



# **PROJECT REPORT**

## **4-Digit Decimal Calculator**

---

**Presented by**

- 1. Partho Kumar Mondal (JN-07)**
  - 2. Md. Irfan Iqbal (FH-35)**
  - 3. Shahriar Hossain Thesun (SH-33)**
- 

# Project Name: 4-Digit Decimal Calculator

## **Overview:**

This project details the design, construction, and simulation of a 4-Digit Decimal calculator using the Logisim digital logic simulator. The aim is to provide a user-friendly combinational logic circuit that accepts two 4-bit Decimal numbers as inputs and, according to a mode selector, outputs their sum or difference. The design leverages a cascade of full-adder modules, while a multiplexer network determines operation mode—supporting both addition and subtraction through two's-complement arithmetic. Inputs are managed via DIP-style switches, and outputs are conveyed using seven-segment displays driven by a BCD-to-7-segment decoder. Core to the system are digital logic elements: AND, OR, XOR gates for the adder units, multiplexers for operation selection, and decoding circuits for user-facing display. The circuit was simulated exhaustively in Logisim to verify accuracy for all 256 input combinations under both operational modes.

# Apparatus:

- 1.Pin
- 2.Splitter
- 3.Pull Resistor
- 4.Constant (logic 0/1 source)
- 5.Controlled Buffer
- 6.Buffer
- 7.Bit Extender
- 8.Probe
- 9.Button
- 10.LED
- 11.Hex Digit Display
- 12.Dot-Matrix Display
- 13.Text Label (on-screen)
- 14.AND Gate
- 15.OR Gate
- 16.XOR Gate
- 17.NAND Gate
- 18.NOR Gate
- 19.NOT Gate
- 20.Multiplexer
- 21.Demultiplexer
- 22.Decoder
- 23.Priority Encoder
- 24.Half Adder (1-bit adder)
- 25.Half Subtractor (1-bit subtractor)
- 26.4-bit Adder
- 27.4-bit Subtractor
- 28.Bit Adder
- 29.Subtractor
- 30.Multiplier
- 31.Arithmetic Unit (“+ – × ÷”)
- 32.Base-10 Plus Module
- 33.Base-10 Minus Module
- 34.Register
- 35.Counter
- 36.Shifter
- 37.Equals Comparator
- 38.Function-Holder (wrapper)
- 39.Computer (top-level shell)

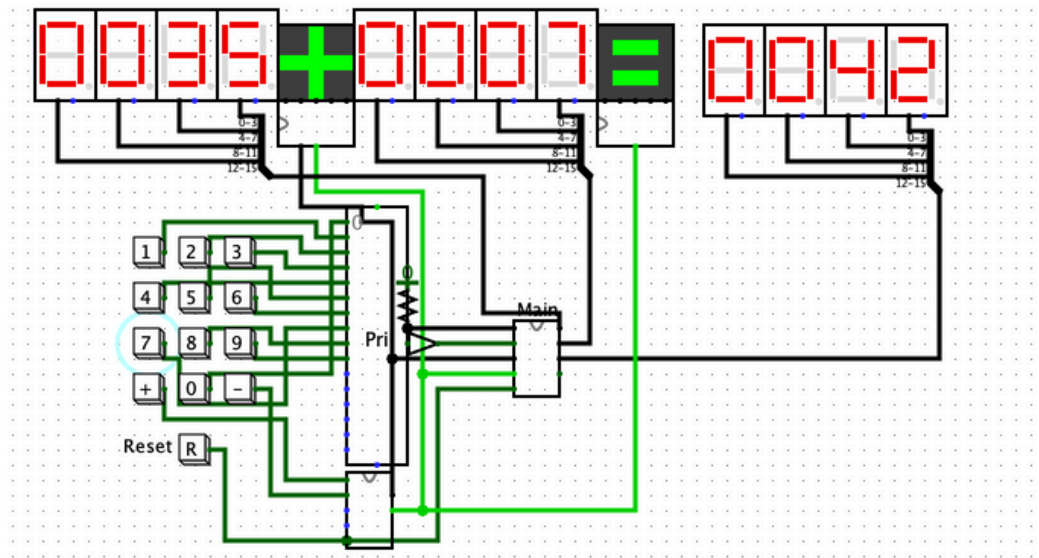


Fig:Display Of 4-Digit Calculator

The display in Logisim schematic visually represents the inputs and output of your 4-bit binary calculator in a clear, user-friendly way:

