	Medium	Simulate expected power consumption; design power rails with margin; include current-limiting resistors and decoupling capacitors.
Software or Firmware Verification	Misalignment between simulated logic and hardware implementation	Low	Medium	Compare VHDL simulation waveforms with real hardware behavior, develop test scripts for each module, and automate regression testing whenever possible.
Schedule or Resource Constraints	Delays due to component lead times or team workload	Medium	Medium	Begin ordering components early; prioritize critical path tasks; schedule weekly progress reviews; reallocate labor to high-risk modules if needed.
Documentation & Demonstration	Insufficient documentation or hardware failure during demo	Low	High	Maintain live documentation on GitLab; perform multiple pre-demo dry runs; prepare backup demonstration video.

Figure 3. Risk Management Table

There were two primary risks that we encountered during this project. The first risk that occurred was Schedule or Resource Constraints. This included delays in receiving components that were needed to assemble or prototype certain designs, such as delays in receiving EEPROMs for the Base Conversion prototype, and delays in getting PCBs manufactured due to issues with the ordering system, which delayed our Clock and Memory Drum PCBs. To mitigate lost time from these delays, the affected team member simply worked on other areas of the project until the delay was resolved. The second main risk that occurred was PCB Design & Fabrication. Several PCBs that we ordered had errors in them with a range of severity. Most of the errors were able to be fixed with jumper wires to correct the logic, but one PCB had to be reordered. In this more severe case, we mitigated the impact by, again, continuing work on other areas.

3.6. Personnel Effort Requirements

Note: Original planned tasks/modules preserved for comparison.

Task	Subtasks	Projected Person-hours	Description
Initial Discovery Work	Reviewing the work of last year's SD team that started this ABC Project	24	It is essential for our team to have a thorough understanding of both the original ABC's design and the progress made by last year's senior design team in recreating it with modern components, in order to undertake this ambitious project effectively.
	Researching and reading about the original ABC	30	
Add/Subtract Module	Logical Design	10	The Add-Subtract Module will be the first module of the ABC that we will recreate. To broaden each team member's experience and appreciation for the module, we will each recreate the logic of the ABC's Add-Subtract mechanism on a breadboard. Then, we will develop structural VHDL code for the model and design the module's PCB using the KiCad software.
	Structural VHDL Design	5	
	PCB Design	16	
	Ordering PCBs & Logistics	2	
	PCB Assembly & Testing	8	
Memory Module	Logical Design	15	This unit will be responsible for storing all 30 coefficients of the equations to be solved.
	Structural VHDL Design	6	
	PCB Design	15	
	Ordering PCBs & Logistics	2	
	PCB Assembly & Testing	4	
Binary	Logical Design	6	This unit will be

Conversion Module	Structural VHDL Design	8	responsible for converting decimal numbers from the decimal input module into binary numbers to be stored on the memory drums.
	PCB Design	12	
	Ordering PCBs & Logistics	2	
	PCB Assembly & Testing	4	
Timing and Control Module	Logical Design	40	This unit is very important as it sets all the timings with the clock and control signals.
	Structural VHDL Design	20	
	PCB Design	24	
	Ordering PCBs & Logistics	4	
	PCB Assembly & Testing	10	
Base-10 I/O Module	Logical Design	20	This unit will take base-10 input and use an add/subtract module operation to store it in the memory unit.
	Structural VHDL Design	10	
	PCB Design	14	
	Ordering PCBs & Logistics	2	
	PCB Assembly & Testing	6	
Base-2 I/O Module	Logical Design	10	This is the last component of the ABC we will be making, and is used to directly input and output the contents of the memory drums.
	Structural VHDL Design	10	
	PCB Design	12	
	Ordering PCBs & Logistics	2	
	PCB Assembly & Testing	4	
Top-Level Assembly	Motherboard Design	30	This will be the final step in assembling all the units on a motherboard. This will take a significant amount of time, as debugging will be difficult and the overall design of
	Motherboard Testing	20	

			all the wires will be complex.

Figure 4. Expected Effort Table

Module	Estimated Effort	Actual Effort
Add/Subtract	41	20
Memory	42	70
Binary Conversion	32	50
Timing & Control * Includes Clock & Program Counter, Control System, and Control Panel	98	172
Base-10 I/O & Base-2 I/O * Covered by Decimal Output and Card Readers/Writers	90	98
Top-Level Assembly	50	40

Figure 5. Actual Effort Table

Two primary factors contributed to the misalignment between our expected and actual effort. One of these factors was the lack of familiarity with the design process to be able to accurately estimate how much effort would be required for any given module. The second factor was that we underestimated the amount of design work that needed to be done to expand on the previous team's work. For example, the estimated effort for the Timing & Control was far below the actual required effort due to a significant amount of historical research that had to be done, since the original diagrams for these modules were lost and the previous team used manual controls, and changing requirements for additional functionalities in the Clock & Program Counter module.

3.7. Other Resource Requirements

1. Technical Resources
    a. Breadboards and TTL Chips
        i. To reference and test.
    b. PCB Design Software
        i. Tools like KiCad designing custom printed circuit boards.
    c. Soldering Equipment
        i. For assembling PCBs with through-hole components.
    d. Logic Analyzers and Oscilloscopes
        i. For debugging signal flow and verifying timing.

    e. Power Supplies
        i. Bench power supplies or regulated DC adapters for powering modules.
    f. LEDs, Switches, and Buttons
        i. To replicate the ABC's original user interface and provide visual feedback.
    g. FPGA Development Boards
        i. For implementing the ABC logic in hardware.
2. Knowledge and Expertise
    a. Historical Documentation of the ABC
        i. Schematics, logic descriptions, and operational principles from the original and replica projects.
    b. PCB Fabrication Knowledge
        i. Understanding of layout constraints, signal integrity, and manufacturability.
    c. FPGA Programming Skills
        i. Familiarity with VHDL or Verilog for implementing logic modules.
3. Human Resources
    a. Client Expertise
        i. Weekly meetings with Dr. Stoytchev for guidance and historical accuracy.
    b. Peer Collaboration
        i. Team coordination for modular development and integration.
4. Software Resources
    a. Simulation Tools
        i. For verifying logic before hardware implementation, such as ModelSim
    b. Version Control Systems
        i. GitLab for managing code and documentation.
    c. Documentation Tools
        i. Google Drive and Google Docs for writing technical reports and user guides.
        ii. Draw.io for creating diagrams

4. Design

4.1. Design Context

4.1.1 Broader Context

Our recreation of the ABC exists within the context of historical preservation, engineering education, and the origins of digital technology. The design aims to make the ABC's

principles, such as binary arithmetic, electronic switching, and capacitive memory, tangible to modern learners. We are designing for educational audiences, primarily university students and instructors, as well as historians and anyone interested in the origins of digital computing. The project contributes to societal needs for STEM education, historical literacy, and preservation of technological milestones. By reconstructing the ABC, we are helping people appreciate the ingenuity of early computing pioneers and understand how their ideas have shaped today's digital world.

The project prioritizes safe, low-voltage electronic design suitable for classrooms. While it does not directly affect public health, it promotes public welfare through educational engagement and technological awareness. We will ensure all electrical components operate at relatively low voltages and are safe for public interaction. The documentation we will develop to explain how to operate our recreation will help make sure it is usable across classrooms and other interested engineers.

The ABC is an important piece of global computing history and a key part of Iowa State University's history of innovation. The project honors the contributions of Atanasoff and Berry while highlighting their work in the computing field. We will ensure our supporting documentation will be educational for learners and visitors of all backgrounds. This project will help create institutional pride and teach users about local contributions to world-changing technology.

Given its educational and small-scale nature, the project's environmental impact is minimal. Efforts will be made to minimize waste in construction and design the replica to be modular and repairable, extending its useful life in educational settings and reducing wasted parts.

The recreation will be designed to be as cost-effective as possible, given the technological complexity of staying true to the ABC's logical design. It will be reproducible by other institutions, schools, or museums thanks to our open-source documentation and construction with widely available electronic components. Its economic value lies in supporting education, research, and cultural tourism rather than commercial profit.

4.1.2 Prior Work/Solutions

Our recreation of the ABC is a continuation of a project from one of the senior design teams last year (sdmay25-29). Their project focused on doing a lot of discovery work to determine the logic behind each of the ABC's modules and how they operated together. The second semester of their project concluded with them making a small-scale recreation of the ABC on breadboards, successfully implementing most of the components of the original ABC [3]. Our work this year utilized the discovery work this team did in order to recreate all of the components of the ABC machine on PCBs and bring it up to its full computational scale (solving systems of 30 linear equations).

In the 1990s, a team of researchers at Iowa State attempted to create a one-to-one replica of the ABC, using vacuum tubes, rotating drums, and other mechanical devices to match the technology used by the original ABC [9]. While this recreation was primarily considered a success, it had an incredibly high budget ($300,000), and the final product was seemingly only able to solve carefully chosen sets of linear equations that are "easier" for the machine to solve. This recreation also severely lacks proper documentation and a technical manual describing how each component was implemented. This significant deficit in documentation of the details of the ABC's operation is something we addressed through our project.

Our project also has many parallels with the i281 Hardware senior design project from 2014. In this project, the team designed hardware to support the i281 CPU for Iowa State's CprE 281: Digital Logic course [1]. While this CPU is very different from the ABC, we followed a similar design workflow as this team: prototyping each component to understand how it functions before designing and fabricating PCBs for each module. In their case, this prototyping was done primarily through creating their logical designs on breadboards. We conducted most of our prototyping in VHDL to ensure thorough testing, reduce human error, and improve efficiency.

4.1.3 Technical Complexity

Our project is sufficiently technically complex due to the high degree of technical research it required and the large number of interdependent components we designed. For each component of the original ABC, we needed to investigate its behavior to determine the logical diagram behind it. Then, we developed our designs in VHDL and tested them to make sure they operated as expected before designing and manufacturing PCBs for each module.

This mix of logical design, programming, and PCB fabrication gave our project a wide breadth of technical complexity. The challenge of staying true to the ABC's original design while designing for modern components and bringing everything together to an operational, full-scale recreation was not be easy. However, we believe that through this unique challenge, we were be able to develop an operational final product that improved on and exceeded the current ABC replicas in existence today.

4.2. Design Exploration

4.2.1 Design Decisions

Described below are three design decisions that we made that are representative of the many choices we had to make and our decision-making process, as they cover broad system design, individual component design, and test design.

1. One key decision we had to make was how many bits each instance of the compute subsystem would handle. The system's bit width determines how many coefficients

the module can handle in parallel. Following the ideation and decision-making process described in the following sections, we decided to use a width of 5 bits and 6 components to meet the required 30 coefficients.

2. Another design decision that we had to make was which historical design to implement for the Decimal Output module. In the original ABC, this system underwent a later revision, resulting in two distinct, historically accurate designs: one described by Atanasoff and the other by Berry. Ultimately, we chose to use the earlier design described by Atanasoff, but designed the decimal output PCB such that it could be used in Berry's design with minimal modifications.
3. A third decision we made was to create a complete VHDL simulation of our designs and to assign a subgroup to develop it, using it to test our other designs. This decision proved highly useful later in the project, as we were able to conduct complete system tests before fully completing the hardware fabrication and assembly, which helped us perform more comprehensive testing within the available time and identify higher-level design flaws earlier.

4.2.2 Ideation

1. There were three main options available for this design decision. The first, and most obvious, choice was to have a 30-bit-wide compute system, one for each supported coefficient. With this choice, all thirty coefficients would be processed in parallel by a single piece of hardware. The second option was to use eight, four-bit-wide components, which would more closely follow the previous team's designs. The third option was to use six five-bit-wide components. This idea was inspired by the ABC's decimal card-reading process, which was limited by the size of punch cards that could hold only five coefficients.
2. There were two options to consider for the design of the decimal output display. The first design described by Atanasoff included a mechanical "borrow" signal, so that some digits would move the previous digit back by one. In this method, the even digits counted the number of subtractions performed during the display algorithm (which was one too many), and the odd digits recorded nine minus the number of additions performed and moved the previous digit back by one. In Berry's updated design, the "borrows" were removed, and the display algorithm was updated so that even digits counted one less than the number of subtractions, and odd digits counted ten minus the number of additions.
3. We had several options for testing our designs. The first option was to continue testing on breadboards, following along with the previous team's process. The second option was to use VHDL for each individual module, without integrating all these components into a full system. Extending this, we could also integrate the individual modules and run a full VHDL simulation of our designs. Finally, our designs could have been implemented on an FPGA for full system and component-level testing.

4.2.3 Decision-Making and Trade-Offs

1. The two main factors considered in this decision were historical accuracy and feasibility. The first option rejected was a 30-bit width, since it is simply too large to implement with the type of ICs and components we were required to use. This left the five- and four-bit options, both of which had some historical precedent. As described earlier, the ABC was split into sets of five to accommodate the limitations of the decimal punch cards. Selecting the five-bit design would thus closely follow this historical pattern. Selecting the four-bit design would support 32 coefficients. Although the original ABC had 32 rings on its drums, 2 were spares, and it was intended to operate with 30. Based on these factors, the five-bit design is more historically accurate. Regarding feasibility, the four-bit design is much simpler, as it aligns with the width of modern components, which are typically powers of two. With the five-bit design, we often had to use a second chip and only use one of its components to get the fifth bit. Since the five-bit option is more historically accurate but the four-bit option is easier to implement with modern components, we decided to do a rough cost analysis of the two designs. Although the five-bit option adds several more underutilized chips, the four-bit design requires two more components to reach the full scale. After analyzing the cost of each option and finding that the five-bit option was slightly cheaper, we decided to use a five-bit width.

5 Bit Option (6 Boards)				
Gate	Chip Name	Unit Price	Quantity	Part Total
2:1 Mux	SN74HCT257N	$0.58	8	$4.61
5 Bit Registar	SN74AHC174N	$0.58	3	$1.74
5 Bit Buffer	SN74HCT541N	$0.58	1	$0.58
6 Bit Full Adder	CD74HC283E	$0.80	2	$1.60
6 Bit RAM	AS6C62256-55PCN	$6.36	2	$12.72
4:1 mux	SN74HC153N	$0.68	3	$2.04
			Board Total	$23.29
			Chip Total	19
			6 Board Total	$139.74
			6 Board Chip Total	114

4 Bit Option (8 Boards)				
Gate	Chip Name	Unit Price	Quantity	Part Total
2:1 Mux	SN74HCT257N	$0.58	5	$2.88
5 Bit Registar	SN74AHC174N	$0.58	3	$1.74

5 Bit Buffer	SN74HCT541N	$0.58	1	$0.58
6 Bit Full Adder	CD74HC283E	$0.80	2	$1.60
6 Bit RAM	AS6C62256-55PCN	$6.36	2	$12.72
4:1 mux	SN74HC153N	$0.68	2	$1.36
			Board Total	$20.88
			Chip Total	15
			8 Board Total	$167.06
			8 Board Chip Total	120

Figure 6. Bit-Width Cost Analysis

2. Both design options can be considered historically accurate, so the decision process focused primarily on implementation feasibility. Since the exact workings of the algorithms and control signals for both designs are not well documented, several simulations were written in Python to reconstruct how the decimal output module worked for both designs. This process yielded two important findings that informed our decision-making. The first is that the design described by Atanasoff would require adding an additional signal to the timing drum that was not present in the historical diagram of this component. Secondly, the design described by Berry had certain edge cases that did not work correctly, but the Atanasoff design handled them correctly. Given the greater reliability of the earlier design described by Atanasoff, we decided to implement it. More details about the two different designs can be found in Appendix 2.
3. Unlike the other design decisions, where only one option was chosen, this decision offered opportunities to implement multiple options into our design process. For example, although breadboard testing was rejected for our overall design testing method, since it would be infeasible at the scale of our project, it was still incorporated into the prototyping and unit testing processes of individual modules. Similarly, stopping at VHDL units, despite requiring less effort, would not have enabled large-scale system testing and was not selected as our overarching test design; it was used to unit test the designs of several individual modules. FPGA and full VHDL simulation testing were the only feasible options for performing the system testing that we desired. However, we discovered some limitations of the FPGA testing method, particularly around data input, that would have limited the testability of some features. As a result, we decided on a full VHDL simulation as a core part of our test design.

4.3. Final Design

4.3.1 Overview

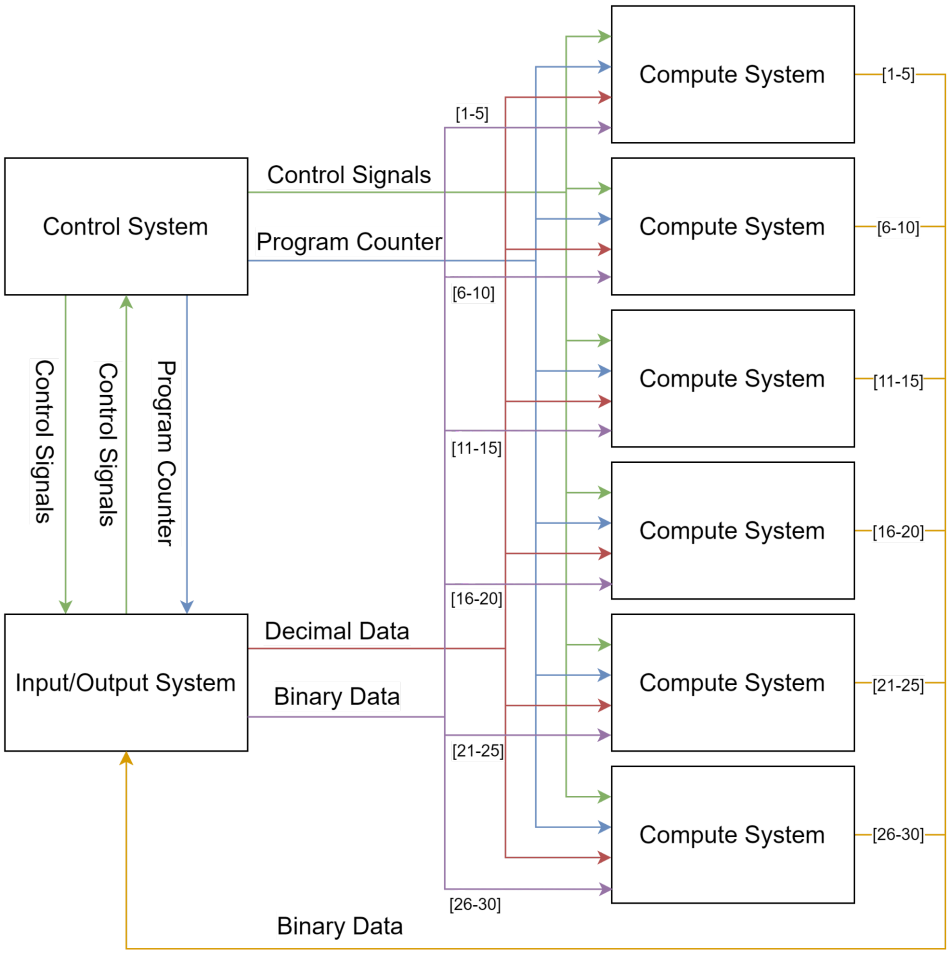


Figure 7. System-Level Diagram

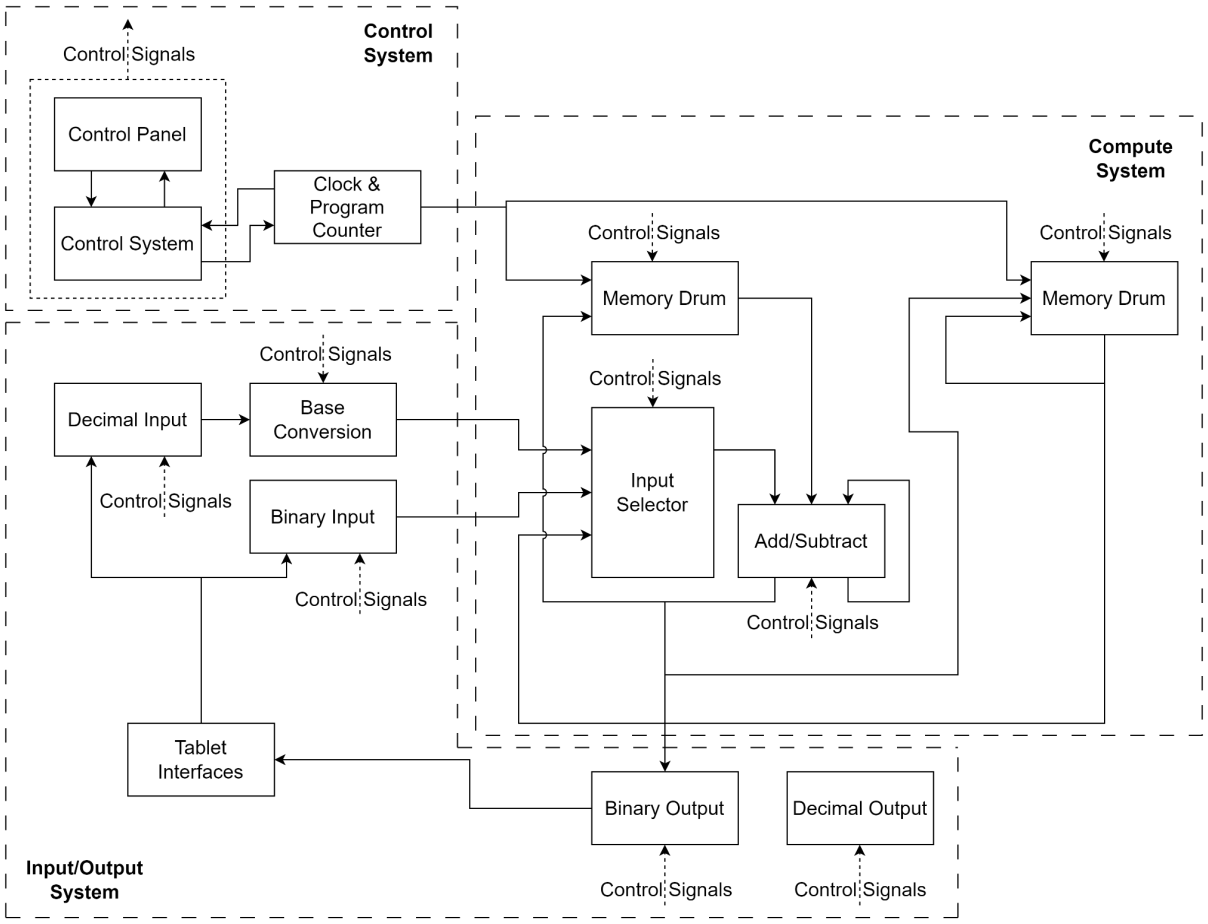


Figure 8. High-Level Block Diagram

The design of our ABC replica is made up of three main systems: the control system, the Input/Output system, and the compute system. Each compute system is identical and merely increases the number of supported variables. The control system is responsible for coordinating and timing the other systems for performing different tasks, and is made up of a Control Panel for user input, the Control System for all of the control logic, and the Clock & Program Counter for timing. The input/output system contains all of the interfaces and supporting modules for inputting data to the computer and receiving the results of computation. Both input and output data can be in decimal or binary form. The compute system is the core of the computer and is responsible for storing the coefficients of the two equations being eliminated and performing the arithmetic operations for the elimination algorithm. This system contains two memory drums, one for each equation, an Add/Subract module for performing computation, and an input selector to control input to the Add/Subtract module.

