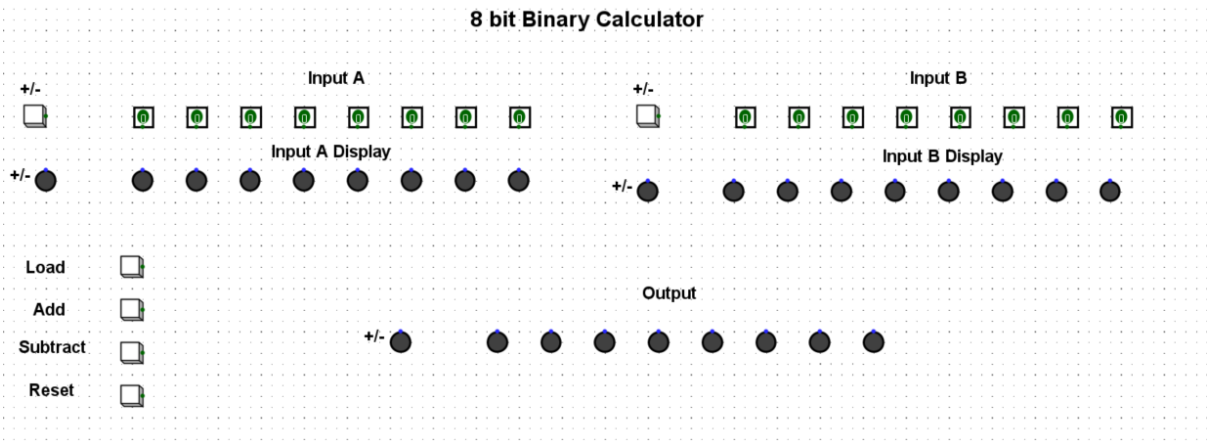
COS10004 Computer Systems Assignment 1

An 8-Bit Binary Calculator for Addition and Subtraction in Logisim



Objective: The objective of this project is to design and implement an 8-bit binary calculator using digital logic circuits in Logisim Evolution v3.8.0. The calculator will perform binary addition and subtraction operations. The solution will be developed in four stages to incrementally build the functionality.

The file calculator.circ provides all the user interface components your assignment will need (shown above). *Your assignment must be built using this file as the interface*. You are encouraged to use multiple circuits to implement specific components however your whole circuit must be operable and displayed using the buttons and display components in this file. That is, your solution should not require any additional interface components (e.g., buttons, displays, pins etc) in order to test the functionality of your circuit. You may adjust the position of components in this file if you need to, but please try to avoid large changes, and keep all UI components together.

Using Sub-circuits

You are highly encouraged to use sub-circuits to build up your solution. For those unfamiliar, take a look at this short video on how to use them effectively: https://youtu.be/BZiut4OUdXo

Logisim Version

Your assignment must be implemented using Logisim Evolution 3.8.0, which can be downloaded from: https://github.com/logisim-evolution/logisim-evolution/releases

This is the version we will test with, and we will not be using any other version, or making special accommodations. If your solution is incompatible with ours, it will not be able to be tested and thus will be ineligible for most marks on offer.

Allowable Logisim Components

Only the following components may be used to develop your solution:

- Logic Gates: AND, OR, XOR, NOR, NOT, NAND, XNOR
- Flip Flops: JK, D, S-R, T
- LEDs

- Clock (only one)
- Hex Digit Display
- Buttons
- Pins (as inputs and for connecting sub-circuits)
- Constants (for setting inputs that will not change)
- Splitter (for using HEX Digit Display)

The use of any other components will be penalised – in particular, you must not use any pre-built circuits such as registers, shift registers, etc).

Assignment Outline:

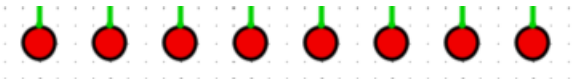
Stage 1: Load and Reset Buttons

Input entry for your calculator will be done using the 8 pins provided for each input (Input A and Input B) in the provided interface. However, these values need to be loaded into registers before calculations can be performed on them. To this end, you are going to implement this functionality, such that when the "Load" button is pressed, the values from the Input A and Input B pins will be loaded into registers.

Specifically, Stage 1 needs you to:

- implement the "Load" button such that when it is pressed, the values represented by the Input A and Input B pins are fed into a dedicated register for each (i.e. a bank of Flip Flops).

- Implement the "Reset" button such that when it is pressed, all registers are set to 0
- Ensure the LED display for each of Input A and Input B reflects the value stored in each register.



- Your solution should include the use of Flip Flops to store both the input values
- Save your solution as "stage1.circ".

Stage 2: Implement the "Add" Button

You are now going to build on your solution to Stage 1, and implement the Addition operation.

