Chittagong University of Engineering and Technology

Department of Electronics and Telecommunication Engineering

ETE 404

VLSI Technology Sessional

Experiment No: 02

Schematic Modeling and Implementation of Modified SAP Architecture in Logisim – 2

Submitted by:

Shamanza Chowdhury

ID: 1908008

Submitted to:

Arif Istiaque Rupom

Lecturer

Dept. of ETE,CUET

May 26,2024

1 Objectives:

- To familiarize with the BUS in a microprocessor system.
- To familiarize with addressing and RAM.
- To familiarize with program execution cycle.

2 Apparatus:

- **Required Software:** Logisim-evolution v3.8.0
- **Required Hardware:** PC

3 Design Process:

3.1 Decoder(4x16):

A decoder is a combinational circuit with n select bits and up to 2^n output lines, where only one output is active at a time based on the input combination. In our case, a 4-to-16 decoder with 4 select lines and 16 output lines will be used to access 16 different addresses in our RAM.To design a 4X16 decoder,the required procedures are given below:

- The design have been initiated with an array of 16 AND Gates where each AND gate is responsible for generating one output line in one specific combination of the input lines('dec_sel').
- When a particular combination of the input lines is present, the corresponding AND gate outputs a high signal (1), activating the respective output line('dec_out').

3.1.1 Diagram:

The schematic of the 4x16 decoder can be seen in the following diagram:

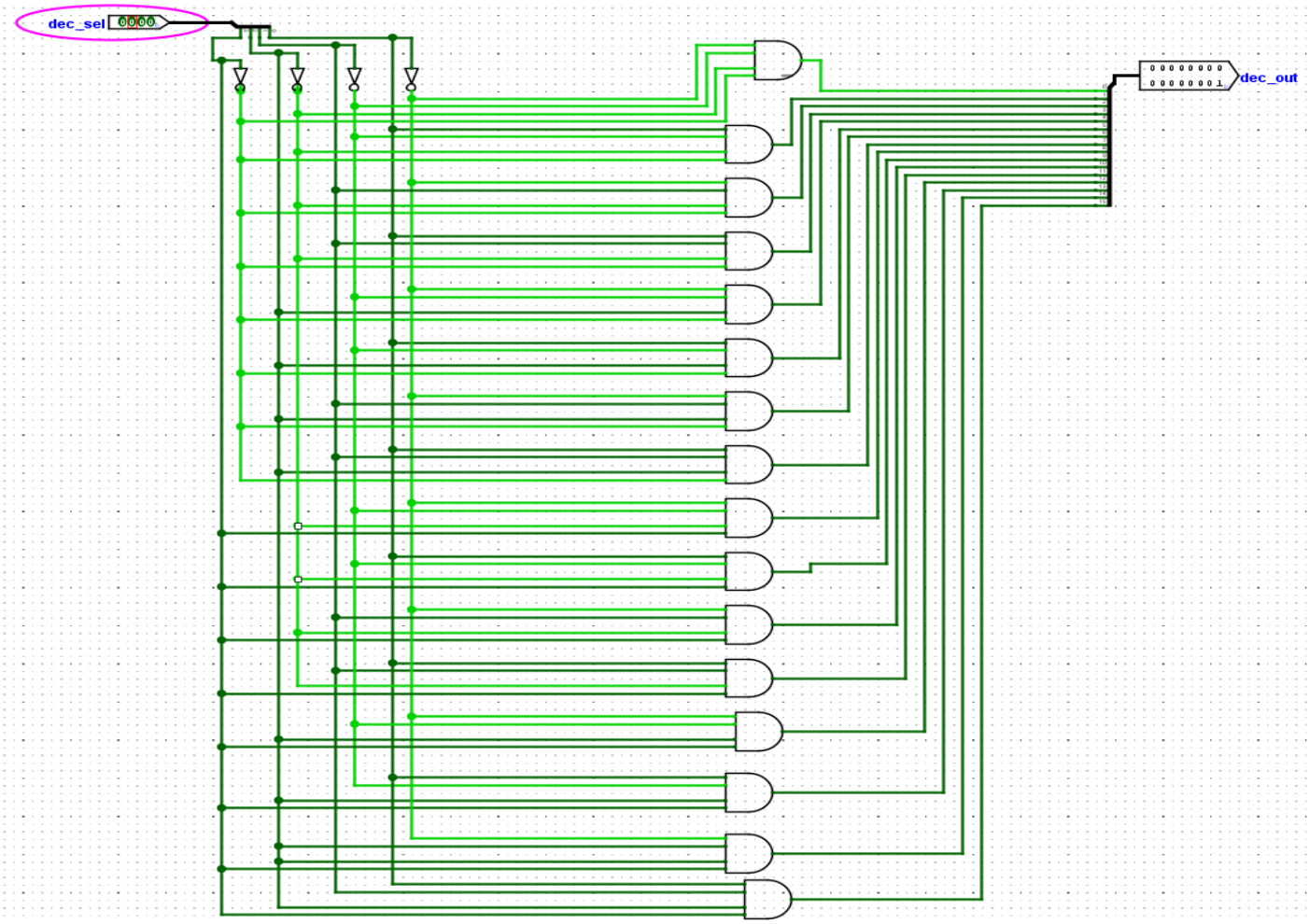

Figure 3.1.1: Schematic of 4x16 Decoder.

