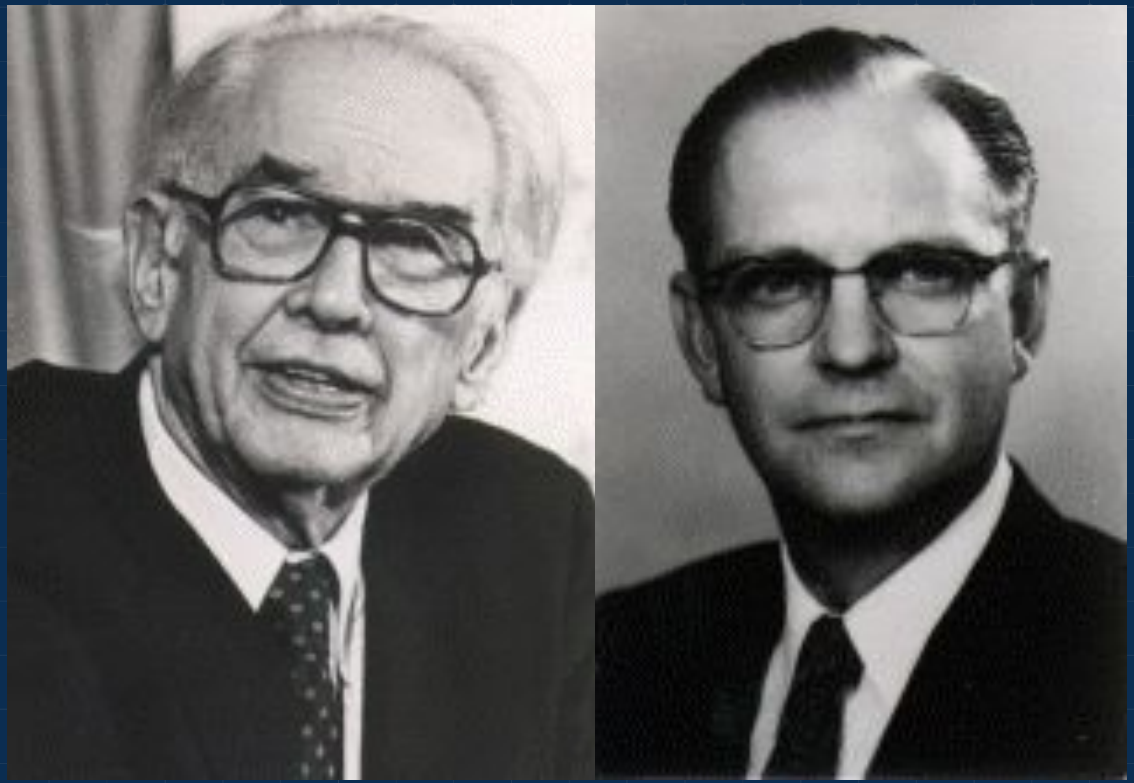
Implementing the ABC Computer with Modern Technology (Phase 2)

sdmay26-12

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

Faculty Advisor: Dr. Stoytchev

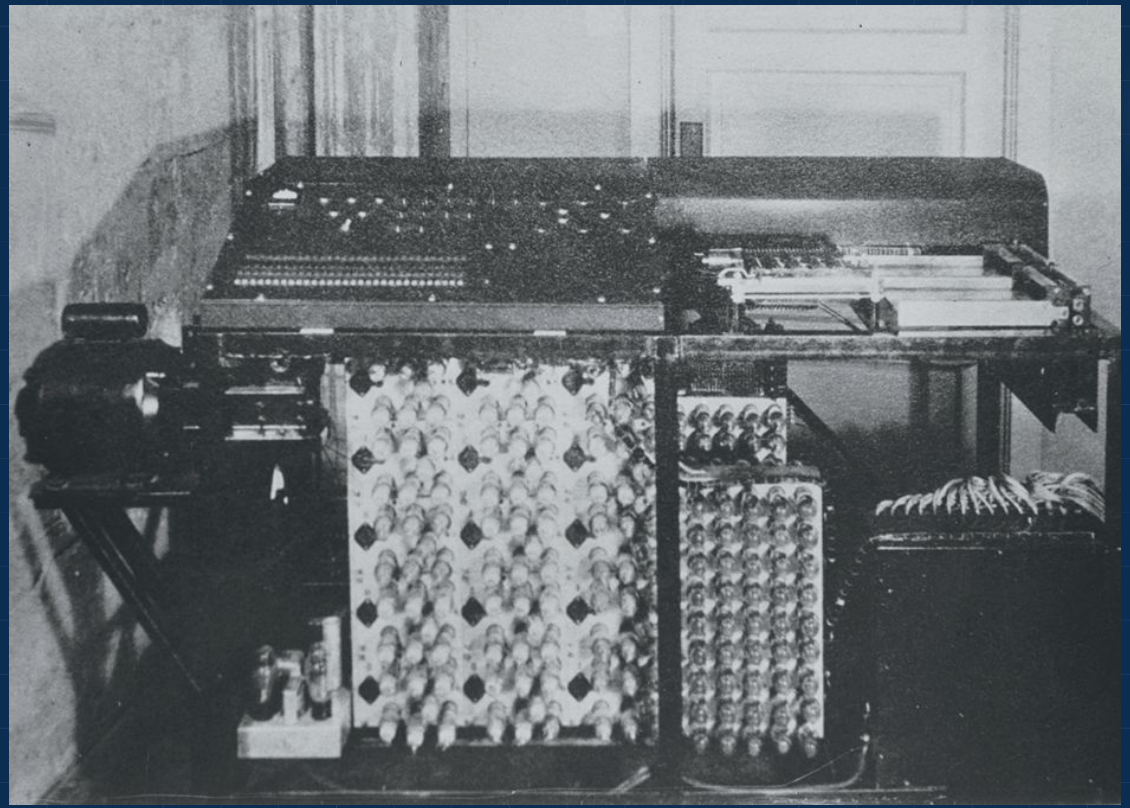
The Creators



John Vincent Atanasoff
https://jva.cs.iastate.edu/jvabio.php

Clifford Berry
https://jva.cs.iastate.edu/berrybio.php

Atanasoff-Berry Computer



ABC, built
1939-1942

https://jva.cs.iastate.edu/operation.php

Technology



Rotating Memory Drums

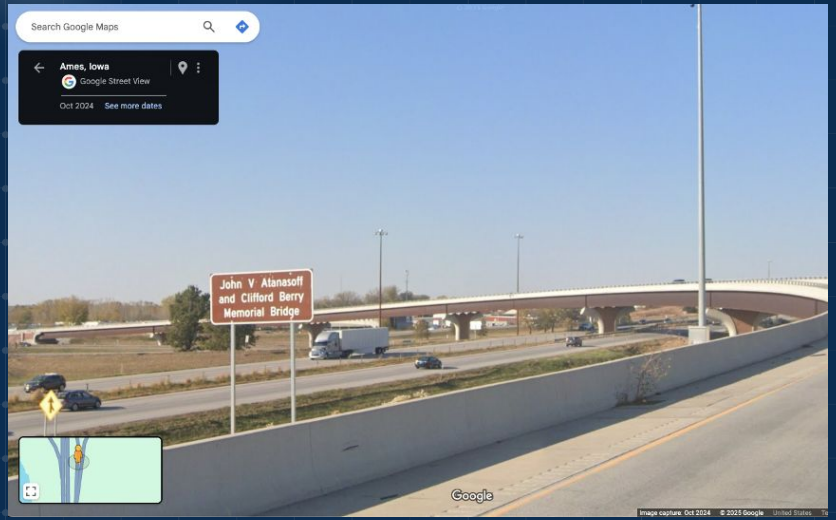
https://jva.cs.iastate.edu/operation.php



Vacuum Tube Circuits

https://en.wikipedia.org/wiki/Atanasoff%E2%80%93Berry_computer

Legacy



Northbound I-35 to Westbound U.S. 30

https://www.google.com/maps



Flyover Bridge with Punch Card Detailing

https://iowadot.gov/consultants-contractors/bridges-structures/featured-bridges/flyover-bridge



Plaque Outside of Physics Hall
https://www.fpm.iastate.edu/maps/memorials/marker.asp?id=16-03

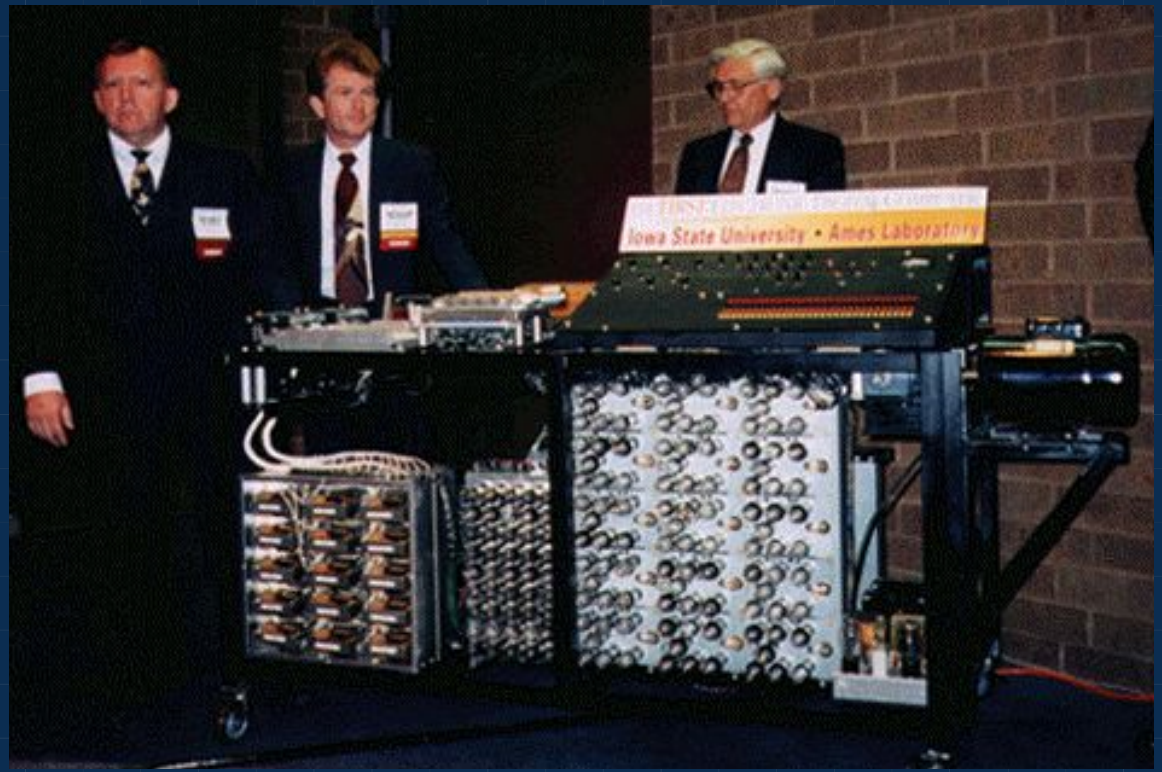
ABC Reconstruction (Ames Lab, 1994-1997)

Successes:

- Original mechanical design
- Looks like the original ABC

Shortcomings:

- $300,000 budget (~$650,000 today)
- Lacks technical documentation
- No longer at ISU



https://jva.cs.iastate.edu/team.php

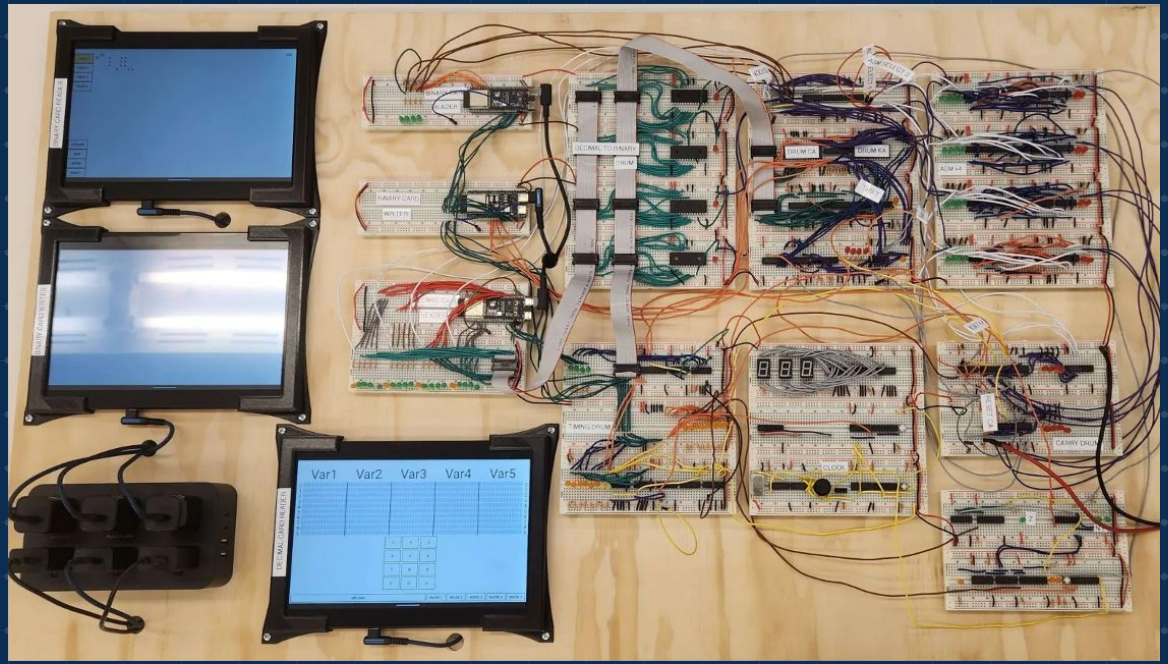
Partial Breadboard Recreation - sdmay25-19

Successes:

- Incredible amount of research and discovery work on ABC's design
- Recreated the logic of most modules on breadboards and integrated them to solve systems

Shortcomings:

- Significantly scaled down (computes 3x3 systems)
- Did not implement all of the modules of the ABC (such as the decimal output)



https://sdmay25-29.sd.ece.iastate.edu/

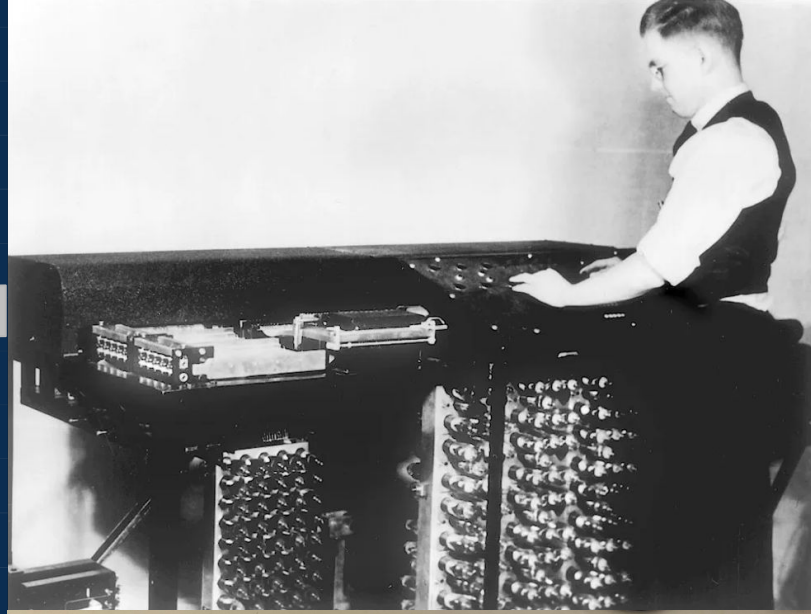
Problem Statement

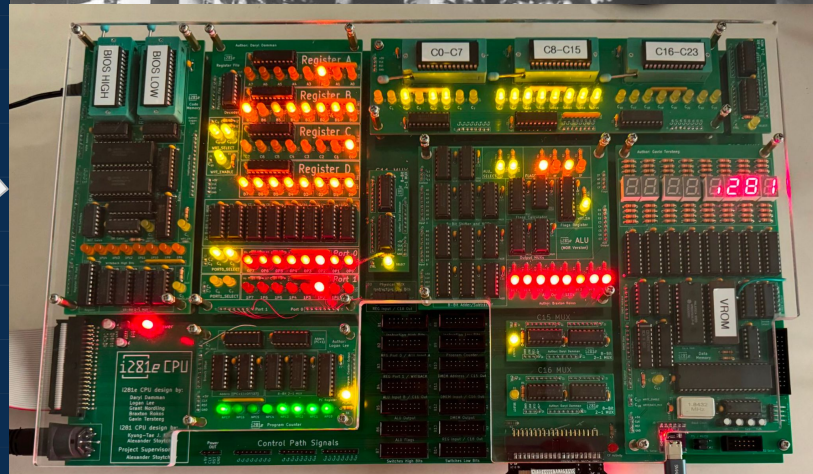
Implement the logical design of the ABC using modern components

Design PCBs for each of the ABC's modules and integrate them onto a motherboard for a full-scale logical recreation of the ABC

Purpose:

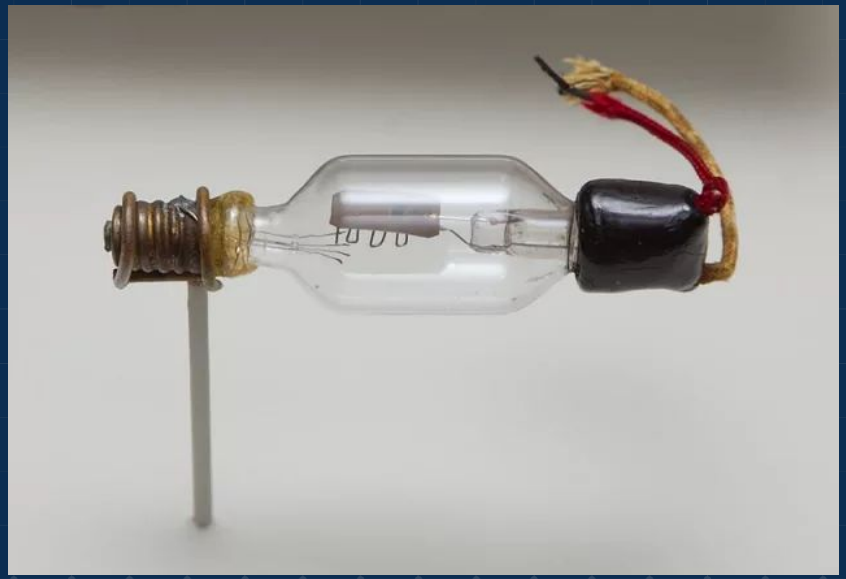
- Restore lost history
- Teach students and ISU visitors about computing logic and design





https://jva.cs.iastate.edu/berrybio.php https://sd-archive.ece.iastate.edu/sdmay24-14/

Functional Requirements



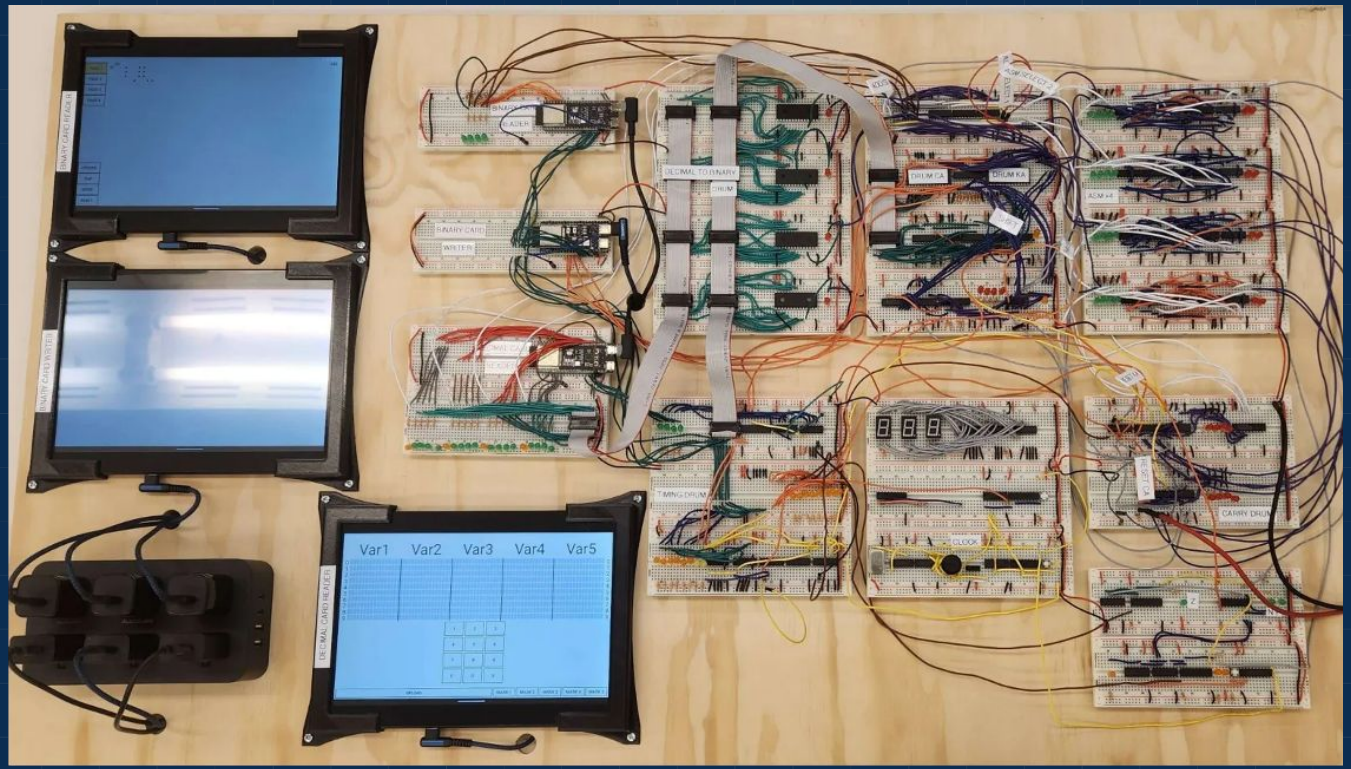
https://www.engineering.com/vacuum-tubes-the-world-before-transistors/



https://twen.rs-online.com/web/p/inverters-ics/0307503?srsltid=AfmBOorOb9zcAIYTTrmtgXFPHACvz-bvUbWgLwtGqh4OPugaa9H8-zyJ

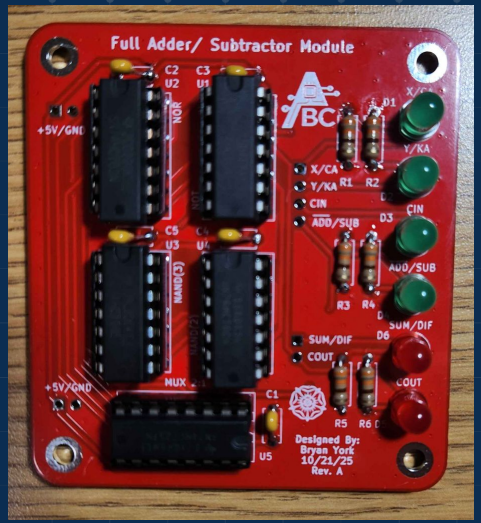
- Be as accurate as possible to original ABC
- Have the ability to solve 29 x 29 systems of linear equations

