DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY

CHATTOGRAM – 4349, BANGLADESH

Experiment No. 1

Schematic Modeling and Implementation of Modified SAP Architecture in Logisim – 1

PRECAUTIONS:

- Students must carefully read the lab manual before coming to lab to avoid any inconveniences.
- Students must carry a flash drive to transfer files and lab manuals.
- Use of mobile phone in lab is strictly prohibited and punishable offence.
- Experiment files must be uploaded to GitHub including home tasks.

OBJECTIVES:

- To familiarize with various parts of the SAP-1 architecture.
- To familiarize with IC requirements and logic design.
- To familiarize with the concept of abstraction.

THEORY:

The SAP-1 computer is a bus-organized computer and makes use of Von-Neumann architecture. It makes use of an 8-bit central bus and has ten main components. A pictorial representation of its architecture is shown below. Each of the individual components that make up this computer are described right after. [1]

SAP-1 Components:

- **Program Counter:** The program counter's job is to store and send out the memory address of the next instruction to be fetched and executed. The program counter, which is part of the control unit, counts from 0000 to 1111 as the program is stored at the beginning of the memory with the first instruction at binary address 0000, the second instruction at address 0001, the third at address 0010, and so on. At the start of each computer run, the program counter is reset to 0000. When the computer run starts, the program counter sends out the address 0000 to the memory and is then incremented by 1. After the first instruction is fetched and executed, the program counter sends the next address 0001 to the memory and again, after that, the program counter is incremented. In this way, the program counter keeps track of the next instruction to be fetched and executed.

- **Input and Memory Address Register (MAR):** The MAR stores the 4-bit address of data or instruction which are placed in memory. When the SAP-1 is running, the 4-bit address is taken from the Program Counter through the W-bus and then stored. This stored address is sent to the RAM where data or instructions are read from.

- **Random-Access Memory (RAM)**: The SAP-1 makes use of a 16 x 8 RAM (16 memory locations each storing 8 bits of data). The RAM can be programmed by means of the address and data switches allowing you to write to the memory before a computer run.

During a computer run, the RAM receives its 4-bit address from the MAR and read operation is performed. In this way the instruction or data word stored in the RAM is placed on the W bus for use in some other part of the computer.

- **Instruction Register:** The instruction receives and stores the instruction placed on the bus from the RAM. The content of the instruction register is then split into two nibbles. The upper nibble is a two-state output that goes into the Controller-sequencer while the lower nibble is a three-state output that is read from the bus when needed.

- **Controller-Sequencer:** The controller-sequencer sends out signals that control the computer and makes sure things happen only when they are supposed to. The 12-bit output signals from controller-sequencer are called the control word which determines how the registers will react to the next positive clock edge. It has the following format: ` CON = $C_p$ $E_p$ ~$L_m$ ~$C_E$ ~$L_i$ ~$E_i$ ~$L_a$ $E_u$ $S_u$ $E_u$ ~$L_b$ ~$L_o$`

- **Accumulator:** The accumulator is an 8-bit buffer register that stores intermediate answers during a computer run. The accumulator has two outputs. The two-state output goes directly to the adder-subtractor and the three-state output goes to the bus. This implies that the 8-bit accumulator word continuously drives the adder- subtractor but only appears on the W bus when E_a is high.

- **Adder-Subtractor:** The adder-subtractor asynchronously adds to or subtracts a value from the accumulator depending on the value of S_u. It makes use of 2's complement to achieve this When S_u is low the output of the adder-subtractor is the sum of the values in the accumulator and in the B-register (O/P = A + B). When S_u is high, the output is the difference between them (O/P = A + B').

- **B-Register:** The B-register is a buffer register used in performing arithmetic operations. It supplies the number to be added or subtracted from the contents of the accumulator to the adder/subtractor. When data is available at the bus and L_b is low, at the positive clock edge, B register gets and stores the data.

- **Output Register:** The output register gets and stores the value stored in the accumulator usually after the performance of an arithmetic operation. The answer that is stored in the accumulator is loaded into the output register through the W bus. This is done in the next positive clock edge when E_a is high and L_o is low. The processed data can now be displayed to the outside world.