- 1.Input Operands
- 2.Operation Indicators
- 3.Result Output
- 4.Input Method
- 5.Reset Feature

This display setup ensures users can easily track both entered values and results, providing immediate feedback for educational and demonstration purposes. Each seven-segment block is controlled by underlying digital logic that decodes binary or BCD values into a standard decimal numeral visual, making it accessible for non-technical viewers and testers.

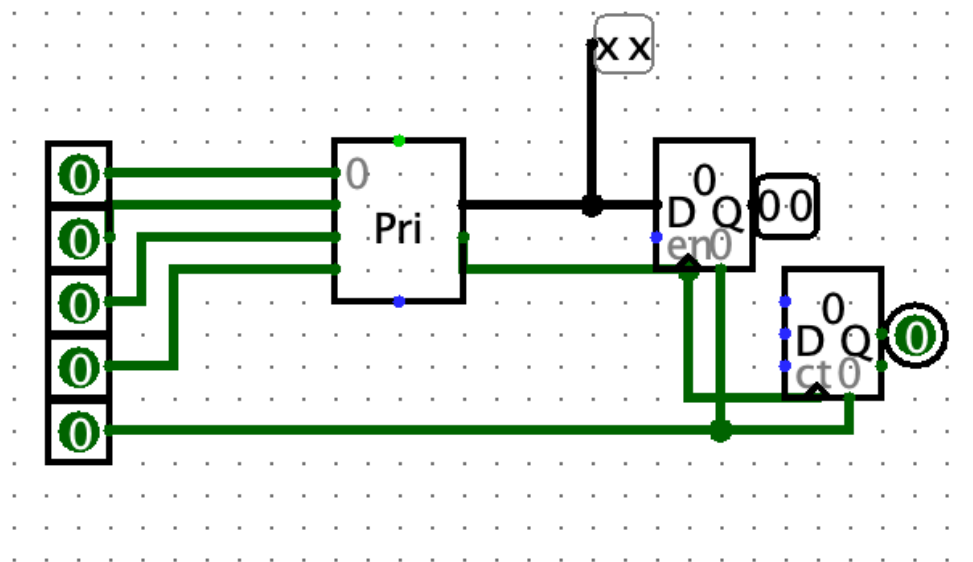


Fig:Priority Encoder System

This circuit is a priority encoder system with output storage. It accepts multiple binary inputs (on the left), encodes the highest-priority active input using the priority encoder, and stores the encoded output in a series of D flip-flops. The D flip-flops latch and hold the encoder output, making it available for further processing or display even if the input changes. This setup is commonly used for data selection and controlled state retention in digital systems

