Chittagong University of Engineering & Technology

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

NAME OF THE EXPERIMENT / REPORT

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

COURSE NO : ETE 404

COURSE TITLE : VLSI Technology Sessional

EXPERIMENT NO. : 01

DATE OF EXP. : 08.05.2023

DATE OF SUBMISSION : 17.05.2023

R E M A R K S

NAME : Touhidul Islam Khan

ROLL NO. : 1808034

LEVEL : 4

TERM : I

SECTION :

GROUP : G-2

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.

Required Software:

1. Logisim

Design:

General Purpose Register:

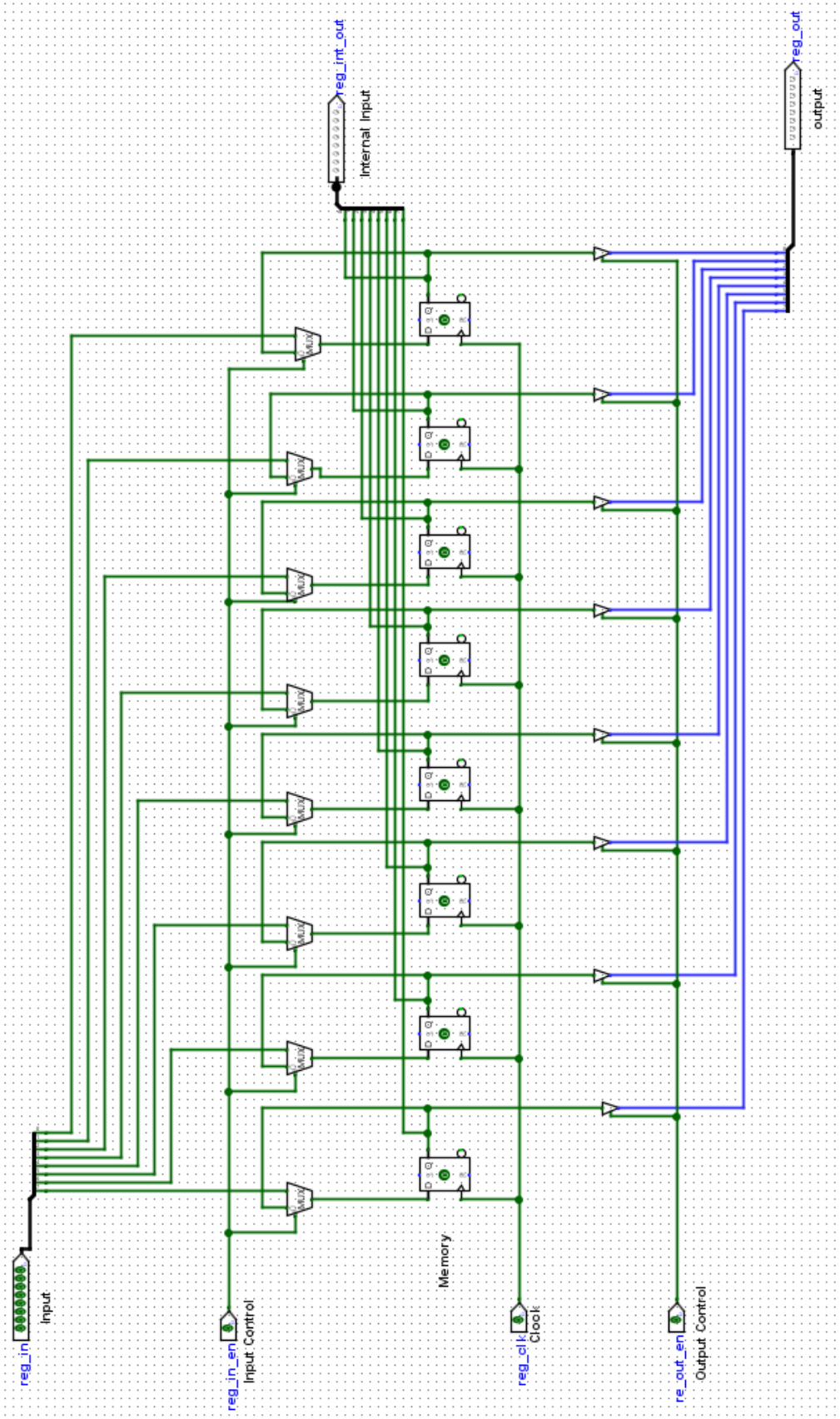

Figure No.01: Schematic of General-Purpose Register

A general-purpose register can be defined as the additional registers present to be used for memory address or data in the CPU. In Figure No.01, a schematic of general-purpose register has been shown. 8 D Flip-Flops are used as memory unit in order to construct an 8-bit register. MUX is used to switch between the present and previous values of the flip-flops controlled by 'reg_in_en' pin which is the write control pin. The write control pin is used to control the data saving of the flip-flops. Splitters are used to connect the input and output to the BUS.

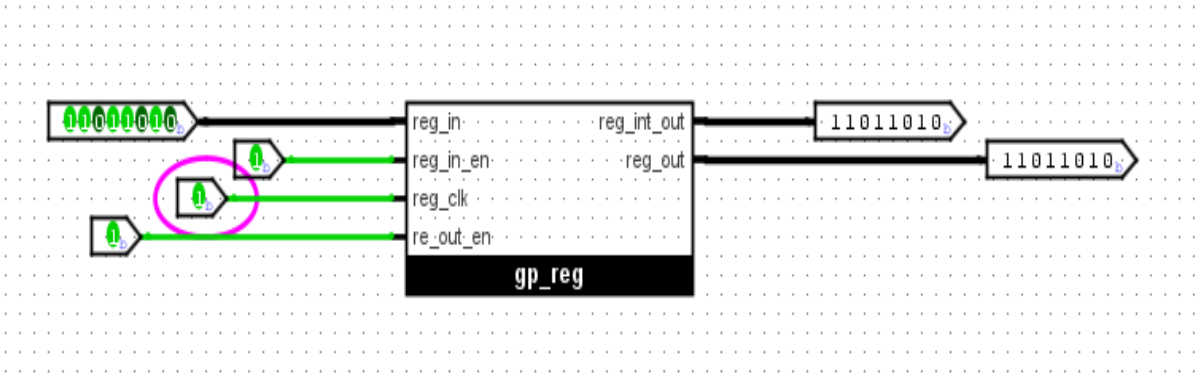

Figure No.02: Example of value saved in register

The register takes input with the clock pulse only when the write control pin is enabled. The 'reg_out_en' is the read control pin which control if data can be retrieved from the register. The 'reg_out' pin shows output of the register only when the read control pin is enabled. The data in the register will not change until the write control pin is enabled and the data cannot be viewed until the read control is enabled even if clock pulse is given.

Arithmetic Logic Unit (ALU):

Arithmetic Logic Unit or ALU in short is a part of the processor which does arithmetic and logical operations between binary numbers. In Figure No.03, we can see the schematic diagram of the ALU. For simplicity only addition and subtraction functionality are designed. The design process begins by taking 8 single bit full adders which are then cascaded. One input comes from the 'alu_in_1' and the other from the XOR gates whose signal comes from the 'alu_in_2'. These are the main two inputs of the ALU. The remaining input of the XOR gate is connected to 'alu_sub' which when enabled helps to invert the input and generate 1's complement. While performing subtraction the same signal is added to carry to generate 2' complement. The final carry goes to the 'alu_carry_out'. The output of the ALU is then connected to the 'alu_out' through a tristate buffer. The 'alu_out_en' controls the buffer that is controls whether the output is shown or not.