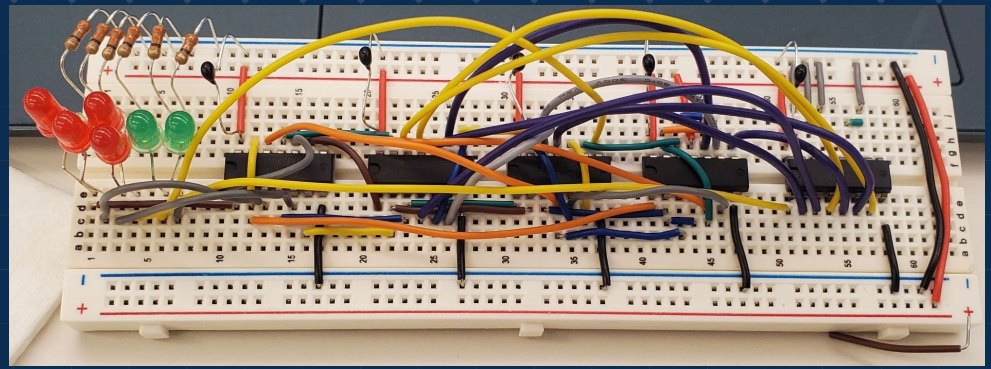
Physical Requirements



- Physical device easily usable by the user
- Model should be modular

User Experience Requirement





- Model should have LED's to show states and functionality
- Easily usable with help of the instruction manual
- Overall is visually appealing

IEEE Standards

- IEEE Std 1149.1™-2013, IEEE Standard for Test Access Port and Boundary-Scan Architecture
- IEEE/ANSI Std 91/91a-1991, IEEE/ANSI Standard Graphic Symbols for Logic Functions
- IEEE/IEC 61691-1-1:2023, Behavioural Languages – Part 1-1: VHDL Language Reference Manual
- IPC-2221, Generic Standard on Printed Board Design



https://www.prnewswire.com/news-releases/ieee-standards-in-interactive-html-now-available-in-ieee-xplore-300271566.html

Design & Implementation

High Level Design

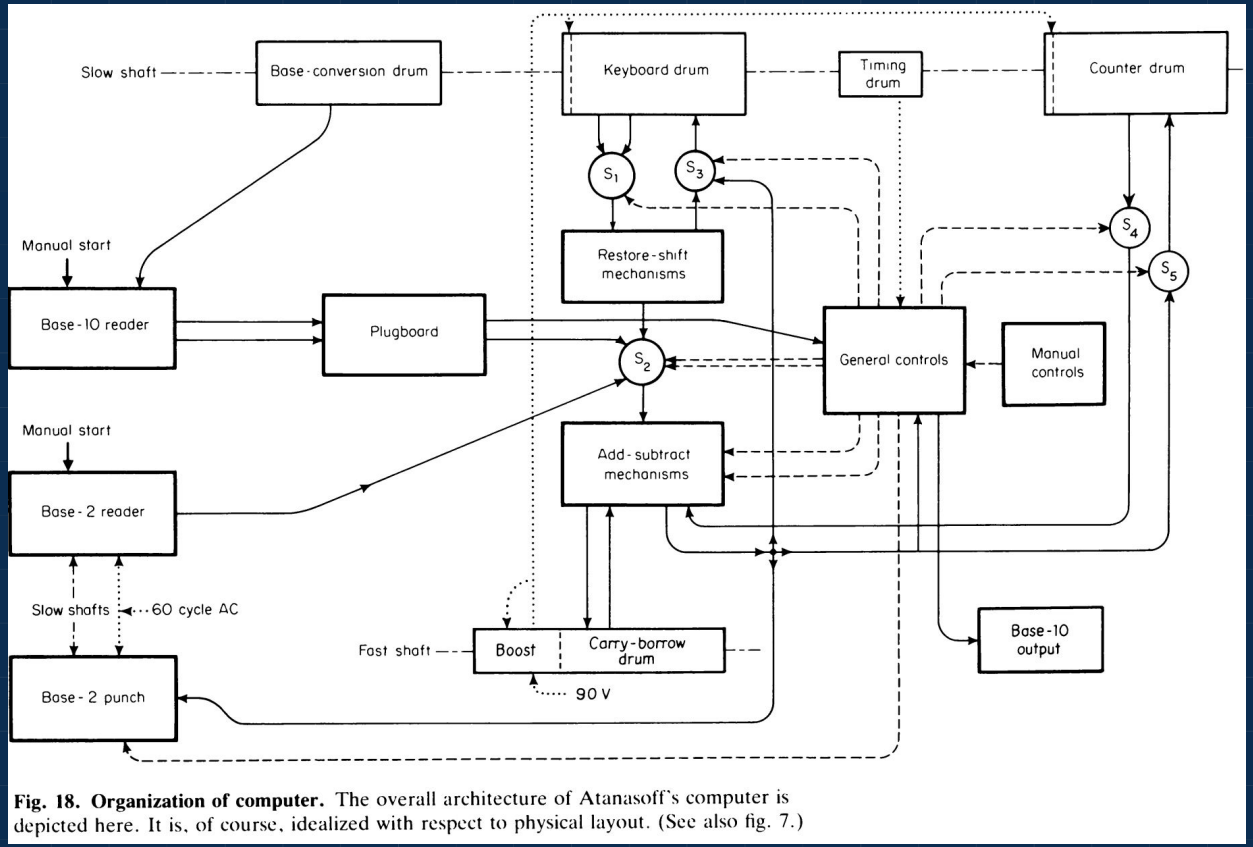
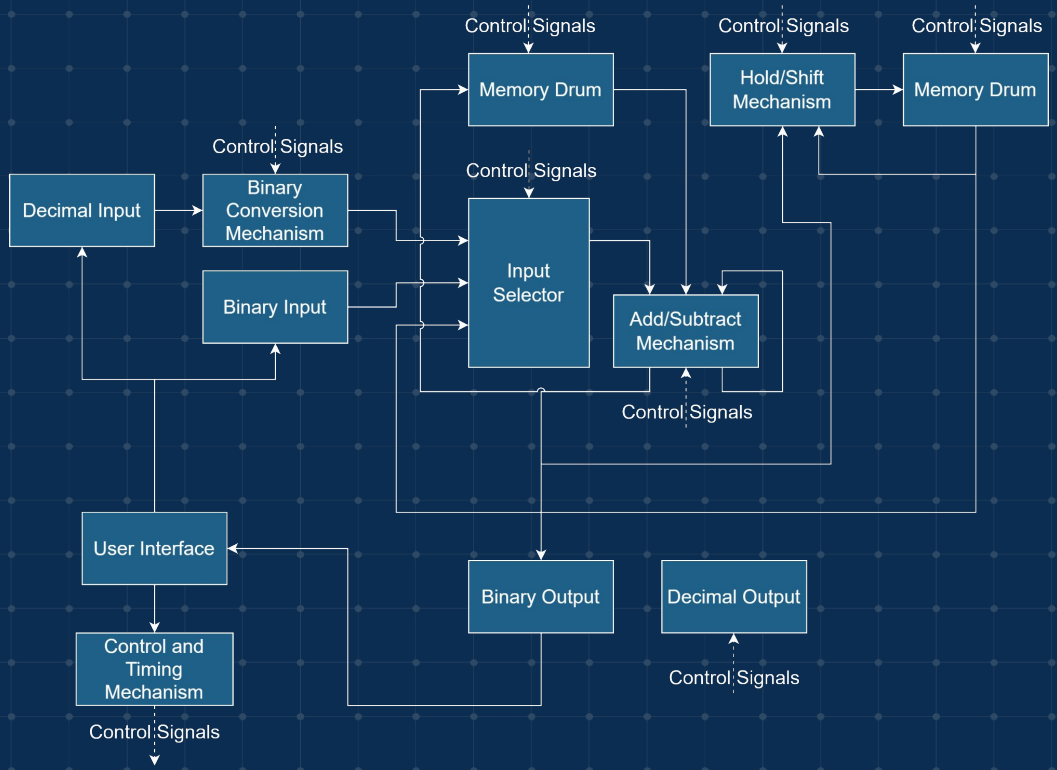


Fig. 18. Organization of computer. The overall architecture of Atanasoff's computer is depicted here. It is, of course, idealized with respect to physical layout. (See also fig. 7.)