3.1.2 Behaviour Analysis:

The 4x16 decoder had been taken 4-bit binary input and decoded it to activate one of the 16 output lines. The functionality of a 4X16 decoder can be tested with the following combination:

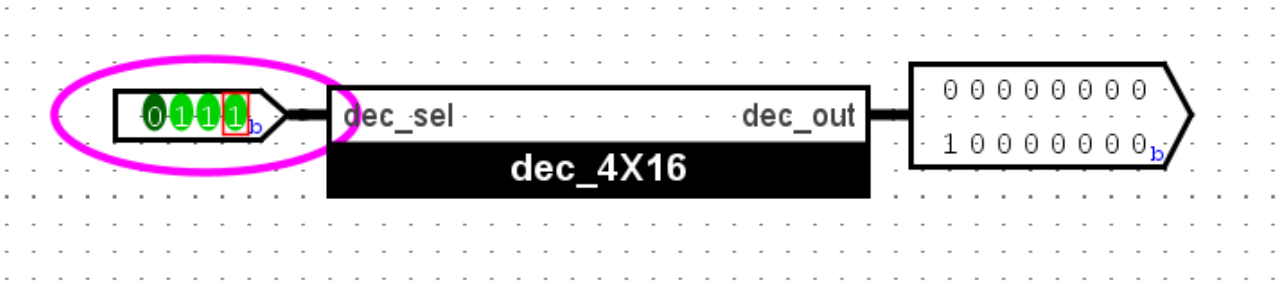

Figure 3.1.2: Testing the 4x16 Decoder.

3.2 Static Random Access Memory(SRAM) :

Static Random-Access Memory (SRAM) is a type of that does not need to be periodically refreshed and used to store each bit.To design a SRAM cell,the required procedures are given below:

- The design consists of a combination of control signals and registers to enable read and write operations.
- The wr_en, rd_en, cs, and clk signals coordinate to control the flow of data into and out of the memory.
- The use of AND gates ensures that data is only written or read when the appropriate conditions are met.
- The first AND gate combines wr_en and cs to produce a signal (reg_in_en) for enabling data writing.
- The second AND gate combines rd_en and cs to produce a signal (reg_out_en) for enabling data reading.

3.2.1 Diagram:

The circuit appearance of SRAM Cell is given below:

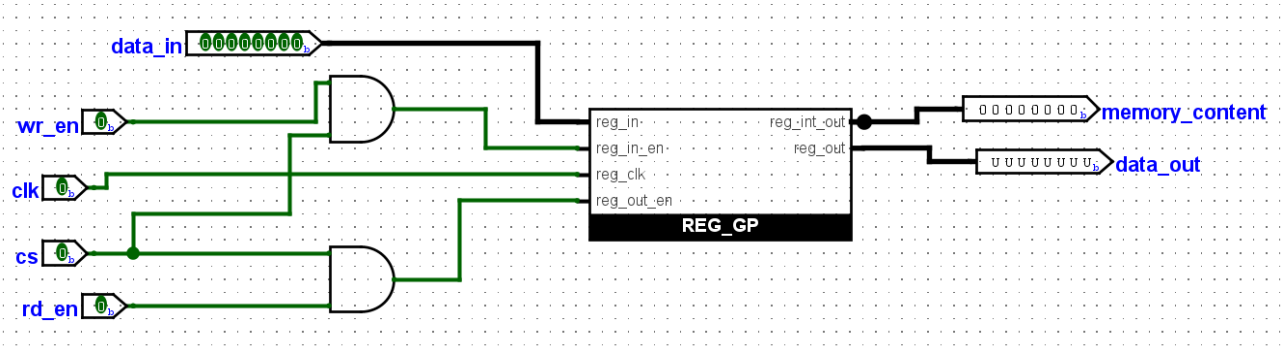

Figure 3.2.1: Circuit Appearance of SRAM Cell.

3.2.2 Behaviour Analysis:

- **Write Operation:**
  - When wr_en and cs are both high (1), the first AND gate outputs a high signal, enabling the reg_in_en line.
  - The 8-bit data from data_in is then written into the register REG_GP on the rising edge of the clk signal.

