CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Electronics and Telecommunication Engineering

VLSI Technology Sessional

ETE 404

Experiment No:01

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

Submitted by:

Shamanza Chowdhury

ID: 1908008

Submitted to:

Arif Istiaque Rupom

Lecturer

Dept. of ETE,CUET

February 22, 2025

1 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.

2 Apparatus:

- Logisim-evolution v3.8.0
- PC

3 Design Process:

3.1 General Purpose Register:

A general purpose register is a collection of flip-flops.An n-bit register is utilized with an n-bit flip flops to store an n-bit word.It can serve as a replacement for the accumulator, register B, or the output register in the SAP-1 architecture.To design a general purpose register,the required procedures are given below:

- The design have been initiated with an array of 8 D Flip-Flops forming memory units.
- Multiplexers have been included to switch between input signals and the previously stored values in the Flip-Flops, under the control of a signal named 'reg_in_en'.
- Output signals have been directed through a sequence of Tristate Buffers, governed by 'reg_out_en'.
- To facilitate better connectivity, splitters have been utilized to link the register's inputs and outputs to the BUS.

3.1.1 Diagram:

The schematic of the general-purpose register can be seen in the following diagram:

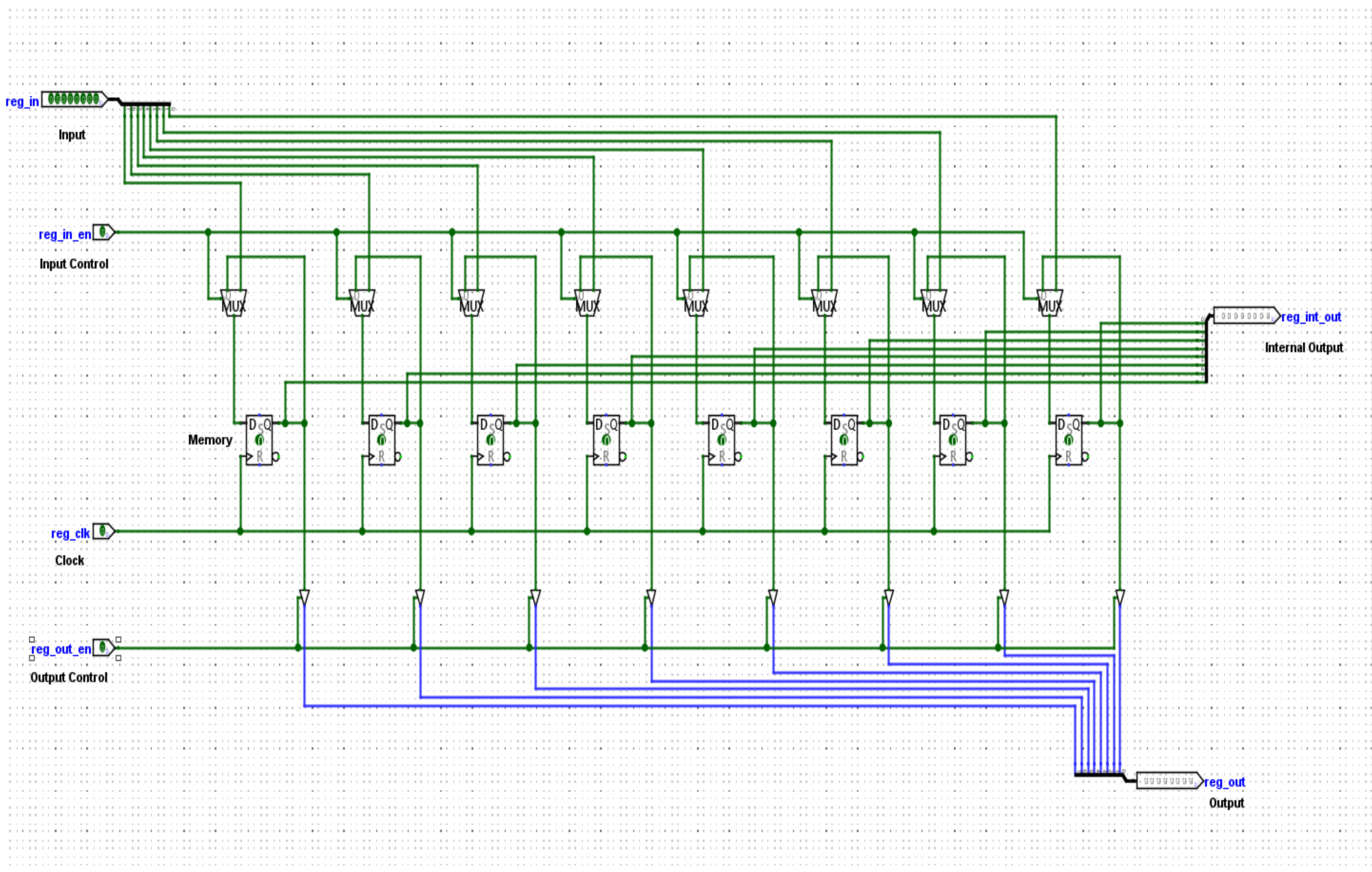

Figure 3.1.1: Schematic of General-Purpose Register.

The circuit appearance of general-purpose register is given below:

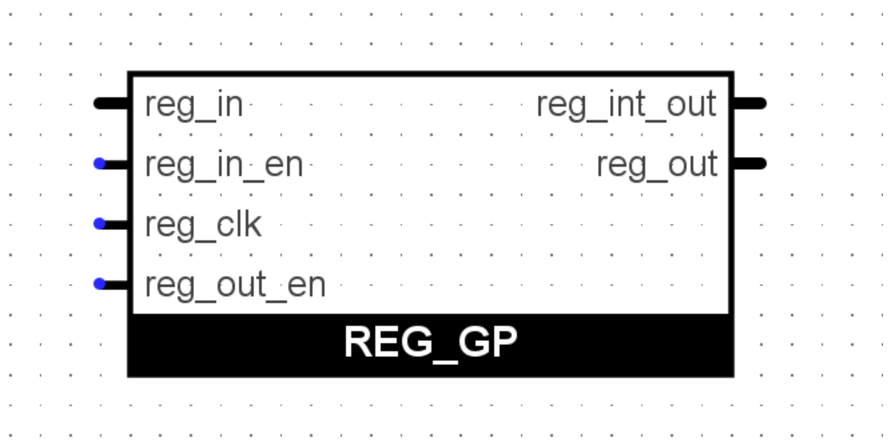

Figure 3.1.2: Circuit Appearance of General-Purpose Register.

3.1.2 Behaviour Analysis:

- It would only have taken inputs when the 'reg_in_en' signal was provided, followed by a clock pulse.
- Whenever a value had been stored in the register, it would have been visible through the 'reg_int_out' pin.
- The output would have been visible through the 'reg_out' pin after the activation of 'reg_out_en' signal.

The functionality of a general-purpose register can be tested with the following combination:

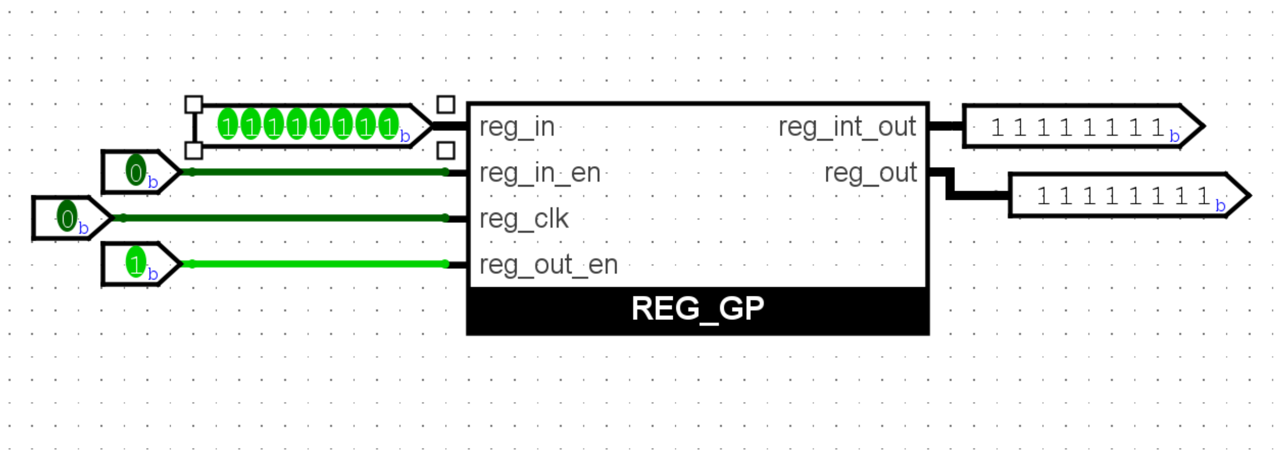

Figure 3.1.3: Testing the General-Purpose Register.

3.2 Arithmetic & Logic Unit :

In SAP-1 architecture,ALU would only include an adder and a subtractor and does all the actual calculations in a processor.To design a arithmetic & logic unit,the required procedures are given below:

- The design initiates by arranging 8 individual single-bit full adders that are subsequently linked in sequence (with the carry output of the lower bit connected to the carry input of the higher bit) to form an 8-bit Ripple Carry Adder. .
- One input side of the adder receives input 1 from register A, while the other input side is supplied through a series of XOR gates.
- These XOR gates receive signals from register B, with additional inputs connected to 'alu_sub' to invert the register B value, generating the 1's complement.
- The same signal is used to influence the carry-in for 2's complement during subtraction.
- The resultant carry-out is directed to 'alu_carry_out', typically sent to the FLAG register.
- The output of the ALU is conveyed through an 8-bit Tristate buffer and linked to 'alu_out'.
- The buffer is controlled by the 'alu_out_en' signal.

3.2.1 Diagram:

The schematic of the ALU can be seen in the following diagram:

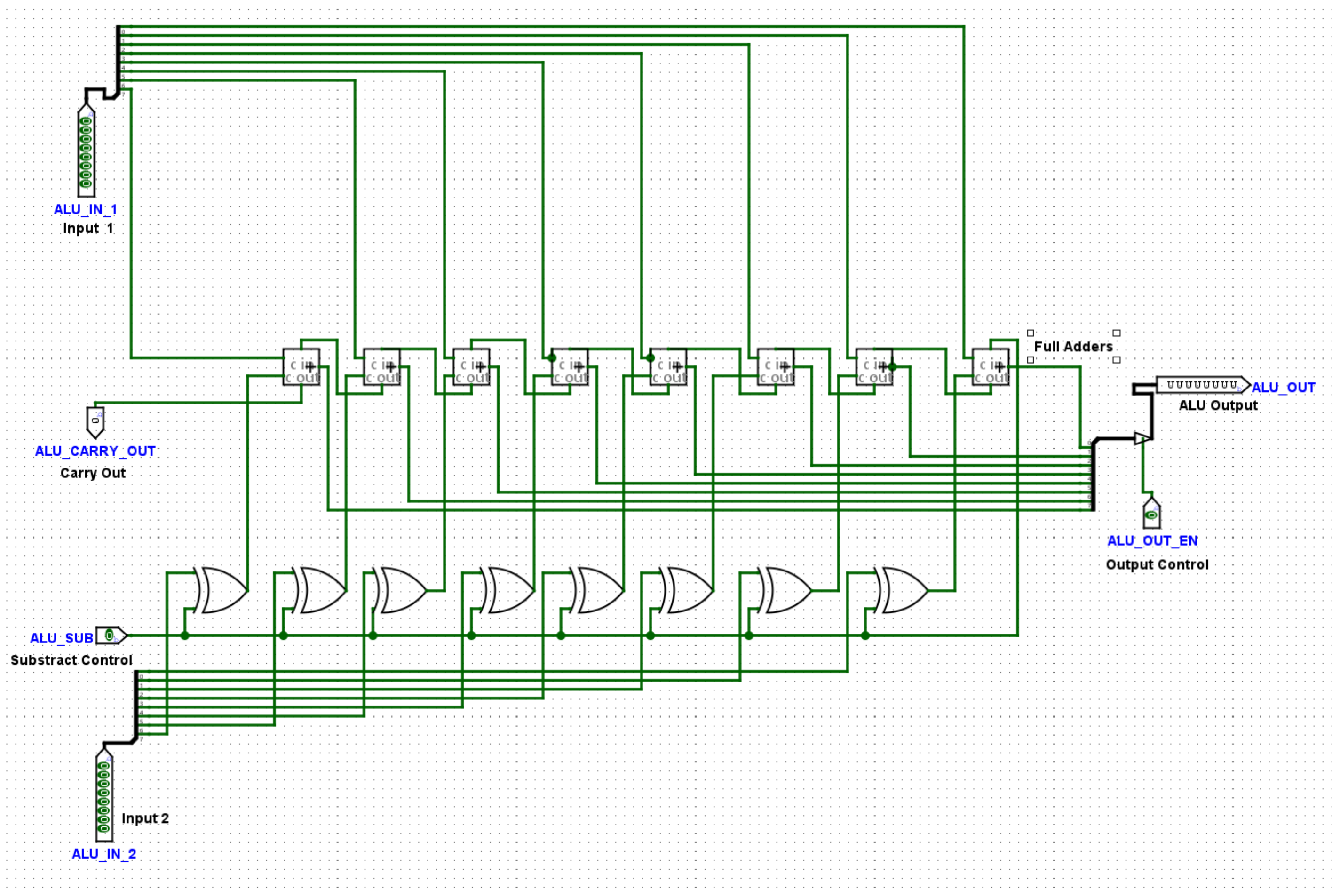

Figure 3.2.1: Schematic of ALU.

The circuit appearance of arithmetic & logic circuit logic is given below:

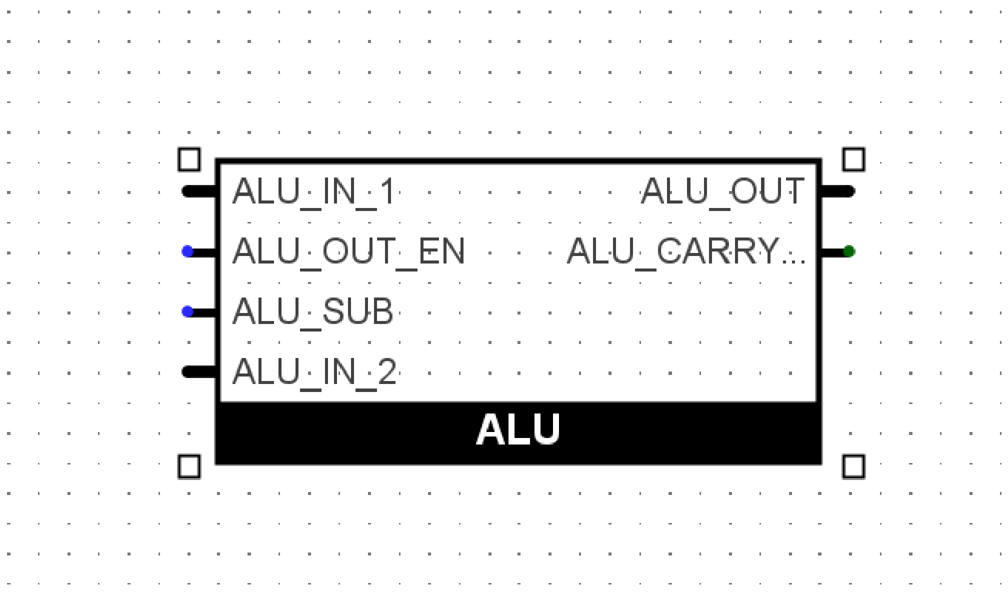

Figure 3.2.2: Circuit Appearance of ALU.