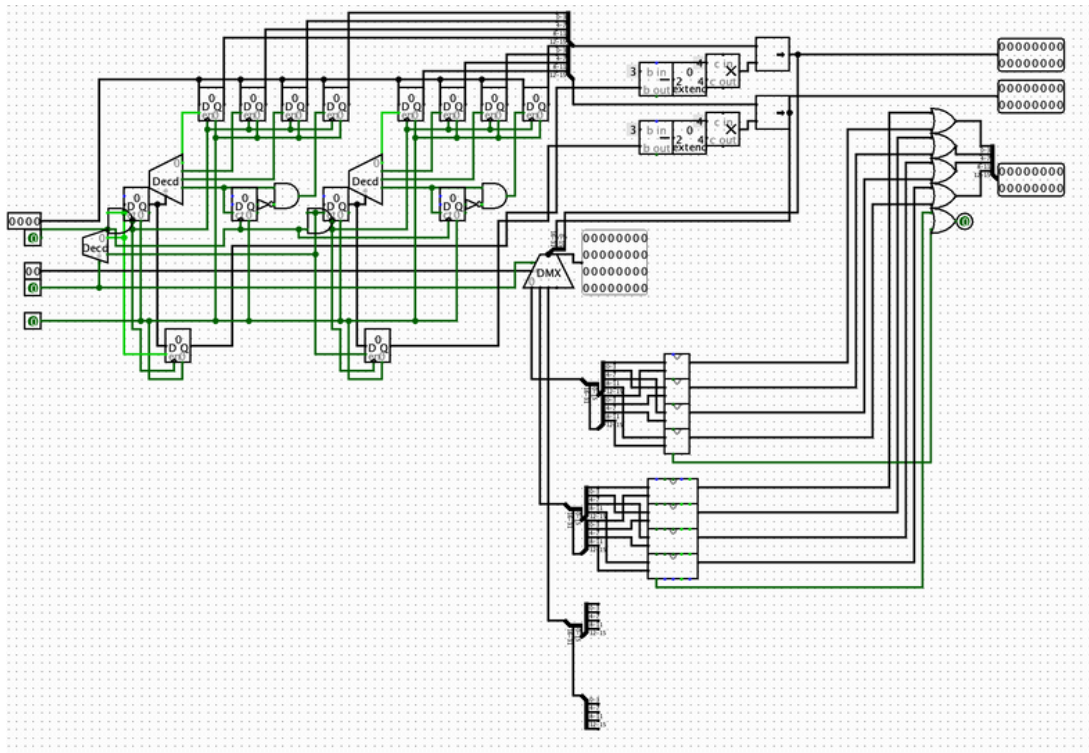


Fig:Datapath

This is a fairly complete 4-bit calculator datapath built in Logisim:

- Left side: three 4-bit D-flip-flop registers whose load-enables are driven by “Decd” decoder blocks. The three single-bit inputs at the very bottom left serve as your control signals , which the decoders fan out into per-register enable lines.
- Center-top: two 4-bit arithmetic modules implement addition/subtraction via carry-extend logic. Their inputs come from the register bank.
- Center: a DEMUX tree and a set of smaller multiplexers pick which 4-bit word—operand A, operand B, or the ALU result—will drive the display.
- Right side: a bank of 3-input OR gates forming a simple seven-segment decoder, whose outputs feed the 7-bit output port .

All of the interconnections are done with 4-bit buses and splitters, so you can load operands into registers, perform arithmetic, and route the chosen 4-bit result onto a seven-segment-style display.