4.3.2 Detailed Design and Visual(s)

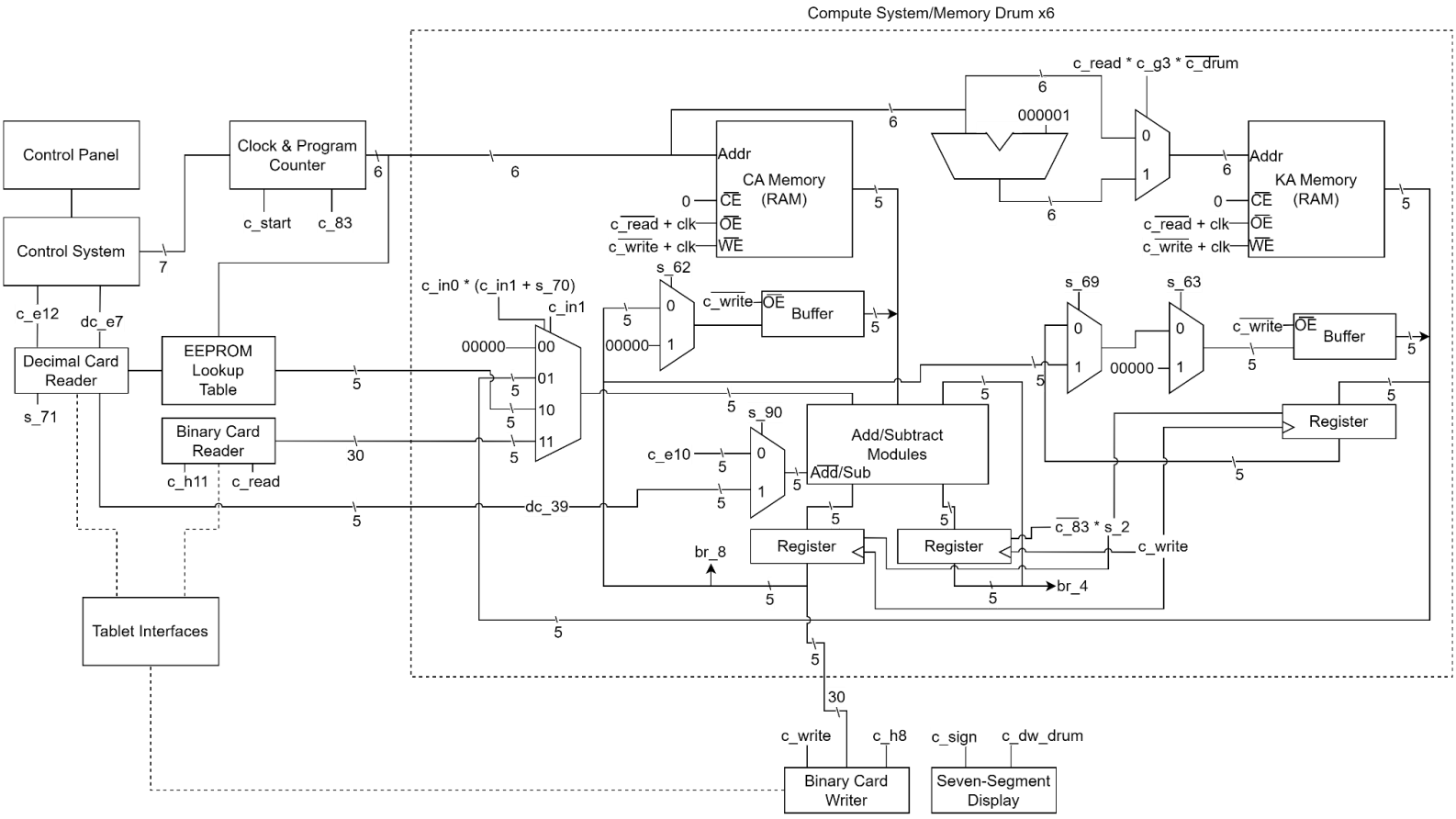


Figure 9. Detailed Block Diagram

Control Panel

Purpose

The purpose of this component is to provide a user interface for controlling the system's operation and viewing its current state via LED indicators.

Input and Output Signals

Input Signals

- c_e4 - [LED] Selected operation is addition
- c_e5 - [LED] Selected operation is subtraction
- c_f4 - [LED] Selected coefficient is negative

- c_f5 - [LED] Selected coefficient is positive
- c_i7 - [LED] Computing is enabled
- c_h11 - [LED] Binary reading is enabled
- c_dec_read - [LED] Decimal reading in progress
- c_dec_write - [LED] Decimal writing in progress
- clk - Clock for debouncing buttons
- br_4[1-30] - The carry out signal from all add/subtract units
- br_8[1-30] - The sum/difference signal from all add/subtract units

Output Signals

All output signals beginning with "s_" are controlled by a switch. Signals beginning with "b_" are controlled by a debounced button.

- s_70[a-f] - One-hot-encoded output to enable a single bank of five add/subtract units
- s_68 - Controls data input to the add/subtract units; low for stored data and high for card data
- s_90 - Controls which system controls the selected arithmetic operation; low for the control system and high for the decimal card reader
- s_71 - Only read the 1's row of the decimal card; this is used to read in the powers of ten for the decimal output algorithm
- s_62 - Clear the CA memory
- s_63 - Clear the KA memory
- s_69 - Transfer the contents of CA memory to KA memory
- s_2 - System reset, active low
- s_1 - System power
- s_89 - Enable the decimal output module
- b_84 - Toggles the selected arithmetic operation
- b_85 - Starts decimal card reading
- b_86 - Starts binary card reading
- b_87 - Starts computation or the decimal output process
- b_88 - Starts binary card writing
- br_4 - The carry signal from the coefficient to eliminate or read out on the decimal output module; selected from the inputs with a jumper
- br_8 - The sum/difference from the coefficient to eliminate or check the signb_85

Design

The LEDs, buttons, and switches are arranged to match the layout of the original ABC.

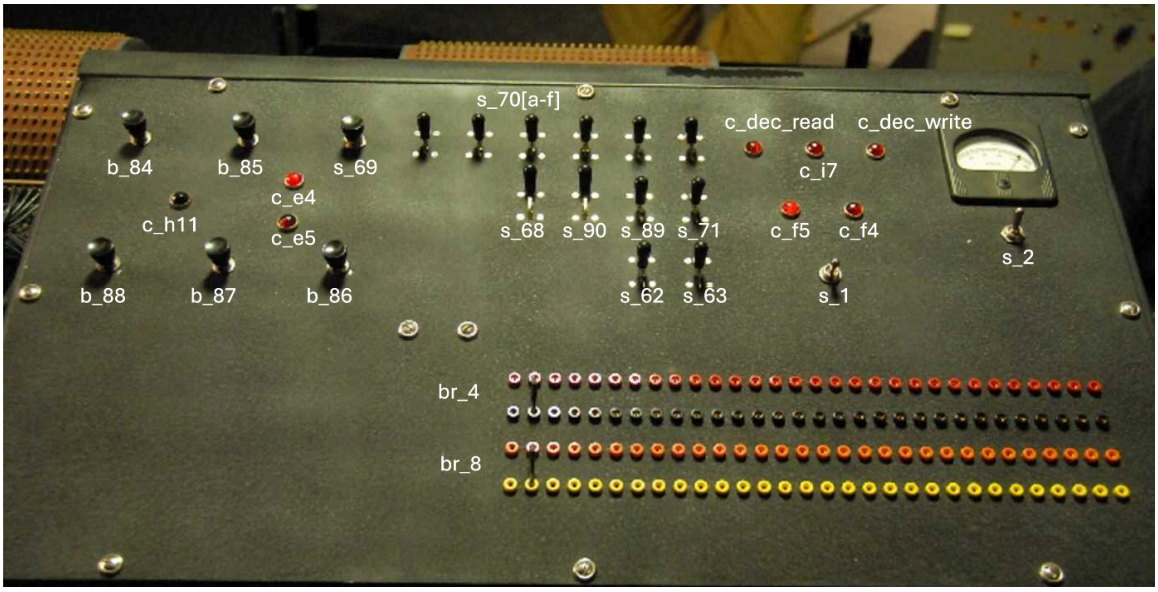


Figure 10. Original ABC Control Panel

Full Control System

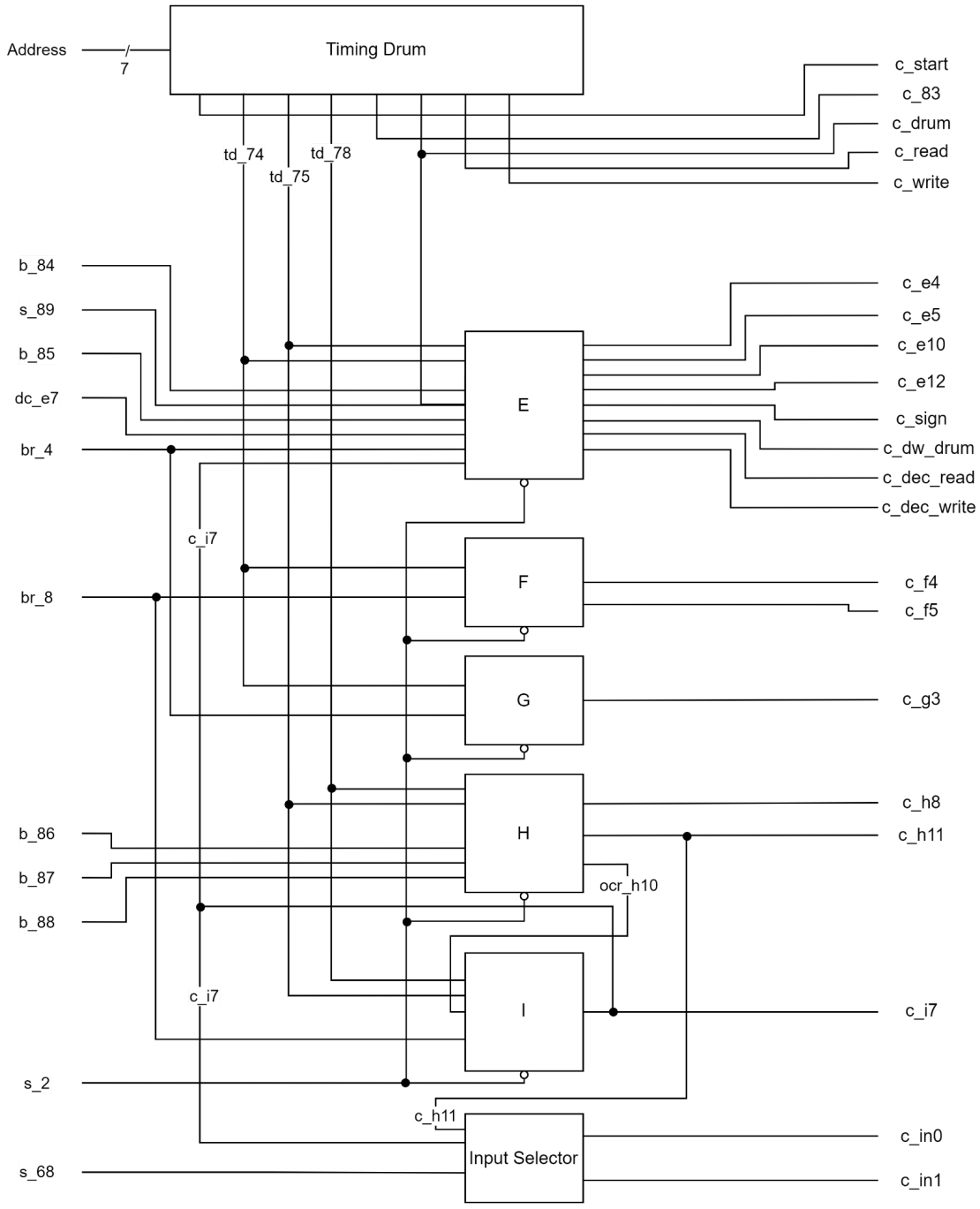


Figure 11. Control System Module

Unit E: Add-Subtract Control

Purpose

The purpose of this component is to automatically control which operation, addition or subtraction, the add-subtract modules are performing during the elimination process and the reading out of decimal numbers. This module also controls moving the decimal card reader to the next column when reading decimal input or outputting decimal numbers. Finally, this module controls the decimal output module.

Input and Output Signals

Input Signals

- Reset_n (s_2) - The default state is to perform addition and disable moving the card reader
- Switch operations (b_84)
- Sign change (br_4)
- +/- Control (td_74)
- Start decimal reading (b_85)
- Escapement (td_75)
- End of card_n (dc_e7)
- Decimal output mode (s_89)
- Computing enable (c_i7)
- Drum rotation (c_drum)

Output Signals

- Add (c_e4)
- Subtract (c_e5)
- Add-Subtract (c_e10)
- Decimal card shift (c_e12)
- Sign change (c_sign)
- Drum rotation (c_dw_drum)
- Decimal reading (c_dec_read)
- Decimal writing (c_dec_write)

Description of Operation

Generating the Add-Subtract Signals

The *Sign change* and +/- *Control* signals are combined with an AND gate to ensure that only the last carry/borrow out from the add-subtract mechanisms is used, since the +/- *Control* signal will only be logic high after the last arithmetic operation. When the number

being added to or subtracted from the number stored in CA memory is positive, a carry/borrow out of 1 indicates that the sign of the number stored in CA memory has changed, so the computer needs to switch from addition to subtraction and vice versa. Thus, a logic high on the combined *Sign change* and +/- *Control* signals toggles the controller's internal state, which determines which operation to perform. Pressing the *Switch operations* button also toggles this internal state. If the internal state is 0, the operation is addition, so *Add* is set to 1 and *Subtract* and *Add-Subtract* are set to 0. If the internal state is 1, the operation is subtraction, and *Add* is set to 0 and *Subtract* and *Add-Subtract* are set to 1.

Shifting the Decimal Card Reader

Pressing *Start decimal reading* enables shifting the card reader to the next column. If this is enabled, the *Escapement* signal is passed through to the decimal card reader as *Decimal card shift*; otherwise, it is ignored. The decimal card reader disables passing this signal through once the end of the card is reached via the *End of card* signal. Alternatively, if *Decimal output mode* and *Computing enable* are both high, which indicates that a decimal output is in progress and sets *Decimal output enable*, then the combined *Sign change* and +/- *Control* signals, which transmit the last carry/borrow out, are sent to the decimal card reader as *Decimal card shift*.

Decimal Output Control

The main function of the part of the controller is to only pass through the *Drum rotation* and combined *Sign change* and +/- *Control* signals to the *Drum rotation* and *Sign change* outputs when a decimal output is in progress. To do this, the input signals are gated by the same combined *Decimal output mode* and *Computing enable* signal used in advancing the decimal card reader.

Design

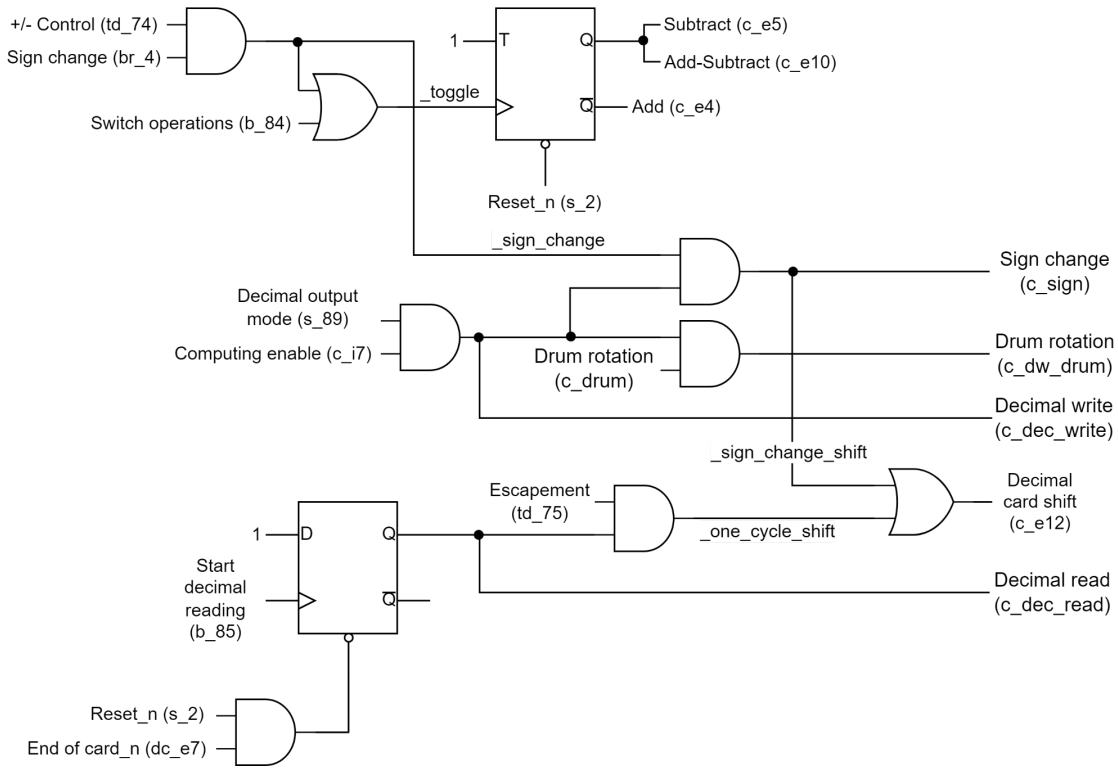


Figure 12. Add-Subtract Control Design

Timing Diagrams

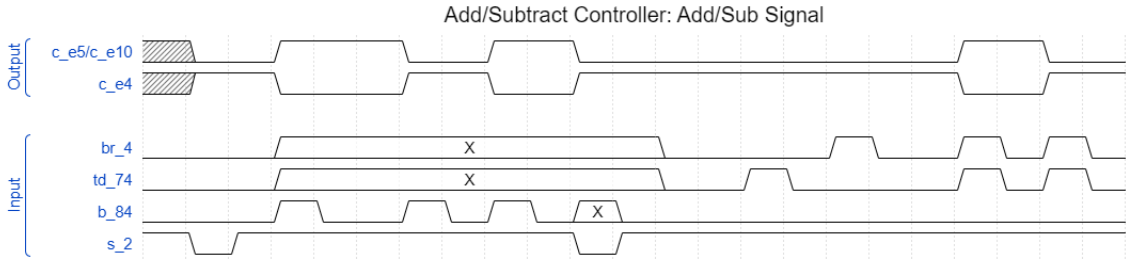


Figure 13. Add-Subtract Control Timing 1

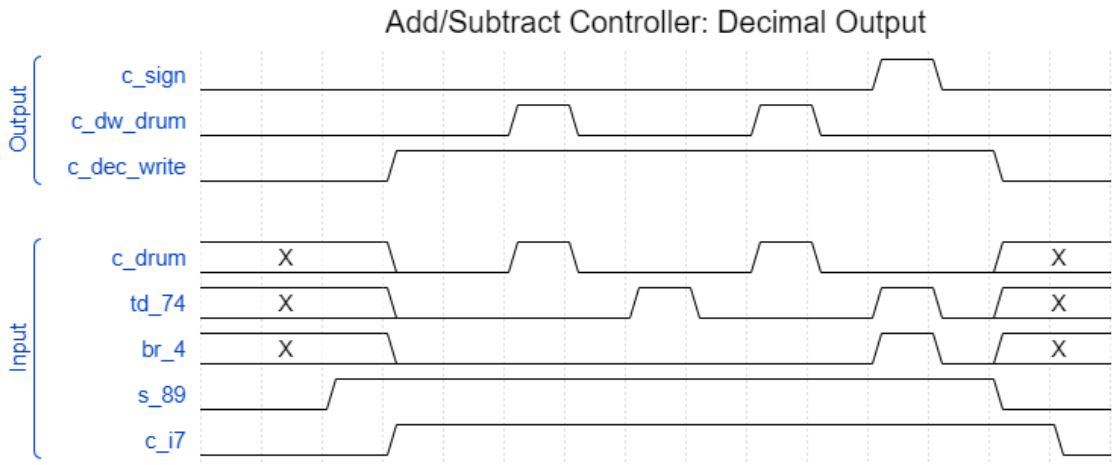


Figure 14. Add-Subtract Control Timing 2

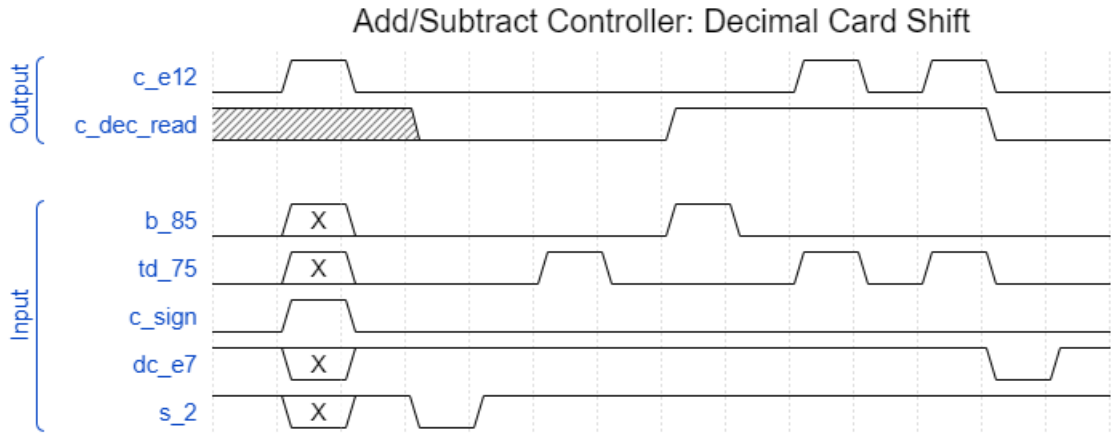


Figure 15. Add-Subtract Control Timing 3

Unit F: Negative Number Indicator

Purpose

The purpose of this component is to display the sign of the selected number that is stored in the CA memory.