3.2.2 Behaviour Analysis:

- After providing two 8-bit inputs to the ALU, the output could have been observed upon activation of the 'alu_out_en' signal.
- The output would have represented addition if 'alu_sub' had been set to 0 and subtraction if it had been set to 1.

The functionality of a arithmetic & logic circuit can be tested with the following combination:

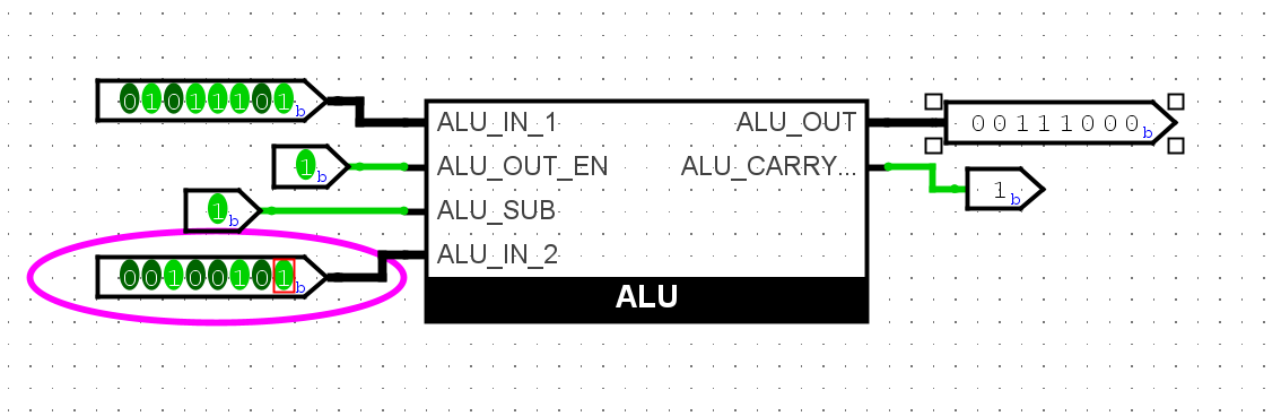

Figure 3.2.3: Testing the ALU.

It could have also verified whether the ALU was functioning properly in conjunction with two registers(Accumulator and Register-B).

- The ALU received one 8-bit input(10100000)from the register(Accumulator) by enabled the ’reg_in_en’ signal followed by a clock pulse.
- After deactivation of ’reg_in_en’ pin, the alu received another 8-bit input(00001111)from the Register-B in the same way.
- The output could have been observed in alu_out (1010111) upon activation of the ’alu_out_en’ signal.
- The output would have represented addition as ’alu_sub’ had been set to 0 .
- alu_carry represented the carry of the operation.