From *The First Electronic Computer* by Burks and Burks

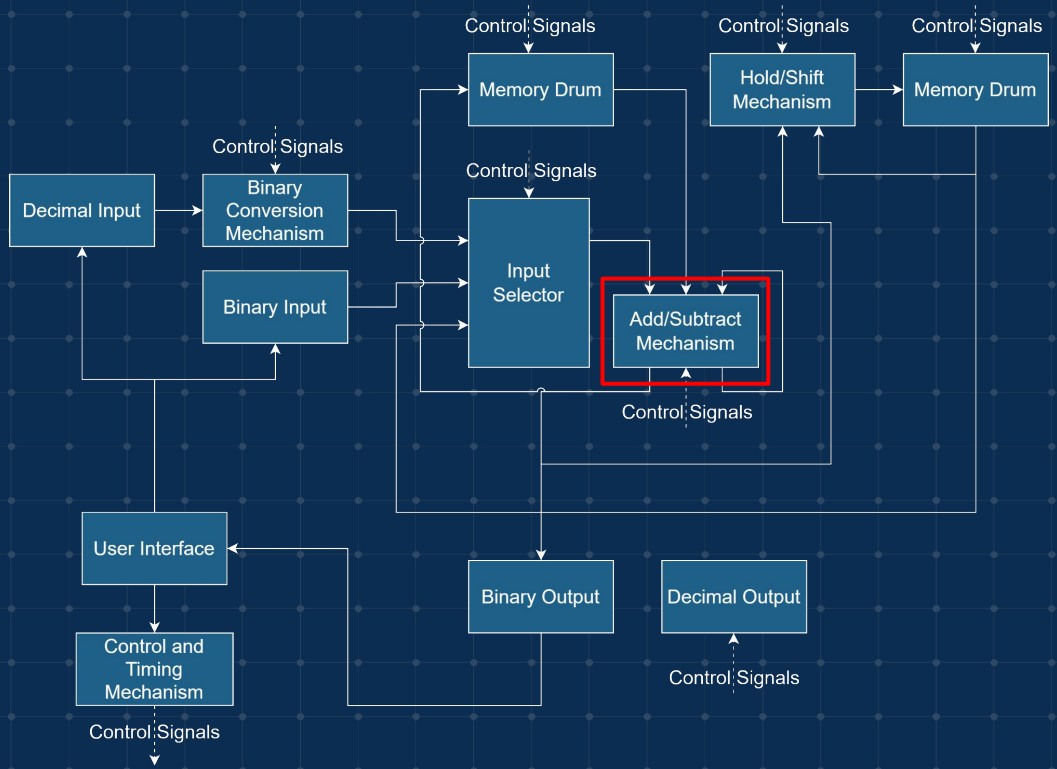
High Level Design



Design & Testing Process

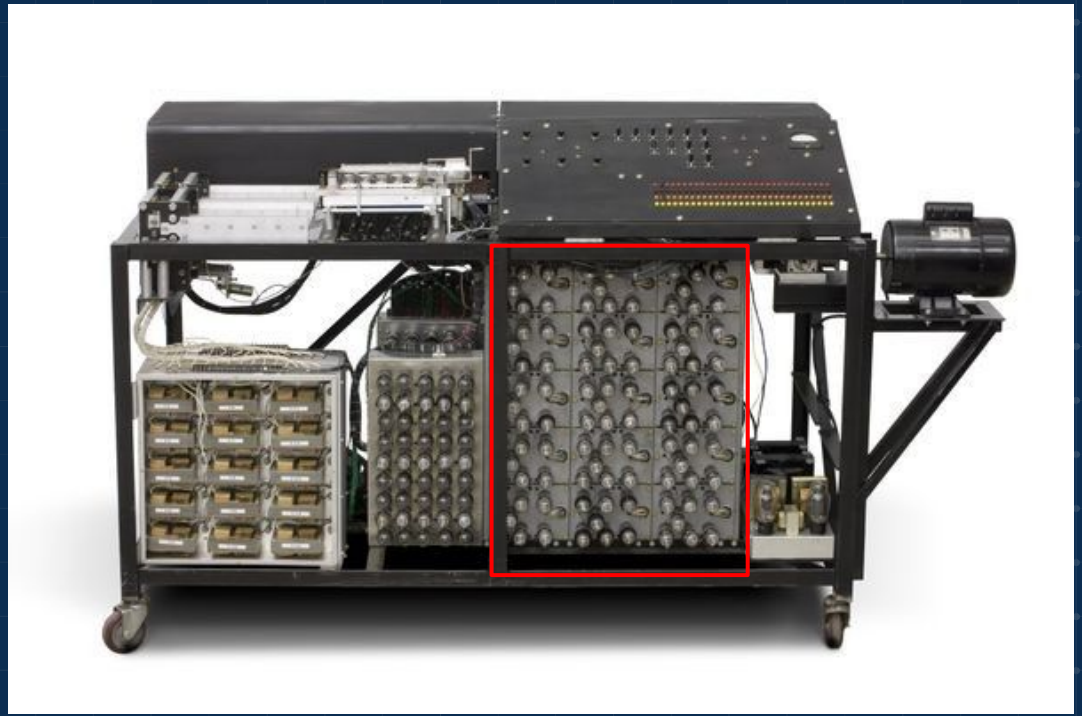


Add/Sub Module

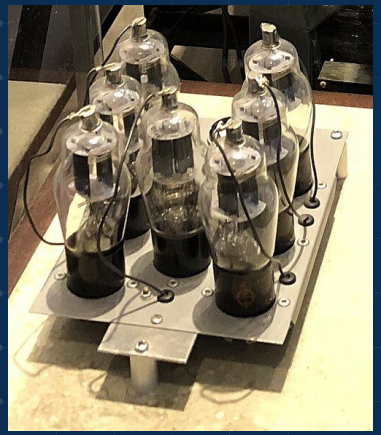


Add/Sub Module: Design

1997 ABC Replica



https://sdmay25-29.sd.ece.iastate.edu/



https://en.wikipedia.org/wiki/Atanasoff%E2%80%93Berry_computer

Add/Sub Module: Design

	Variable Inputs			Control Settings		Intermediate Switching Points												Outputs	
	Addend α_i	Augend β_i	Carry in γ_i	f	g	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_{10}	P_{11}	P_{12}	P_{13}	Sum σ_i	Carry Out γ_{i-1}
	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	0	0	0	0
	0	0	1	0	1	1	1	0	0	1	0	1	1	0	1	0	0	1	0
	0	1	0	0	1	1	0	1	0	1	0	1	1	0	1	0	0	1	0
	0	1	1	0	1	1	0	0	0	1	1	0	1	1	1	0	1	0	1
	1	0	0	0	1	0	1	1	1	0	0	1	1	1	0	0	0	1	0
	1	0	1	0	1	0	1	0	1	1	0	1	0	1	1	0	1	0	1
	1	1	0	0	1	0	0	1	1	1	0	1	0	1	1	0	1	0	1
	1	1	1	0	1	0	0	0	1	1	1	0	1	1	0	0	1	1	1
Height	0					1			2			3	4		5			6	

(a) Addition

	Variable Inputs			Control Settings		Intermediate Switching Points												Outputs	
	Minuend α_i	Subtrahend β_i	Borrow in γ_i	f	g	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_{10}	P_{11}	P_{12}	P_{13}	Difference σ_i	Borrow Out γ_{i-1}
	0	0	0	1	0	1	1	1	0	0	0	1	1	1	1	0	0	0	0
	0	0	1	1	0	1	1	0	0	1	0	1	1	0	1	1	0	1	1
	0	1	0	1	0	1	0	1	0	1	0	1	1	0	1	1	0	1	1
	0	1	1	1	0	1	0	0	0	1	1	0	1	1	1	1	0	0	1
	1	0	0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1	0
	1	0	1	1	0	0	1	0	1	1	0	1	0	1	1	0	0	0	0
	1	1	0	1	0	0	0	1	1	1	0	1	0	1	1	0	0	0	0
	1	1	1	1	0	0	0	0	1	1	1	0	1	1	0	1	0	1	1
Height	0					1			2			3	4		5			6	

(b) Subtraction

Fig. A.18. Truth table for the add-subtract mechanism

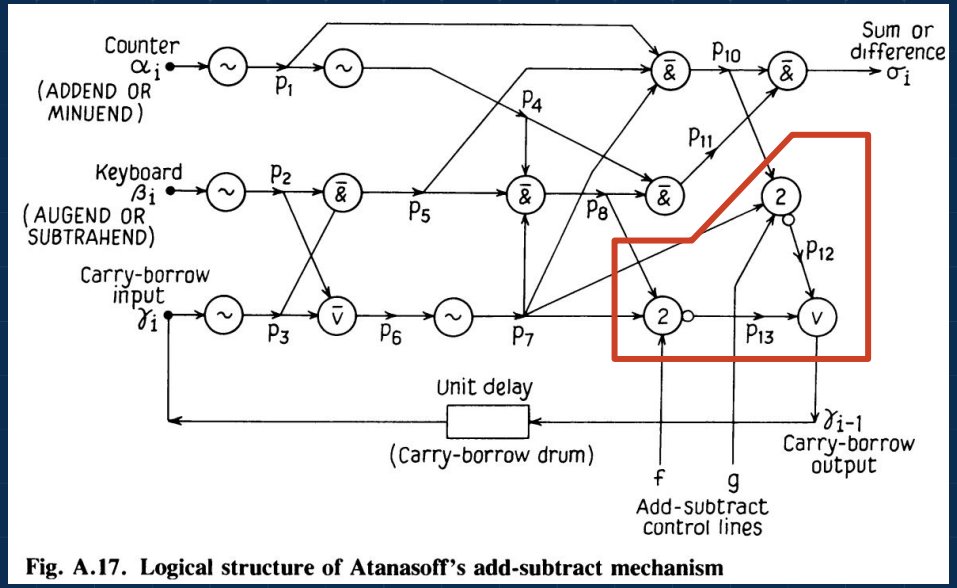
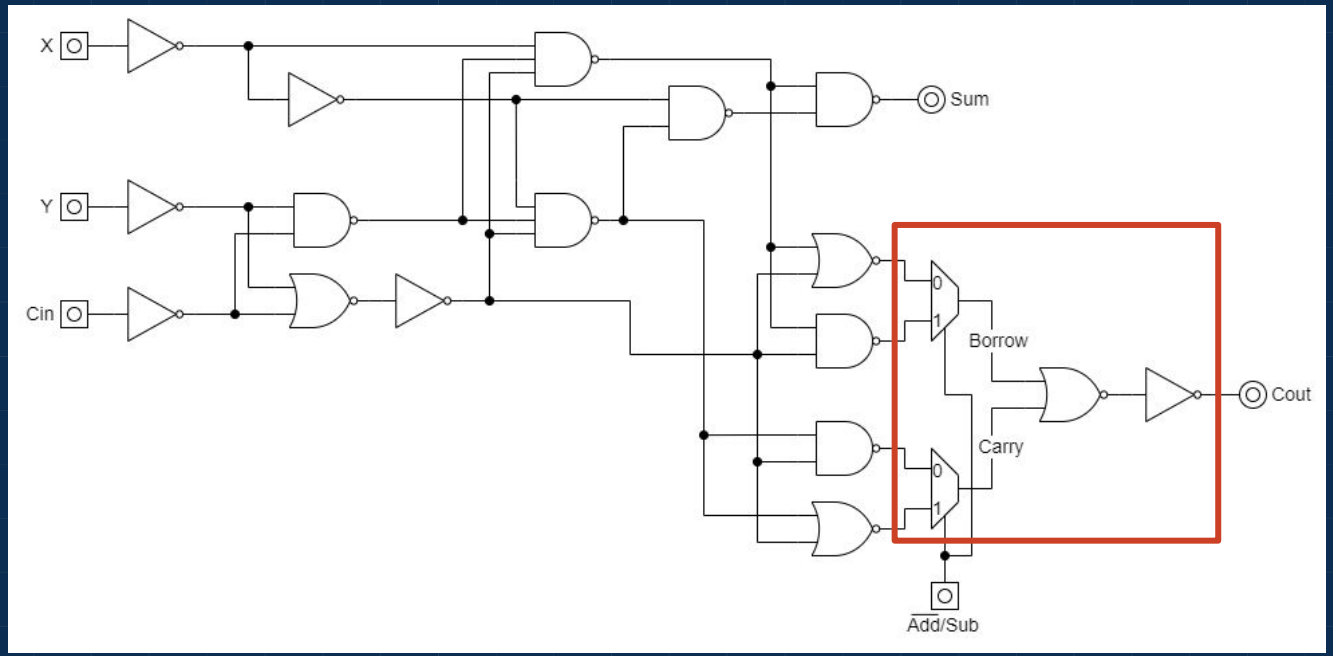


Fig. A.17. Logical structure of Atanasoff's add-subtract mechanism

From *The First Electronic Computer* by Burks and Burks

Add/Sub Module: Design



Add/Sub Module: Design Verification

```
entity AddSubModule is
port (x, y, cin, AddSub: in std_logic; Sum, Cout : out std_logic);
end entity AddSubModule;


architecture Structural of AddSubModule is

    -- Declare internal signals
    signal P1, P2, P3, P4, P5, P6, P7, P8, P10, P11,
    P12, P13, m1A, m1B, m4a, m4b : STD_LOGIC;
        begin
        P1 <= NOT x;
        P2 <= NOT y;
        P3 <= NOT cin;
        P4 <= NOT P1;
        P5 <= P2 NAND P3;
        P6 <= P2 NOR P3;
        P7 <= NOT P6;
        P8 <= NOT (P4 AND P5 AND P7);
        P10 <= NOT(P1 AND P5 AND P7);
        P11 <= P4 NAND P8;
        m1a <= P7 NOR P10;
        m1b <= P10 NAND P7;
        m4a <= P7 NAND P8;
        m4b <= P8 NOR P7;
        P12 <= ((NOT AddSub) AND m1a) OR (AddSub AND m1b);
        P13 <= ((NOT AddSub) AND m4a) OR (AddSub AND m4b);
        Cout <= NOT (P12 NOR P13);
        Sum <= P10 NAND P11;


end Structural;
```

Addition Implementation

F = X + Y + Cin

Subtraction Implementation

F = (X-Y) -Cin

Inputs:

X => 1'b0

Y => 1'b1

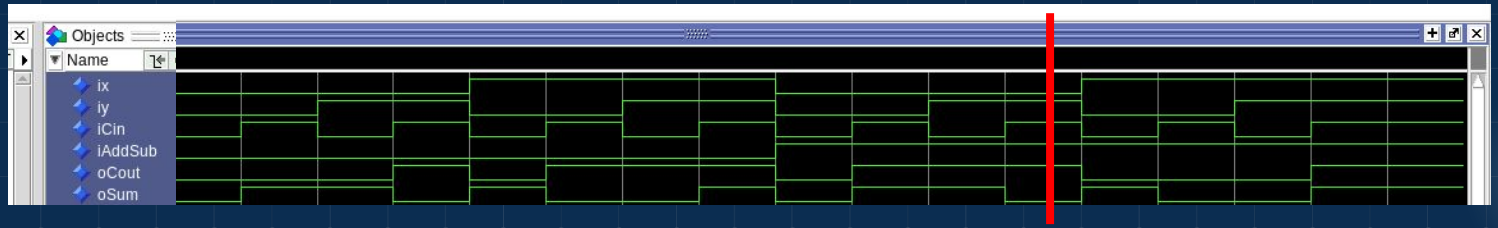
Cin => 1'b1

Add/Sub => 1b'1

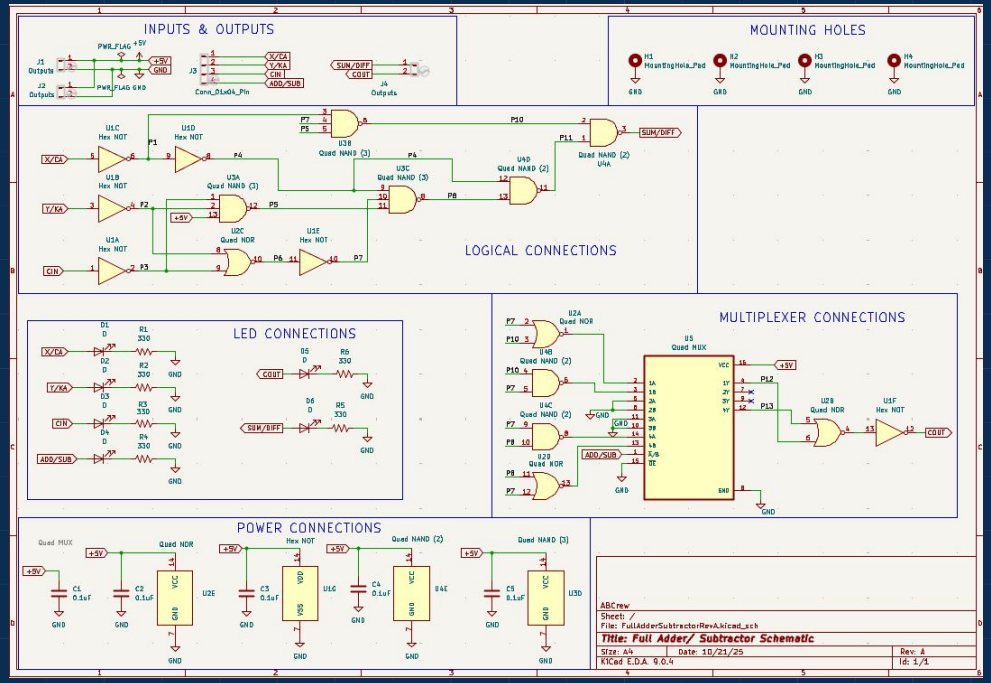
Outputs:

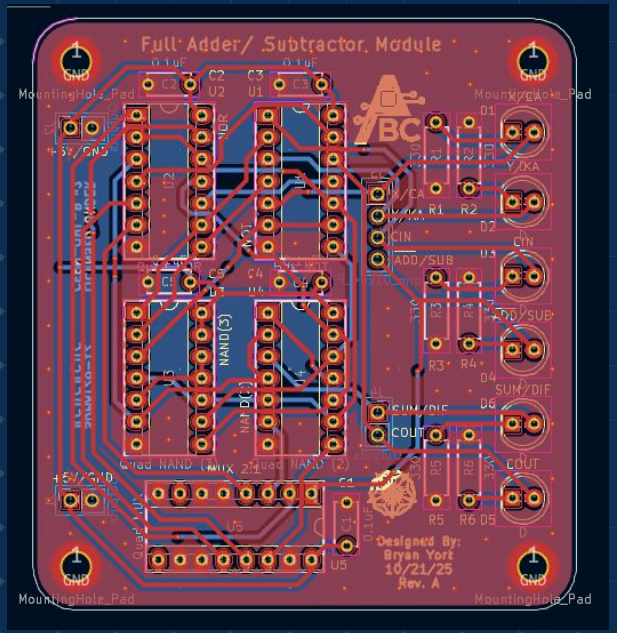
Cout => 1'b1

Sum => 1'b0

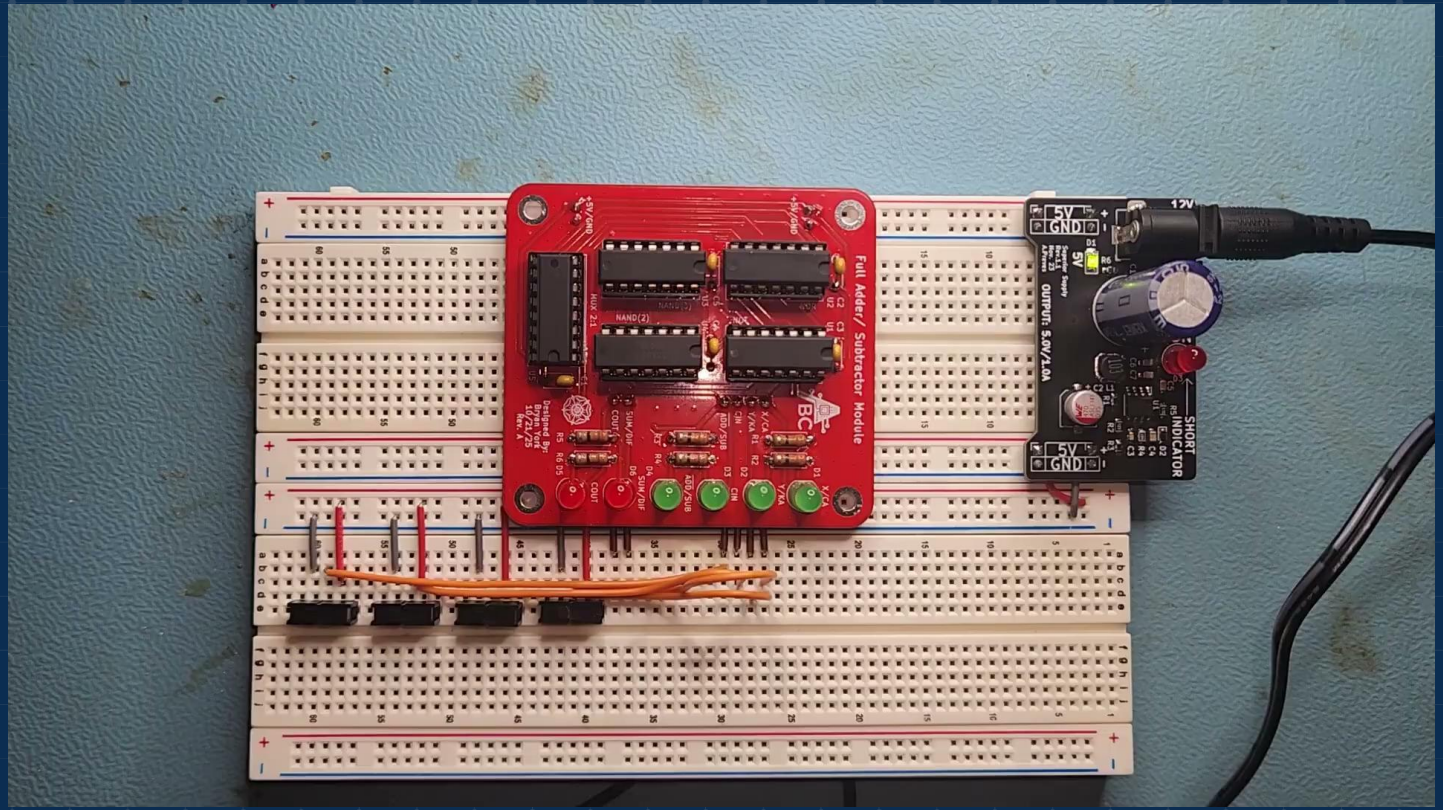


Add/Sub Module: PCB Design



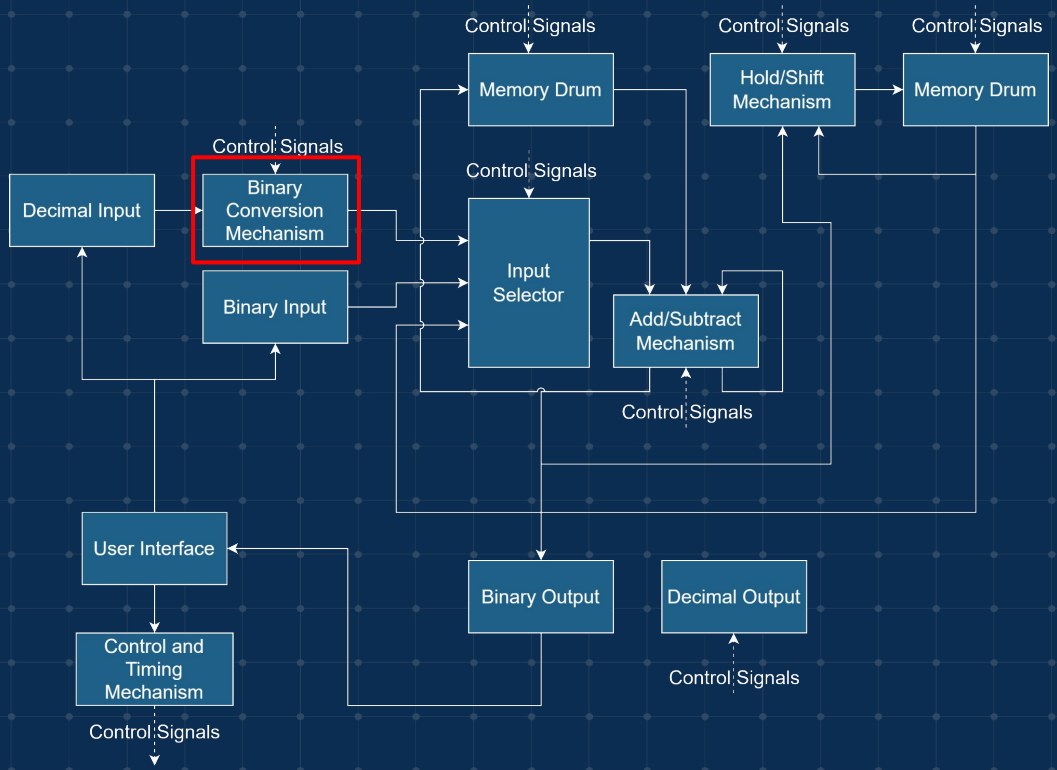


Add/Sub Module: PCB Verification

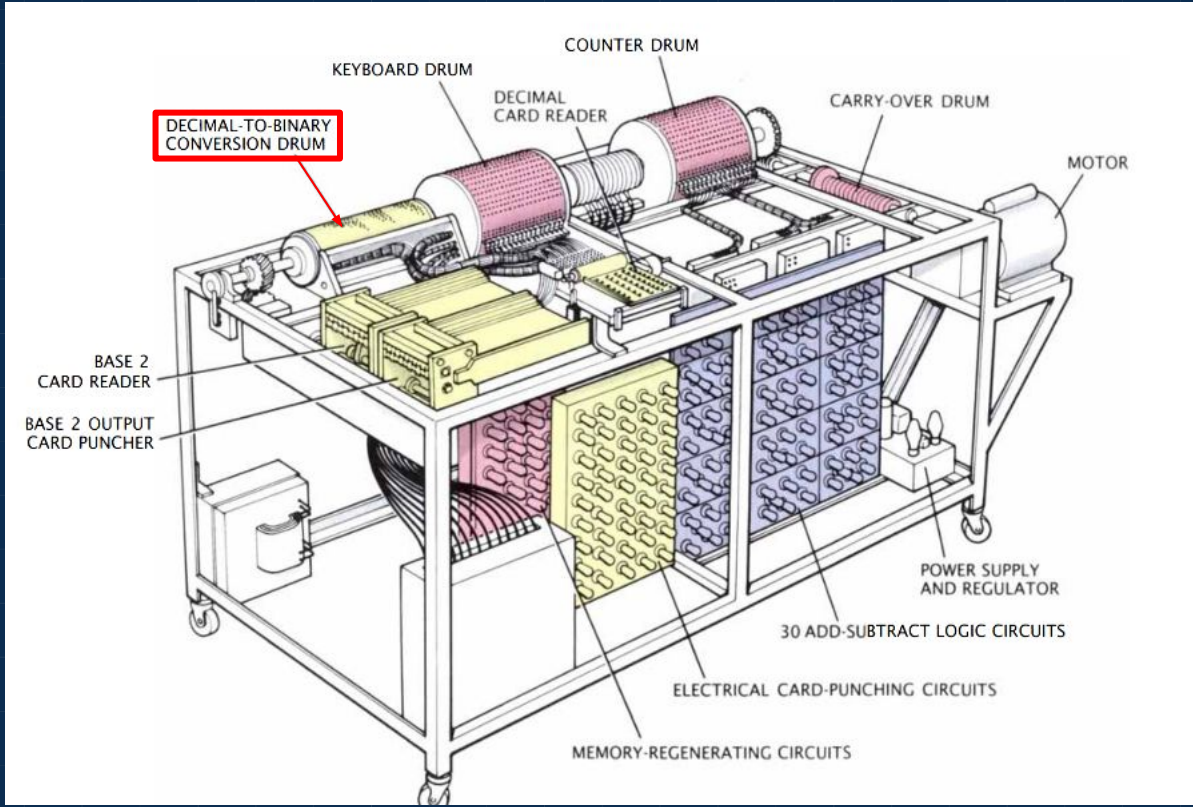


Progress on Additional Modules

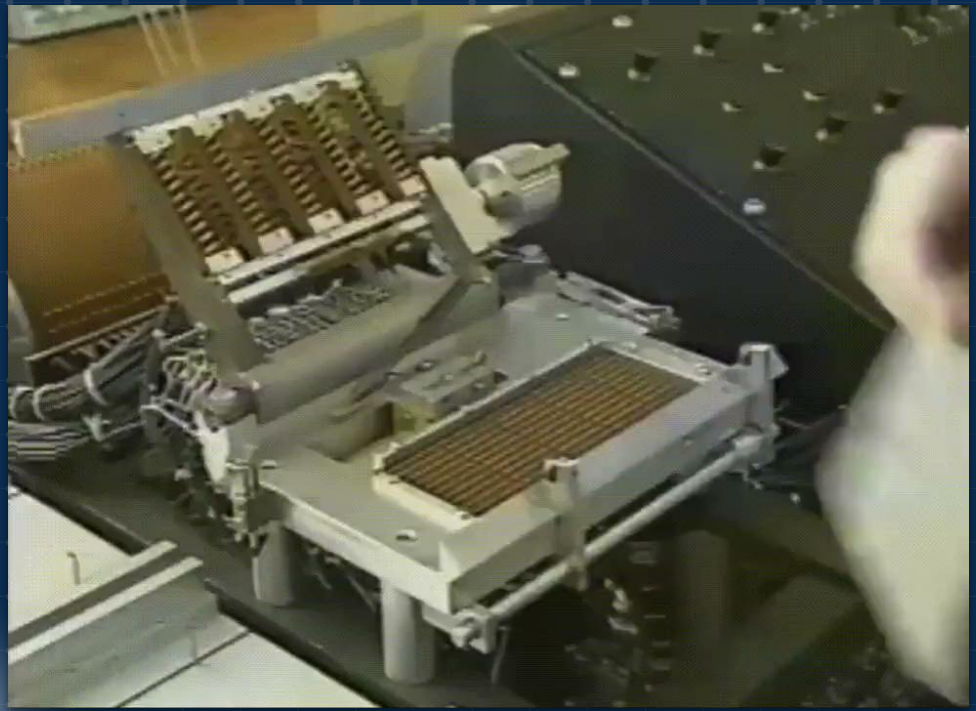
Decimal to Binary Conversion



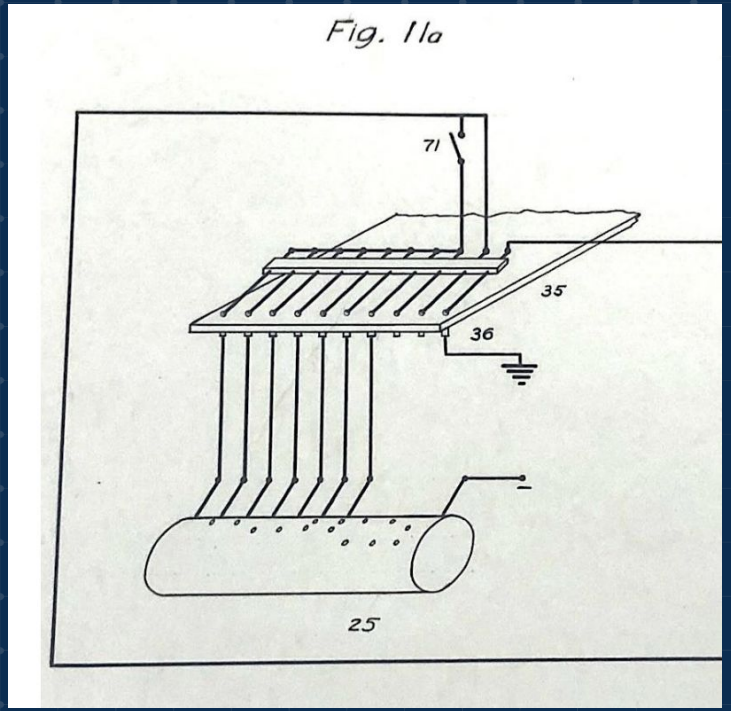
Decimal to Binary Conversion



Decimal to Binary Conversion



https://jva.cs.iastate.edu/reconstructionoperation.php



From *The First Electronic Computer* by Burks and Burks

Decimal to Binary Conversion

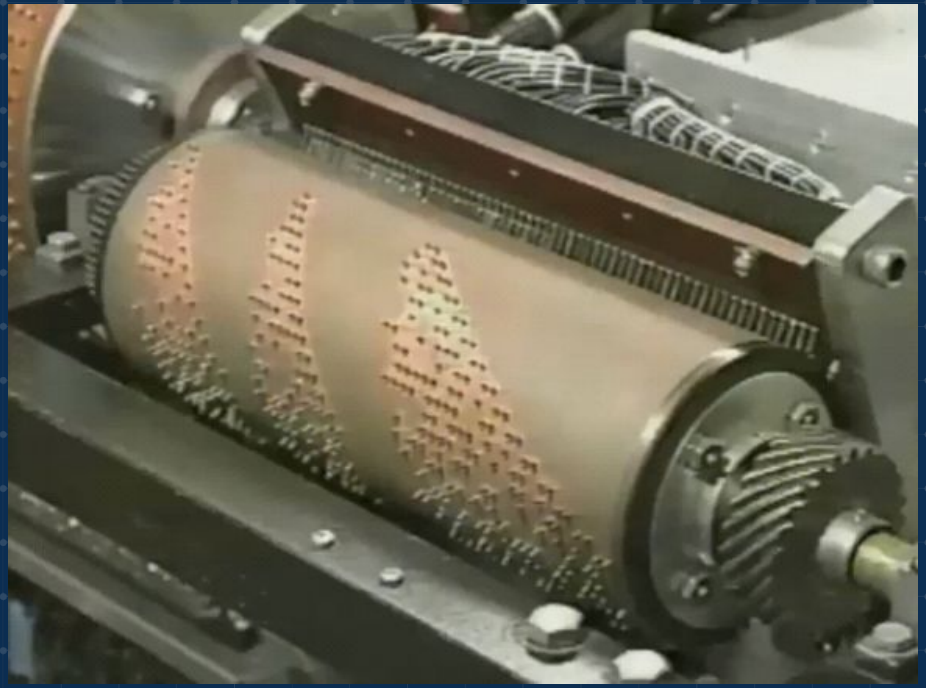
TABLE 3. Pattern of Decimal-Binary Conversion Drum

	Hundreds ($\times 10^2$)									Tens ($\times 10^1$)									Units ($\times 10^0$)								
	9	8	7	6	5	4	3	2	1	9	8	7	6	5	4	3	2	1	9	8	7	6	5	4	3	2	1	
The array continues to the left for twelve more decimal digits.																												2^{48} • • • • • 2^{10}
	●	●	●	●																								2^9
	●	●			●	●	●																					2^8
	●		●		●	●		●																				2^7
				●	●			●	●	●	●	●																2^6
		●	●		●		●		●				●	●	●													2^5
			●	●	●	●				●	●		●	●		●	●											2^4
			●	●			●	●		●			●		●	●		●	●	●								2^3
	●		●		●		●		●			●	●			●	●				●	●	●	●				2^2
										●		●		●		●		●			●	●			●	●		2^1
																			●		●		●		●		●	2^0

A large dot indicates the presence of a stud, for a "1."
A blank cell indicates the absence of a stud, for a "0."
Compare table 2 and its representation of 875

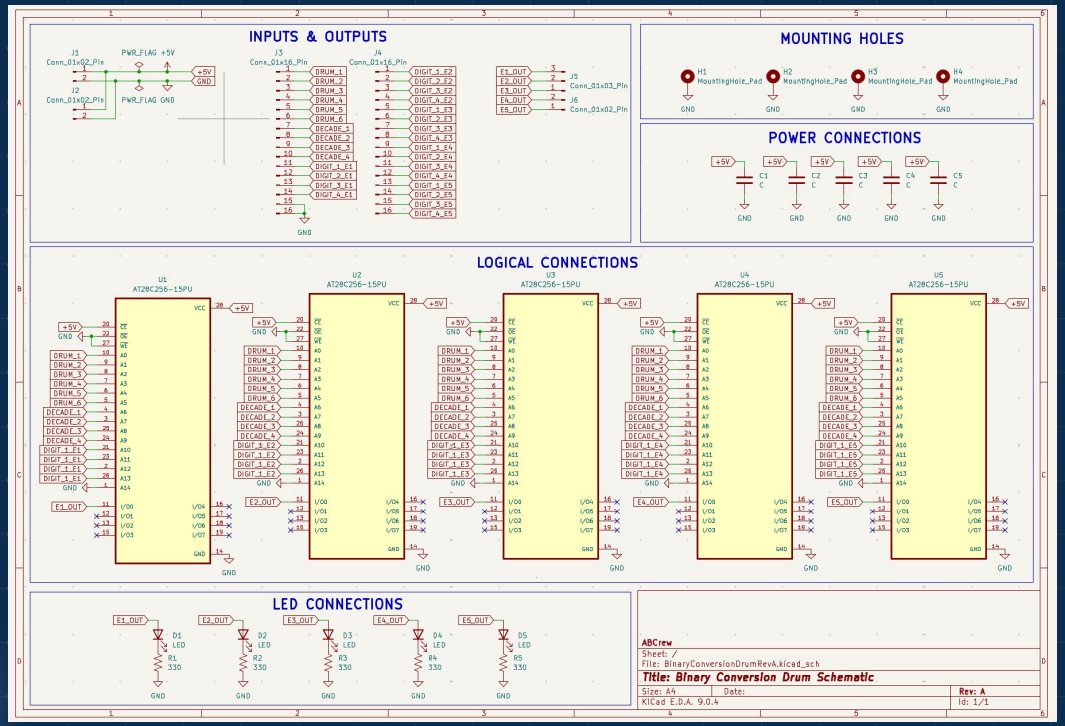
From *The First Electronic Computer* by Burks and Burks