Input and Output Signals

Input Signals

- Reset_n (s_2) - The default state is to indicate a negative number
- Negative number indicator (td_74)
- Bit input (br_8)

Output Signals

- Negative (c_f4)
- Positive (c_f5)

Description of Operation

The primary functionality of this controller is to store the most significant bit of the selected coefficient in CA memory, indicating the number's sign. On the positive edge of *Negative number indicator*, this controller reads the value to be written back to CA memory on *Bit input*. *Negative number indicator* is timed such that this will happen when the register *Bit input* reads from contains the most significant bit of the number. This will be overwritten each full cycle of the machine (0-59).

Design

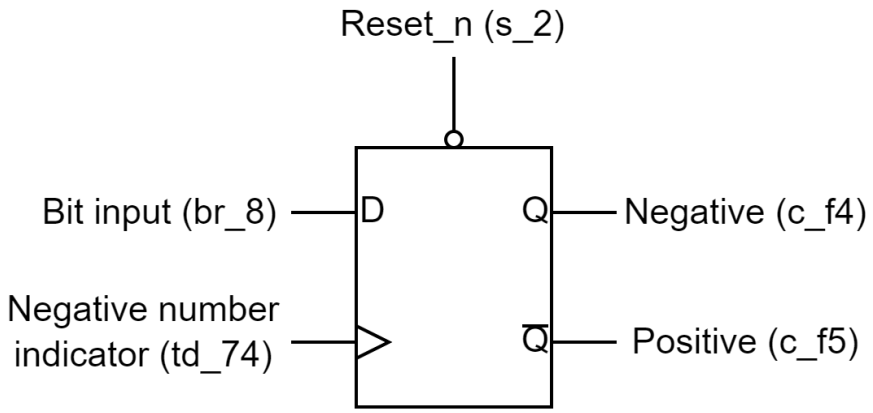


Figure 16. Negative Number Indicator Design

Timing Diagram

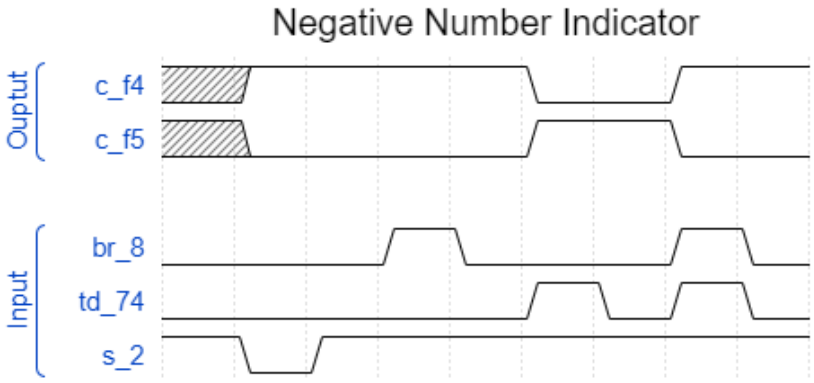


Figure 17. Negative Number Indicator Timing

Unit G: Shift Control

Purpose

The purpose of this component is to automatically instruct the hold/shift mechanism when to hold and when to shift.

Input and Output Signals

Input Signals

- Reset_n (s_2) - The default state is to hold
- Shift control (td_74)
- Sign change (br_4)

Output Signals

- Shift (c_g3)

Description of Operation

The primary functionality of this controller is to store the last carry/borrow out from the add-subtract mechanisms of the selected coefficient, indicating a sign change as described earlier. On the positive edge of *Shift control*, this controller reads the value in the carry drum register on *Bit input*. *Shift control* is timed such that this will happen when the register *Bit input* reads from contains the carry/borrow out of the last operation. This will be overwritten each full cycle of the machine (0-59).

Design

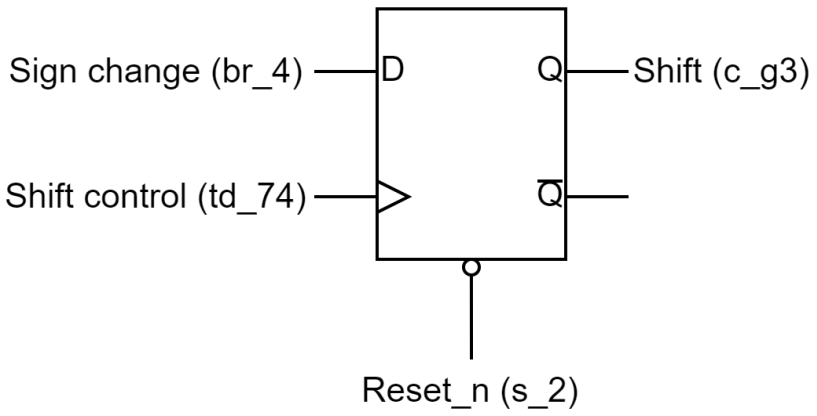


Figure 18. Shift Control Design

Timing Diagram

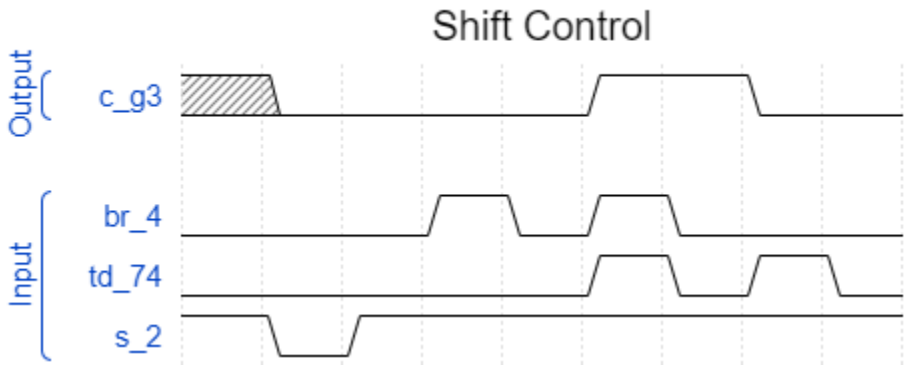


Figure 19. Shift Control Timing

Unit H: One-Cycle Relays

Purpose

The purpose of this component is to ensure that certain operations begin at the start of a full cycle of the machine, regardless of when the user requested the operation, and to perform those operations for a single cycle of the machine. This controller provides this functionality for binary card writing, binary card reading, and elimination.

Input and Output Signals

Input Signals

- Reset_n (s_2) - The default state is to have all outputs disabled
- Start binary write (b_88)
- Start binary read (b_87)
- Start elimination (b_86)
- One-cycle switch open (td_75)
- One-cycle switch close (td_78)

Output Signals

- Binary write enable (c_h8)
- Binary read enable (c_h11)
- Elimination enable (ocr_h10)

Description of Operation

For each pair of input buttons and output enable signals, the operation is identical. When a button is pressed, a signal is stored to indicate that a start has been requested. Nothing

happens until the *One-cycle switch close* signal becomes high, indicating that a system cycle is about to begin, and the corresponding output signal is set high. When the output signal is high and the *One-cycle switch open* signal becomes high, indicating the end of a system cycle, the state for this pair is reset, disabling the output.

Design

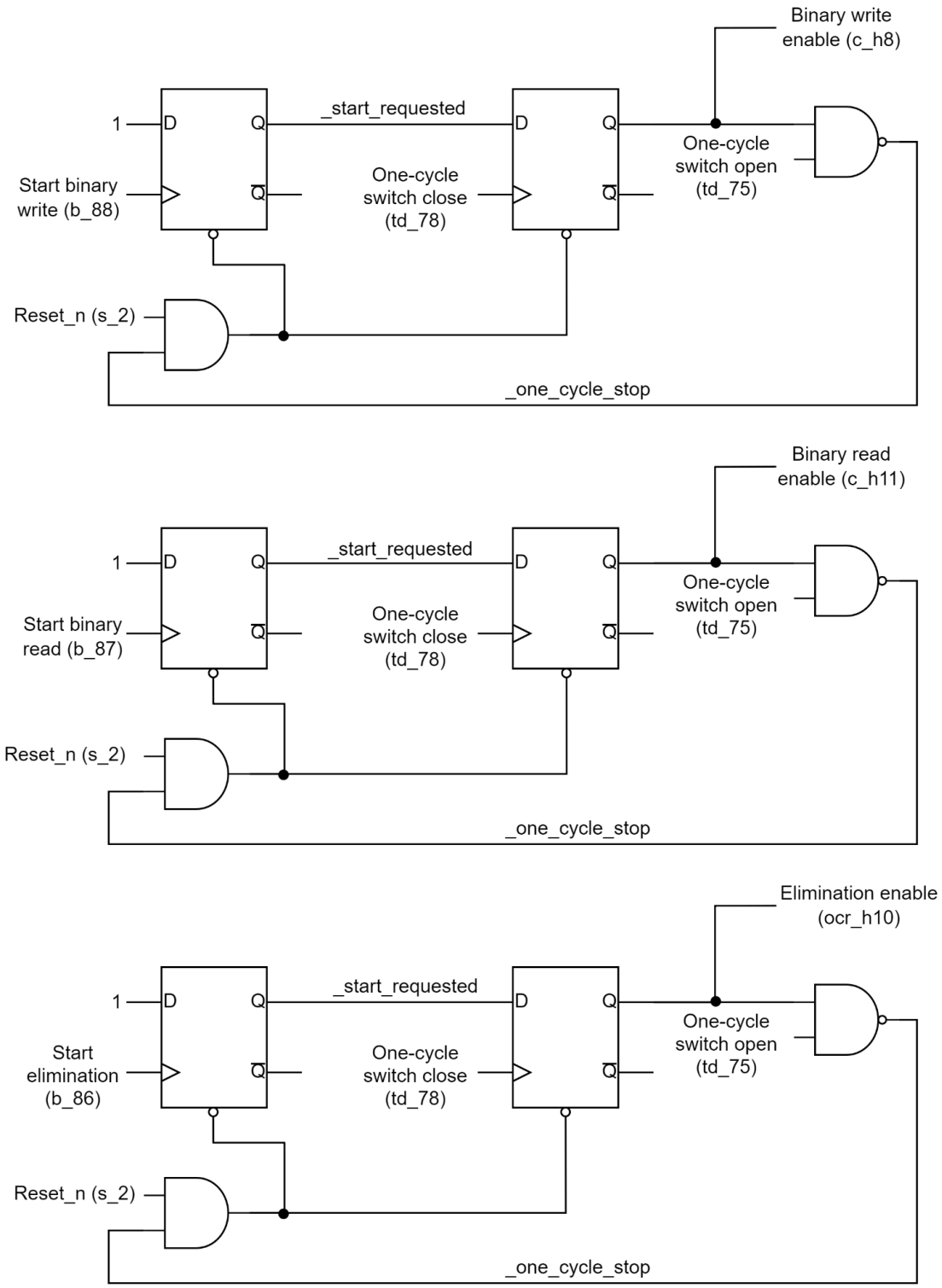


Figure 20. One-Cycle Relays Design

Timing Diagram

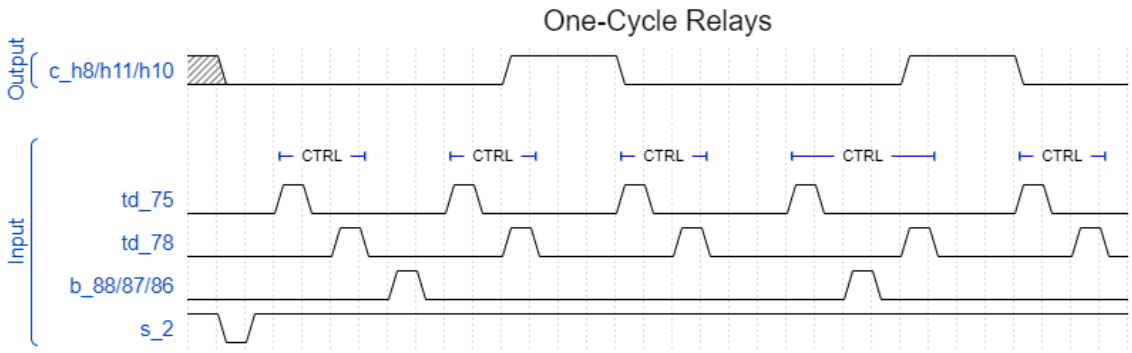


Figure 21. One-Cycle Relays Timing

Unit I: Division Control

Purpose

The purpose of this component is to start computation when requested and stop the elimination process once the selected coefficient has been reduced to zero.

Input and Output Signals

Input Signals

- Reset_n (s_2) - The default state is to stop the computing process
- Bit input (br_8)
- Elimination enable (ocr_h10)
- Division control open (td_75)
- Division control close (td_78)

Output Signals

- Computing enable (c_i7)

Description of Operation

Starting Computation

When *Elimination enable* becomes 1, the controller updates its state to indicate that computation has started. Once computation has started, *Computing enable* is set to 1, and the process continues until a zero is detected. *Running* always displays the same value as *Computing Enable*.

Detecting Zero

The controller continuously receives the new value stored in the CA memory via the *Bit input* signal. If, at any point during computation, this signal becomes high, indicating a 1 in the result, the controller updates its state to indicate that the number is not zero. A 1 indicates the number is not zero, and a 0 indicates the number is zero. When *Division control open* becomes high at the end of a system cycle, this state is used to determine if computation should continue. When *Division control close* becomes high at the start of a new system cycle, this internal state is reset to 0. Thus, the controller by default stores that the number is zero unless it receives a 1 on *Bit input* during computation.

Stopping Computation

During the checking period (between the *Division control open* and *Division control close* signals), if the controller's state indicates that the number is not zero, then computation will continue as normal. If the state indicates that the number is zero, then *Computing enable* is set to 0, and the state indicating that computation has started is reset to 0.

Design

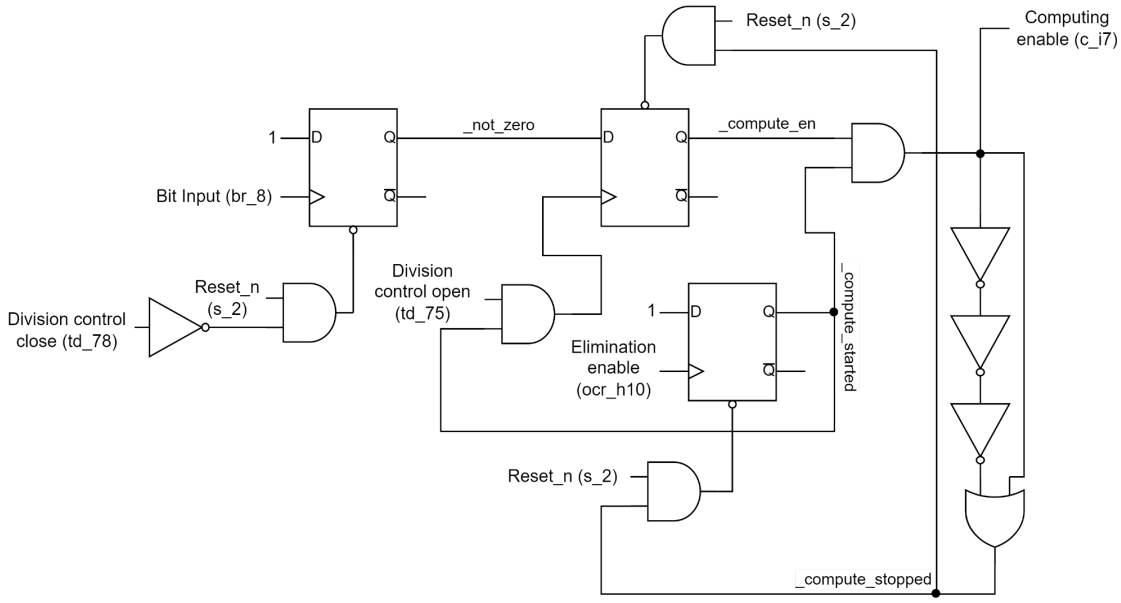


Figure 22. Division Control Design

Timing Diagram

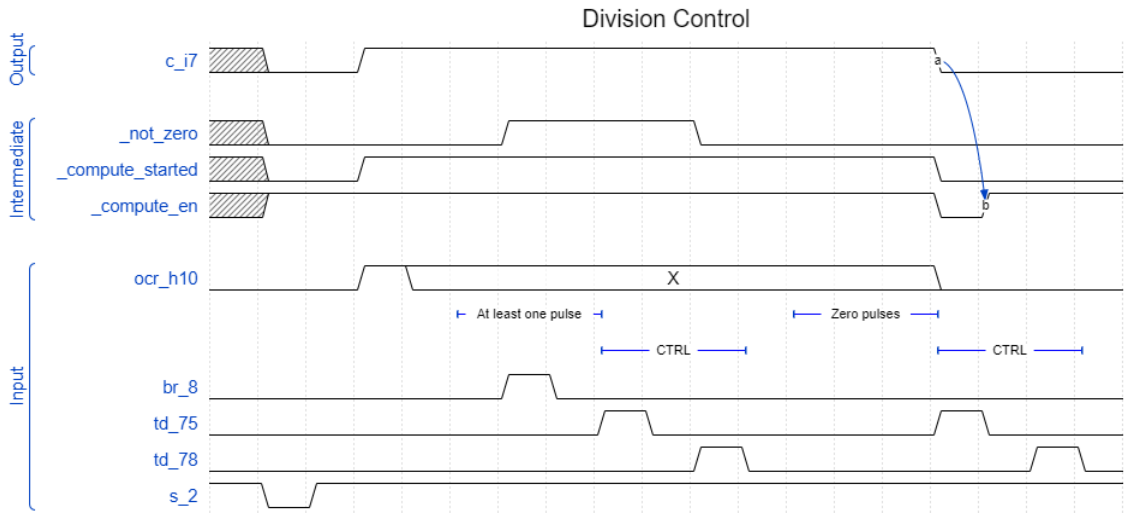


Figure 23. Division Control Timing

Timing Drum

Purpose

The purpose of this component is to provide the signals that other controllers use to ensure they act at the correct time in the full cycle of the machine.

Input and Output Signals

Input Signals

- Drum position - a 7-bit signal from the program counter

Output Signals

- c_start
- td_74
- td_75
- td_78
- c_83
- c_drum
- c_read
- c_write

Description of Operation

This module uses an EEPROM lookup table to output the appropriate values to all output signals based on the current *Drum position*.

Design

See the timing drum spreadsheet here for the EEPROM table contents.

Input Selector

Purpose

The purpose of this component is to generate the control signals for the multiplexer that selects the input to the add-subtract mechanisms.

Input and Output Signals

Input Signals

- s_68
- c_i7
- c_h11

Output Signals

- c_ino
- c_in1

Description of Operation

This component is split between the control system and the individual, 5-bit memory drum components. The circuit in the control system generates preliminary versions of *c_ino* and *c_in1*, and the portion on the memory drums incorporates their respective *s_70x*. The four inputs to the add-subtract mechanisms are given the following positions on the multiplexer:

- Zero: 00
- KA: 01
- Decimal card: 10
- Binary card: 11

The *s_68* signal switches between card inputs (either decimal or binary) and KA memory inputs (or zero if elimination has stopped), so it is used directly as *c_ino*. This signal is also

used to control a multiplexer that switches between *c_i7* and *c_h11*. If *s_68* is 0, then *c_in1* is *c_i7*. In this state, the input will either be zero or from the KA, depending on whether the computer is eliminating. If *s_68* is 1, then *c_in1* is *c_h11*. In this state, the input will come from the decimal or binary card, depending on whether the binary card reader is enabled. On the memory drum, *c_in0* is modified such that when *s_70x* is off for a given memory drum and the decimal card input is selected, the selected input becomes zero. This ensures that the contents of a decimal card are read into a single 5-bit bank.

Design

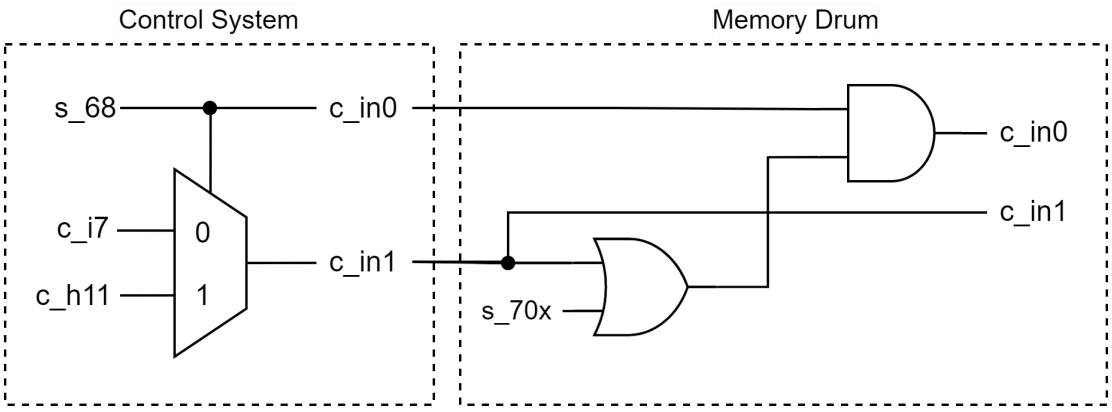


Figure 24. Input Selector Control Design

Clock & Program Counter

Purpose

The purpose of this module is to provide a clock signal for debouncing and timing, and a program counter to indicate the system's position within a computation cycle.