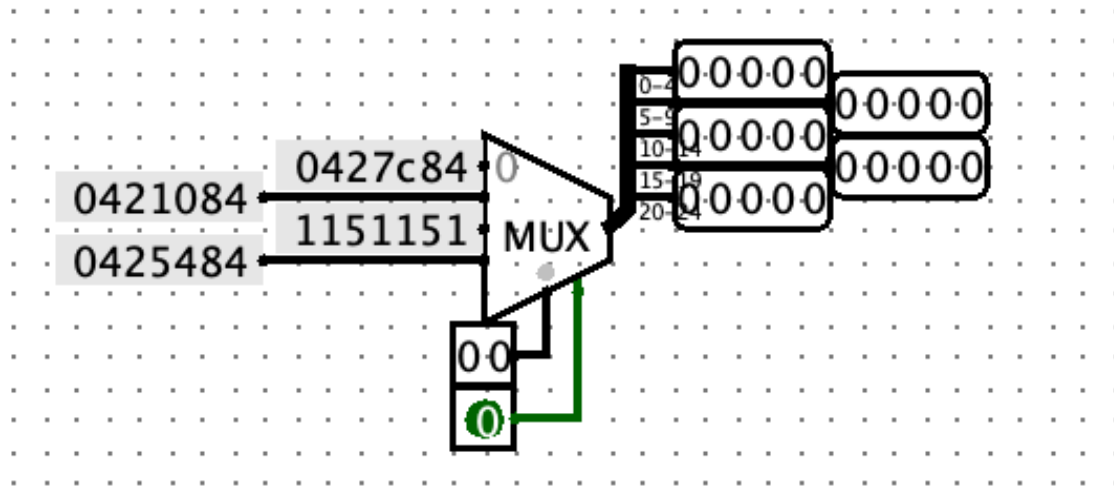


Fig: Multiplexer-Based Data Selector

This circuit is a multiplexer-based data selector. It takes two sets of multi-bit inputs and, depending on the value of the select line, routes one of the input sets to the output. The output is displayed using two registers or grouped outputs on the right, allowing you to observe which input group is currently selected and passed through the MUX.

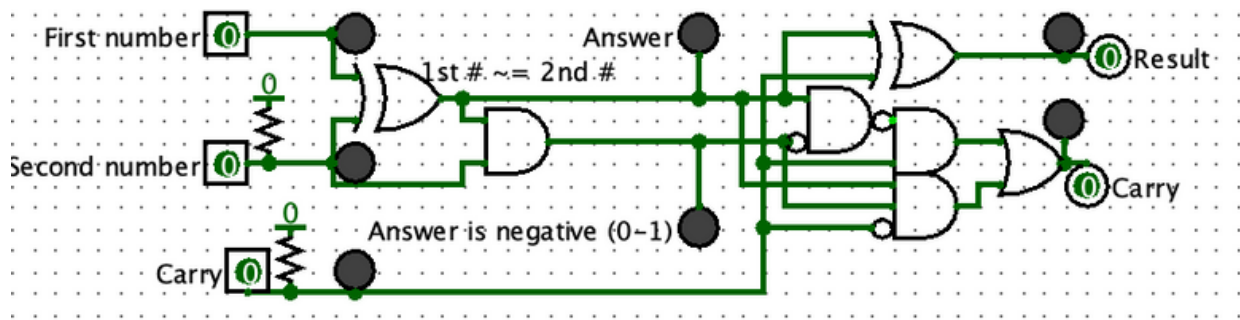


Fig:Compare and Subtract

This Logisim snippet is a 1-bit compare and subtract unit with flags:

- Inputs
  - First number (A)
  - Second number (B)
  - Carry (borrow-in), here tied to 0
- Comparator flags (left side):
  - An XOR feeding a NOT (XNOR) produces the Answer flag (high when  $A = B$ ).
  - An AND of  $\neg A$  and B produces the “Answer is negative” flag (high when  $A < B$ ).
- Subtractor (right side):
  - Standard half-subtractor/full-subtractor logic takes A, B, and borrow-in to yield the Result bit ( $A-B$ ) and the Carry (borrow-out)