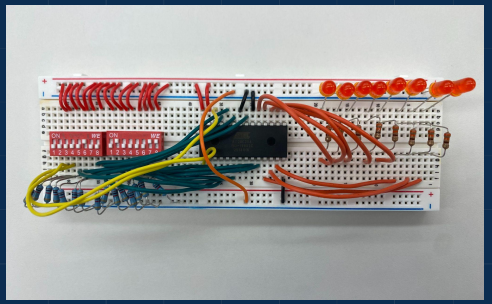
Decimal to Binary Conversion

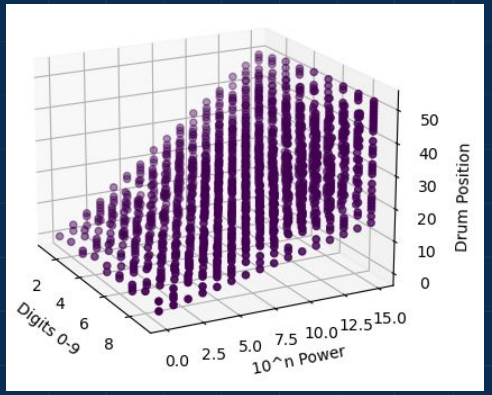


https://jva.cs.iastate.edu/reconstructionoperation.php

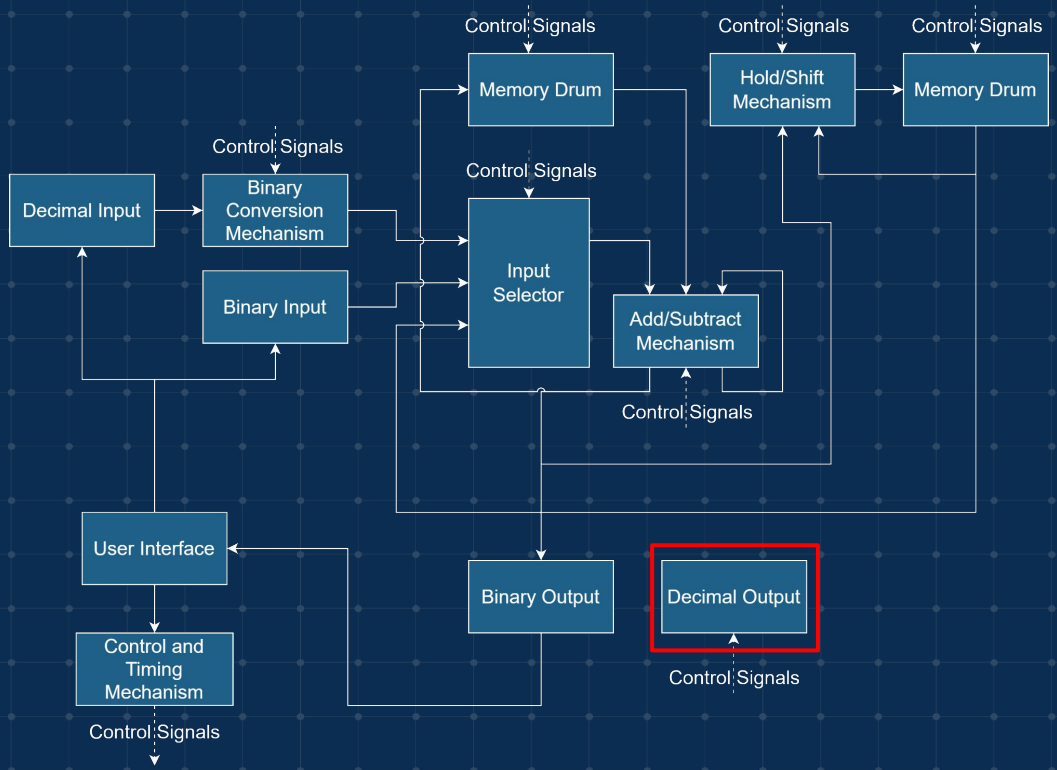
Decimal to Binary Conversion







Decimal Output



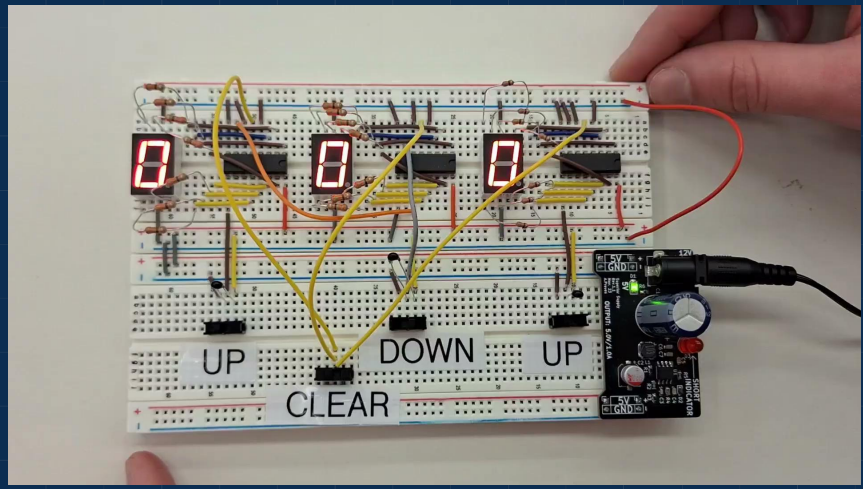
Decimal Output



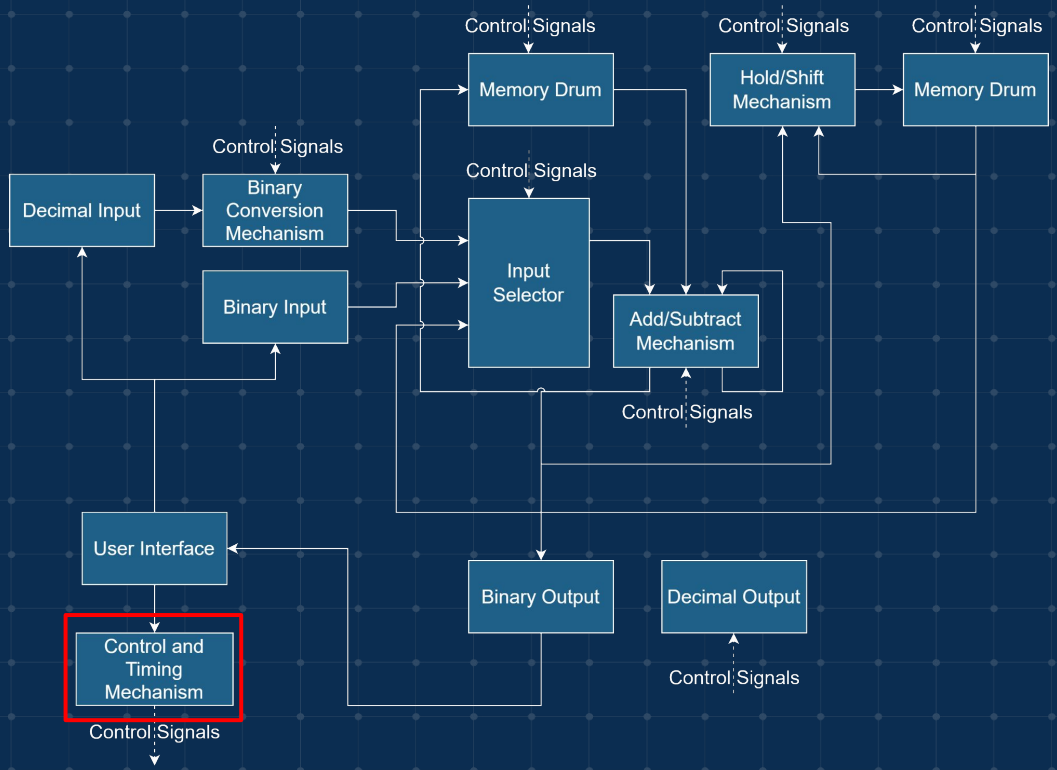
https://jva.cs.iastate.edu/reconstructionoperation.php

Decimal Output

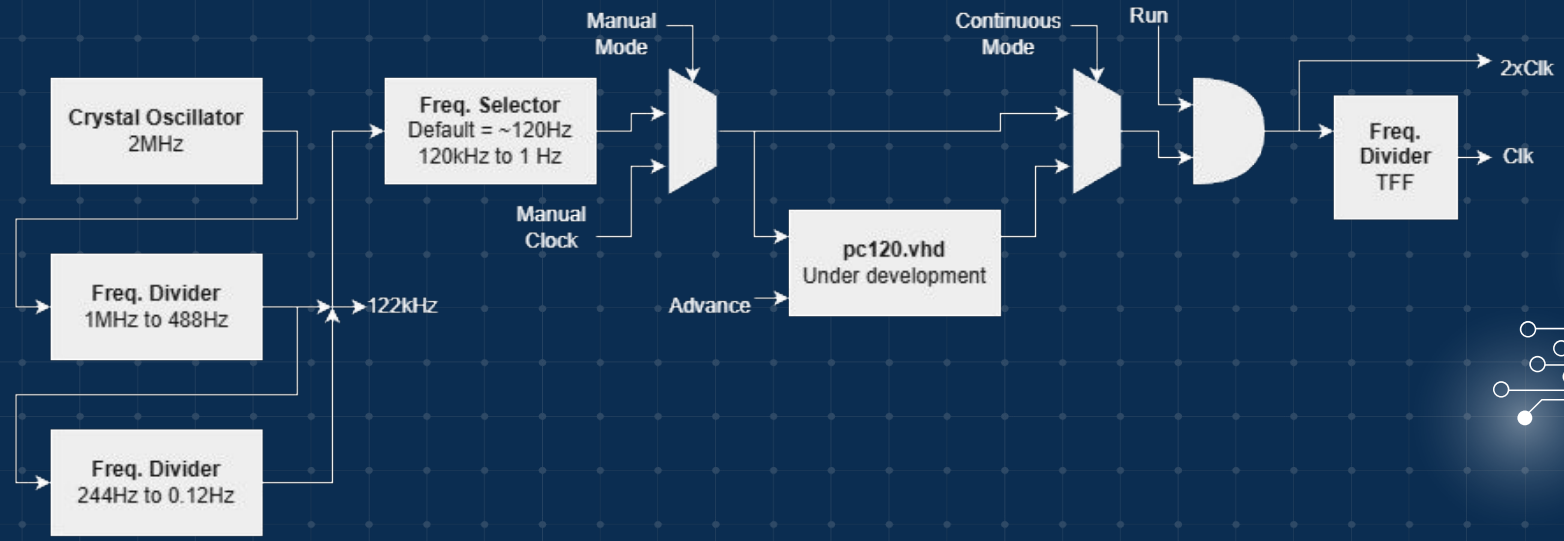
Stored Value	
82	
	Subtract 100, advance a dial
-18	
	Sign change, switch to the next dial
	Add 10, advance a dial
-8	
	Add 10, advance a dial
2	
	Sign change, switch to the next dial
	Subtract 1, advance a dial
1	
	Subtract 1, advance a dial
0	
	Zero, terminate