Input and Output Signals

Input Signals

- s_2 - System reset
- c_start - Indicates the start of a computation cycle
- c_83 - Indicates the end of a computation cycle

Output Signals

- clk - Clock signal
- Program counter - 7-bit number (0-119)

Description of Operation

A computation cycle consists of all the operations required to perform one arithmetic operation and to update the control signals. Each cycle consists of 60 positions, each with a read and write phase, for a total of 120 different parts. The first 50 positions are used for computation, one for each bit of the stored numbers, and the last 10 positions make up the control period. This module provides the *Program counter* to indicate the current position in the control cycle. Some modules, such as the Control System need to know the exact position and phase so they use the entire 7-bit output. Other modules, like the memory drum, only need to know the position, since they receive information about the phase from the control system, so they use the upper six bits of *Program counter*. The *clk* signal oscillates at a frequency twice that of the least significant bit of *Program counter*.

To aid debugging and demonstration, this module supports multiple operating frequencies and modes. The oscillator output is sent through a series of counters to divide the frequency into several options, which can be selected with a rotating switch. There are switches to halt the module, manually reset it, and enable manual or batch mode. In manual mode, inputs from a button can be used as *clk* and for incrementing *Program counter*. In batch mode, the module completes one computation cycle and then halts.

Design

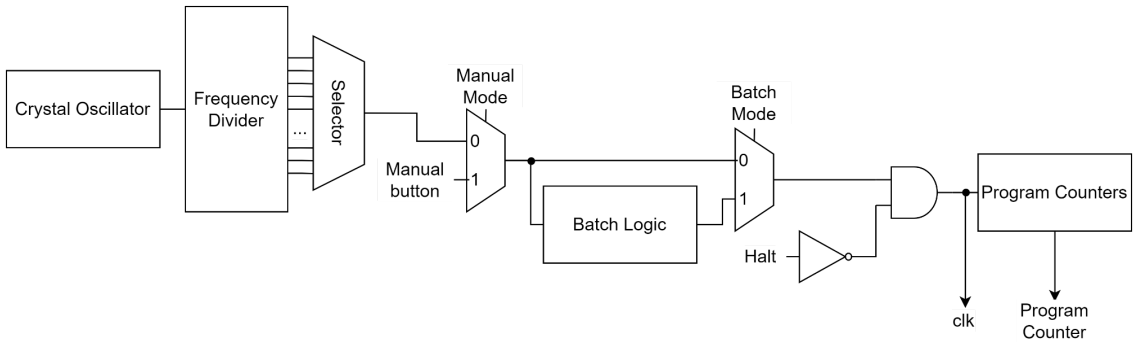


Figure 25. Clock & Program Counter Design

Card Readers/Writers

Purpose

The purpose of the base conversion module is to convert the decimal numbers read from decimal punch cards into binary numbers that the ABC can perform calculations with.

Input and Output Signals

Input Signals

- c_read - Read phase
- c_write - Write phase
- c_e12 - Shift the decimal card reader
- c_h8 - Binary card writer enable
- c_h11 - Binary card reader enable
- s_71 - Only read the 1's digit of the decimal card; used to read in the powers of ten for the decimal output algorithm
- Binary write data - The 30-bit column to write to a binary card

Output Signals

- dc_e7 - End of decimal card
- dc_39[a-e] - The sign of each decimal card number
- Digit_C[1-5] - The 4-bit binary representation of the digit (0-9) on the decimal card
- Decade - The 4-bit binary representation of which power of 10 the digit is for
- Binary read data - The 30-bit column of the binary card

Description of Operation

Decimal Card Reader

Each decimal card contains 80 columns of the digits 0 through 9. These 80 columns are divided into 5 groups of 16, so the computer can read in 5 different coefficients simultaneously. The first column of each section is used to determine the sign of the coefficient. If the 0 space is punched, then the number is negative. For all other columns, if the digit is 0, no space is punched; otherwise, the appropriate space for the digit is punched. The decimal card reader initially outputs all zeros for *Decade* and *Digit_C[1-5]*. On the first high pulse of *c_e12*, dc_39[a-e] is set for each of the five coefficients on the card. This signal controls the arithmetic operation performed by the Add/Subtract units. If the number is positive, the signal will be low, which will add the number to the memory contents (which should have been cleared to zero). If the number is negative, the sign will be high, subtracting the number from the memory contents. Further pulses of *c_e12* will decrement *Decade* (set to 15 on the second pulse) and set *Digit_C[1-5]* to the digit in the next column of the decimal card. When the card reader receives a pulse on *c_e12* that would decrement *Decade* to -1 the *Decade* and *Digit_C[1-5]* outputs are cleared, and *dc_e7* is set to high until the card reader is reset. If *s_71* is high during this process, the card reader will only set the output of *Digit_C[1-5]* if the punched digit is 1. This is used along with special mask cards that have all 15 1's punched out for only one of the five coefficients to read in the powers of ten used to output the selected coefficients on the decimal display.

Binary Card Reader

Binary cards are 30-by-50 grids that represent the entire contents of a memory module. If the content of memory is a 1, then there will be a hole punched at that location; otherwise, there will be no punch. Initially set to all zeros, the binary card reader will set *Binary read data* to the value of the next column of the binary card when it receives a high pulse on *c_read*. After the end of the card is reached, *Binary read data* will be set to all zeros. If *c_h11* is low, *c_read* is ignored, and the card reader will not output any data.

Binary Card Writer

The binary card writer operates the same as the binary card reader, except that on a high pulse on *c_write*, the value of *Binary write data* is read in and stored. The card reader will not read in data unless *c_h8* is high.

Design

Each card reader or writer is an ESP32 development board running software that implements the described functionality. Each ESP32 is also paired with an Android tablet that provides a graphical interface for viewing, creating, or selecting the cards. For decimal cards, users can punch a card in the Android app and then upload it to the ESP32 over BLE. For binary cards, when the binary card writer receives all the binary card data, it transmits it to the binary card writer Android app via BLE, where it is visualized as a grid of punches. This Android app will then relay the data using traditional Bluetooth to the binary reader Android app. The binary reader Android app will store written cards, which can be selected for BLE upload to the binary card reader ESP32.

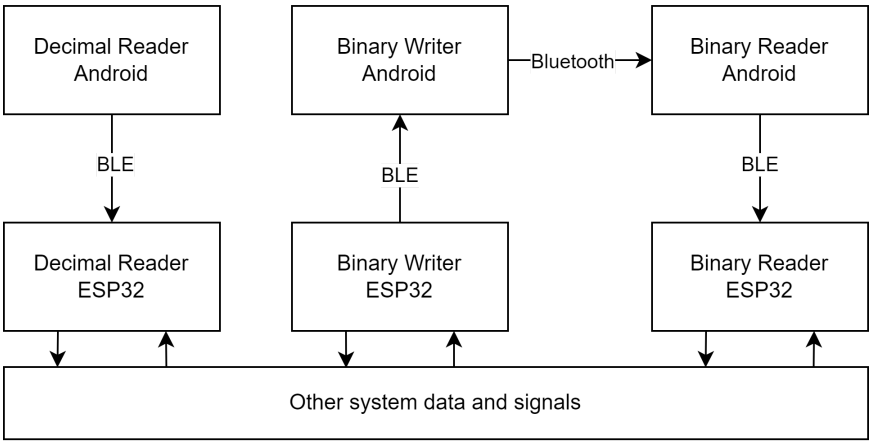


Figure 26. Card Readers/Writers Design

Base Conversion Module

Purpose

The purpose of the base conversion module is to convert the decimal numbers read from decimal punch cards into binary numbers that the ABC can perform calculations with.

Input and Output Signals

Input Signals

- Digit_C[1-5] - The 4-bit binary representation of the digit (0-9) from the decimal card reader
- Decade - The 4-bit binary representation of which power of 10 the digit is for
- Drum - The current address for the memory drums

Output Signals

- Out_C[1-5] - The current bit of the converted binary number

Description of Operation

This component converts 5 numbers in parallel, one for each coefficient on the decimal punch cards. The conversion module operates on the principle that any n-digit decimal number can be represented in the form $d_{n-1} \times 10^{n-1} + d_{n-2} \times 10^{n-2} + \ldots + d_1 \times 10^1 + d_0$. For this component, *Digit_Cn* is d_i and *Decade* is the corresponding exponent of 10.

Instead of actually computing $Digit_Cn \times 10^{Decade}$ and then converting that number into binary, a lookup table stored in an EEPROM is used. Since the Add/Subtract mechanisms operate in a bit-serial fashion, only one bit of the converted binary number can be output at a time, so the *Drum* signal is used to select which bit to output based on the active memory address. Thus, if *Drum* was equal to decimal 10, meaning the result of the computation would be written to the tenth row of memory, then the tenth bit of the converted number would be output to *Out_Cn*. For the EEPROM lookup table address, the four most significant bits are *Digit_Cn*, the next four bits are *Decade*, and the final six bits are *Drum*. Using these lookup tables, the conversion module can thus output a bit-serial binary version of $d_i \times 10^i$.

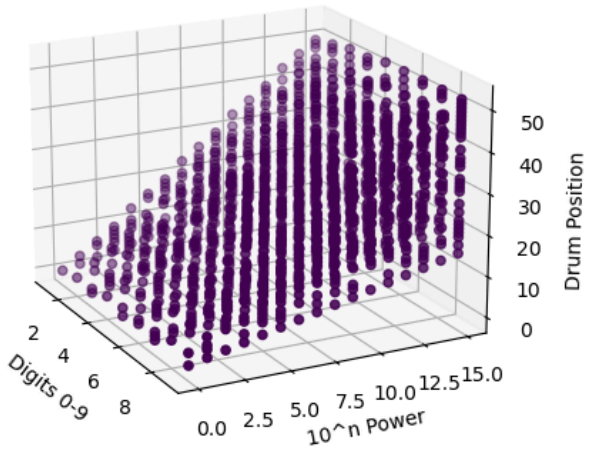


Figure 27. Base Conversion Table Visualization

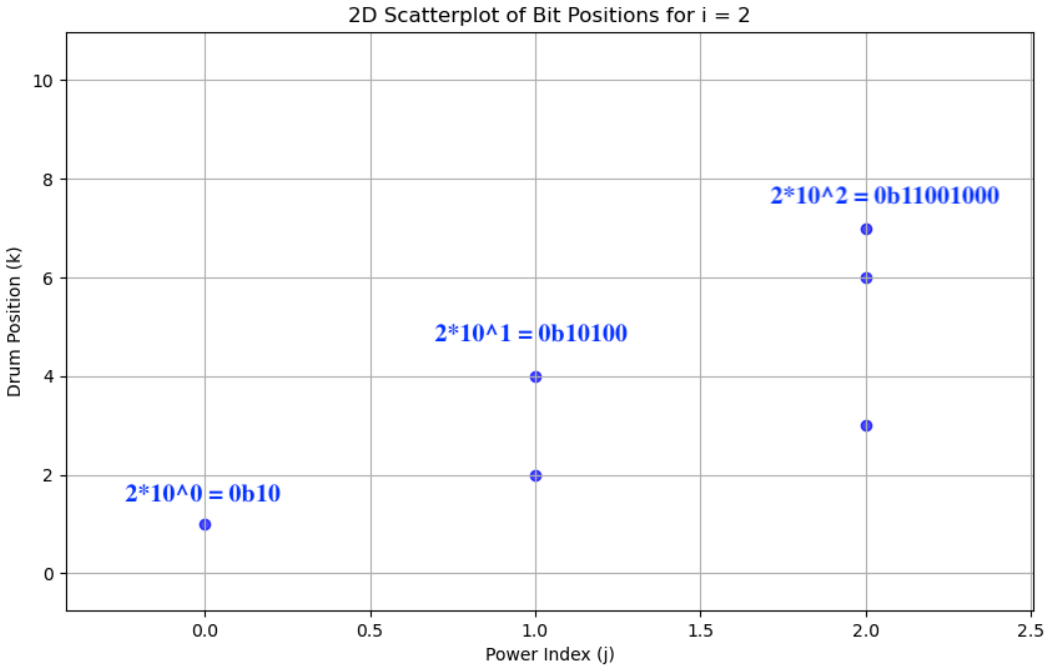


Figure 28. Base Conversion Table Slice

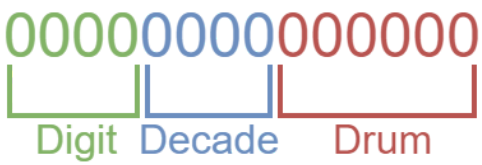


Figure 29. Base Conversion EEPROM Address Format

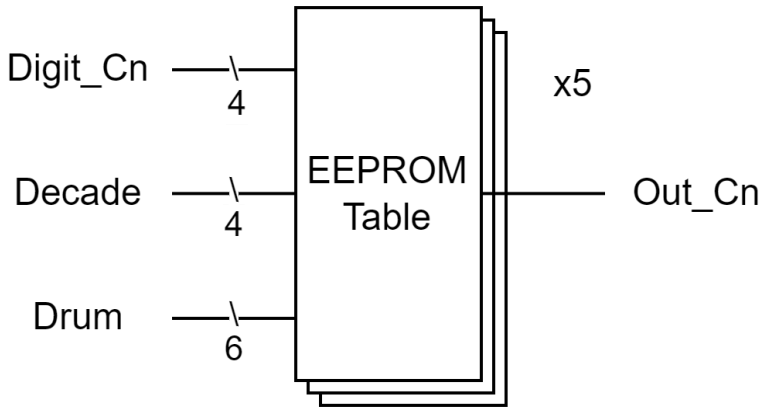


Figure 30. Base Conversion Module

Decimal Output Module

Purpose

The purpose of the decimal output module is to display one of the stored coefficients in human-readable decimal format. The module can display up to 15-digit decimal numbers

Input Signals

Input Signals

- c_sign - Indicates that a sign change occurred during the last operation
- c_dw_drum - Indicates that the last arithmetic operation has completed

Description of Operation

This module operates on the principle that any n-digit decimal number can be represented in the form $d_{n-1} \times 10^{n-1} + d_{n-2} \times 10^{n-2} + \ldots + d_1 \times 10^1 + d_0$. Starting with $n = 15$, the machine repeatedly subtracts 10^{n-1}, which is read in from the base conversion module using a special mask card on the decimal input module, until the value of the selected coefficient changes sign, indicated by *c_sign*. Each performed operation, indicated by *c_dw_drum*, increments or decrements the $n - 1$ digit display. Where n is odd, the display counts up, and where it is even, the display counts down (wrapping from 0 to 9). When a sign change is detected, n is decremented and the arithmetic operation is switched from subtraction to addition and vice versa. This process is repeated until an operation results in the stored value becoming 0. Switching operations after a sign change is detected will result in 10^{n-1} being subtracted one too many times. To correct the displays for these digits (the ones that count up), the borrow signal generated by the next display wrapping

from 0 to 9 is used to decrement them. To display a negative number, the process is the same, except the first arithmetic operation performed is addition instead of subtraction.

Design

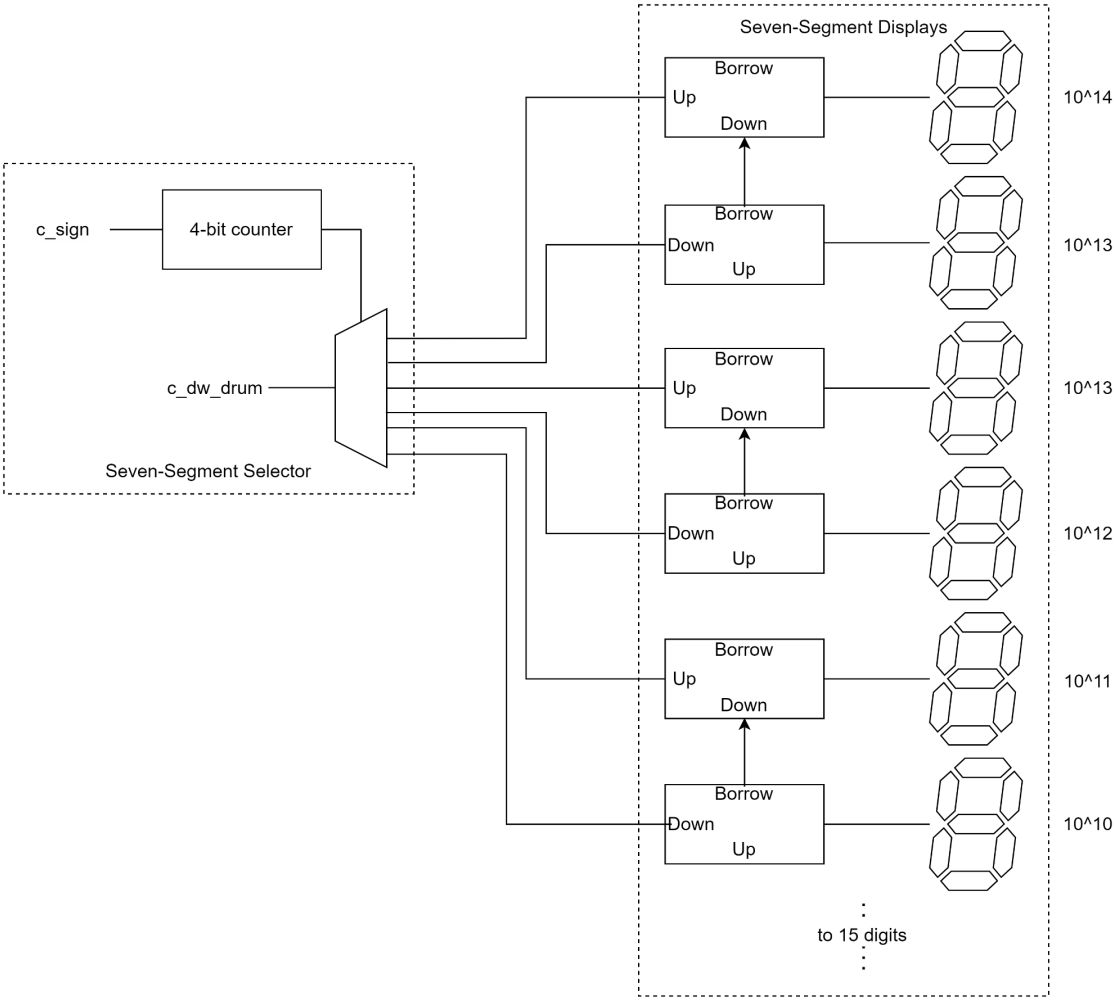


Figure 31. Decimal Output Module Design

Add/Subtract Module

Purpose

The purpose of the add/subtract module is to perform 1-bit addition or subtraction with carry or borrow.

Input and Output Signals

Input Signals

- X - The addend or minuend of the operation
- Y - The addend or subtrahend of the operation
- Cin - The carry or borrow into the operation
- *Add/Sub - The operation to perform; low for addition, high for subtraction

Output Signals

- Sum/Dif - The sum or difference of the operation
- Cout - The carry or borrow out of the operation

Design

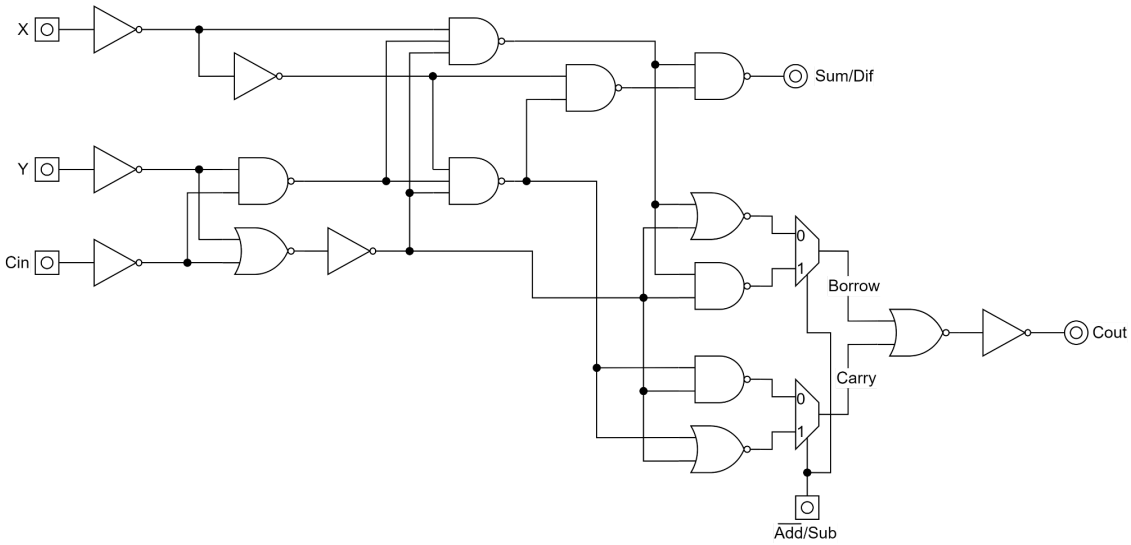


Figure 32. Add/Subtract Module Design

Memory Drum

Purpose

The purpose of the memory drum is to store and control the flow of data into and out of the Add/Subtract modules. This component comprises the entire compute system, except for the Add/Subtract modules.

Input and Output Signals

Input Signals

- Decimal card data - The 5-bit output of the base conversion module

- Binary card data - 5 bits of the input data from the binary card reader
- dc_39[a-e] - The sign of each decimal card number
- Address - The upper 6 bits of the program counter
- clk - Clock signal
- s_70x - Enable this bank of 5 Add/subtract modules
- s_90 - Controls which system controls the selected arithmetic operation; low for the control system and high for the decimal card reader
- s_62 - Clear the CA memory
- s_63 - Clear the KA memory
- s_2 - System reset, active low
- s_69 - Transfer the contents of CA memory to KA memory
- c_e10 - Controls the selected arithmetic operation; low is addition, high is subtraction
- c_drum - Signals the end of a computation cycle, indicates address 49
- c_g3 - Request a right shift of the value in KA memory
- c_83 - Clear registers, active high
- c_read - Read phase
- c_write - Write phase
- c_in0 - First control line for the input selector
- c_in1 - Second control line for the input selector

Output Signals

- br_4[1-5] - The carry out signal from each add/subtract unit
- br_8[1-5] - The sum/difference signal from each add/subtract unit

Description of Operation

Memory

Each memory drum has two separate memories named CA memory and KA memory after the original Counter Abacus and Keyboard Abacus. Each memory stores 5 50-bit, two's complement numbers in a bit-serial fashion. In this format, *Address* 0 stores the least significant bit of each number and *Address* 49 stores the most significant bit. Thus, the system reads and writes a stored number one bit at a time, but 5 numbers in parallel.