Specifically, you are required to:

- Implement the "Add" button by designing a full 8-bit adder circuit that takes as inputs, the values stored in Input A and Input B registers from Stage 1

- Ensures the output of the addition calculation is displayed using the “Output” LED display.
- Save your working solution of all stages up to Stage 2 in "stage2.circ".

Stage 3: Signed Integers Using 2’s Complement

Most calculators allow positive or negative numbers to be entered. You’re going to implement this by implementing a “+/-“ button for each input binary number (i.e. a separate “+/-“ button for each), allowing the user to specify if the value is negative or positive. You will be using 2’s Complement representation to handle signed integers, and so be sure to first watch **Lecture 4.2 (Week 4 Module) for more details on 2’s Complement.**

Building on Stage 2’s solution:

- Implement the “+/-“ button for each input binary value, such that when it is pressed, it toggles between the number being positive (by default) or negative. Use a labelled LED for each input binary value to indicate when the input is positive (LED off) or negative (LED on).
- When the “+/-“ button is pressed, the current value of the corresponding binary input value should be converted to it’s 2’s Complement representation. That is, you should implement a circuit that performs 2’s Complement conversion on an 8 bit input binary string whenever the “+/-“ button is pressed.
- Ensure the 8 LEDs displaying each binary input value show the correctly converted version.
- Ensure the addition calculation (when the “Add” button is pressed) operates on the correctly converted version of the inputs.
- Note that as we are using 2’s Complement, the value range of your calculator will be between -128 and +127.
- If the output result is a negative number, ensure the +/- LED is on (otherwise off)
- In your video, explain how the "+/-" button is implemented in your circuit
- Save your working solution of all stages up to Stage 3 in "stage3.circ".

Stage 4: Implement Subtraction Using 2's Complement

Building on Stage 3’s solution:

- Implement the "subtract" button for subtraction operations on the two binary input numbers.
- To implement subtraction, you should utilise your “2’s Complement” conversion circuit from Stage 3 to convert the number to be subtracted. That is, if the calculation to be performed is:

  01001100 (76) – 00110011 (51),

  then “00110011” should first be converted to its 2’s Complement representation: “11001101” (-51), and then added to 01001100 (76), yielding the result “00011001” (25).
- Combine the 8-bit adder circuit with the subtraction functionality.
- Detail the process of negating numbers using 2's complement and performing subtraction in your video.

- Include a labelled LED with the 8 output LEDs that is turned on when the result of the calculation is negative, and off is the result is positive.
- Save your working solution of all stages up to Stage 4 in "stage4.circ".

Final Submission:

- Include four separate circuit files, each representing one stage of development. Each stage file should include all the functionality required to test that stage, in Logisim Evolution v3.8.0
- Prepare a 5-minute video (no more than 5 minutes!) demonstrating each stage of the calculator's evolution. In the video, outline:
    - the key features of each design stage
    - the rationale behind your design decisions, emphasising your understanding of the implemented solution.
    - a critical reflection on your design and its functionality (you are encouraged to point out its flaws and any errors in its behaviour).

Assessment Criteria:

- Successful completion of each stage according to the specified requirements (**60% of total mark**)
    - Completion of Stage 1: (up to 15 of the 60 marks available)
    - Completion of Stage 2: (up to 30 of the 60 marks available)
    - Completion of Stage 3: (up to 50 of the 60 marks available)
    - Completion of Stage 4: (up to 60 of the 60 marks available)
- Quality of the solution (**20% of total mark**)
    - Clarity and modularity of design
    - Efficient layout and use of components
    - Readability: use of labels and easy to find/use UI components.
    - Innovation/elegance of solution
- 5-minute video demonstration and reflection (**20% of total mark**)
    - Quality and clarity of the video
    - depth of understanding and critical reflection on the design and implementation

    Note that your video should be well planned and structured. It should make succinct, well thought out points about your design.

Submission

Your completed submission must be made through Canvas - (Go to Assignment 1 under "Assignments" before the due date/time).

Everyday day late will incur a *10% deduction*.

Each *submission must be zip file containing*:

- the actual Logisim files (.circ source files) for testing. Each file MUST be labelled stageX.circ, where X is the stage completed.
- The video file or a file providing a link the file online. Please make sure of you are providing your video as a link, *that the file is not made publicly accessible (i.e.., it should only be accessible to those with the link)*

Academic Integrity

This is an individual assessment task and it is required that you work alone on your solution for assignment. This means you *must not share your solution with any other student*, *or make any part of your solution publicly available available online*. Markers will be cross-checking work, and will expect to see progress being made on assignments during the dedicated lab classes.

Any breaches of academic integrity will immediately attract a mark of 0 for the assignment, and probable further disciplinary actions in accordance with Swinburne's Academic Integrity policies and procedures.