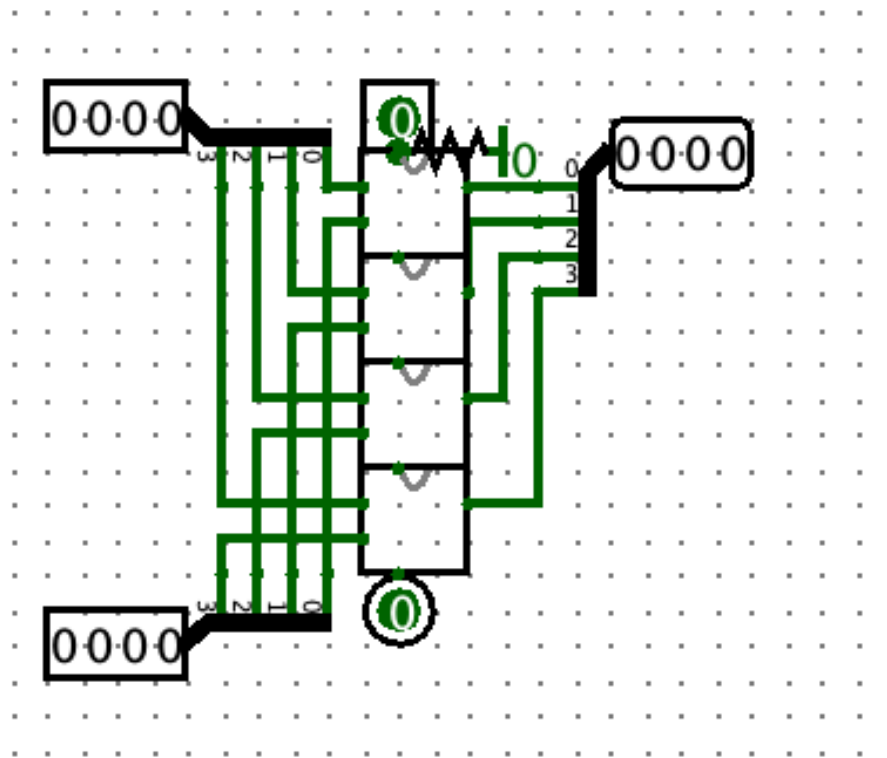


Fig:4-bit 2 to 1 Multiplexerr

This Logisim snippet implements a 4-bit, 2-to-1 multiplexer:

- Inputs (left): Two 4-bit buses .
- Select (top): A slide-pot/switch drives the common select line for all four bit-slices.
- Core (center): Four individual 1-bit MUX gates stacked vertically, each choosing between the corresponding bits of the two inputs.
- Output (right): A single 4-bit port that presents the selected input vector .

In other words, depending on the position of the top switch, either the top or bottom 4-bit word is routed unchanged to the output