Shifting

There are times during computation when the contents of the KA memory need to be divided by two, which is accomplished with a right shift. To achieve this, there is a shift mechanism that takes in *Address* and passes either *Address* or *Address* + 1 to the address input of the KA memory. When *c_g3* is high, which means a shift is required, the shifted address is used during the read phase (when *c_read* is high) and not for address 49 (when

c_drum is high). The address is shifted only during the read phase, so that during the write phase, the data is written to the previous address. So the read phase reads the data at *Address* + 1, and the write phase writes the data back to *Address*. The shifted address is not used at address 49 so that the sign is preserved during the shift.

Input Selector

The signals *c_in0*, *c_in1*, and *s_70x* are used to select one of four inputs to send to the Y input of each Add/Subtract unit. See the Input Selector section of the Control System to see how these signals are used.

Data Flow

For each *Address* there is a read phase and a write phase. During the read phase, the contents of the address are read from memory, potentially modified with the Add/Subtract modules, and then stored in a register until the write phase. During the write phase, the stored value passes through a buffer and is written back to the address. CA memory is the only memory that can be modified by the Add/Subtract modules. The read value from KA memory is simply sent to the input selector to be used in calculations, stored in a register, and written back through the buffer. For CA memory, the read value is first supplied to the X inputs of the Add/Subtract units, which compute a new value using the data from the input selector, before being stored in the register. The carry/difference outputs of the Add/Subtract units are also stored in a register to be used during the next computation. The contents of these two registers are sent to the outputs *br_8* and *br_4*, respectively. All three registers can be cleared with *s_2* or *c_83*. To clear memory, multiplexers controlled by *s_62* and *s_63* are used to send 0 to the buffers instead of the register contents. To transfer the contents of CA memory to KA memory, a multiplexer controlled by *s_69* is used to switch between sending the KA's register contents or the Add/Subtract's sum/difference register contents to the KA's clear multiplexer. The logic shown in Figure x. is used to correctly time the enables and clocks of the registers, buffers, and memory as shown in Figure x.

Design

See Figure 9. Detailed Block Diagram for the full design of the memory drum. Additional details are shown below.

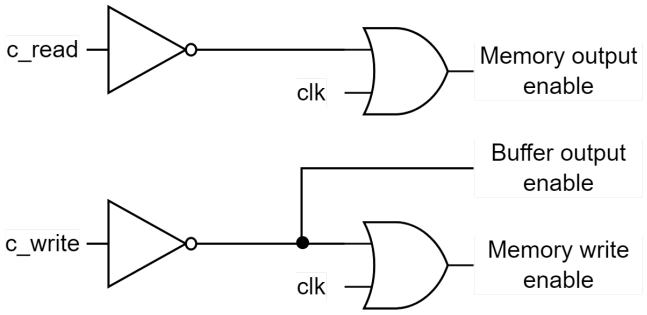


Figure 33. Memory Drum Timing Logic

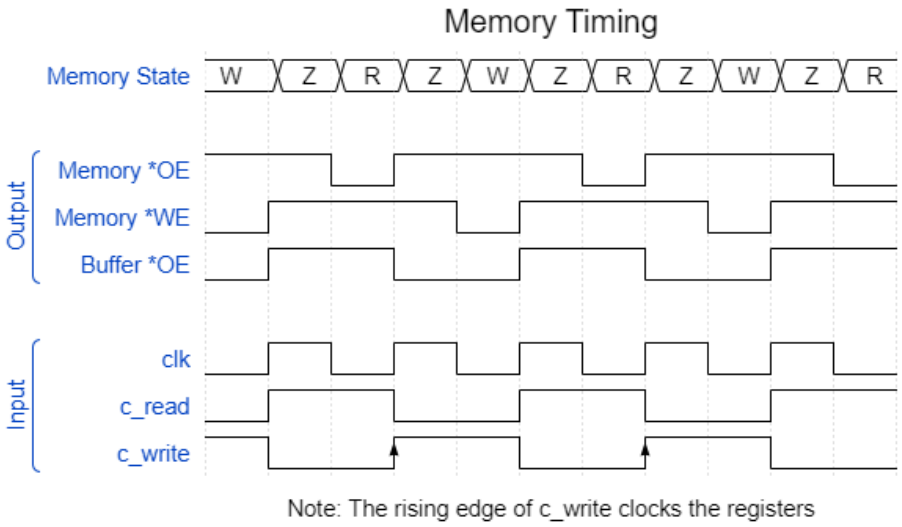


Figure 34. Memory Drum Timing Diagram

	Bits							
Address	0	1	2	3	4	5	6	7
000000 (0)								
...								
...								
...								
...								
110001 (49)								
110010 (50)								
...								
111011 (59)								
...								
111111 (63)								

Unused
Empty for control period
Used for data

Figure 35. Memory Address Space

4.3.3 Functionality

Our design is intended to have a large plywood board that holds all of the PCBs and other components so that it can be easily displayed in a classroom. The design features three tablet for data inputs and outputs, since we did not use physical punch cards, but digital ones. Each tablet displays a binary or decimal punch card on the screen and provides the user interface for selecting and creating the cards as needed. There is also a seven-segment decimal display for users to view the output value from the ABC and a control panel for inputting and viewing control signals. The device features a number of LEDs to indicate exactly what is happening in the circuits, allowing the user to view the intermediate values and states of the system.

An example of a typical user experience with the ABC begins with the user loading/programming the first decimal card on the display by entering the coefficients of the equation into the display. This card is then uploaded to the ABC and read into memory when the user presses the button to do so. After transferring this data to the second memory with a switch on the control panel the process is repeated with the second equation. Once all the values are inputed, the user will choose a value to start eliminating using Gaussian elimination, select it on the control panel, and press a button to start the elimination process. This process is repeated until only one coefficient remains, after which the user can use the control panel to output the result to the decimal display. For a more detailed description of the operation of the system, see Appendix 1.

4.3.4 Areas of Challenge

We faced many challenges throughout the design process in the areas of technical progress, satisfying requirements, and meeting user needs. Challenges to technical progress included finding ICs with the needed functionality or number of components, and the time delays between ordering our PCBs and receiving them. To overcome the first challenge, we incorporated chip research into the initial design process so that the finalized design could be implemented with available chips. Designs were adjusted if necessary to accommodate this challenge. To overcome the second challenge, we tried to parallelize tasks as much as possible to minimize downtime. However, the combined time to receive the first version of a PCB, find and fix a design error, and receive the second version of the PCB still made reordering PCBs largely infeasible for our timeline. We sought to mitigate this issue through design testing and reviews prior to manufacturing, and by prioritizing the implementation of fixes on the PCB using jumper wires as needed.

Satisfying the requirements, mainly that our replica maintain historical accuracy, was the primary challenge in this project. Our replica is intended to function with the same logic and general design as the original ABC. However, documentation of this historic logic and design is limited, so our design process also included historical research to reverse-engineer and fill in the gaps in how the original ABC worked. The main way that we overcame this challenge was through careful research and examination of the manuals and diagrams that still exist. Even after this, elements of the original design remained unknown. In these cases, we chose a design that created the same functionality, correctly interfaced with the known design, and was our best guess of how it would have been implemented, given the surrounding context.

The last area of challenge in meeting user needs focused on balancing the user experience and visuals with the power and space needed for them. One of the main needs of our users is to understand the machine's state and view the values of all the different signals. This becomes challenging for components with a large number of inputs or outputs, like the binary card reader and writer. Both of these have 30 input or output signals to be visualized. However, this would require 30 LEDs per component, greatly increasing our system's power requirements and creating much larger, and therefore more expensive, PCBs just to hold all the LEDs. In this case, we decided to compromise by visualizing only five signals at a time (users can select which Memory Drum board they wish to visualize).

4.4. Technology Considerations

This project combines modern hardware with historical logic to rebuild the Atanasoff-Berry Computer in an accurate, durable, and educational form. It uses TTL logic chips, through-hole components, socketed ICs, and KiCad-designed PCBs verified through VHDL simulation before fabrication. TTL logic provides stable performance and keeps the system faithful to early digital designs. Through-hole parts and sockets make the boards easy to assemble, repair, and understand during demonstrations. The trade-off is space

and power use, since discrete logic takes up more space and consumes more current than integrated options.

VHDL simulation allows the team to confirm circuit logic before committing to hardware. This reduces wasted materials and ensures functionality before manufacturing. KiCad is the main tool for schematic and PCB layout, chosen for its open-source flexibility and compliance with IPC-2221 standards. Electrical and design rule checks built into KiCad help prevent layout errors before fabrication.

ESP32 modules manage user input and output by replacing physical punch cards with a Bluetooth-connected mobile interface. This preserves the interactive behavior of the ABC while keeping it practical for classroom or exhibit use. The main drawback is synchronization, since the ESP32 operates on a separate clock, requiring careful timing alignment with the logic system.

Alternative technologies like FPGAs were evaluated but not chosen. While they would simplify the build, they would remove the physical logic that makes the project valuable for teaching and historically accurate. The design remains focused on clarity and visibility, with LEDs showing signal states and all major components large enough for hands-on demonstration and classroom viewing.

5. Testing

5.1. Unit Testing

Each module and its PCB implementation was considered a single unit for testing. We tested our proposed logic for each module of the ABC in VHDL or a breadboard prototype before fabricating PCBs to ensure our approach was functional. This enabled us to verify our designs quickly and to reduce our development costs by catching design flaws before fabrication. We also tested our fabricated PCBs individually using breadboards and control wires to simulate how each unit of the ABC functions in its respective place in the full machine. Where feasible, we used an exhaustive truth table to test each unit. For modules with more complex input where this is infeasible, we used a combination of boundary value and functional testing to cover as many cases as possible. This unit testing was performed to ensure that each module of the ABC works on its own, and narrowed down the possible issues when testing the modules together and doing full system testing.

Tools used for physical unit testing include oscilloscopes for analyzing timing signals, and function generators for supplying controlled input patterns. Bench power supplies and multimeters were used to validate voltage levels, continuity, and connector pinout accuracy. Collectively, these methods ensured that all electrical and logical components of

the unit operated reliably when integrated into the final system.

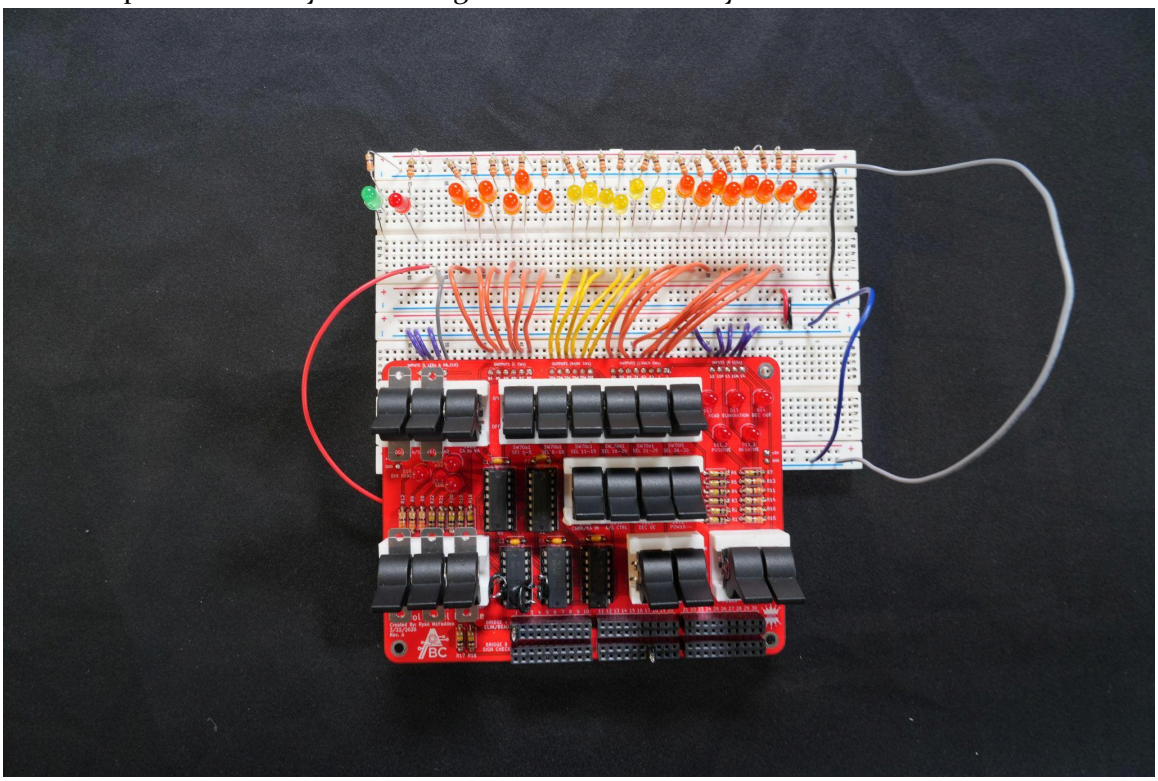


Figure 36. Control Panel Unit Tester

5.2. Interface Testing

The final product for this design consists of multiple PCBs, or submodules, mounted onto a larger motherboard PCB. Each board connects through defined interfaces such as shared data lines, address lines, control signals, clock inputs, and regulated power rails. The submodules are designed for easy swapping, which allows them to be inserted onto the main motherboard or tested individually on a breadboard. This capability supports verification that interface signals, such as binary outputs from the EEPROM being sent to the add/subtract module or timing pulses from the control circuitry updating the decimal output module, maintain correct logic levels and align with expected timing. By connecting multiple submodules together, various input combinations can be tested to ensure they propagate correctly through the system and match expected behavior based on the ABC architecture. Interface validation is also performed before PCB fabrication by modeling each subsystem in VHDL. Simulating logic behavior, timing interactions, and architectural dependencies helps verify that the design is feasible using the selected electronic components. This early-stage testing prevents conflicts between submodules and reduces the risk of errors before committing to PCB manufacturing and assembly.

Interface testing largely used the same tools used in the unit testing phase, but for different purposes. Interface testing was also used to ensure that long trace lengths on the motherboard do not introduce unintended signal coupling or degrade logic thresholds.

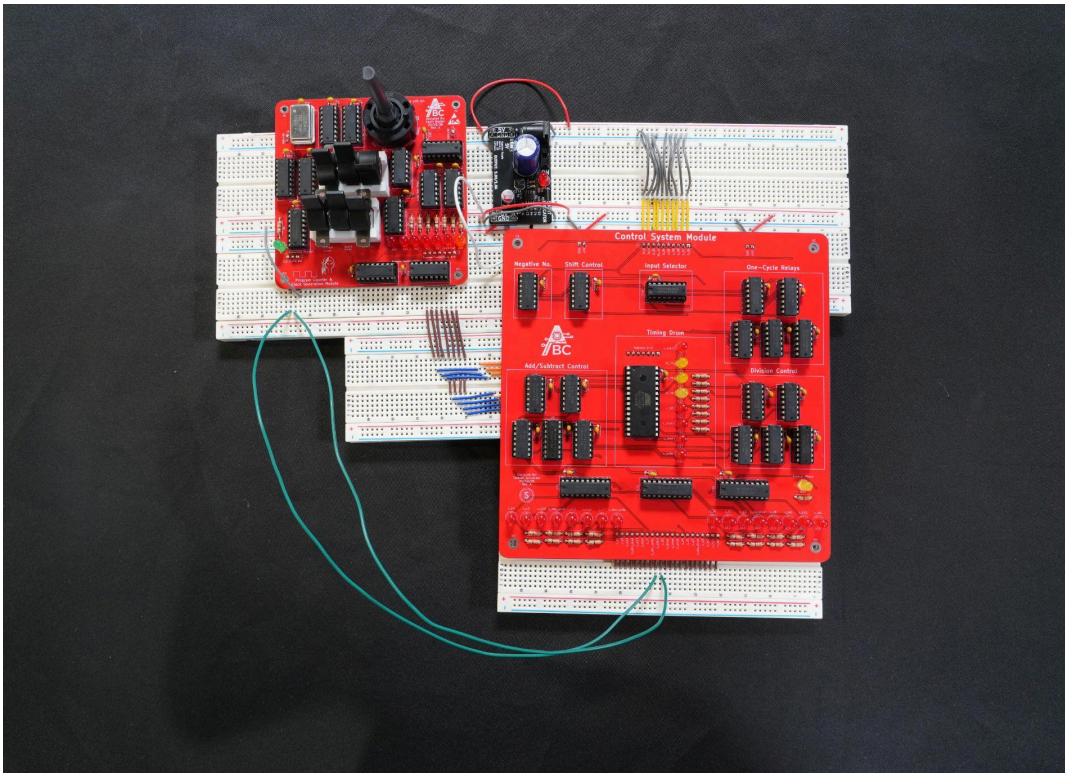


Figure 37. Clock - Control System Interface Tester

5.3. Integration Testing

Although there are several different integration paths, three are the most critical as they provide the essential functionality required to solve a system of linear equations like the original ABC. The first path is the input-to-memory path, which is required to input data into the ABC for computation. This path utilizes the decimal card input, binary conversion drum, add/subtract module, one of the memory drums, and the various routing components that correctly direct the signals between these modules. The second path is the computation path for performing the Gaussian elimination algorithm, which includes both memory drums, the add/subtract module, the control system, and the connecting components. The final critical path is the memory-to-output path which is essential for users to see the results of computation. This path is made up of one memory drum, the add/subtract module, the decimal output module, and the routing components.

Testing these integration paths used similar tools and methods as the other testing phases. Each integration path was tested using a set of test values determined using boundary value and functional testing methods to ensure a wide range of cases are covered. For virtual integration testing, the VHDL simulation of each component in a critical path was connected together, and waveforms were used for the test values and results. For physical integration testing, each PCB module was connected together with breadboards, and the tools mentioned above were used to generate and read the electrical signals for the input

and output. Pictured below is a partial example of the data input path integration testing setup, with the tablet user interface not shown.

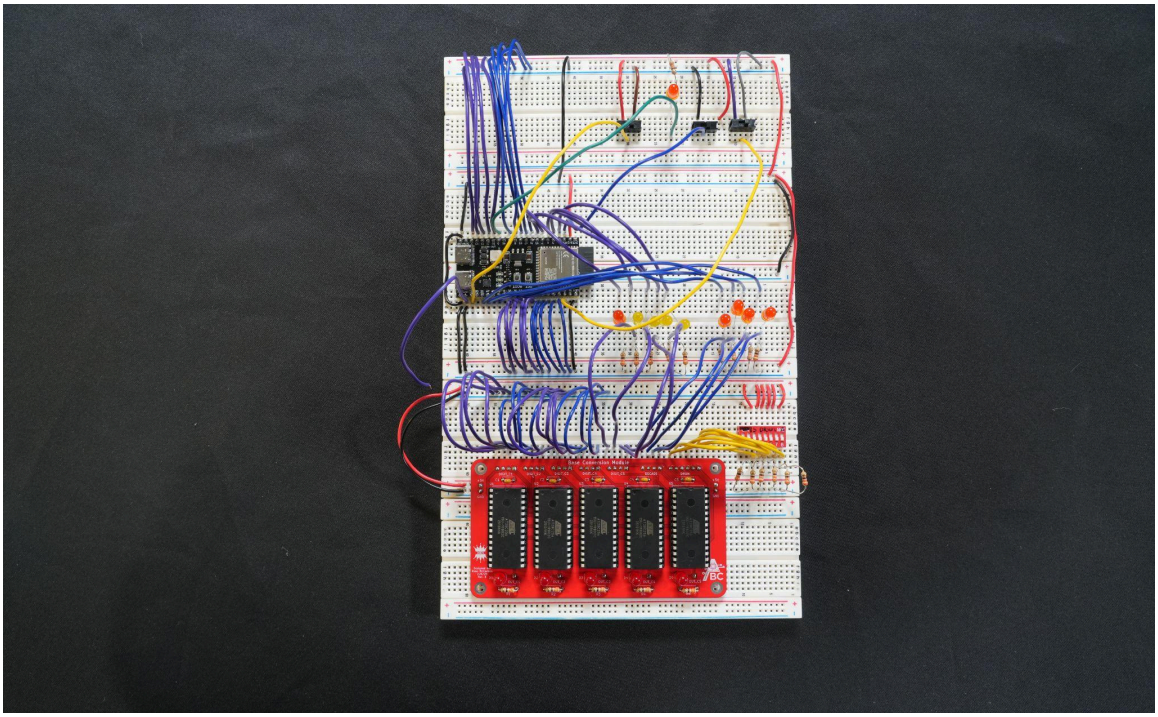


Figure 38. Partial Data Input Integration Tester

5.4. System Testing

System testing was performed once all modules passed unit, interface, and integration testing. The purpose of this stage was to validate the complete ABC replica as a unified system, ensuring that the hardware modules, control logic, and user interface operate together to meet the functional and user experience requirements. At this level, testing focuses on the end-to-end behavior of the system, verifying that it can correctly solve systems of linear equations, display intermediate states through LEDs, and provide accurate outputs through the I/O modules and user interface.

The methodology for system testing involved executing representative workflows that mirror how the ABC was originally intended to operate. This included entering equations through the input modules, observing the computation step by step, and confirming that the final solution matched the expected results. The complete VHDL simulation of our designs was also used for system testing prior to the completion of the physical computer, allowing us to validate our designs earlier in the development process.

When it comes to full-system verification testing using VHDL, we were able to fully replicate the ABC in VHDL and run through the steps to solve a 4-by-4 system of linear equations. This was done by fully implementing all physical units of the ABC in VHDL, creating an overall design file, and running the design through a testbench in QuestaSIM.

This testbench was made to fully test all the operations the ABC can perform. Such as adding, subtracting, shifting, inputting decimal cards and converting to binary, and inputting and exporting binary cards. Also, while completing full system testing, we found two distinct errors/bugs in our overall design. The first one was the issue with not shifting or adding the most significant sign bit of a negative number in our drum. We fixed this by adding a new control signal to the drum's write enable to ensure we don't lose the sign bit. The second issue we found was with the system's clock/counter. We realized in the testing that we were using the seventh bit of a counter as the clock and read/write, which would make it so that we would only get a positive clock edge of write, so in read, we would not actual be able to read anything. So we needed an 8th bit, and it needed to be for the clock to fix this issue. Overall, the VHDL system testing was a huge success and was necessary to ensure the ABC functioned correctly.