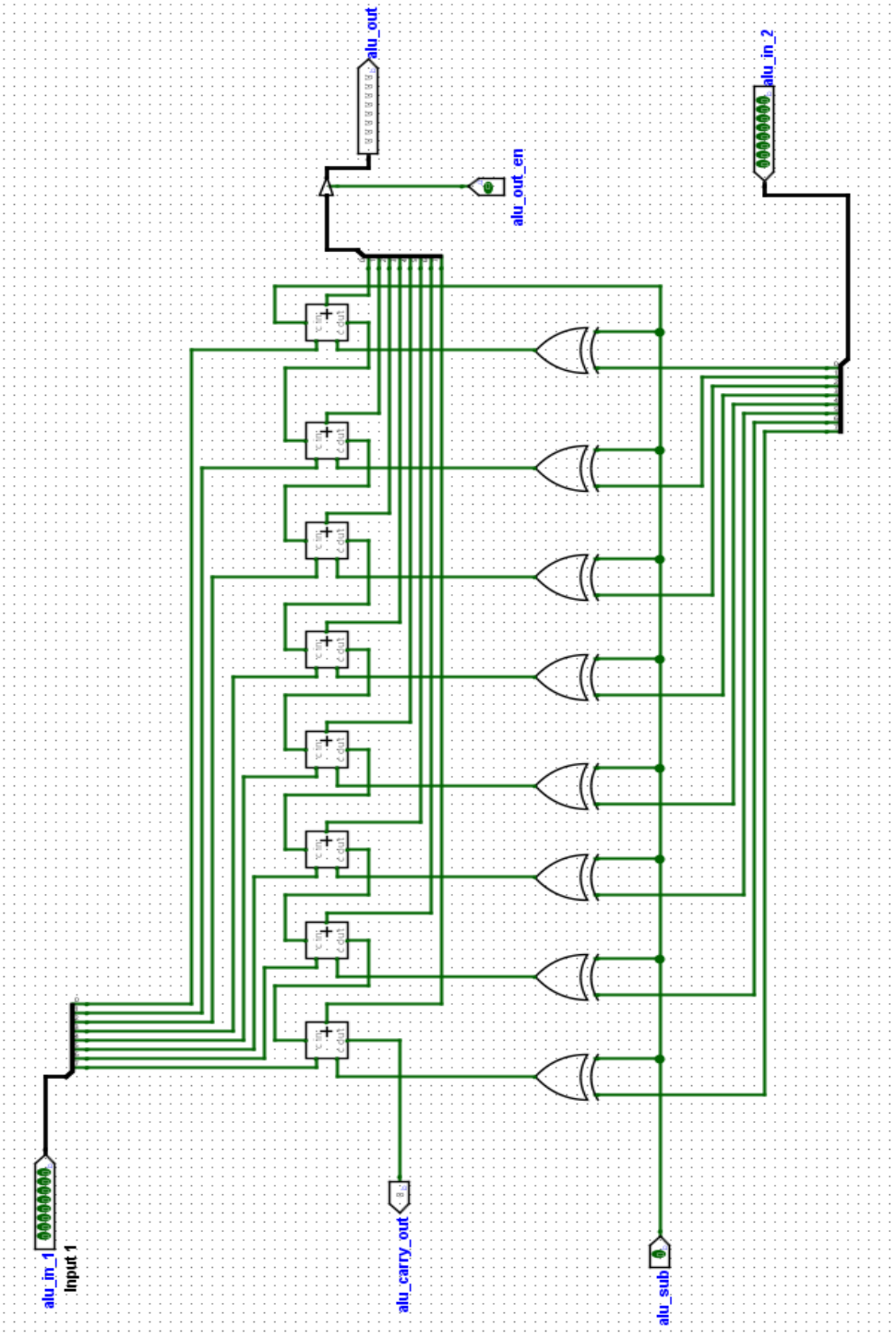

Figure No.03: Schematic of ALU

When input is given to the ALU, by default it performs the addition but when the ‘alu_sub’ is enabled it performs subtraction. The output can be seen if ‘alu_out_en’ is enabled otherwise not.