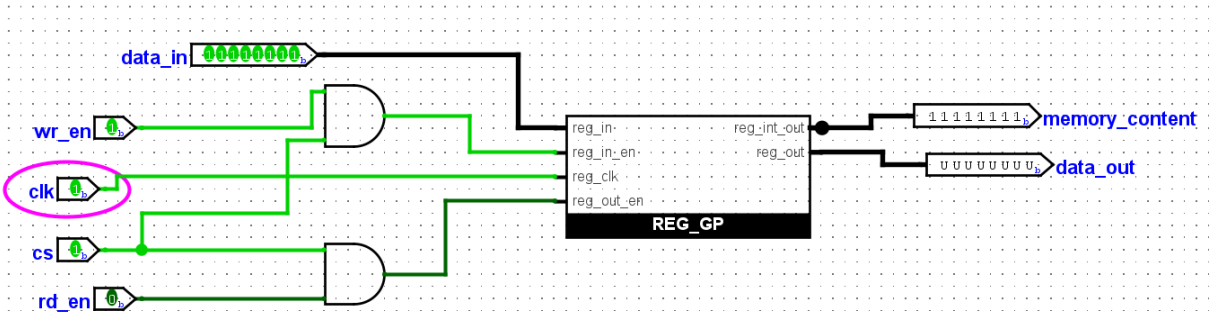

Figure 3.2.2: Testing the SRAM Cell during write operation..

- **Read Operation:**
  - When rd_en and cs are both high (1), the second AND gate outputs a high signal, enabling the reg_out_en line.
  - The data stored in the register REG_GP is then placed on the data_out line.

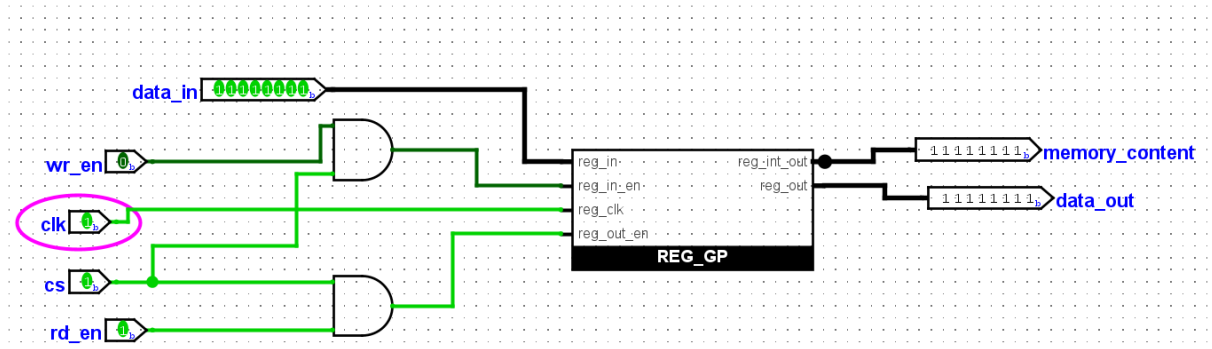

Figure 3.2.3: Testing the SRAM Cell during read operation.

3.3 Random Access Memory(RAM) :

RAM is a memory bank accessible to the processor for use during program execution. Before any program is executed, it is first loaded into RAM.To design a RAM ,the required procedures are given below:

- The design consists of a combination of a register,a 4x16 decoder,16 sram cells.
- All components has one same clock for synchronization.

3.3.1 Diagram:

The circuit appearance of RAM is given below:

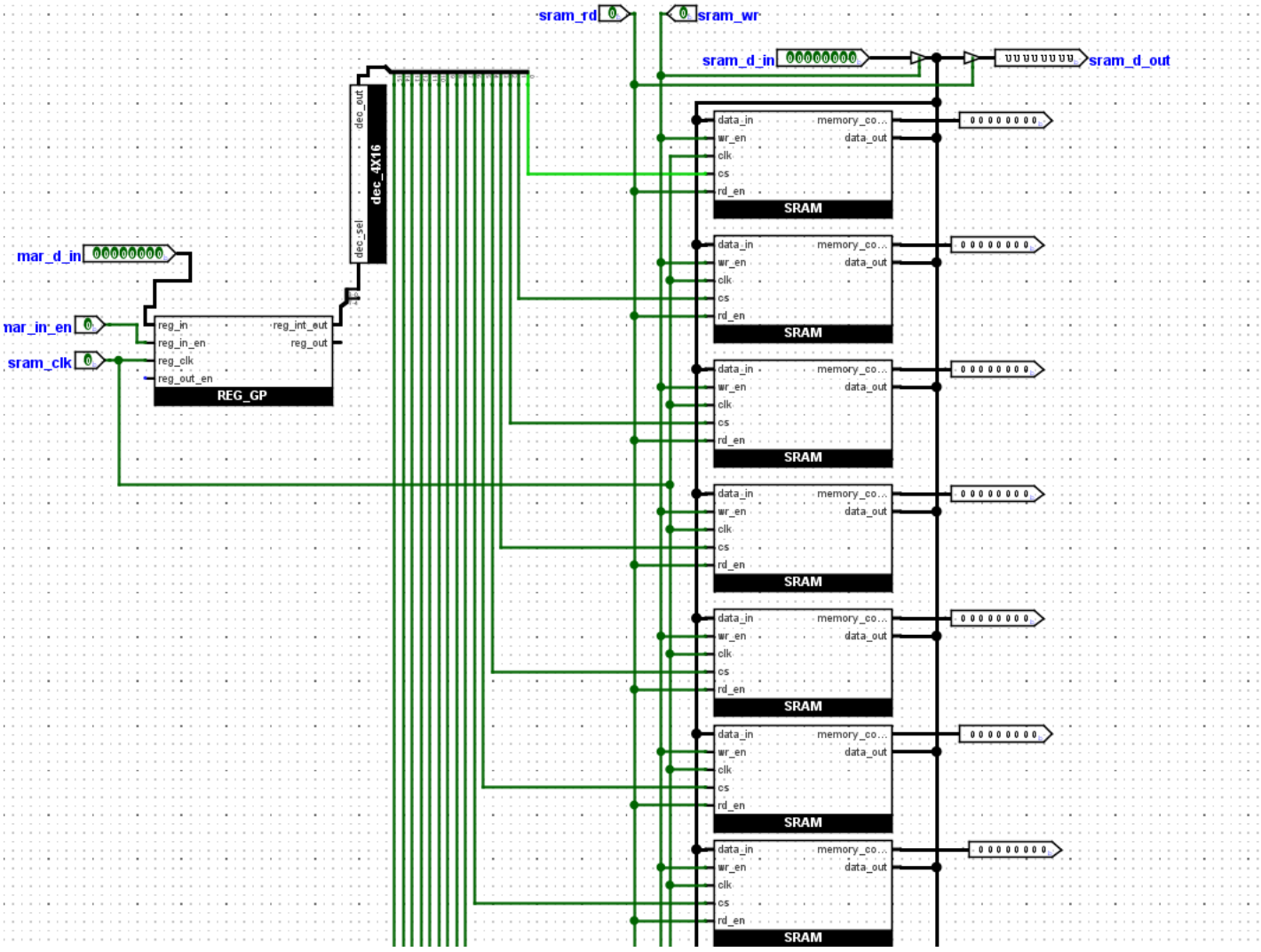

Figure 3.3.1: Circuit Appearance of RAM part-I .