5.5. Regression Testing

The progressive nature of our project and testing process helped to ensure that new additions do not break old functionality. Due to constraints discussed in section 3.1, our project is not taking an iterative approach for creating our PCB modules. The result of this decision is that we gradually accumulated completed PCBs, which had two implications for testing. First, previously tested units and interfaces were not modified. This, by nature, reduces the likelihood of regression. Second, the different stages of testing (unit, interface, integration, etc.) were completed as the PCB fabrication process finishes. After finishing the fabrication and unit testing of a new module, we performed interface and integration testing with this new module and a set of already tested previous modules. Regression testing was incorporated into this step of the testing process by including previously successful test cases, in addition to the test cases for the new module. Because regression testing was incorporated into the other testing stages, it was performed using the same tools and techniques as those other stages.

5.6. Acceptance Testing

Since many of our design requirements were provided by our client and since he will be the primary user of our project's results, our client played an important role in our acceptance testing process. Acceptance testing was conducted at the end of the logical design, PCB design, and fabrication phases. At the end of each of these phases we presented our designs or PCBs to our client to ensure that they meet all functional and non-functional requirements before proceeding to the next development phase. Performing this acceptance testing early and often ensured that our design and implementation met the requirements and ensured that decisions that would have moved our project away from meeting the requirements were quickly identified and corrected.

5.7. User Testing

User testing will focus on evaluating whether the ABC replica is understandable, operable, and useful to its intended audience: computing instructors, students, and campus visitors. Since the project's purpose extends beyond raw functionality, user testing will evaluate

clarity, ease of interaction, and the educational value of the system. As discussed above, out client will be the primary user representative of the computing instructors user group. As such, our user testing in this area focused on ensuring that he was able to perform various complex tasks using the provided instructions and noting areas where the instructions were insufficient, the number of user mistakes or misinterpretations, and overall understanding of the ABC operation. Future user testing will need to be conducted do address the other groups of potential users, and could include methods such as student surveys to gauge the effectiveness our replica in teaching computing history and concepts.

5.8. Testing Results

Our testing processes have revealed a variety of defects that can be broken up into three categories: cosmetic, easily fixable, and significant. Cosmetic defects are defects that do not affect the functionality of our product but could provide misleading or incorrect visuals to our users. Examples of cosmetic defects include silkscreen errors in the Control Panel module and Memory Drum where the labels of switches or registers are swapped. These defects were discovered during the unit testing process of the respective PCBs. Another cosmetic defect, the LED indicators the the select lines of the input selector being swapped, was found during interface testing between the Control System module and Memory drum. Increasing in severity from cosmetic defects are easily fixable defects which include defects that affect the functionality of the system but can be easily fixed by either software changes or minor wiring changes to the back of the PCB. This was the most common category of defect found by our testing process. Examples of hardware defects in this category include a missing reset pin output on the Control System, found during unit testing, and the need for a faster clock signal for the Memory Drum relative to other components, found by our VHDL system testing. Software defects in this category include the card readers/writers shifting past the end of cards during the ends of the control period, which was found during data-path integration testing. The final category of defects are significant defects which require major changes to a component's logic or reordering a PCB. Our testing processes have revealed a few significant defects including incorrect clock logic for halting after a full compute cycle (found during unit testing), a single decimal card controlled sign input to the Memory Drum instead of five independent signals (found during interface testing with the decimal card reader), and a reversed chip footprint on the Base Conversion module (found during unit testing). Other defects, their category, and their resolution are described below.

Defect	Category	Testing Stage
Silkscreen typo and flipped register labels on the memory drum	Cosmetic	Unit
Input selector select line LEDs are flipped on the memory drum	Cosmetic	Interface
Memory drum only has one input for dc_39x control signal	Significant	Interface
Memory drum requires a faster clock signal than the read/write signals	Fixable	System
Motherboard swaps the power and ground pins for the control system	Fixable	Interface
Motherboard program counter pins with the clock are offset by one pin	Fixable	Interface
Control system is missing a reset pin input	Fixable	Unit
LED labels on the control panel are swapped	Cosmetic	Unit
Batch mode on the clock does not stop after one full compute cycle	Significant	Unit
Flipped footprint for the EEPROM chips on the base conversion module	Significant	Unit
Control panel debouncing clock needs to be decoupled from the manual mode of the clock	Fixable	Integration
Clock reset is active high but system reset is active low	Fixable	Unit
Binary card software shifts too many times	Fixable	Integration

Figure 39. Testing Results Defect Table

6. Implementation

At this point all module designs have been implemented into hardware. For the control subsystem, each module was implemented as a separate PCB. The control system was implemented as a single PCB, but each control unit was kept in a separate group and marked with silkscreens.

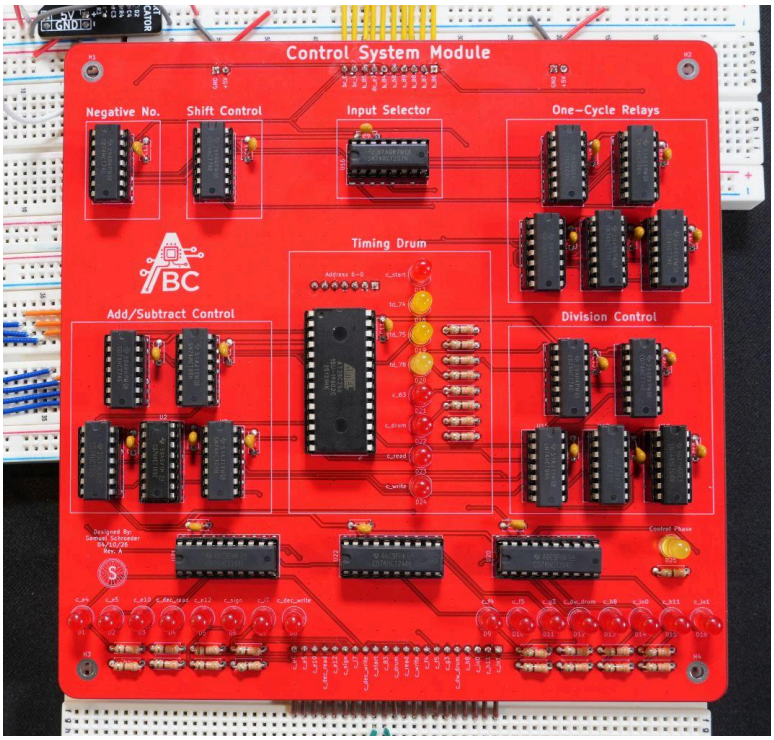


Figure 40. Control System Implementation

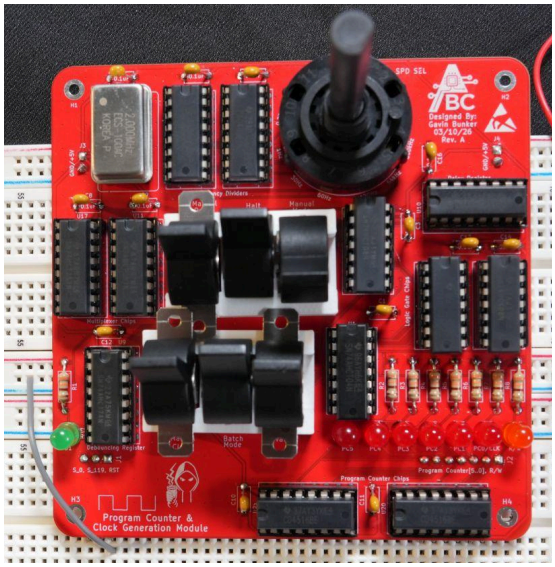


Figure 41. Clock & Program Counter Implementation

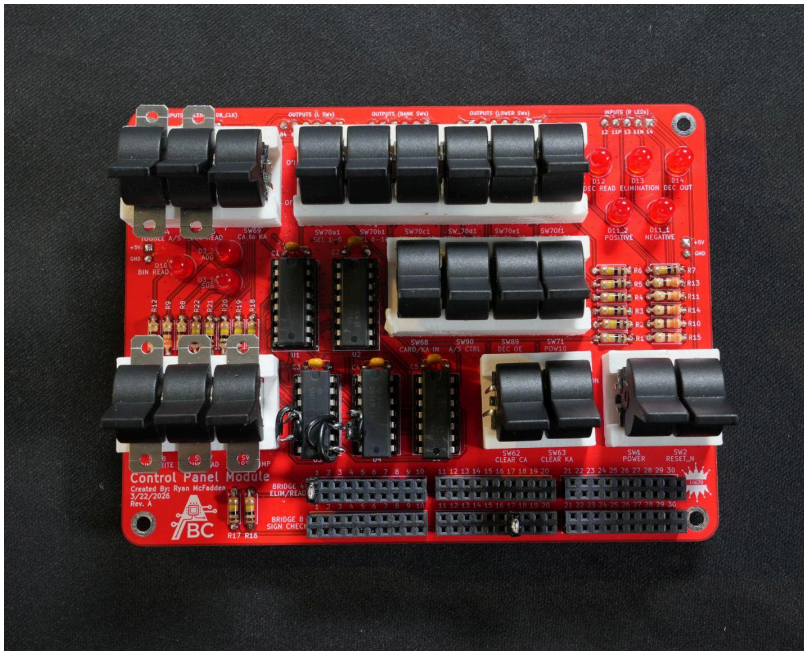


Figure 42. Control Panel Implementation

For the Input/Output subsystem, the decimal output module and base conversion module have been implemented into PCBs and the card readers and writers were implemented as ESP32 development boards that communicate to an Android app interface over Bluetooth. A PCB to hold the ESP32 boards as well as the system's power supply units has been designed and ordered; however, we have not yet received the manufactured PCB. All of the software for both the Android and ESP32 apps was mostly implemented by the previous team. Our part in implementing these components was adapting their software to fit with our 5-bit width and different control mechanisms.

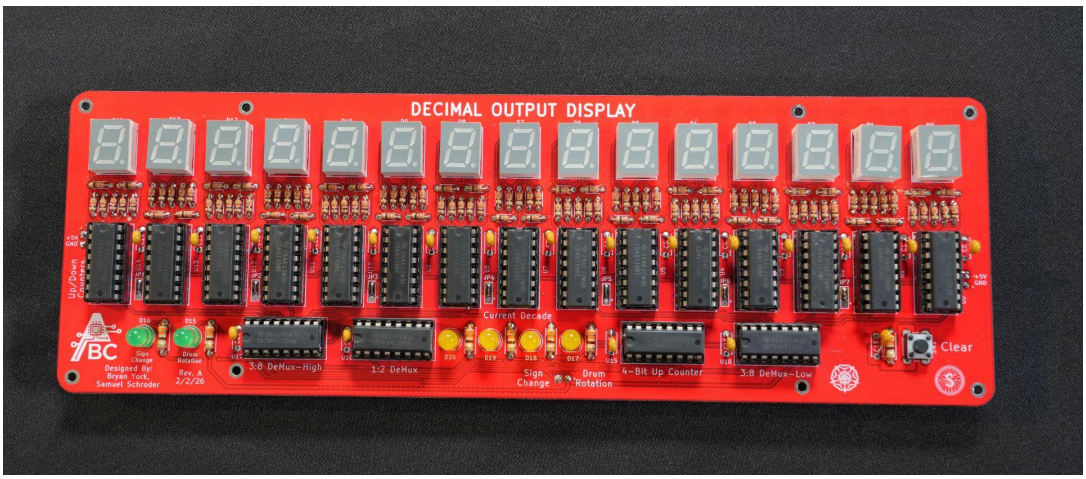


Figure 43. Decimal Output Implementation

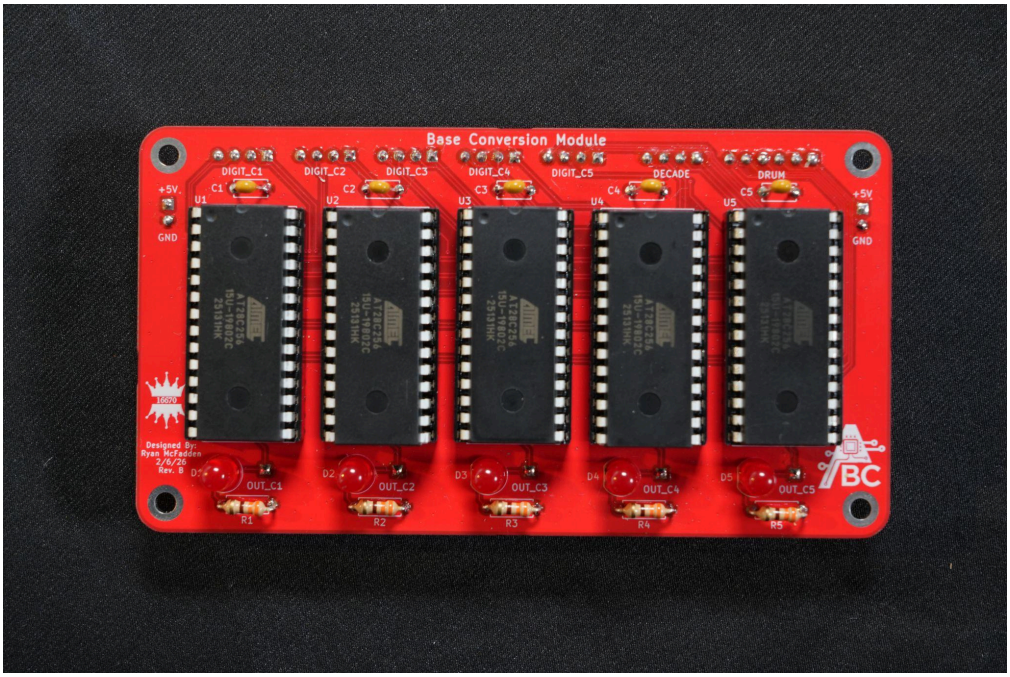


Figure 44. Base Conversion Implementation

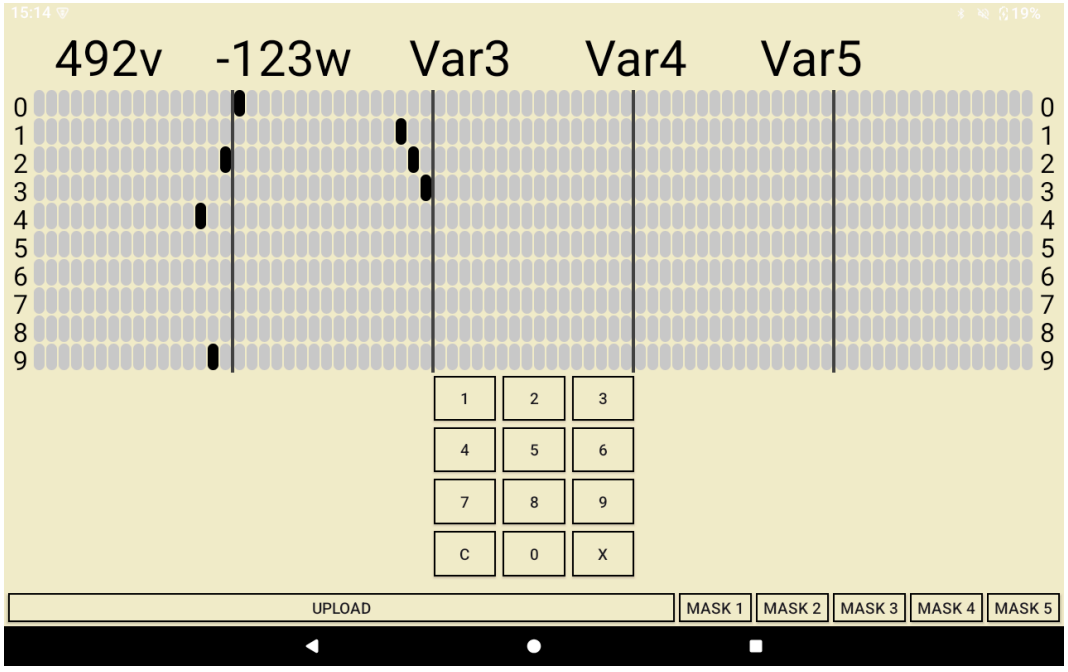


Figure 45. Decimal Card Reader Interface

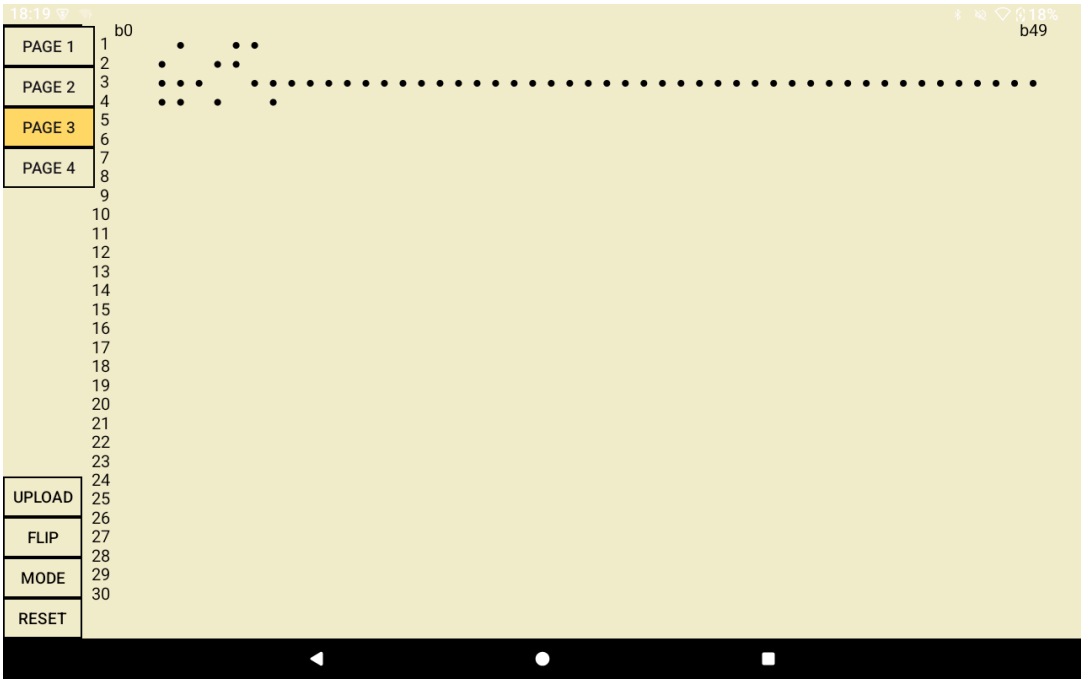


Figure 46. Binary Card Reader Interface

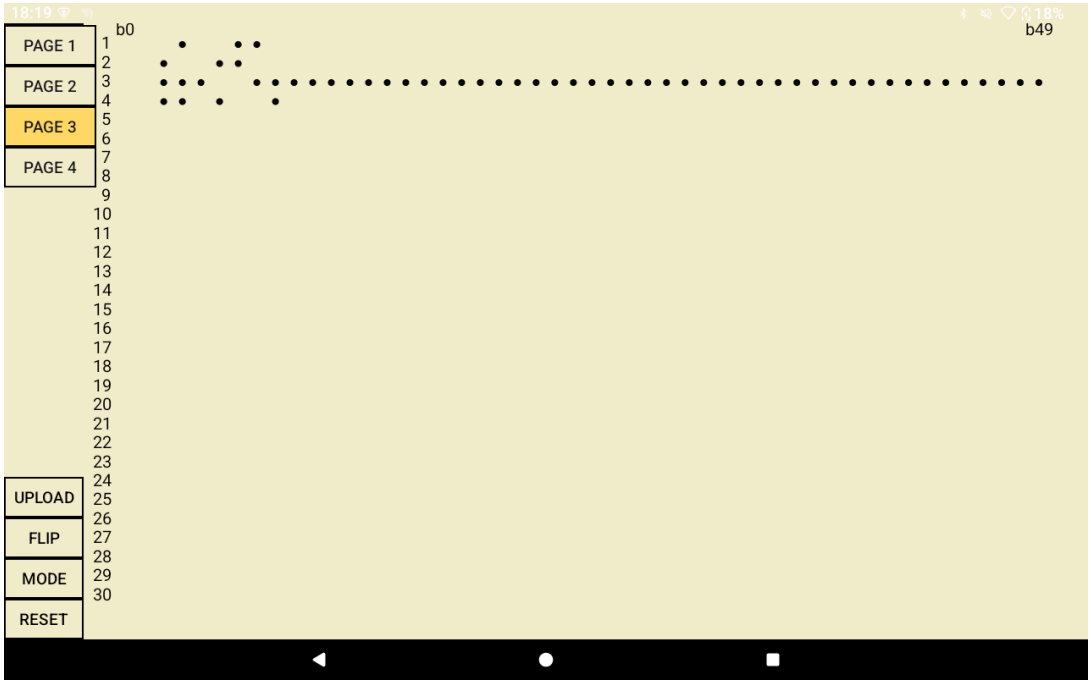


Figure 47. Binary Card Writer Interface

The compute subsystem has been implemented as two PCBs that stack on top of each other. The Add/Subtract module was implemented as a separate PCB, and everything else was implemented into the Memory Drum PCB, which has sockets for five of the

Add/Substract PCBs to plug into itself. For the full-scale implementation six copies of this PCB would be assembled.