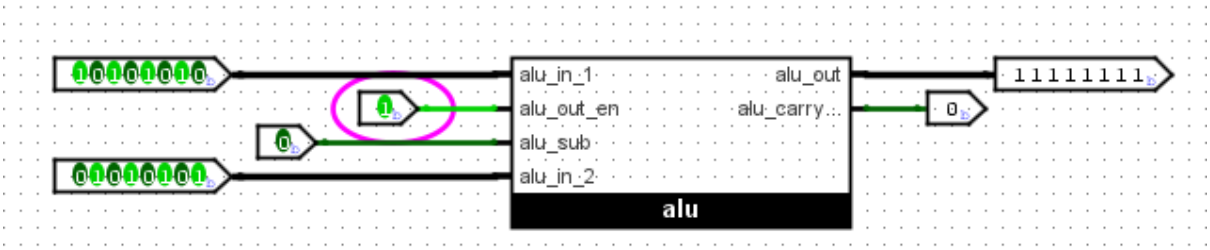

Figure No.04: Example of ALU operation

When two 8-Bit Registers are connected to the input of the ALU then the summation and subtraction is performed between the values stored in the registers.

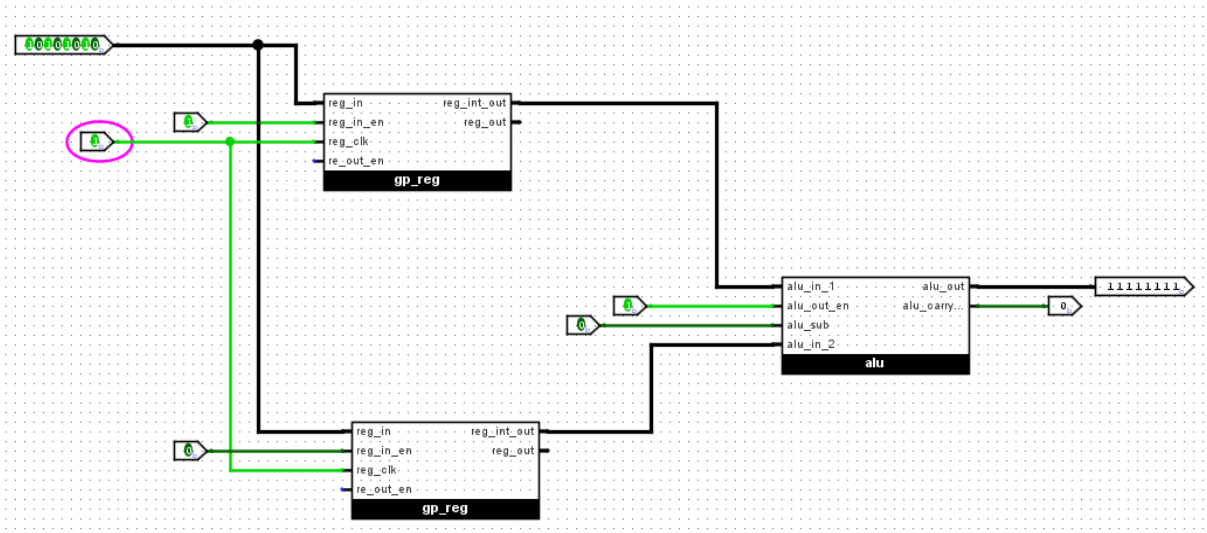

Figure No.05: ALU along with registers.

Program Counter:

The program counter in the SAP-1 architecture provides the address of the next instruction to be fetched and also keeps track of the part of code that is running. In our design we have used asynchronous counter using JK Flip-Flops. The reason behind using JK flip-flop is that it can toggle between its state if both inputs are high depending on the clock pulse. The ‘pc_en’ or program counter enable pin controls whether the counter is active or not. The ‘pc_reset’ pin is used to reset the counter and ‘pc_out_en’ pin controls whether the counter will give any output or not.