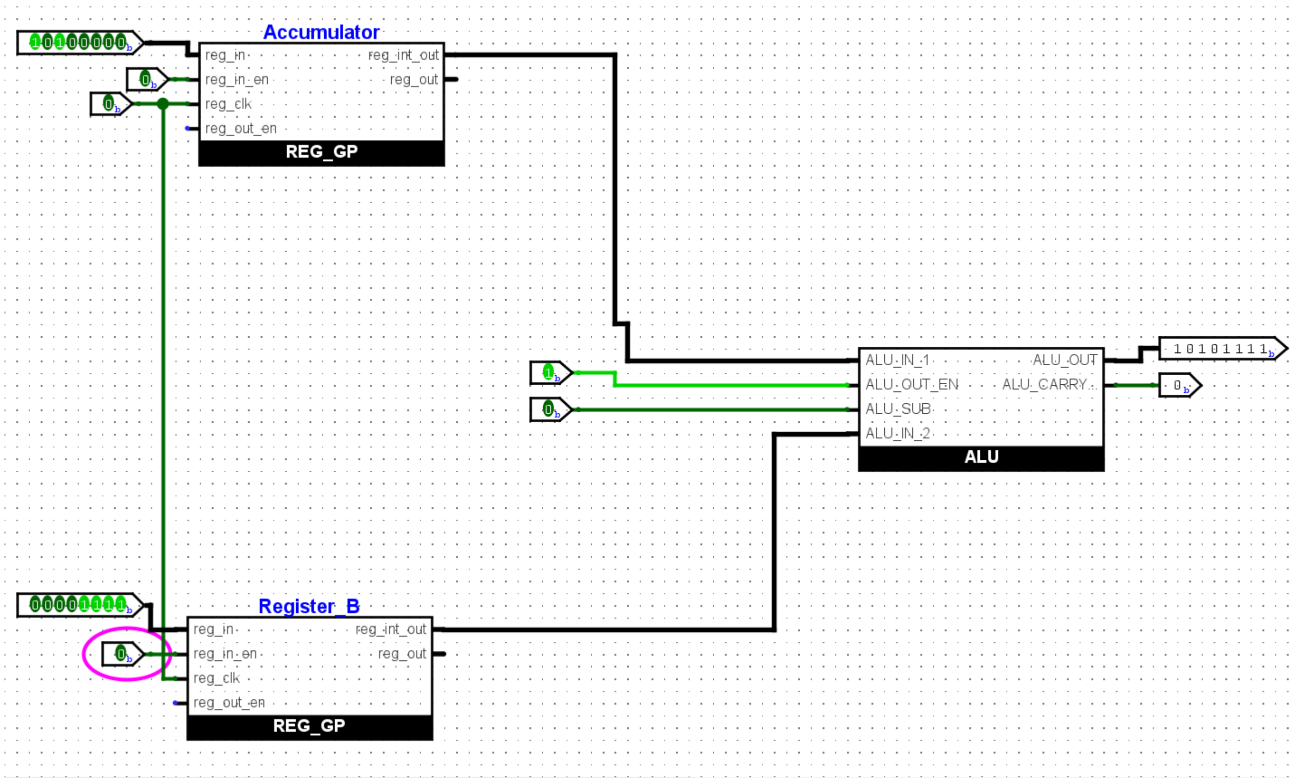

Figure 3.2.4: Testing the ALU along with two registers.

4 Home Task

4.1 Program Counter:

The program counter, a component of the control unit, stores and transmits the memory address of the next instruction to be fetched and executed. It incrementally counts from 0000 to 1111 at the beginning of a computer run, starting with address 0000, which is then incremented by 1.To design a program counter,the required procedures are given below:

- The design have been initiated with an array of 4 JK Flip-Flops forming an asynchronous counter.
- The ”pc_en” pin determined whether counting was enabled or not,followed by clock pulse that was connected to the flip flop containing the LSB.
- The ”pc_reset” pin could have been employed to reset the counter.
- Output signals have been directed through a sequence of Tristate Buffers, governed by ’pc_out_en’.
- To facilitate better connectivity, splitters have been utilized to link the register’s inputs and outputs to the BUS.

4.1.1 Diagram:

The schematic of the program counter can be seen in the following diagram:

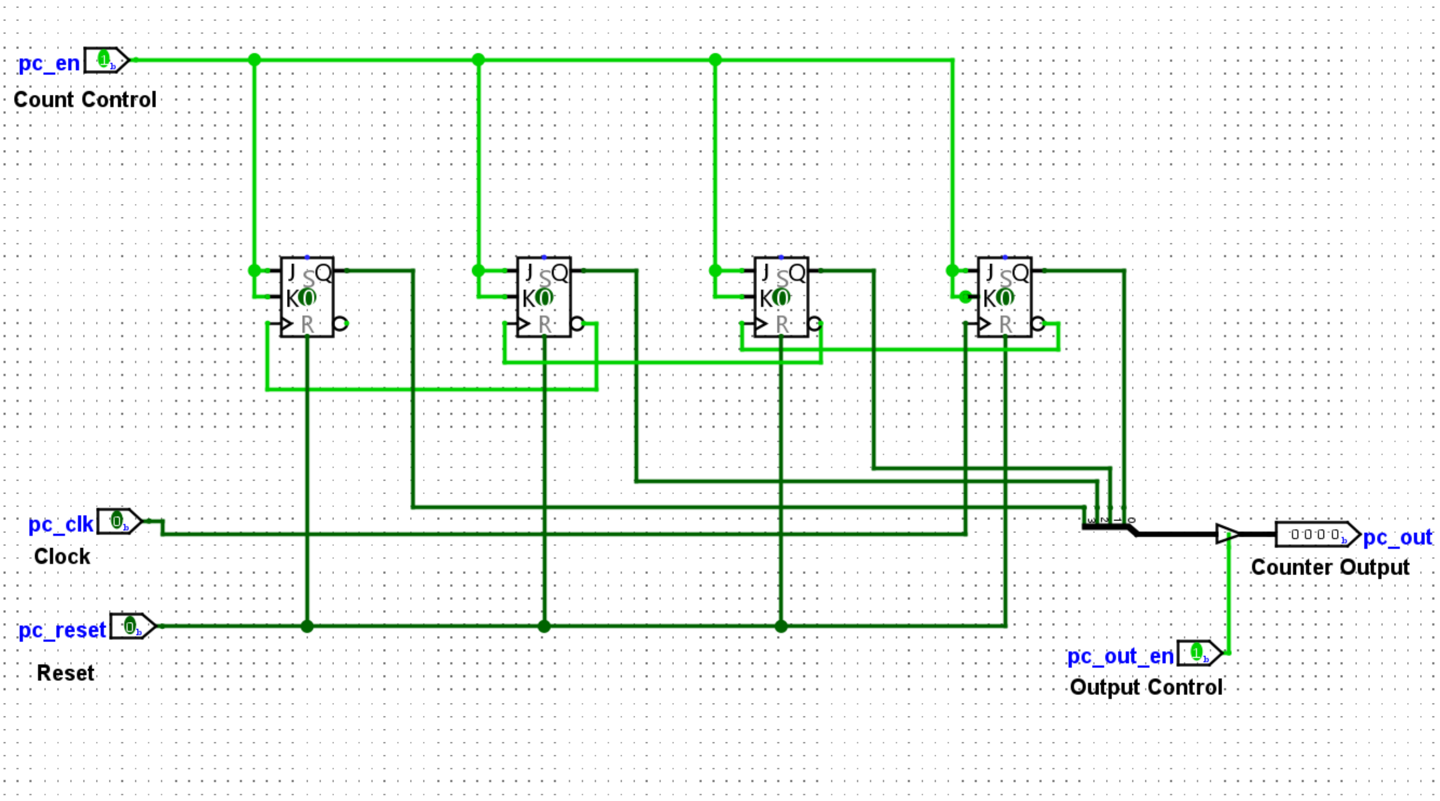

Figure 4.1.1: Schematic of Program Counter.

The circuit appearance of program counter is given below:

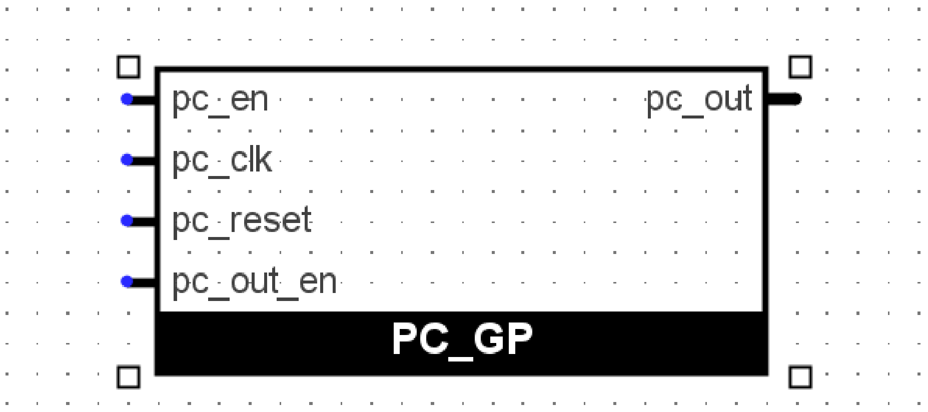

Figure 4.1.2: Circuit Appearance of Program Counter.