- **Binary Display:** The binary display is row of eight light emitting diodes (LEDs). The binary display shows us the contents of the output by connecting each LED to the output of the output register. This therefore enables viewing of the answer transferred from the accumulator to the output register in binary.

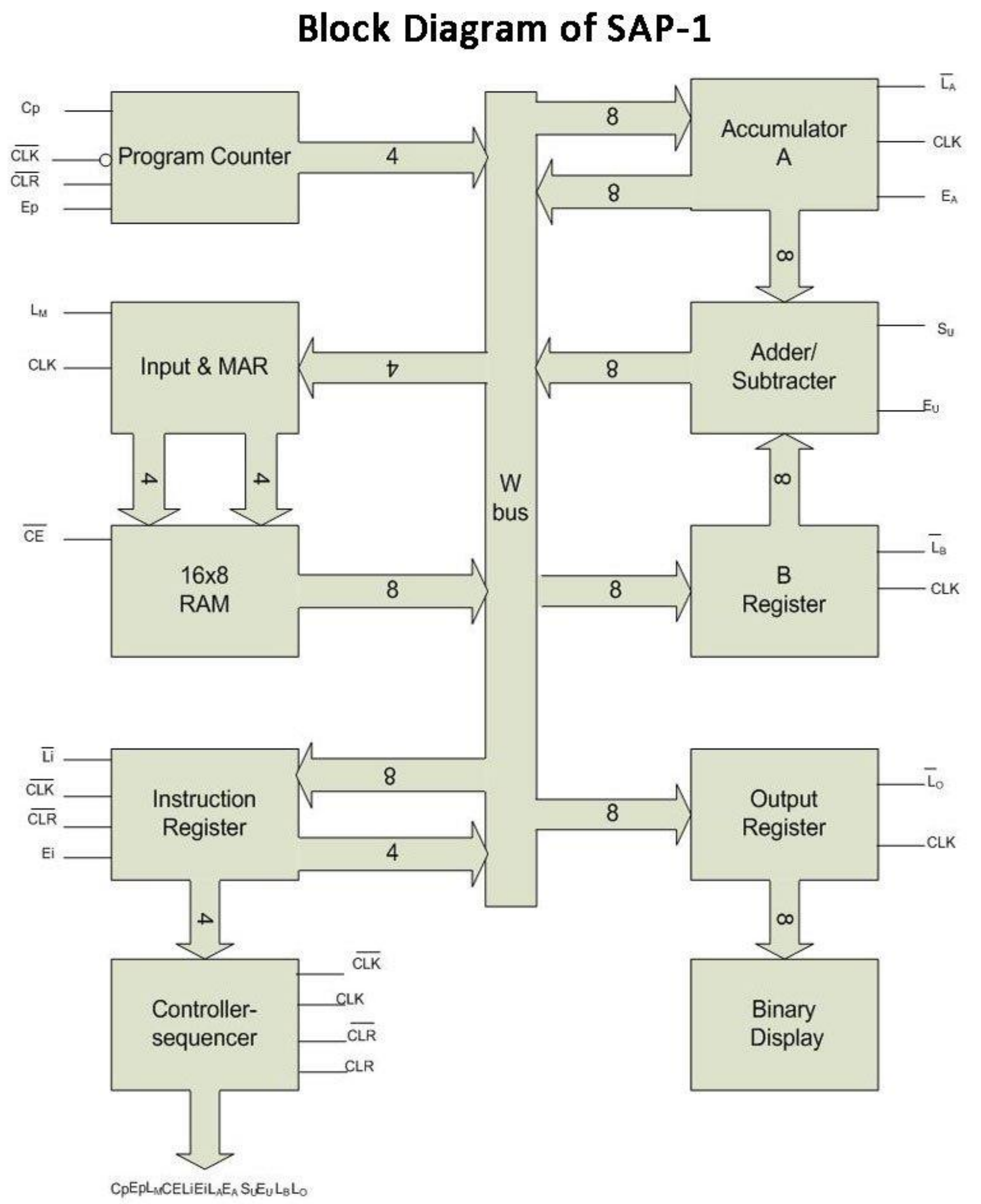

Figure 1: Block diagram of SAP-1 architecture [2].