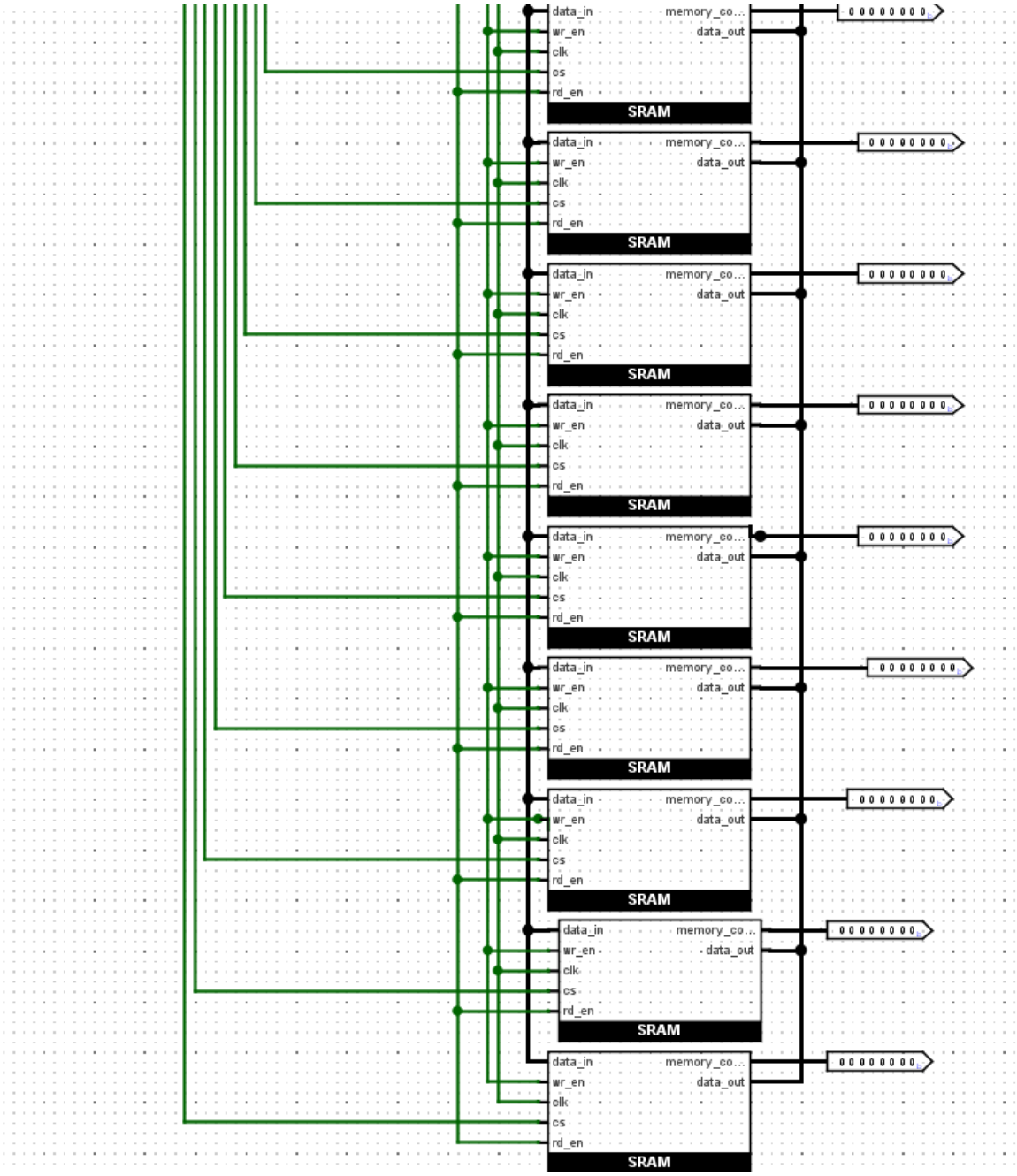

Figure 3.3.2: Circuit Appearance of RAM part-II .

3.3.2 Behaviour Analysis:

First, mar_d_in was provided along with an enable signal and a clock pulse, saving reg_int_out as the decoder's input. The decoder's outputs were connected to the CS pins of each SRAM cell, activating the corresponding CS pin.

- **Write Operation:**
  - When sram_d_in and sram_wr were enabled with a clock pulse, the data was stored in the activated SRAM cell.

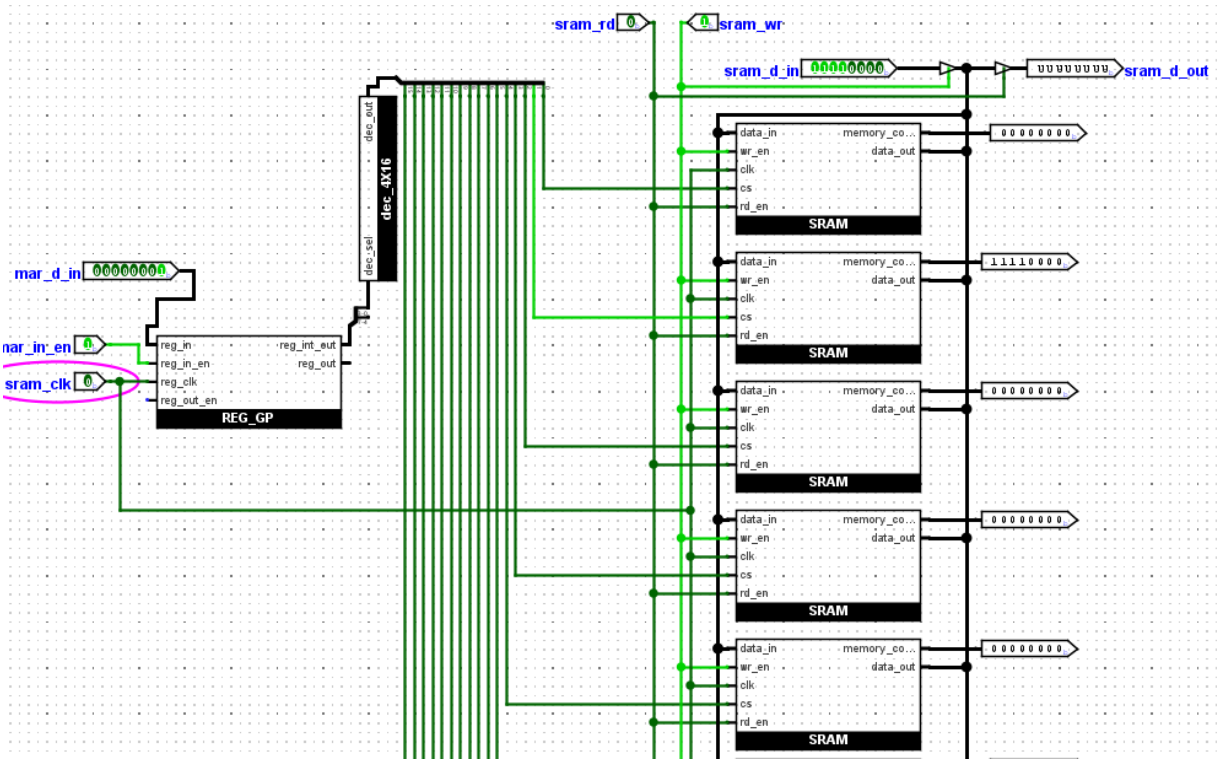

Figure 3.3.3: Testing the RAM during write operation.