4.1.2 Behaviour Analysis:

- It would have taken counts when the 'pc_en' signal was provided, followed by a clock pulse.
- After each clock pulse, the counts have been incremented by 1.
- The "pc_reset" pin have been employed to reset the counter,typically initiated at the start of program execution(0000).
- The output would have been visible through the 'pc_out' pin after the activation of 'pc_out_en' signal.

The functionality of a program counter can be tested with the following combination:

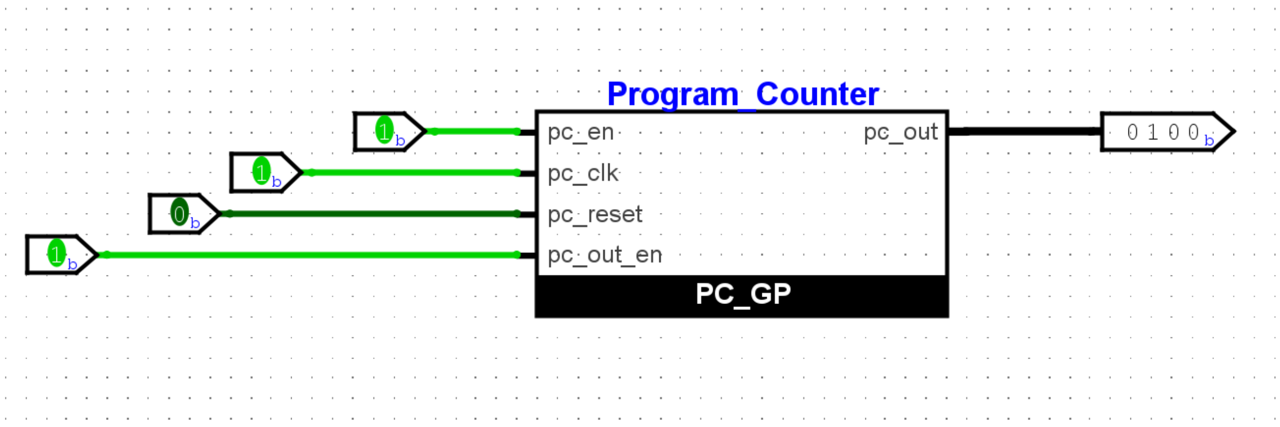

Figure 4.1.3: Testing the Program Counter.

5 Discussion:

- In this experiment, the design and functions of key components in the SAP-1 architecture: the General Purpose Register, Arithmetic & Logic Unit (ALU), and Program Counter had been discussed.
- Testing procedures had been provided to verify proper operation of the components in the SAP-1 architecture.
- The General Purpose Register, serving as a storage unit for data in n-bit words, was meticulously explained as a collection of flip-flops. Its operational aspects, such as input acceptance, value storage, and output generation, were thoroughly examined.
- By utilizing an 8-bit Ripple Carry Adder and XOR gates for subtraction, the ALU had been capable of handling addition and subtraction operations.
- Moreover, the significance of the program counter in managing memory addresses for instruction execution has been emphasized.
- Furthermore,once the circuit was designed, its behavior had been simulated by applying inputs to certain components and observing the outputs.
- There were some issues with the circuit's functionality,it had been debugged by inspecting signal paths, identifying logical errors, and making necessary adjustments for incompatible widths(changing data bits of the components).
- In conclusion, the experiment had successfully achieved its objectives of familiarizing with various facets of the SAP-1 architecture through the utilization of Logisim-evolution software.

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/.