<u>DESIGN PROCESS:</u>

General Purpose Register:

Let us first design a general-purpose register that could be used in place of the accumulator or register B or even the output register. The design begins with the memory units made of 8 D Flip-Flops. A set of MUX are then added to the inputs of the Flip-Flops which switch between the input and previous value stored in the Flip-Flops and is controlled by $'reg\_in\_en$'. The outputs are sent through a set of Tristate Buffers and are controlled by $'reg\_out\_en'$. The inputs and outputs of the register are connected through splitters for easily connecting it to the BUS. The schematic can be seen in figure 2. We can also edit the appearance of the circuit in Logisim. The generated appearance can be seen in figure 3.

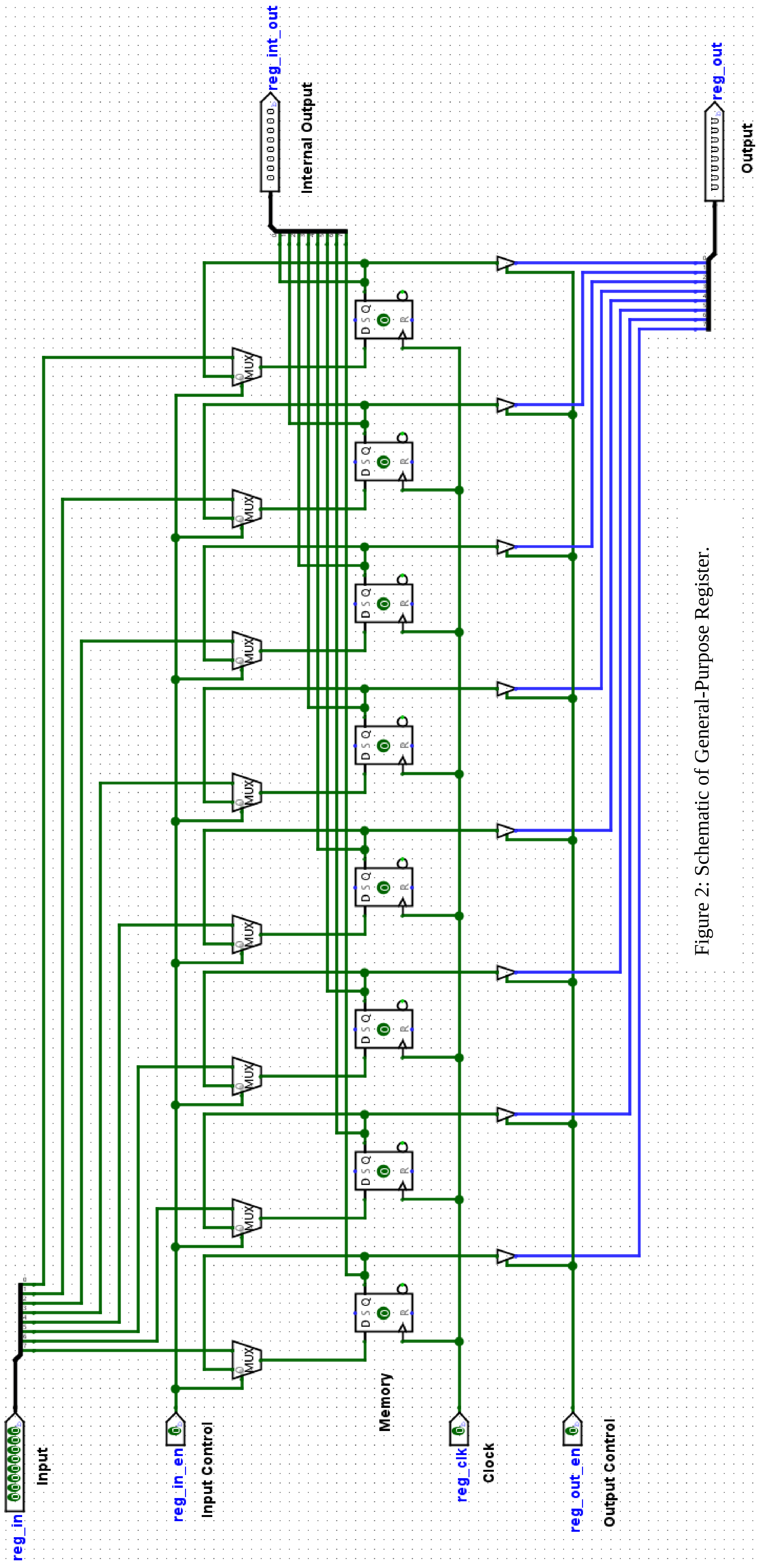

Figure 2: Schematic of General-Purpose Register.