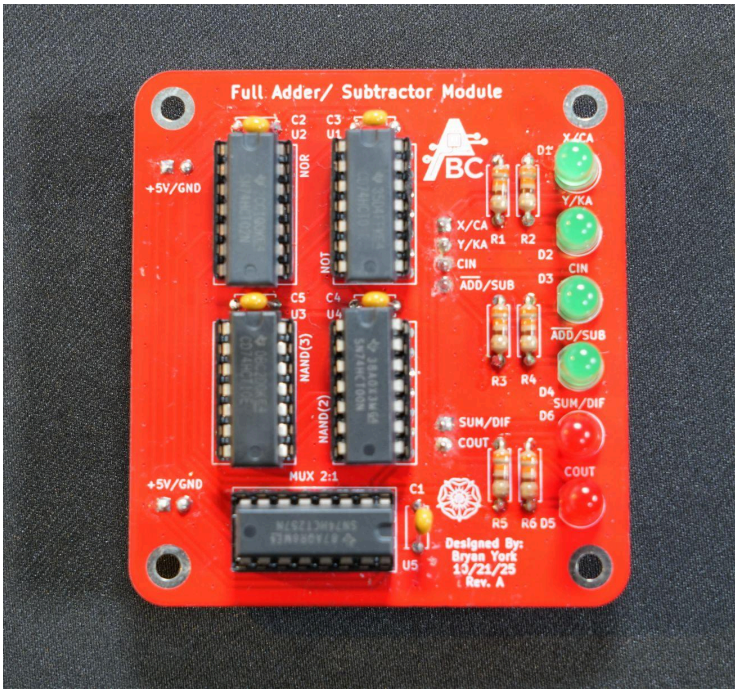


Figure 48. Add/Subtract Module Implementation

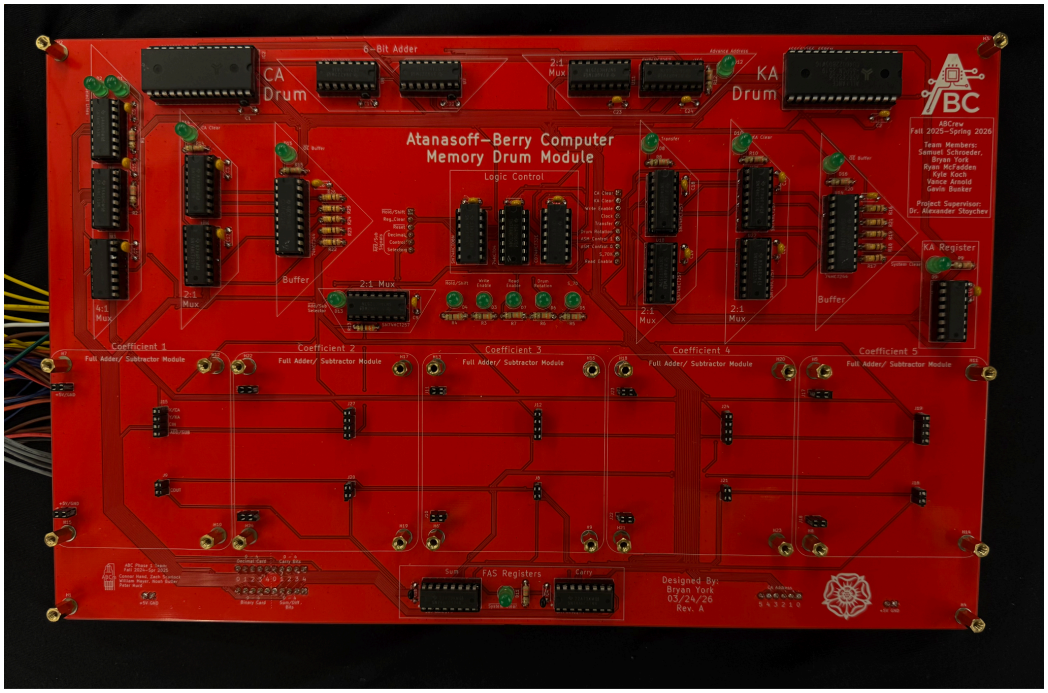


Figure 49. Memory Drum Implementation

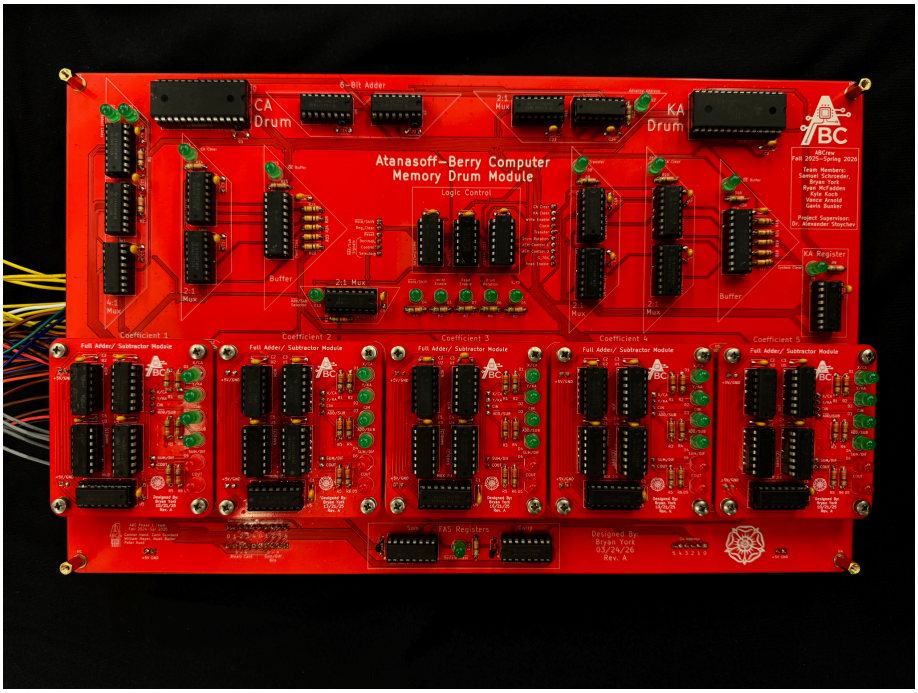


Figure 50. Assembled Memory Drum Module Implementation

Finally, each of these components were connected together onto a larger motherboard. All PCBs of the control and input/output subsystems directly plug into this motherboard, and the compute subsystem connects over a bus of wires, which could be duplicated to support all six memory drum PCBs.

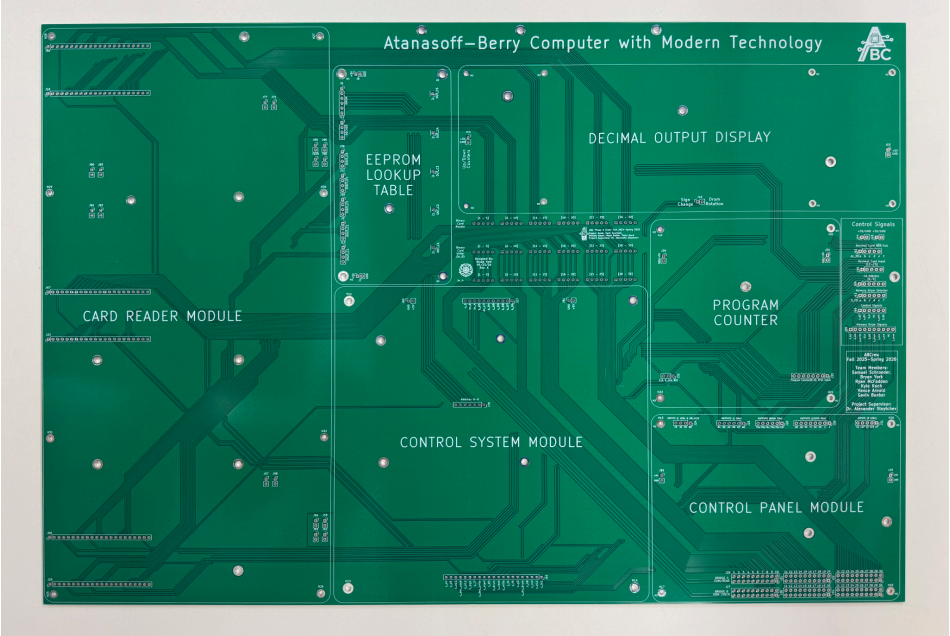


Figure 51. Motherboard Implementation

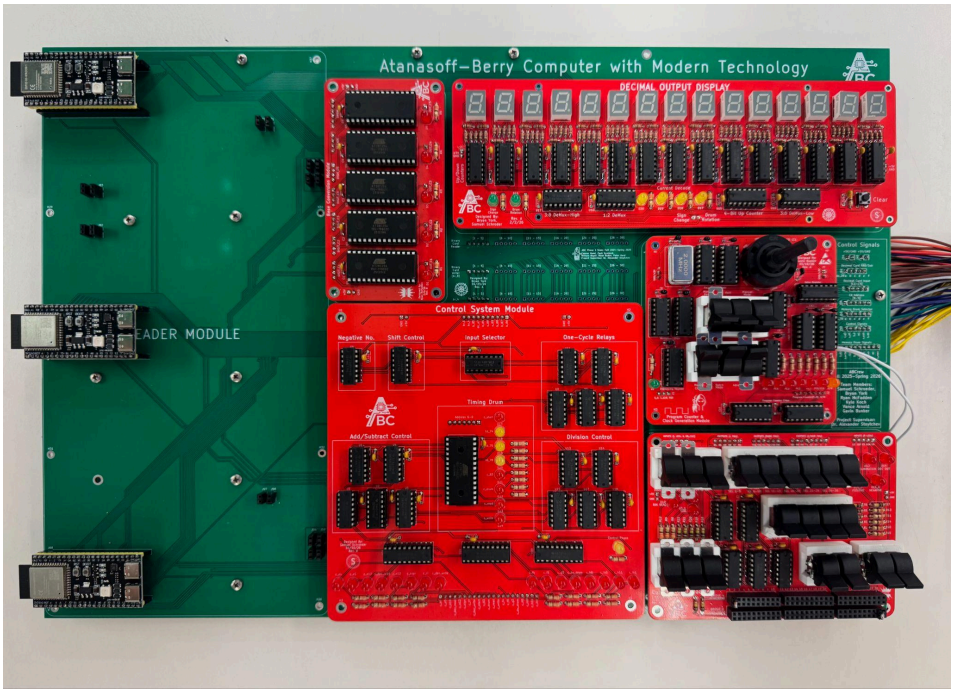


Figure 52. Assembled Motherboard Implementation

6.1. Design Analysis

One of the biggest advantages with this implementation style was the ability to design and fabricate each of these subsystems as individual modules separately. This was a huge advantage when it came to physically testing each design one at a time and making changes as needed. Printing these smaller PCBs more often ended up saving the group both financially and time. Doing this allowed our team to break up each system of this project into smaller subsystems and focus on the design of these smaller scale systems. This also meant our team could create these smaller PCBs, which are cheaper to manufuacuter, and testing them physically through out the whole year instead of waiting until one large board was ready for manufacturing and ready to print.



Figure 53. Memory Drum Module Side Profile

Another key electment of this stackup design was the power testing, or seeing how each module would react, electrically, when supplied with power. Incorrectly connecting components, such as swapping the low and high power pins, or connecting two outputs together on the same line, could end up completely destroying the chips connected to these modules. By testing these smaller modules one at a time, we could help restrict any potencial errors to these smaller PCBs and the fewer amount of chips on this board. If the team had decision to use fewer but larger boards, than the risk if an error occurred would be far greater.

To connect our motherboard to our memory module, individual wires were used to connect each lien to one another. While our team understood this would create a large number of connections between the modules and be messy to work with, this was the simplest way to ensure these modules could be connected properly and also be quick to implement as the semester came to a close. Another way that these two modules could have been connected together would have been by using a ribbon cable to run these signals as a group with looks cleaner and less mess to potencail cause other issues between the boards.

7. Ethics and Professional Responsibility

7.1. Areas of Professional Responsibility/Codes of Ethics

Area of Resposibility	**Definition**	**Relevant Item from IEEE Code of Ethics**	**Relation to Our Work**
Work Competence	Perform work of high quality with integrity	1.6: "...undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations"	Assign tasks inline with our individual competencies (i.e. hardware design, VHDL development) and ensure proper mentorship when someone is working on a task in an area they do not have previous experience in
Financial Responsibility	Design at reasonable costs, delivering a quality return on investment	1.1: "...to strive to comply with ethical design and sustainable development practices..."	Keep fabrication costs in mind when making PCB design decisions, and solder carefully to avoid wasting parts
Communication Honesty	Report work truthfully and understandably	1.5: "to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic in stating claims or estimates based on available data, and to credit properly the contributions of others"	Document our work and designs accurately through this design document and the technical manual we are developing
Health, Safety, & Well-being	Minimize risks to the safety, health, and well-being of users, designers,	1.1: "to hold paramount the safety, health, and welfare of the	Ensure safe hardware design, designing at low voltages, and

	and clients	public…"	soldering solid connections
Property Ownership	Respect the property and ideas of others, giving credit where credit is due	1.5: "…and to credit properly the contributions of others"	Properly credit the work of previous groups who attempted to recreate the ABC, particularly the work of sdmady25-29, whose work and research we are building much of our design off of
Sustainability	Protect the environment and natural resources	1.1: "…to strive to comply with ethical design and sustainable development practices…"	Thoroughly test our designs before fabricating PCBs for them to minimize wasted PCB development and shipping
Social Responsibility	Produce a product that will benefit society	1.2: "to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies…"	Ensure our recreation implements the logical design of the original ABC as accurately as possible, even when it would be more convenient to change the original design, to ensure historical accuracy

Figure 54. Code of Ethics Table

An area on this chart where our team has performed well is in Communication Honesty, as we have been honest in our documentation of designs and in our communications with the client, and have based design decision on the principle of honestly recreating the original ABC's designs to the best of our ability. Our team could have improved in the category of Financial Responsibility by creating a more organized inventory system so that we could accurately evaluate all components we currently had, to minimize leftover components and have fewer larger component orders to minimize shipping costs.

7.2. Four Principles

	Beneficence	Nonmalificence	Respect for Autonomy	Justice
Public health, safety, and welfare	Design safe circuits that operate at low voltages to promote user safety	Design does not support unsafe or harmful practices (e.g., sourcing parts)	We listened to our client and avoided making choices for them during design	Our open-sourced design promotes education equality
Global, cultural, and social	Educating others about computing history and logic, and logically recreating this device that has been lost to history	Avoiding misinforming others by keeping our design logically accurate to the ABC	Enable others to recreate and customize our designs by documenting and sharing them well	Give credit to all sources used and work from previous teams
Environmental	Ordering parts intentionally and not fabricating more parts than we need	Avoid wasting PCB boards by designing intentionally and soldering carefully.	We will not directly disturb any natural ecosystems through our design	Implementation would not disturb one habitat over another
Economic	Design could promote the success of Iowa State University's admissions department, aiding the university financially	Design would not disrupt the economy	Users recreating this design could fabricate a different number of modules based on their budget	Implementation would not unduly financially affect some people

Figure 55. Ethical Principles Table

An area of this table that is important to our project is beneficence in the global, cultural, and social context, because our project's end goal of promoting computing education and recreating lost history aligns well with this principle-context pair. One area in which our

project has a weakness is environmental nonmaleficence, as it utilizes natural resources without a positive or mitigating effect. The scale of our environmental impact is relatively minimal, though, so we believe it is outweighed by the overall positive impact of this project.

7.3. Virtues

7.3.1 Group Virtues

- Determination
    - At the very start of this project, all members understood the scope of the project and knew that hard work would be required to fully implement it. Understanding these requirements, all team members are committed to implementing the full ABC system and documenting each step of the design process. Providing regular updates and encouragement, as well as helping other team members, supports the virtue of determination within our team.
- Collaboration
    - Due to the large nature of this project and the complexity of each individual component, collaboration is essential. Collaboration, both internally as team members and externally with our advisor and first-hand resources, has been and will continue to be an important part of our project. Collaboration not only speeds up our design process but also ensures a better understanding of the project by all members involved. To support this virtue within our team, we hold multiple weekly meetings and maintain open communication, which provides many opportunities for collaboration.
- Documentation
    - Upon reviewing the limited documentation and replicas of the ABC created in the past, it became essential for us to learn from these findings and document all steps in our design process, including research, troubleshooting, and implementation. This way, our team, as well as future teams, will be able to pick up where we left off, spend less time finding the same information, and more time unlocking new clues and findings about the original ABC. Viewing documentation as a required final deliverable and peer reviewing documentation will help promote this virtue.

7.3.2 Individual Virtues

Vance Arnold - Team Facilitator, Parts & Purchasing

- Virtue Demonstrated: Collaboration
    - Collaboration ensures that the project benefits from diverse skills and perspectives, and it prevents bottlenecks by distribution responsibilities.
    - I have facilitated communication between team members, coordinated purchasing needs, and ensured that everyone has the resources required to complete their modules. By anticipating partner needs and clearly framing progress, I've contributed to keeping the team aligned.
- Virtue Not Yet Demonstrated: Confidence
    - Confidence allows me to take initiative, trust my technical decisions, and contribute more fully to design discussions. It also helps reduce hesitation when facing unfamiliar tasks.
    - I plan to build confidence by studying the original ABC's design more deeply, asking clarifying questions when I feel unsure, and practicing schematic design with feedback from teammates. Over time, this will help me feel more capable and engaged in the technical aspects of the project.

Samuel Schroeder - Team Leader

- Virtue Demonstrated: Commitment to quality
    - This virtue is important to me because I value doing things right the first time, and I know that reliably delivering high-quality work reflects on my standards and creates better long-term solutions.
    - I have demonstrated a commitment to quality through my reviews and revisions of all course deliverables, such as status reports and design document sections. I have also demonstrated this virtue in my feedback to team members when designing and reviewing modules, which helped us create a high-quality end product.
- Virtue to be Improved: Collaboration
    - The virtue of collaboration is important to me because it is an essential part of engineering projects. Most projects are far too complex for a single person to complete, so effective collaboration is a requirement of solving these challenging engineering problems.
    - To improve this virtue in my future engineering work and foster more effective collaboration, I can work to better ensure that all team members feel confident in sharing their ideas and feedback, and I can take more time to explain my work and ideas to others so that they understand my work and can provide their feedback.

Ryan McFadden - Scribe, Client Liason

- Virtue Demonstrated: Effective Communication
    - Effective communication is vital to successful work in any field, but particularly in engineering. When designing and verifying complex systems with many moving parts, it's important for engineering teams to stay on the same page and be able to effectively collaborate and learn from each other's experience.
    - I have demonstrated a commitment to effective communication through consistently communicating my designs, current questions, and status of my work to my teammates and our faculty advisor. In addition, I have ensured that our designs and next steps have been shared with our client on a weekly basis to make sure our approach was meeting his expectations for the product.
- Virtue Not Yet Demonstrated: Clear and Thorough Documentation
    - Documentation can play a big part in whether a project is able to be successfully improved upon by future collaborators. When designing our Base Conversion Module and Control Panel, I was not thorough in documenting my design and the discovery work I conducted on these modules.
    - To improve this virtue in my future engineering work, I plan on developing a habit of taking short notes as I work, particularly emphasizing any design decisions I made that day or anything I learned, in order to be able to compile more accurate and thorough documentation once I complete or handoff any given project.

Bryan York - PCB Lead

- Virtue Demonstrated: Hard Work
    - Throught the various modules that I have created for this project, I believe that I has demonstrated my ability to work hard to ensure that the designs and fabricated PCBs would function as intended and without interfacerence from one another. Through my time in soldering and testing these boards, we have been able to have multiple unique modules to implement some of the various functions of this project.
- Virtue Not Yet Demonstrated: Effective Communication
    - When problems arose during component sourcing and reviewing documentation, I often found myself waiting until our next team meeting to address these issues with the full team. Doing this ultimately wasted precious time needed to complete this project.

    - As this second semester was winding down, and I became busy fpcusiong on some of the bigger modules for this project, I found myself not devoted as much time as needed to review some of the other PCB modules that were being created. Because of this, there were simple mistakes which could have been avoided once they were received from production. Learning from these mistakes, I will be sure raise any concern about about not being able to complete all the task assigned to myself and asking for help in completing these tasks.

Gavin Bunker - Manual Lead

- Virtue Demonstrated: Critical Thinking
    - This is important to me as one of the key takeaways of a senior design project is to critically think about how to meet a requirement set forth by a client.
    - I have demonstrated critical thinking in various parts of the project, with the most prevalent example being the Clock Generation and Program Counter Module. I designed this module from scratch and continue to find new optimizations and improvements in its design as requirements evolve.
- Virtue Not Yet Demonstrated: Clear and Thorough Documentation
    - This is important as it will allow us to effectively turn over our work to others to either improve upon it or operate it. We will only be working with and operating this prototype until May, after which we will need to ensure that there is thorough documentation so others do not have to reverse engineer it, as we have had to do for many parts of last year's project.
    - This will be demonstrated after I finalize the clock generator module and write operating instructions for it, as well as explain how it functions in the technical manual.

Kyle Koch - VHDL Lead

- Virtue Demonstrated: Determination to learn
    - This virtue is very important, as in this project, we were thrown into an already underway project and had to learn about it while also picking up where the last group left off.
    - I had to spend considerable time learning how the ABC works and is designed, as well as reviewing the previous group's work and picking up where they left off in the design process.
    - Also, in completing the full VHDL design of the ABC, I had to learn new methods and techniques that VHDL offered to both implement

the design as closely as possible to the physical model and simulate its operation in QuestaSIM.

- Virtue Not Yet Demonstrated: Collaboration
    - I think collaboration is a very important part of the senior design project, but often you fall into the routine of working on it by yourself rather than with others. I was in charge of the full VHDL simulation, which was mostly code, so it was hard to collaborate with others and discuss problems.
    - I tried my best to be more of a team player as we got closer to the end, as I was missing the collaborative aspect of this class a bit. I have also learned that, a lot of the time, talking through my problems with others can help me identify my errors and bugs, which will help me in the workforce after this class is over.

8. Conclusions

8.1. Summary of Progress

All team members of this project feel that it represents a wide variety of accomplishments by the team and the legacy of the Atanasof Berry Computer. Over the past year, our team has demonstrated that recreating the original mechanical device is feasible electrically. This PCB model of the ABC demonstrates that a full logical recreation of the original system can be restored to the present day. Our team originally started with only a couple of physical documents describing how the original system and the 1990's replica. Even then, most of these documents are only a small part of the original documents they were a part of. This made understanding how the original system works quite a challenge. However, even this loss of knowledge did not halt progress, and as the semester progressed, our team was able to narrow in on how the original computer functioned logically. This restoration of information is a great victory for the legacy of Atanasoff and Berry and for the history of computing.

Another accomplishment on the physical side of this project was that our team modularized each component from the original project into multiple PCB modules that could be assembled together. This alone is a great step in the overall goal of getting a 30-coefficient recreation of the original design. Since our design is meant to be scalable in groups of 5 coefficients per memory module, future students interested in this project have a strong foundation to build on and continue working to make this dream a reality.