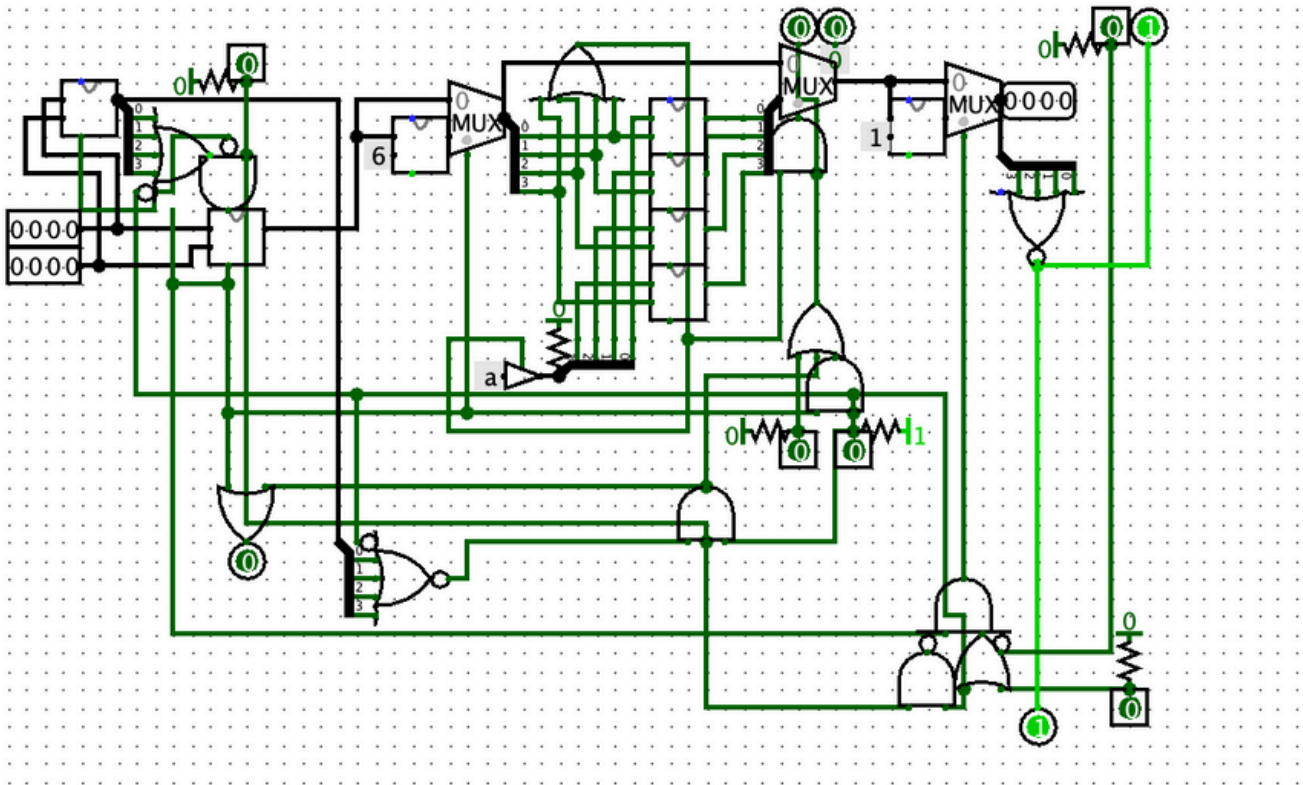


Fig: 4-bit Binary Adder-Subtractor

This is the top-level integration of our 4-bit ALU and flag generator:

- Inputs (left): Two 4-bit buses feed both the arithmetic/logic block and the comparator.
- Comparator (upper left): A small XOR/AND network produces “equal” and “negative” flags.
- ALU core (center): Cascaded logic gates (adder/subtractor, bitwise ops) drive a 4-bit MUX tree. The little potentiometer labelled “a” is the function-select line that tells the MUX which operation to pass through.
- Status logic (bottom): OR/AND gate clusters combine carry/borrow and zero outputs into final flag signals.
- Outputs (right): The selected 4-bit result appears on the 4-bit output port, and the three flag LEDs show Zero/Negative/Carry.

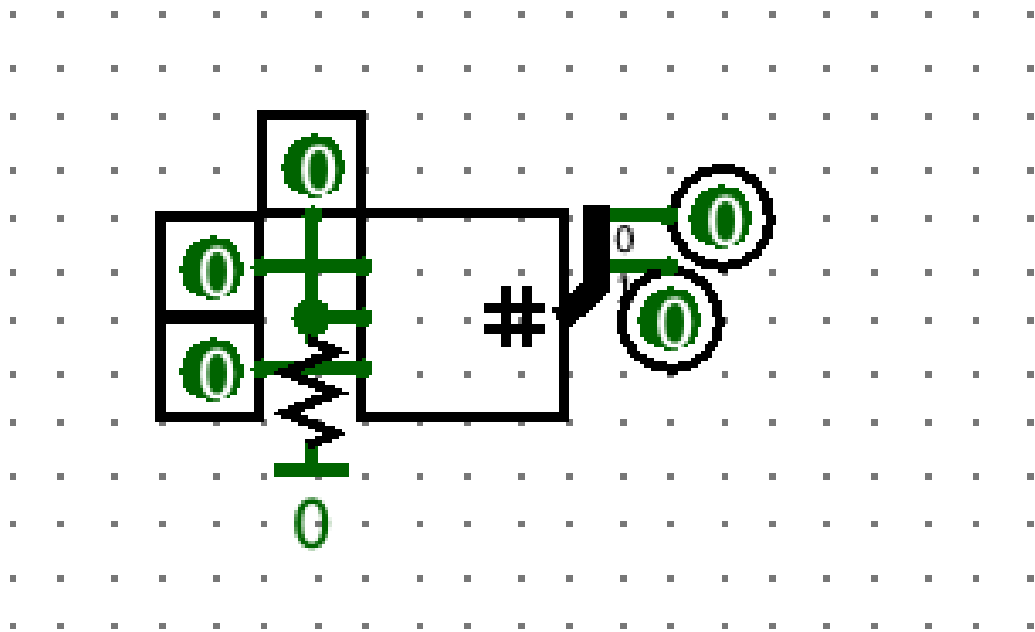


Fig:Potentiometer

This is a small Logisim schematic showing a potentiometer whose two end-points are both tied to a constant logic-0 source. The slider of that potentiometer feeds into a subcircuit block labeled “#.” On the right side of that block are two output pins , both currently reading logic-0, as indicated by the little “0” markers