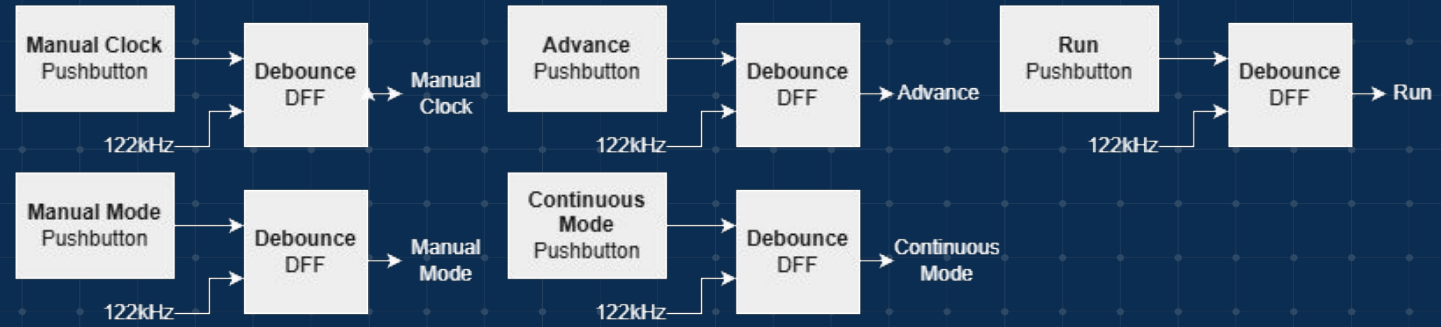


Clock Circuit and Program Counter



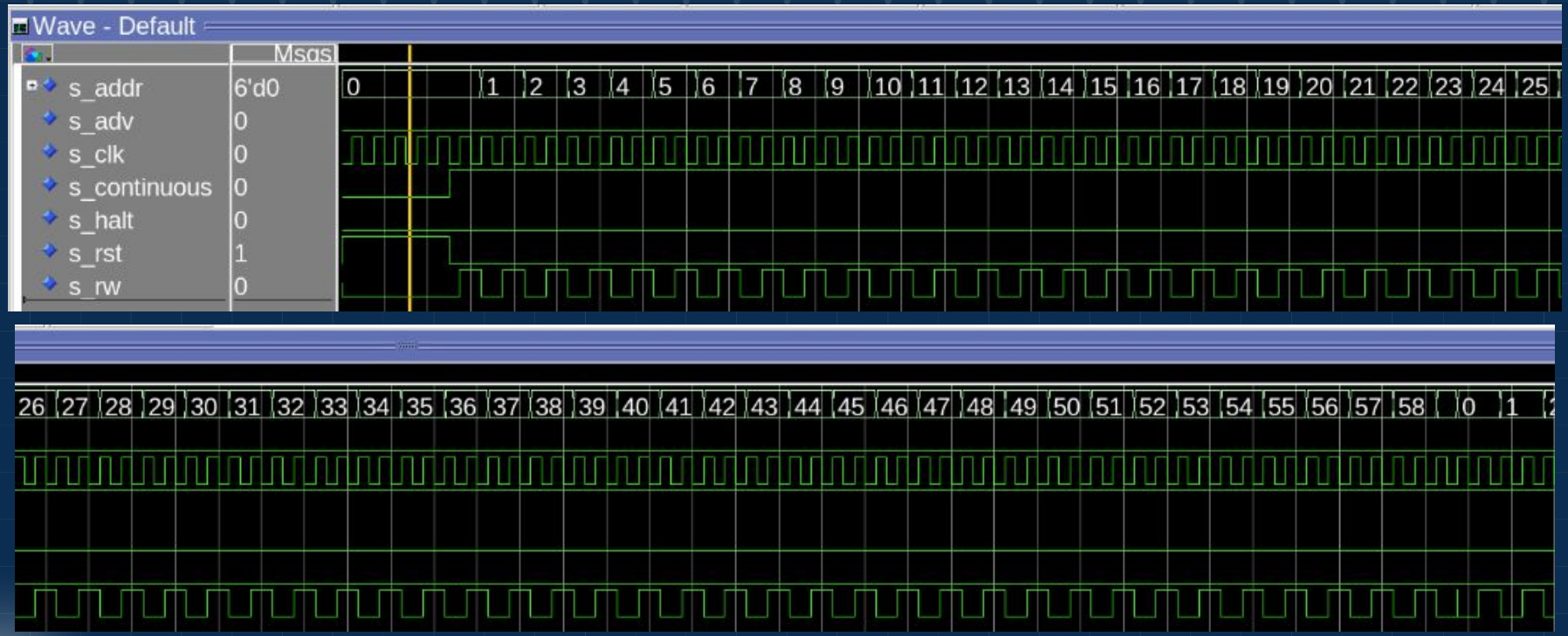
Not Finalized - Under Active Development





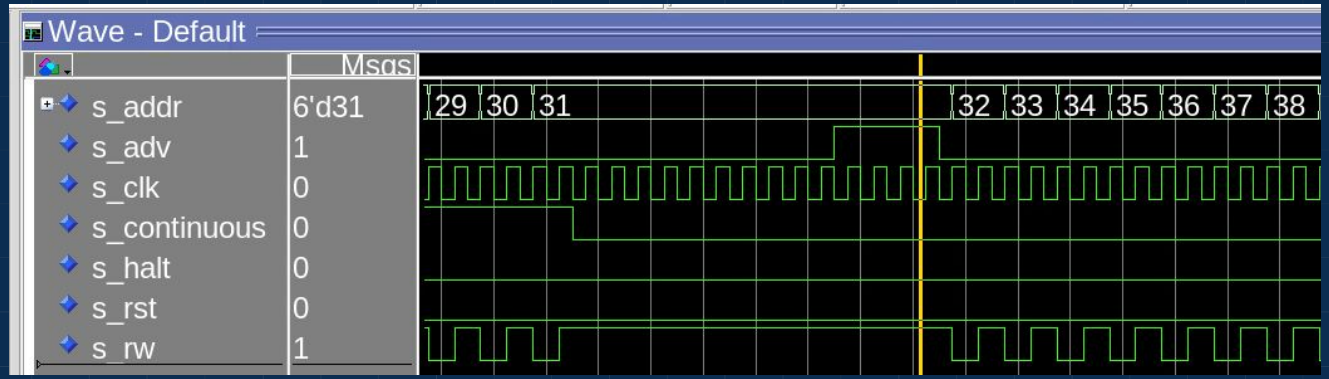
Continuous Functionality

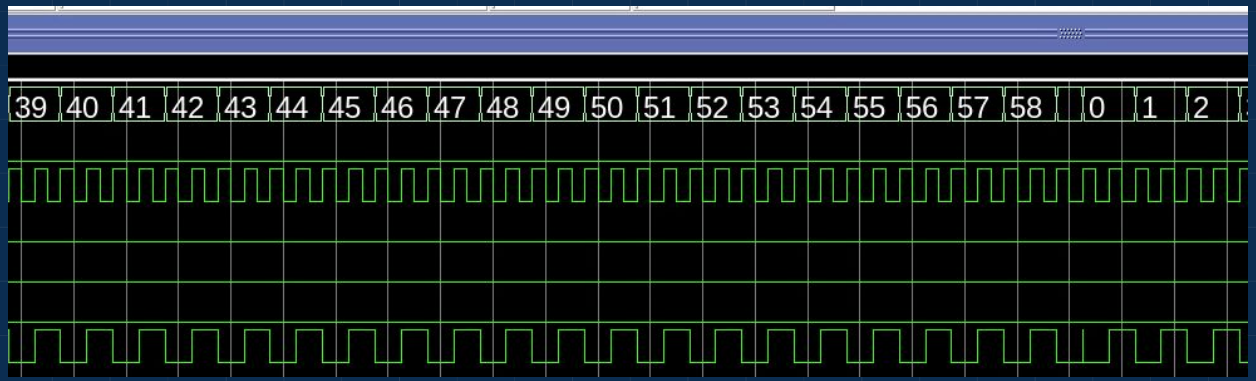
Under Development



Advance Functionality

Under Development

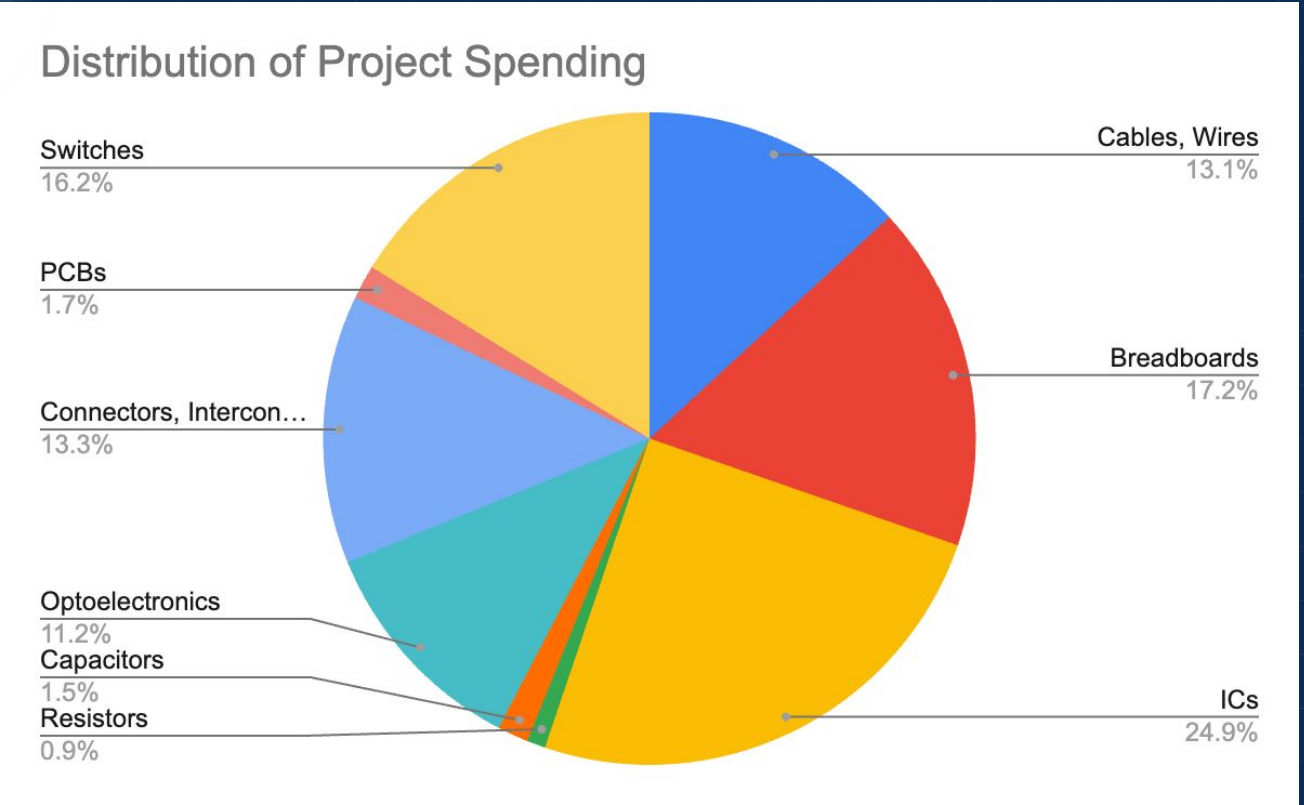




Logistics & Next Steps

Budget Overview

To-Date Spending:
~$600



Next Steps & Takeaways

Short-Term

Finish testing & PCB design for in-progress modules

(clock, decimal output, & decimal conversion)

Long-Term

Design and integrate the remaining modules

Document our designs in a technical manual

Develop full system in VHDL

Lessons Learned

Ordering parts early and accurately to avoid implementation delays

Recreating the complex, sparsely-documented design of the ABC is time consuming

Future Considerations

Identify a realistic scope for our end-of-semester deliverables

Establish thorough & efficient testing methods for gradual integration of all modules

Thank you for your time!

Questions?