Another major accomplishment of our team was creating a functioning VHDL simulation of our design to accurately test the logical design for this project. This VHDL simulation came in handy multiple times on this project, as it helped us simulate and verify which designs would and would not work the way we wanted. Our team decided to have a dedicated subteam to work on this simulation. This decision also allowed us to recreate every PCB module as a software VHDL version, which, when implemented together, accurately solves a 3x3 system of linear equations and handles all internal and external signals. This simulation was a great milestone in verifying that our design is functionally accurate and will also work in hardware.

Our team had many more accomplishments and successes that could be written here in this document. However, each success demonstrates each member's dedication to this project and their hope to see the recreation of the first computer. We anticipate this project will be an important piece of the ever-changing legacy of both Atanassoff and Berry here at Iowa State University and in the world of computing.

8.2. Value Provided

By working on this project, regardless of its completion, we have shown that the recreation of the original Atanasoff-Berry Computer is feasible and have provided a great deal of insight into this piece of history, which was once lost. For example, by diving deeper into

the interworkings of the different sections laid out in the Burks & Burks guide, our team was able to go into more detail about how each component was mechanically connected in the original system to better understand the effect of one signal on another. This information is vital in understanding how we have gone from the creation of the first computer to modern-day computers, which is also an important part of teaching computer architecture. This makes this project far more valuable than the cost to buy the components and fabricate the PCBs. This project adds more value to the legacy of the ABC and to Iowa State University, too, for the path it paved in computing technology that we have today. Furthermore, the main goal of this project is to serve as a teaching aid for students and professors, helping them better understand computer architecture and what it takes to build a computer, both today and historically. This project enables this by visualizing the data being passed in and the different internal flags that are raised to make it all work. Because of all of these systems, the value added to the legacy of the ABC and to this prestigious University itself is far greater than our team first realized when starting this project and goes far beyond the cost of the parts needed to complete this project.

8.3. Next Steps

There are three primary next steps for this project. The first is to create a second revision of the system to fix the defects identified during testing, implement identified design improvements, and create a more unified, polished product. Following this, the replica can be expanded to the full scale by assembling all six memory drum PCBs. Another desired feature that was discussed, but not addressed due to time constraints, was to create a method of automated control such that the user does not have to manually push the various buttons, move cards around, or other time consuming operations. Part of this feature would include general improvements to the look and stability of the software interfaces.

9. References

[1] B. Rokos, D. Damman, G. Tersteeg, G. Nordling, L. Lee, "i281 Hardware" ece.iastate.edu, [Online]. Available: https://sdmay24-14.sd.ece.iastate.edu/. [Accessed Oct. 28, 2025]

[2] Burks, A. R., & Burks, A. W., The First Electronic Computer: The Atanasoff Story. Ann Arbor, MI: University of Michigan Press, 1988. [Accessed 17 Sep. 2025]

[3] C. Hand, Z. Scurlock, W. Mayer, N. Butler, P. Hurd, "Implement the ABC Computer with Modern Technology," ece.iastate.edu, [Online]. Available: https://sdmay25-29.sd.ece.iastate.edu/. [Accessed Oct. 28, 2025]

[4] C. Berry, Manual for the ABC, Iowa State Pyhsicis Department, 1941. [Accessed 20 Sep. 2025]

[5] IEEE Std 1149.1™-2013, *IEEE Standard for Test Access Port and Boundary-Scan Architecture*, IEEE Computer Society, May 13, 2013.

[6] IEEE Std 802.15.1-2005, *IEEE Standard for Information Technology – Local and metropolitan area networks*, IEEE Computer Society, June 14, 2005.

[7] IEEE/IEC 61691-1-1:2023, Behavioural Languages – Part 1-1: VHDL Language Reference Manual, IEC/IEEE, Oct. 2023.

[8] IPC-2221, Generic Standard on Printed Board Design. Northbrook, IL: Association Connecting Electronics Industries (IPC), 1998.

[9] J. Gustafson, "Reconstruction of the Atanasoff-Berry Computer," johngustafson.net, [Online]. Available: www.johngustafson.net/pubs/pub57/ABCPaper.htm. [Accessed 28 Oct. 2025].

[10] Appendices

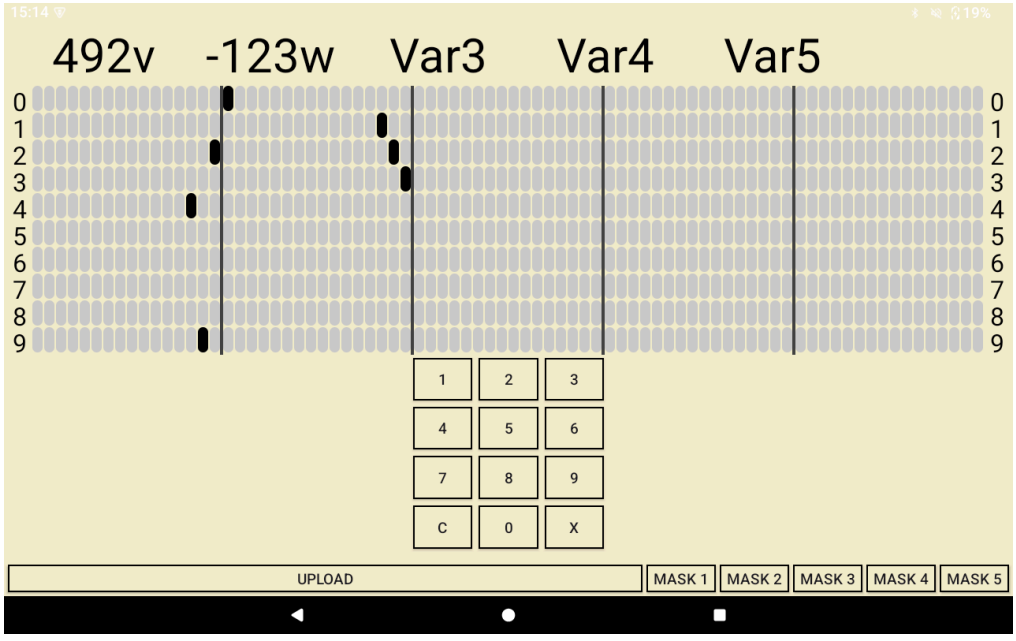
[a] Appendix 1 – Operation Manual

[i] Powering On

1. After connecting the system to power switch SW1 to the up position.
2. Briefly switch SW2 to the up position to reset the system, then change it back the the down position.
3. Clear both drums by setting SW62 and SW63 to the up position for at least one full cycle.

[ii] Reading Decimal Cards

1. Press the reset button by the decimal card reader.
2. On the decimal card reader tablet, enter the coefficients into the interface, pressing the “X” button after each coefficient. When finished press “Upload”.



3. Set SW68 to the up position to select card inputs to the Add/Subtract modules.
4. Set SW90 to the up position to let the card reader control the arithmetic operation.
5. Select which memory drum to read the card into by switching one SW70 to the up position.
6. Press button 85 to start the process.

7. Transfer the contents of CA to KA by setting SW69 to the down position for at least one cycle, then resetting it.
8. Repeat the process to read in the second equation.

[iii] Computing

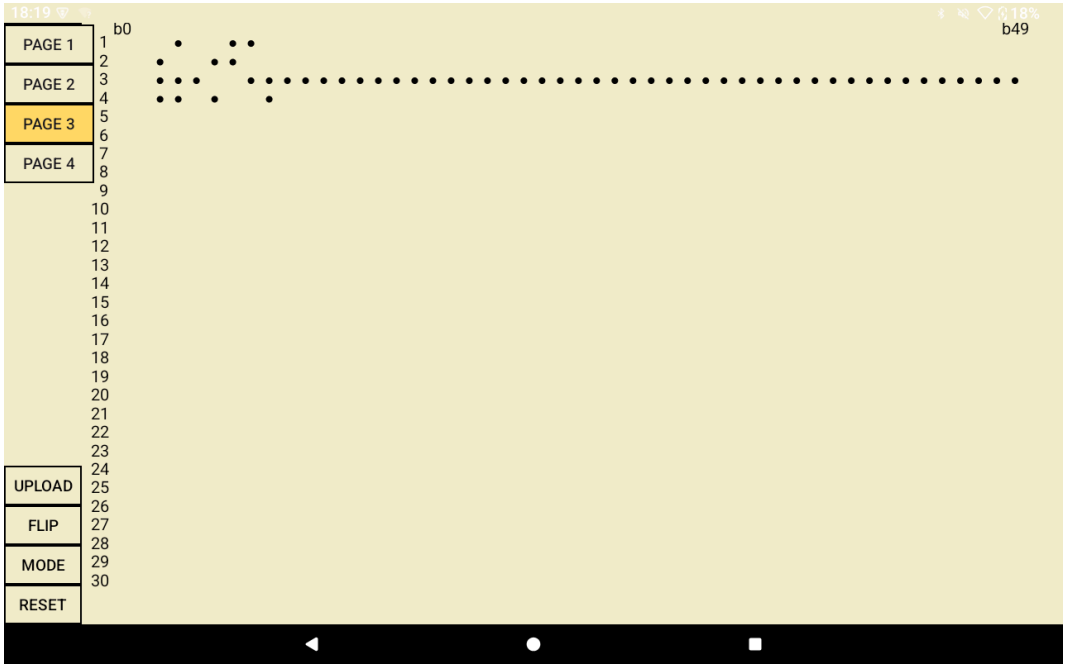
1. Set SW68 to the down position to select the KA inputs.
2. Select the coefficient to eliminate by placing a jumper in BRIDGE 4.
3. Set SW90 to the down position.
4. Use button 84 to set the operation to ADD if the signs of the coefficient to eliminate are different and SUB if they are the same.
5. Press button 86 to start computation.

[iv] Binary Writing

1. Push button 88 to write the contents of CA to the binary card

[v] Binary Reading

1. Select the card to read on the binary card reader tablet and press “Upload”.



2. Clear the drums if needed using SW62 and SW63.
3. Set SW68 to the up position to select card inputs to the Add/Subtract modules.
4. Push button 87 to start the read process.

[vi] Decimal Reading

1. Press the reset button on the decimal output module.

2. Select the coefficients to be read using bridge 8 and bridge 4. Upload the appropriate mask card from the decimal card reader and select the bank with the coefficient using SW70.
3. View the sign of the number after one full cycle.
4. Set SW71 to the up position to only read in the powers of ten values from the decimal card reader.
5. Set SW90 to the down position.
6. Press button 86 to start the reading process.

[b] Appendix 2 – Alternative/initial version of design

[i] Alternative Decimal Output Design

The design shown below was a preliminary design implementing Berry's description of the decimal output dials. Instead of having a mechanical borrow signal between certain dials, the display algorithm was adjusted and the dials that would require being moved back displayed zero twice. This design sought to implement this "double zero" functionality by blocking the first clock cycle for the appropriate counters.

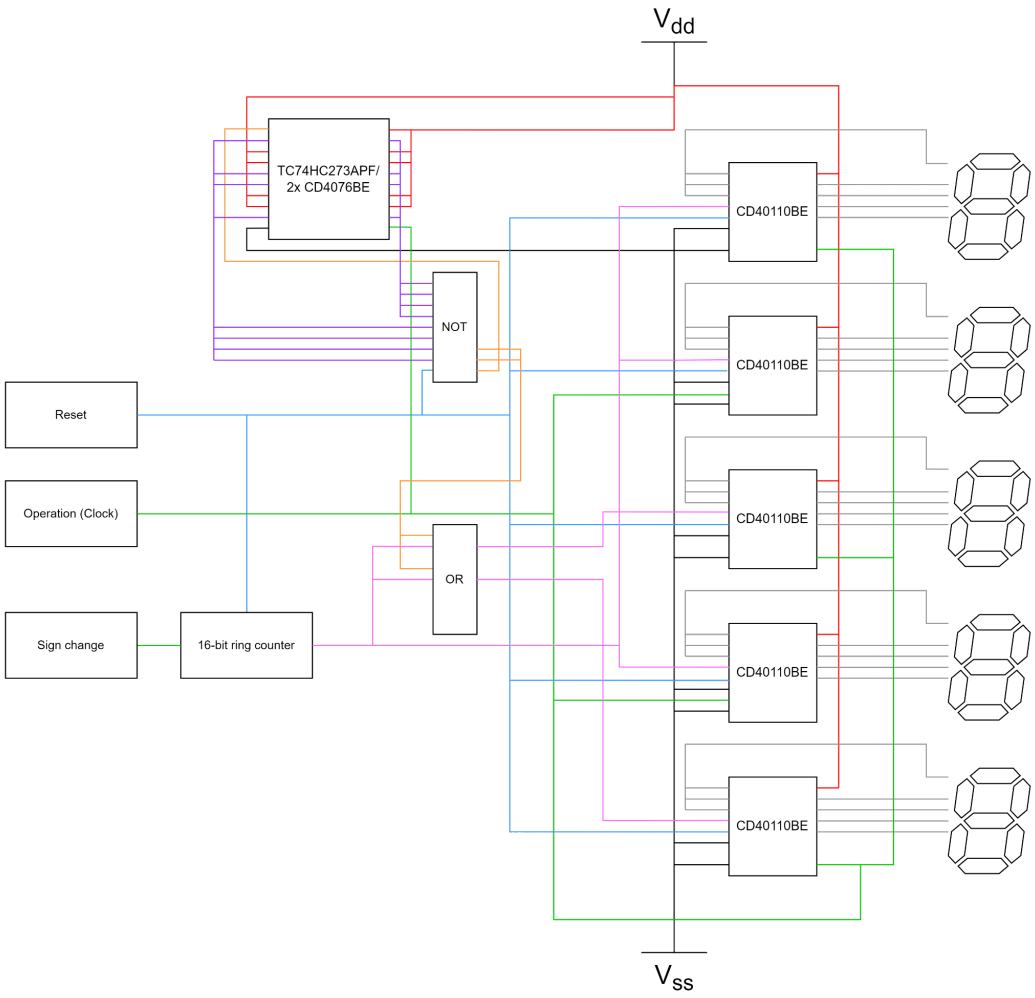


Figure 56. Alternative Decimal Output Design

[c] Appendix 3 – Other considerations

Thank you to the previous year's team (https://sdmay25-29.sd.ece.iastate.edu), Connor Hand, Zach Scurlock, William Mayer, Noah Butler, and Peter Hurd, for their initial work on the project including designs for the Add/Subtract module,

preliminary full-system designs, historical research into the ABC, and their card reader software interfaces.

[d] Appendix 4 – Code and Materials

Application code, KiCAD files, VHDL code, and other documentation can be found at https://git.ece.iastate.edu/sd/sdmay26-12

[e] Appendix 5 – Team Contract

Team Members

- Gavin Bunker
- Ryan McFadden
- Samuel Schroeder
- Vance Arnold
- Bryan York
- Kyle Koch

Required Skill Sets for Your Project

- **Digital logic and electrical circuit design** - Creating logic designs from available functional and technical descriptions will be essential for accurately recreating the ABC with modern technology. In addition to logical elements, our modules will also contain various other electrical components, like LEDs and seven-segment displays, for clearly showing the state and operation of the ABC.
- **PCB design with KiCAD** - All of our logical and electrical designs will need to be turned into PCBs for manufacturing and final assembly.
- **Soldering** - Once we receive our manufactured PCBs, individual components will need to be soldered on to complete the manufacturing process.
- **VHDL simulation** - Individual components and our design as a whole will be simulated in VHDL to verify correctness before PCB implementations of them are purchased.
- **Software development** - A user interface application will be developed that both accurately represents the ABC's user interface and simplifies some of the manual operations for easier demonstrations.

Skill Sets Covered by the Team

Skill	Covering Team Member(s)
Digital logic and electrical ciruit design	Everyone
Soldering	Bryan York, Gavin Bunker
PCB design with KiCAD	Bryan York
VHDL simulation	Kyle Koch, Vance Arnold, Ryan McFadden, Gavin Bunker
Software development	Samuel Schroeder

Figure 57. Skills Table

Project Management Style Adopted by the team

For this project, we have adopted a modular waterfall project management style, where each component or module follows the waterfall model, but there may be several active waterfalls at any given time. For more information about the selected project management style, see section 3.1.

Individual Project Management Roles

For this project, we have two initial project management roles: project owner and project manager. Dr. Stoytchev, our client and faculty advisor, will serve as the project owner by specifying the broader goals and deadlines and prioritizing which modules should be developed first. Samuel Schroeder will serve as the project manager by performing more of the day-to-day coordination of team members and individual tasks.

Team Contract

Team Members

1) ____Gavin Bunker_____ 2) ____Ryan McFadden_____

3) ____Samuel Schroeder_____ 4) ____Vance Arnold_____

5) ____Bryan York_____ 6) ____Kyle Koch_____

Team Procedures

Team meetings will occur after our senior design class ends on Tuesdays for an hour in person at the library. Other weekly sessions will be held between all students and Dr. Stoytchev, faculty advisor, in the senior design lab in Coover 1301 at 2:00 pm on Thursdays.

These times are subject to change on the agreement between all parties involved to allow for better attendance or as otherwise noted by the student team.

Any communication that needs to occur outside of these team meetings will be conducted over our preferred method of communication, Discord for student team questions and comments, or via email when needing to involve the Faculty advisor.

Decisions will be made on various criteria depending on the subject matter that the decision will affect. A breakdown of the hierarchy of this process is outlined below and agreed upon by all student team members by signing this contract at the bottom of the document.

Changes requested by the client shall be enacted and worked into the project

a. Decisions regarding an individual's or a subgroup's (ex, hardware or software) responsibility are decided by the members in that subgroup (not involving other subgroup votes).
b. A majority vote is needed for decisions requiring a group or subgroup, with tiebreakers being decided by the project leader.
c. Only members present may vote.
d. Any individual may call for a vote.
e. Every individual present must vote.

The student team member designated as the recorder will record all action items, business conducted, and any other important legislation brought up during any meetings. All notes will be taken and shared on Google Docs within our shared Google Drive folder so all students can view them.

Participation Expectations

A student team member may miss meetings when necessary, provided they give at least 12 hours' notice. Members are allowed a maximum of three absences without extenuating circumstances and cannot have consecutive absences without a valid reason.

All students are expected to meet deadlines set by the project leader. Each individual is responsible for communicating if they believe a deadline is unrealistic or anticipate a delay.

If a topic requiring a student's attention is mentioned in the preferred communication method (Discord), the student must respond within 24 hours.

Students are committed to fulfilling the requirements established by the client and the project leader as outlined.

Leadership

1. Student team members have all been assigned unique roles to fulfill, which will all benefit the group throughout this project. These students and their roles are outlined below:
    a. Team Leader: Samuel Schroeder
    b. Technical Manual Lead: Gavin Bunker
    c. PCB Lead: Bryan York
    d. VHDL Lead: Kyle Koch
    e. Client Liaison, Recorder: Ryan McFadden
    f. Team Facilitator, Parts & Purchasing: Vance Arnold

2. Strategies for supporting and guiding the work of all team members:

To maintain a productive learning environment and group dynamic, all criticism is to be given constructively. There will be no non-constructive criticism given without subject to penalties in the "Consequences for Not Adhering to Team Contract" section of this contract, or a majority of the members present at the given meeting see cause to inflict. Constructive criticism should be given when appropriate and promptly.

During team meetings, each team member will state their most significant contribution (1) since their last meeting, with a brief overview. If they made no noticeable progress, they will discuss why that was.

Collaboration and Inclusion

An outline of each student team member, along with the assets brought to the project team, is given below:

a. Gavin Bunker: Non-traditional student with over 7 years of active duty in the Navy and experience operating a nuclear reactor onboard an operational submarine.
a. Ryan McFadden: A variety of experience in multi-disciplinary engineering, incorporating mechanical design with software and electrical hardware. Good communicator and team worker.
b. Vance Arnold: Computer engineering student with hands-on experience in the hardware and software industry, with skills including coding, embedded, and general problem-solving. Prioritize communication and cooperation.
c. Bryan York: Electrical Engineering student who specializes in VLSI and Hardware design. I like to be hands-on and always ensure I work towards the end goal.
d. Kyle Koch: Computer Engineer with hands-on experience in electrical and Robotic engineering with a focus on Automation. I enjoy working on hardware design on circuits, and computer components.
e. Samuel Schroeder: Software Engineer with experience in software development for embedded software to high-level languages. I have an interest and expertise in cybersecurity, system architecture, and scripting.

Each member will have the opportunity to bring up subjects of their choice at the conclusion of the team meeting schedule and before concluding the meeting. Issues regarding collaboration or inclusion should be brought up for discussion by the relevant team member during the team meeting, as stated in the section "Participation

Expectations" above. Any individual found to be in breach of the outlined contract, and any amendments voted and passed by the group, will have the following occur:

1. The group will take an initial vote on whether to warn the individual.
2. The group may also vote on whether to notify the instructor following a warning to the individual.
3. The instructor may not be notified unless the vote is unanimous or they have at least two warnings.

Goal-Setting, Planning, and Execution

Below is a list of goals set forth by the faculty advisor assigned to this project, the project outline document given to the student team members when signing up for projects at the beginning of the year, and other team and individual goals set forth by the student team members.

1. Team goals for this semester:
    a. Complete the project as outlined by the client on the schedule defined by the project leader.
2. Strategies for planning and assigning individual and team work:
    a. The team leader's main task will be coordinating the team and setting forth meeting plans. They are to assign individuals deadlines and timelines, and collaborate with them to ensure the deadlines are reasonable.
3. Strategies for keeping on task:
    a. The team facilitator will ensure that the meetings are time-effective. The project leader will follow up with individuals on project deadlines.

Consequences for Not Adhering to Team Contract

Any individual found to be in breach or in violation of the outlined contract, and any amendments voted and passed by the group, will have the following occur. Continuous violations of the agreement will be subject to the same penalties as the other student team members and faculty advisor, as needed, working their way through the given leadership structure outlined in the surrounding EE/CPRE 491/492 class structure until any problems are resolved.

1. The group will take an initial vote on whether to warn the individual.
2. The group may also vote on whether to notify the instructor following a warning to the individual.
3. The instructor may not be notified unless the vote is unanimous or they have at least two warnings.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) Bryan York _____ DATE: 03 September 2025

2) Vance Arnold_____ DATE: 09 September 2025

3) Ryan McFadden_____ DATE: 09 September 2025

4) Samuel Schroeder _____ DATE: 09 September 2025

5) Gavin Bunker_____ DATE: 11 September 2025

6) Kyle Koch _____ DATE: 13 September 2025