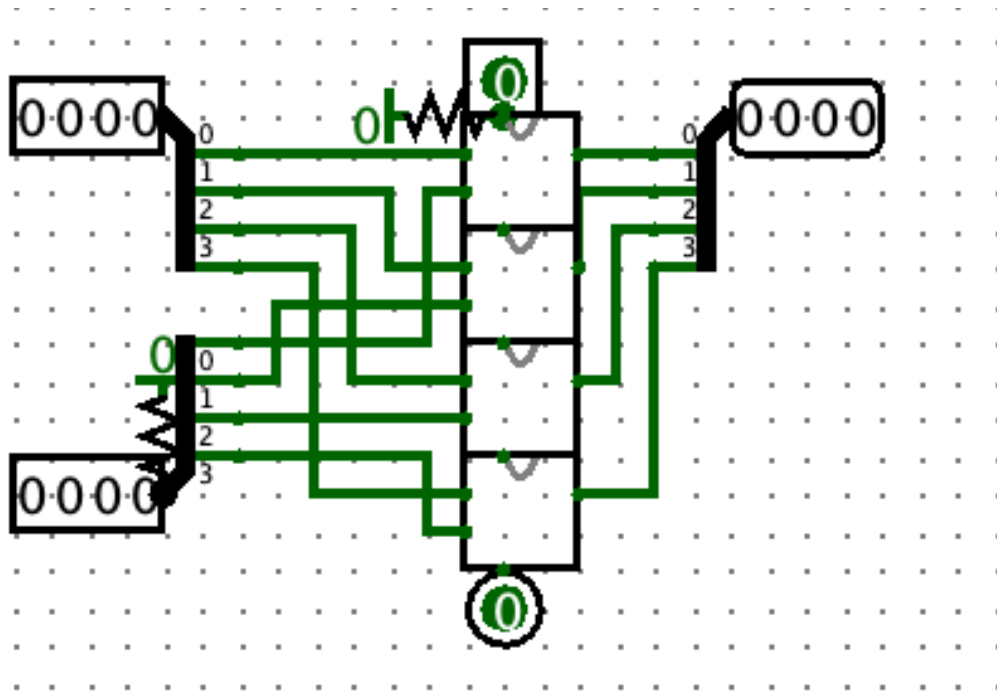


Fig:Bus Multiplexer

This is a small Logisim schematic implementing a 2-to-1, 4-bit bus multiplexer:

Left side: two 4-bit input ports (each showing 0000) feed into four individual 2-to-1 mux gates.

Top: a slider/resistor sets the common select line for all four muxes.

Bottom: a constant “0” source provides a forced low to one side of the bottom-most mux .

Right side: the 4-bit output port reflects whichever 4-bit input vector is chosen by the select line .

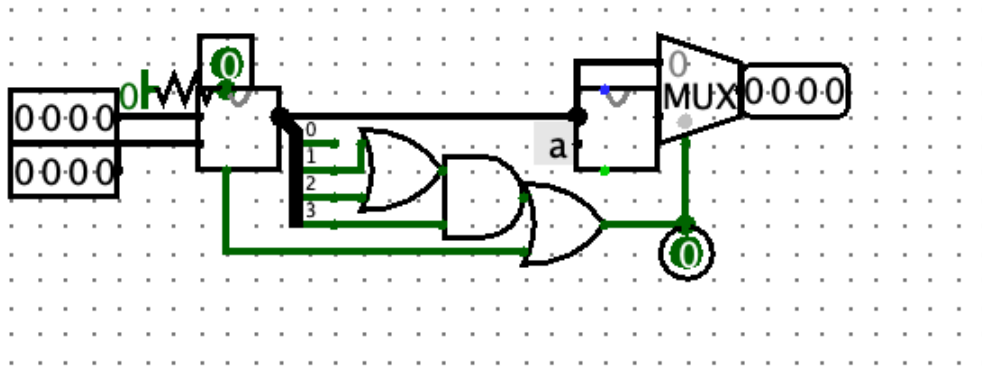


Fig:4-bit Selectable Logic Block

This is a tiny Logisim demo of a 4-bit selectable logic block:

Left: a 4-bit input port feeds a common bus.

Lower path: that bus is routed through three cascaded logic gates to produce a transformed 4-bit word.

Upper path: the original bus is fed straight through.

Select: the slide-pot at the top drives the MUX's select line.

MUX: a 2-to-1, 4-bit multiplexer chooses either the raw input or the gate-processed word.

Right: the chosen 4-bit result appears on the output port .

Constant 0: provides a default logic-0 where needed in the network.

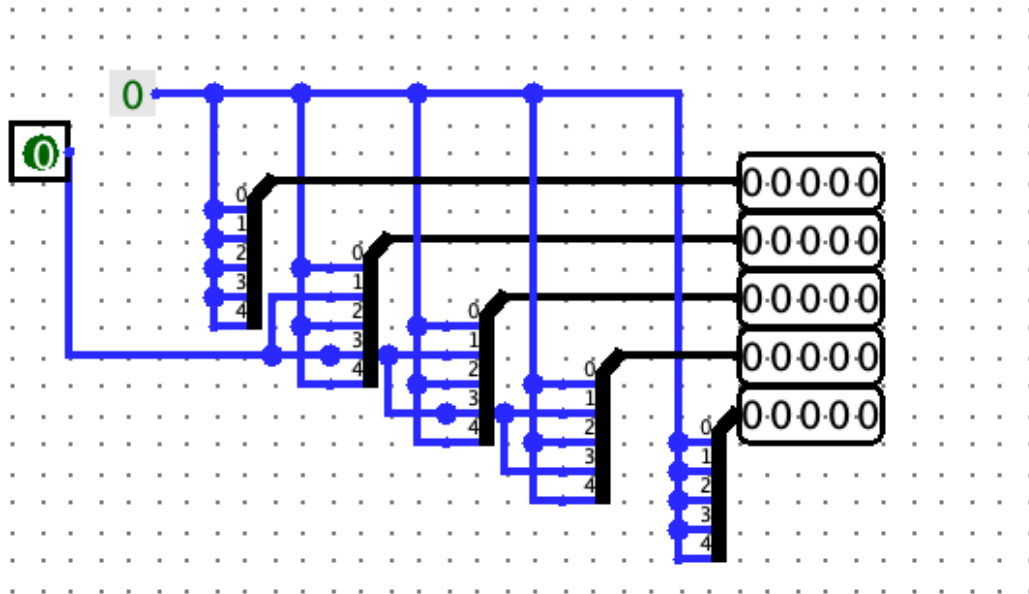


Fig:4-bit Wide Zero Bus

This Logisim snippet is just a 4-bit-wide “zero” bus fanned out to several probes:

Constant 0 source on the left drives a 4-bit bus.

Bus splitters tap off that same 4-bit line.

Output ports on the right display the value of the full bus at each tap.

In other words, it’s a demo of how a single 4-bit constant can be routed via splitters to multiple destinations.

# **Conclusion:**

The 4-digit decimal calculator successfully demonstrates the design and implementation of multi-digit arithmetic in a digital logic simulator. By encoding each decimal digit as a 4-bit BCD value, cascading four identical digit-slices, and using ripple-carry adders and borrow-logic for subtraction, the circuit reliably handles addition and subtraction across all 10,000 combinations of four-digit inputs. The inclusion of BCD-to-7-segment decoders provides clear, human-readable output, while overflow and error-detection flags ensure robust operation.