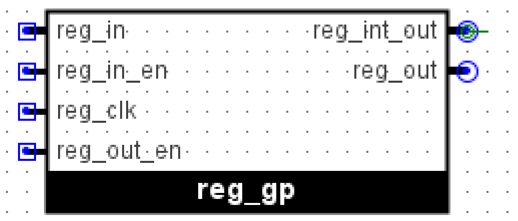

Figure 3: Circuit Appearance of General-Purpose Register.

The behavior can now be tested using the following combination.

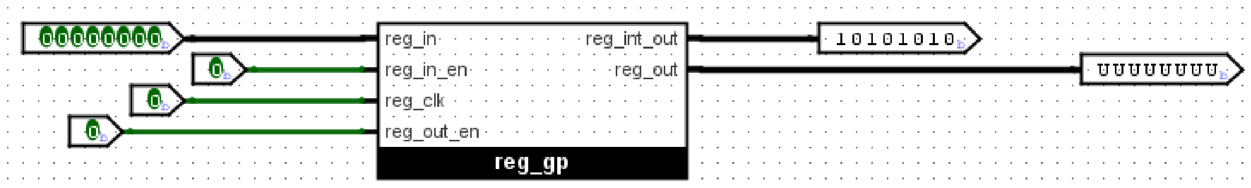

Figure 4: Testing the General-Purpose Register.

Test whether the register you made works as intended. It should only take inputs when the $'reg\_in\_en'$ is given followed by a clock pulse. Whenever a value is saved in the register, it should be visible through $'reg\_int\_out'$. It will be visible through the $'reg\_out'$ pin when the $'reg\_out\_en'$ is given.

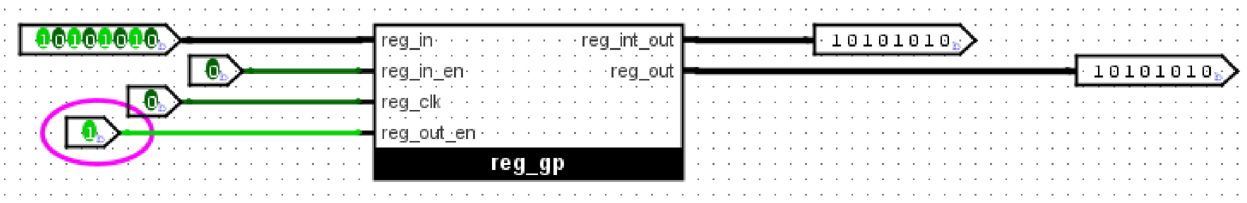

Figure 5: Example value saved in the register.

Our next job is to design the ALU.

ALU:

The ALU or Arithmetic & Logic Unit does all the actual calculations in a processor. For simplicity, out ALU will only include an adder and a subtractor. You may any other functionality as you wish. The process begins with placing 8 single bit full adders. These full adders are then cascaded (connecting carry of lower bit to carry in of higher bit) to create an 8-bit Ripple Carry Adder. One side of the adder is added input 1 coming from register A and the other input is taken through a set of XOR gates, whose signal comes from register B. The other inputs of the XOR gates are connected to $'alu\_sub'$ and is used to invert the value coming from register B to generate 1's complement. Same signal is added to carry in for 2's complement while using this pin for subtraction. The final carry out goes to $'alu\_carry\_out'$ and is generally sent to the FLAG. The output of the ALU is transferred through a Tristate buffer (8 bit in this case) and connected to $'alu\_out'$. The buffer is controlled by the $'alu\_out\_en'$. The final schematic can be seen in figure 6. The circuit appearance can be seen in figure 7.