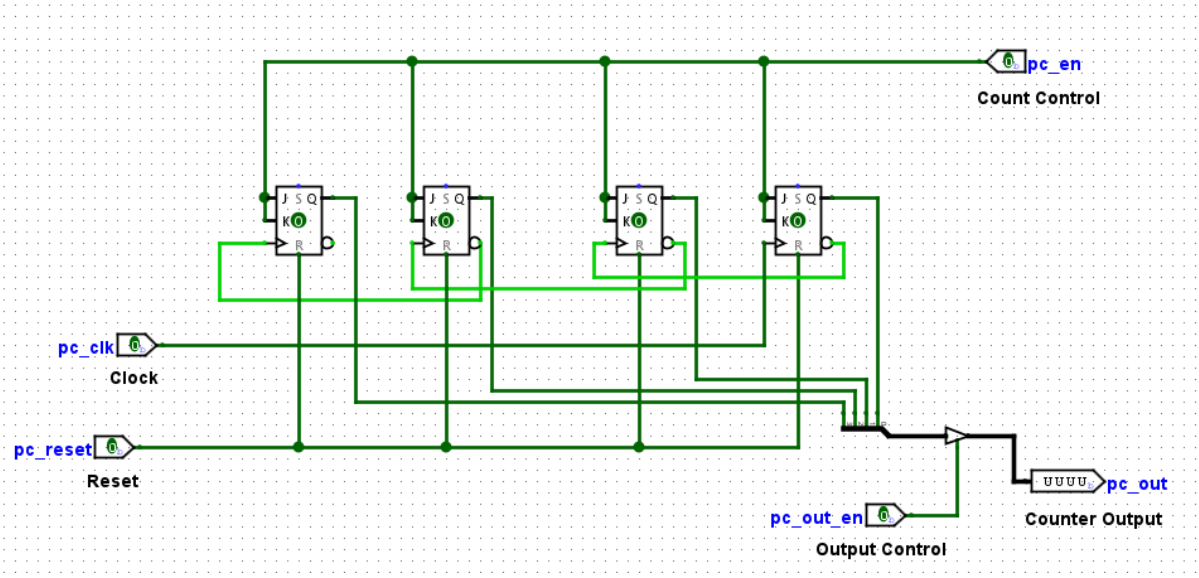

Figure No.06: Schematic of Program Counter.

When the 'pc_en' is on only then the counter counts for every clock pulse. If there is no pulse through the 'pc_clk' then the counter will not count.

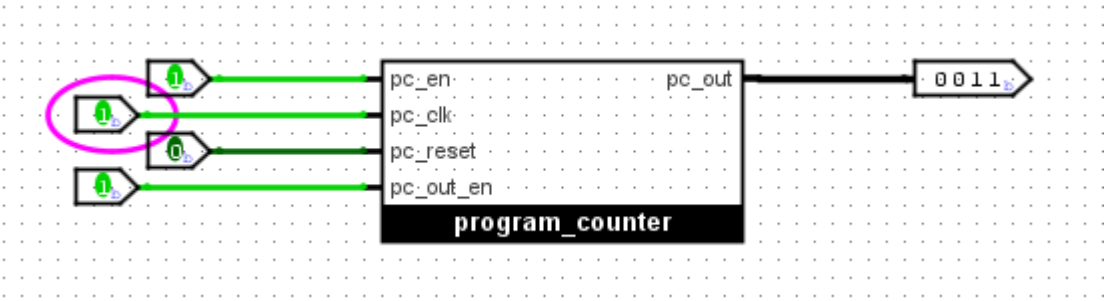

Figure No.07: Program counter operation.

Discussion:

1. Design of register, ALU and program counter was learnt and implemented in this experiment.
2. If uncontrolled buffer was used instead of controlled buffer, then the read control will not be established.
3. Connection of the wires were made carefully so that they do not touch each other and form cross connection.
4. The register did not store value when there was no co-ordination between the clock and the write control pin i.e., if write control is enabled and there is no clock pulse then the register will not store any value and vice-versa.
5. Addition and subtraction between two 8-bit numbers was performed to check the correctness of the ALU design.
6. The counter counts along with clock pulse and increase one in each pulse.