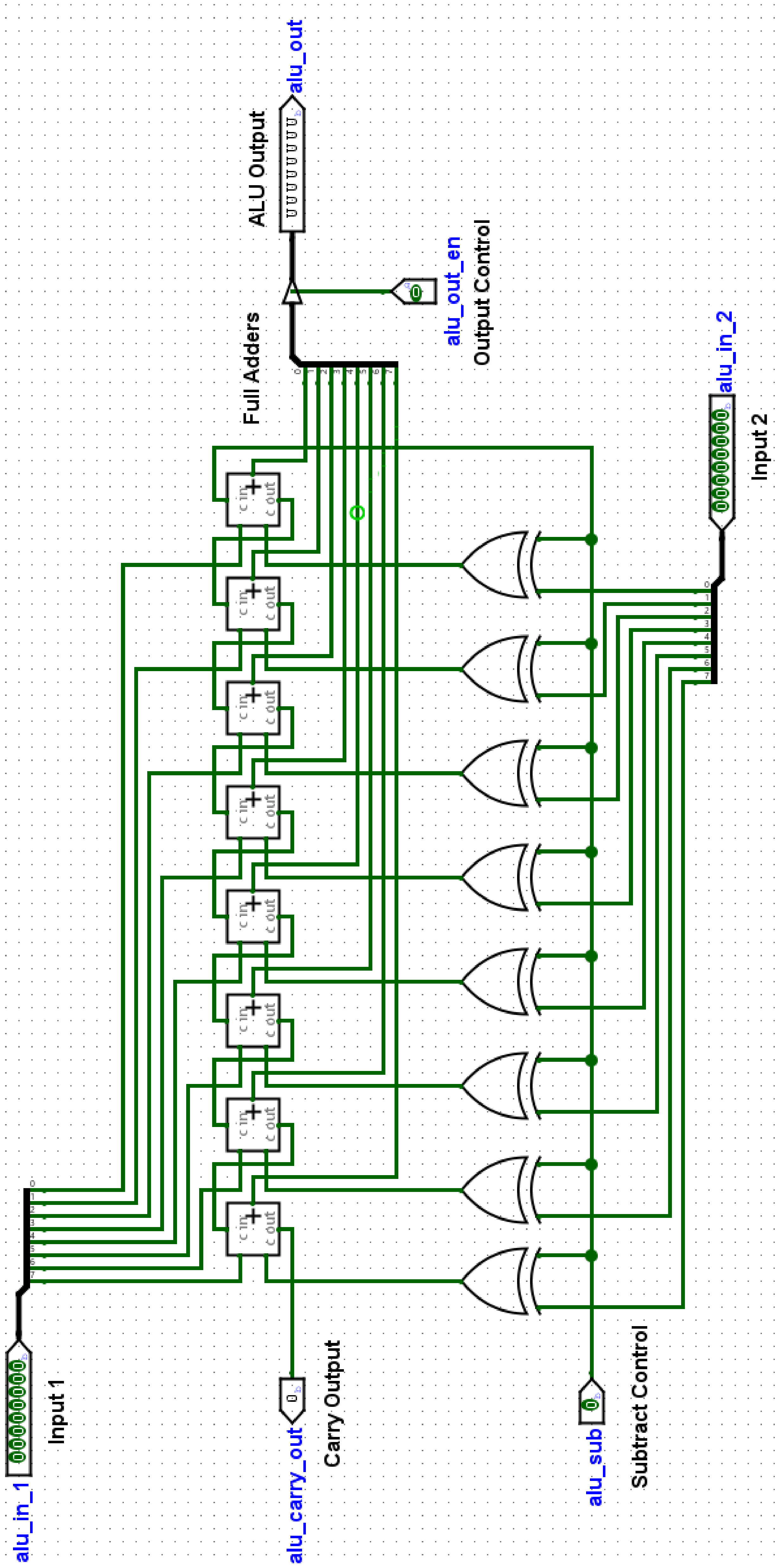

Figure 6: Schematic of ALU.

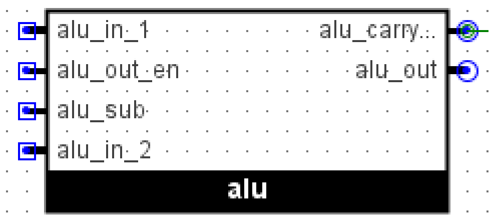

Figure 7: Circuit appearance of ALU.

The behavior can be tested in the same manner as the registers.

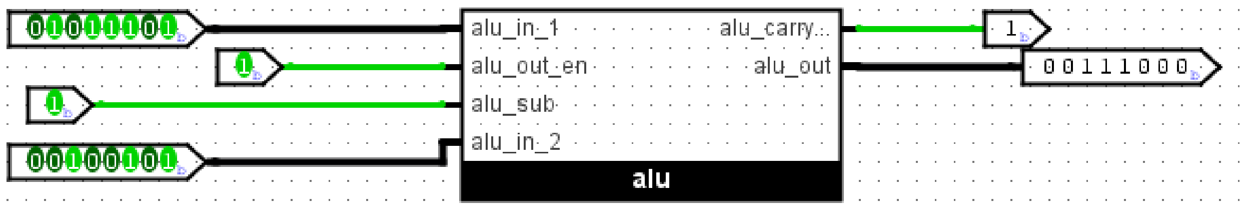

Figure 8: Testing the ALU.

Once two 8-bit inputs are given to the ALU, the output can be seen if $'alu\_out\_en'$ is given. The output will be addition if $'alu\_sub'$ is 0 and subtraction if its 1. At this point we can also verify if our ALU is working properly in conjunction to our registers. This can be tested in the manner shown in figure 9.

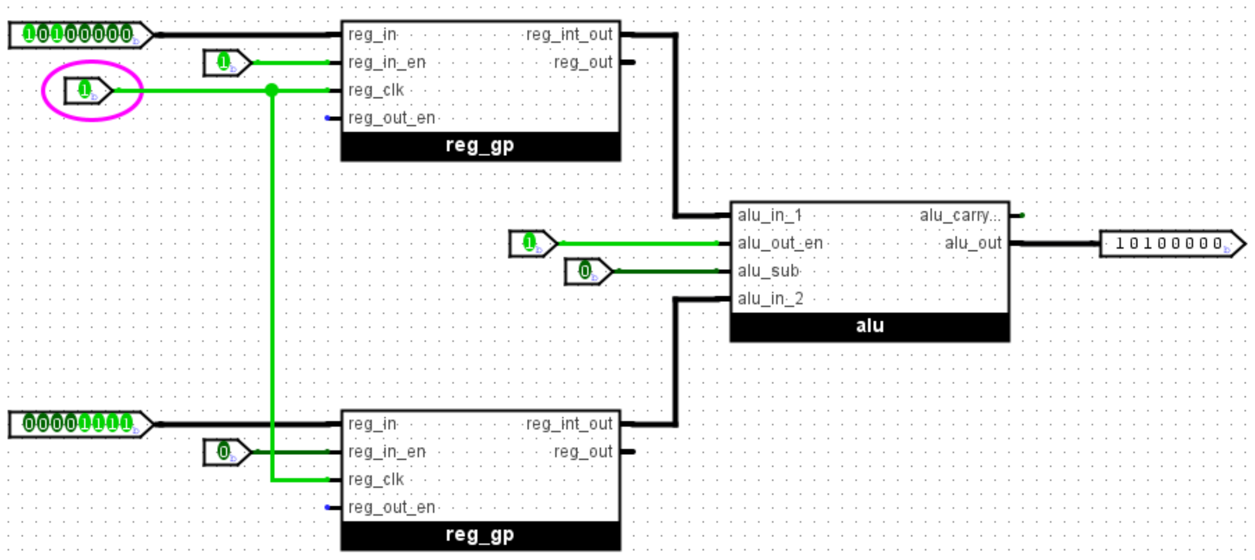

Figure 9: Testing the ALU along with the Registers.

<u>HOME TASK:</u>

Program Counter:

The program counter is responsible for providing the address for the next instruction to be fetched. The program counter keeps track on which part of the code is currently running and what will be the next instruction to be executed. In our design we will be using an asynchronous counter built using JK Flip-Flops. The "pc_en" pin controls whether the counting is enabled or not. "pc_reset" can be used to reset the counter and "pc_out_en" controls the output of the program counter. The schematic of the program counter can be seen in the following diagram:

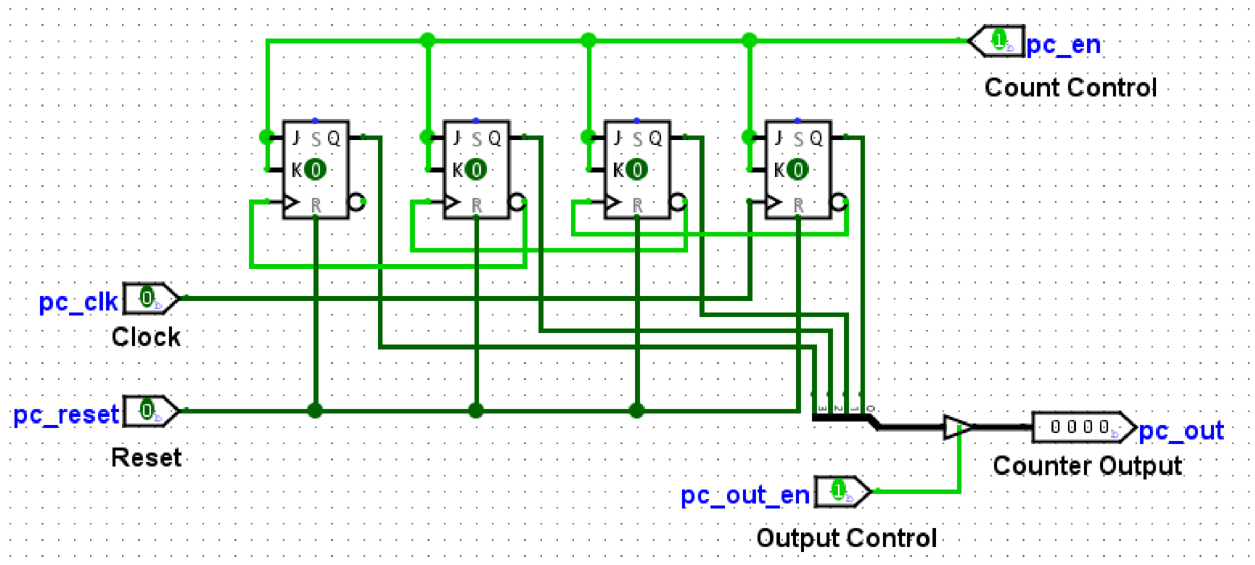

Figure 10: Schematic of Program Counter.

The behavior can be tested using the following arrangement:

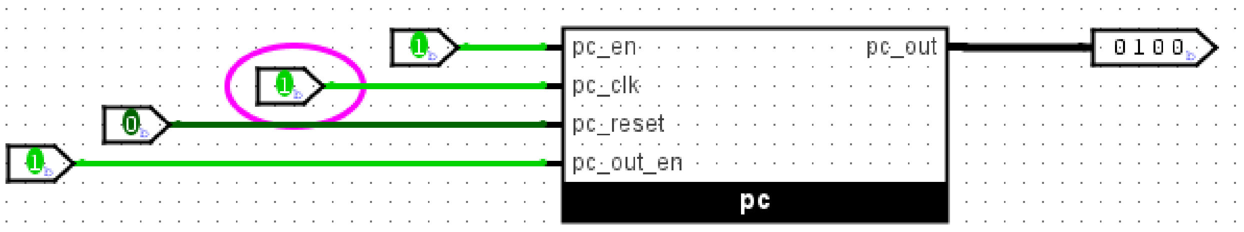

Figure 11: Testing the Program Counter.

Rubrics:

Criteria	Below Average (1)	Average (2)	Good (3)
Formatting	Report is not properly formatted, missing objectives, discussions and references.	Report is somewhat formatted but missing proper references.	Report is properly formatted.
Home Task	Home task was not addressed.	Home task was properly addressed.	
Writing	Writing is poor and not informative. It does not address the design decisions and contains high plagiarism.	Writing is average and original. Design decisions are somewhat explored.	Writing is excellent with every design decision adequately explored.
Diagram	Diagrams are of bad quality and unreadable.	Diagrams are clear and of high quality.	

References

[1] karenOk, "SAP-1-Computer," [Online]. Available: https://karenok.github.io/SAP-1-Computer/.

[2] D. R. Bhujel, "Education for ALL," [Online]. Available: https://deeprajbhujel.blogspot.com/2015/12/sap-1-architecture.html.