- **Read Operation:**
  - When sram_rd was enabled with a clock pulse, the stored data was read from RAM through sram_d_out..

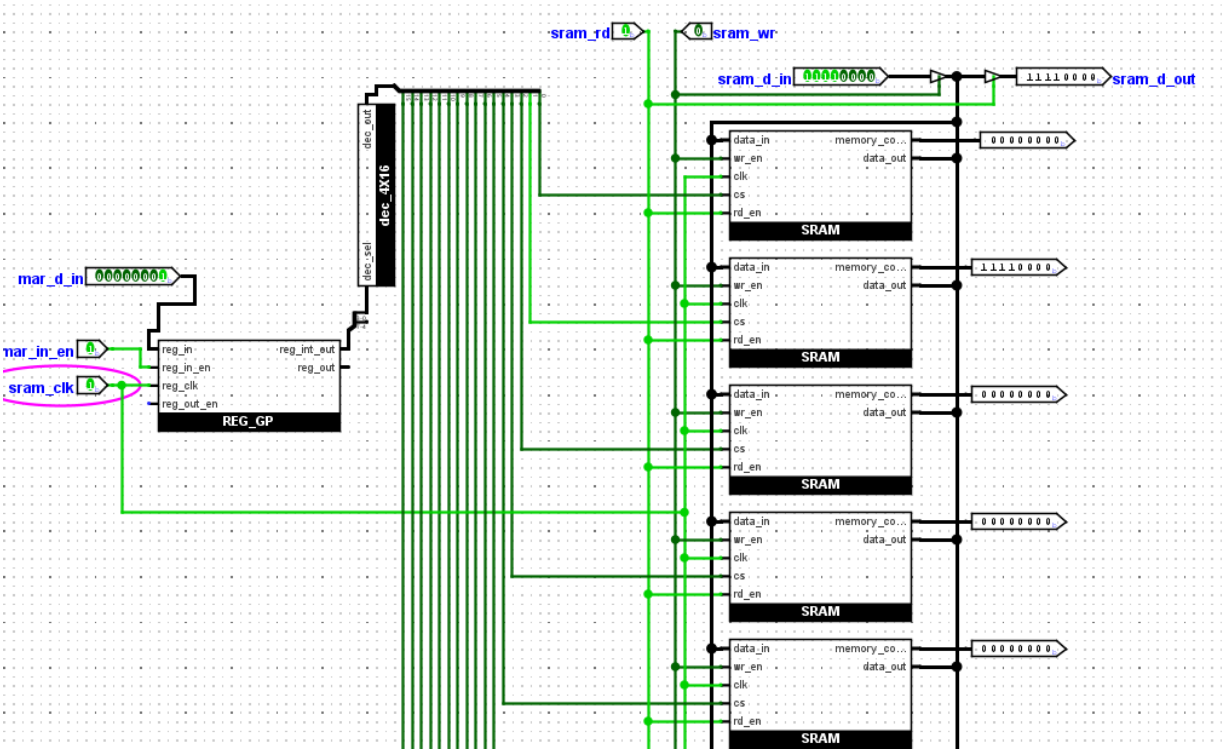

Figure 3.3.4: Testing the RAM during read operation.

3.4 Instruction Register:

The instruction register contains the instruction that is about to be executed. It divides the BUS data into two 4-bit segments: the upper 4 bits are the opcode, while the lower 4 bits are the operands or addresses.This division enables the CPU to decode and execute instructions efficiently. The process includes fetching the instruction from memory, decoding it to identify the operation and operand/address, and then executing the operation.

3.4.1 Diagram:

The circuit appearance of instruction register is given below:

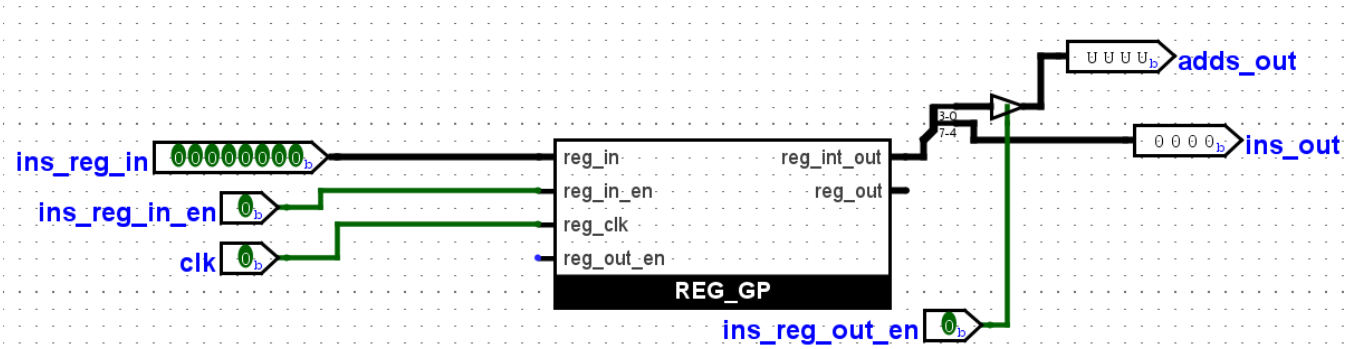

Figure 3.4.1: Circuit Appearance of Instruction Register.

3.4.2 Behaviour Analysis:

- Input to the instruction register was given as higher 4 bits as opcode, lower 4 bits as operand.
- With input enable and a clock pulse, the data was saved.
- When output enable was active, the output was visible.
- The lower 4 bits specified the memory address to be accessed.

The functionality of a 4X16 decoder can be tested with the following combination:

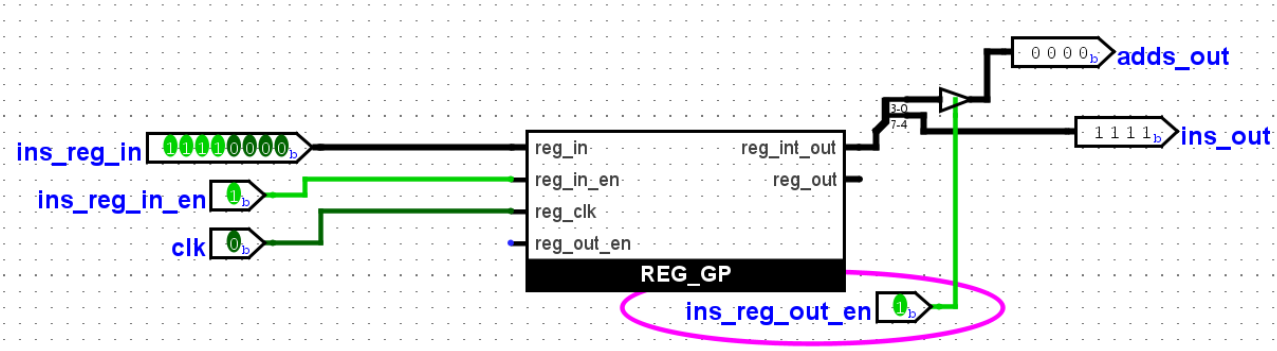

Figure 3.4.2: Testing the Instruction Register.

4 Final Layout of SAP Architecture:

At this stage, we had assembled all parts of the processor as shown in figure 3.5.1. Various parts of the processor had been connected through the central BUS. The control pins of all the subcircuits had been tunneled to a unified location for ease of manipulation. A debug control and debug data pin had also been added to the bus to program the RAM. At this stage, we needed one final component: the control sequencer or control unit. This control unit controlled the control BUS to manipulate the various control pins to ensure proper operation.

4.1 Diagram:

The schematic of the SAP Architecture can be seen in the following diagram:

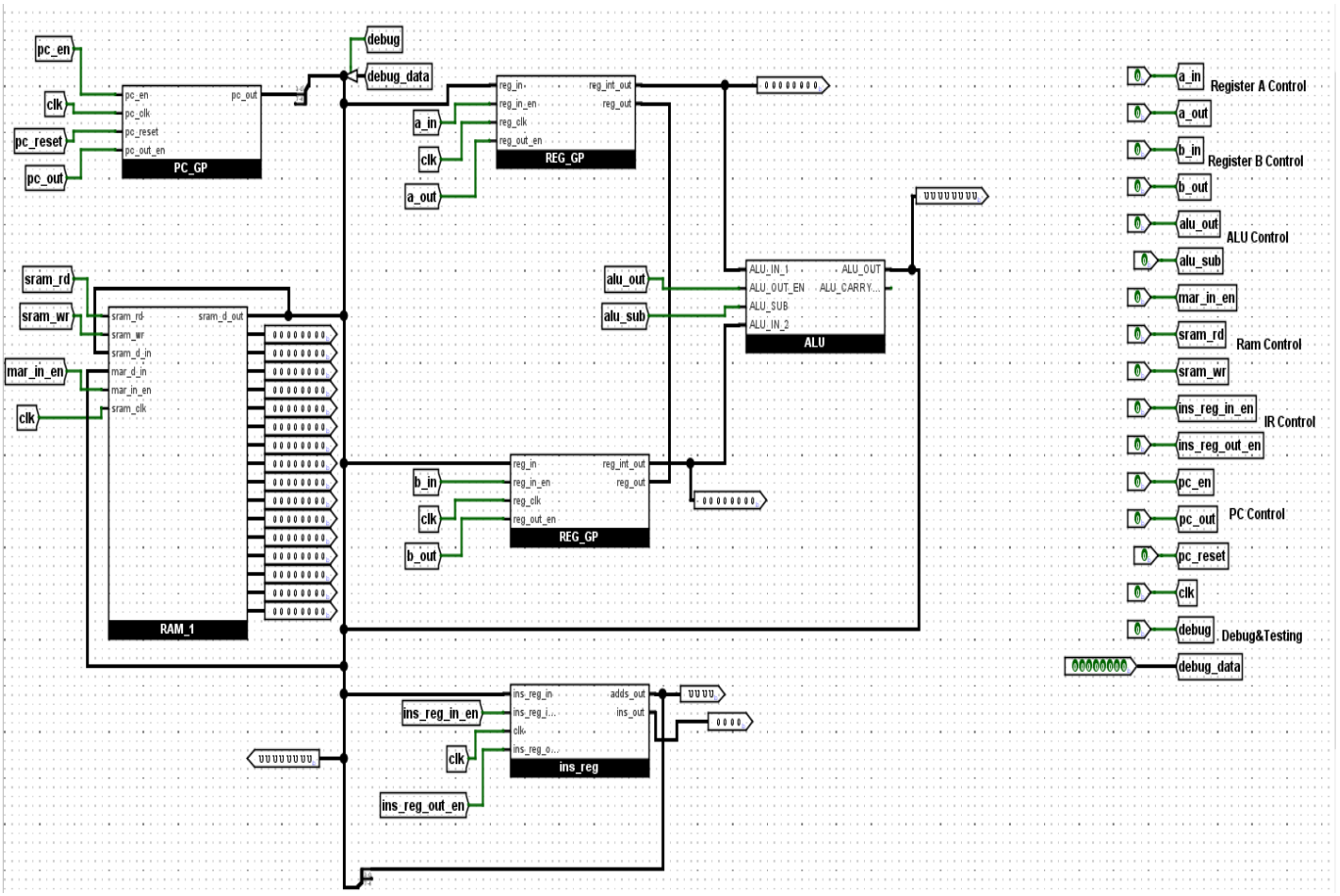

Figure 4.1.1: Final Schematic layout of SAP Architecture.

4.2 Behaviour Analysis:

4.2.1 Load Data into Register A: LDA10

To load register A with the memory contents of location 10, use the instruction with Opcode 0001 (load A) and operand 1010 (location 10). This instruction is stored in the first RAM location, and the value to be moved to register A is placed in memory location 10.

Programming Steps:

- Initiating the program by activating the debug pin and resetting the program counter. With the introduction of debug data 00000000, the bus was promptly loaded. Subsequently, activating ma_in_en and clock pulse, facilitated the loading of data onto the memory address register, followed by its deactivation. So,the intended operation to be executed at address 0000 0000.
- Setting the debug_data to 0001 1010, the subsequent action involved toggling the sram_wr pin and providing a clock pulse. Following this, the sram_wr pin was deactivated, allowing for the observation of the data stored in address 0000.
- To identity the location10 ,the debug_data was set to 0000 1010, the next step entailed toggling the mar_in_en pin and delivering a clock pulse. Upon completion of this action, the mar_in_en pin was promptly deactivated.
- Adjusting the debug_data to 0000 0111, the subsequent action involved toggling the sram_wr pin and providing a clock pulse. Following this, the sram_wr pin was deactivated, allowing for the observation of the data stored in address 1010. Finally, the debug pin was turned off, marking the completion of the programming sequence.

Fetch:

Before proceeding, the program counter was reset to 0000. During the fetch stage, which comprised three T-states, each followed by a clock pulse.

- **T1:** The pc_out and mar_in_en pins were toggled, and a clock pulse was provided, allowing the address of the next instruction to be executed to be sent from the PC to the MAR.
- **T2:** The sram_rd and ins_reg_in_en pins were toggled, and a clock pulse was delivered, ensuring that the instruction register (IR) was loaded with the opcode and operand.
- **T3:** the pc_en pin was toggled, and a clock pulse was provided, resulting in the incrementation of the counter's value to 0001.After each T-state, the control pins were promptly turned off.

Decode:

This state had no T states or clock pulse were provided as it was a pure combinational circuit.So,only combinational logic were executed here.

Excute:

- **T1:** the address to be fetched into the MAR was saved by toggling ins_reg_out_en and mar_in_en, followed by delivering a clock pulse. The subsequent deactivation of the control pins ensured the completion of this action.
- **T2:** The memory contents stored at address 1010, which were 0111, were read and saved in register A by toggling sram_rd and a_in, along with providing a clock pulse. The control pins were then turned off to conclude this stage.
- **T3:**T3 had remained unused for the LDA operation.

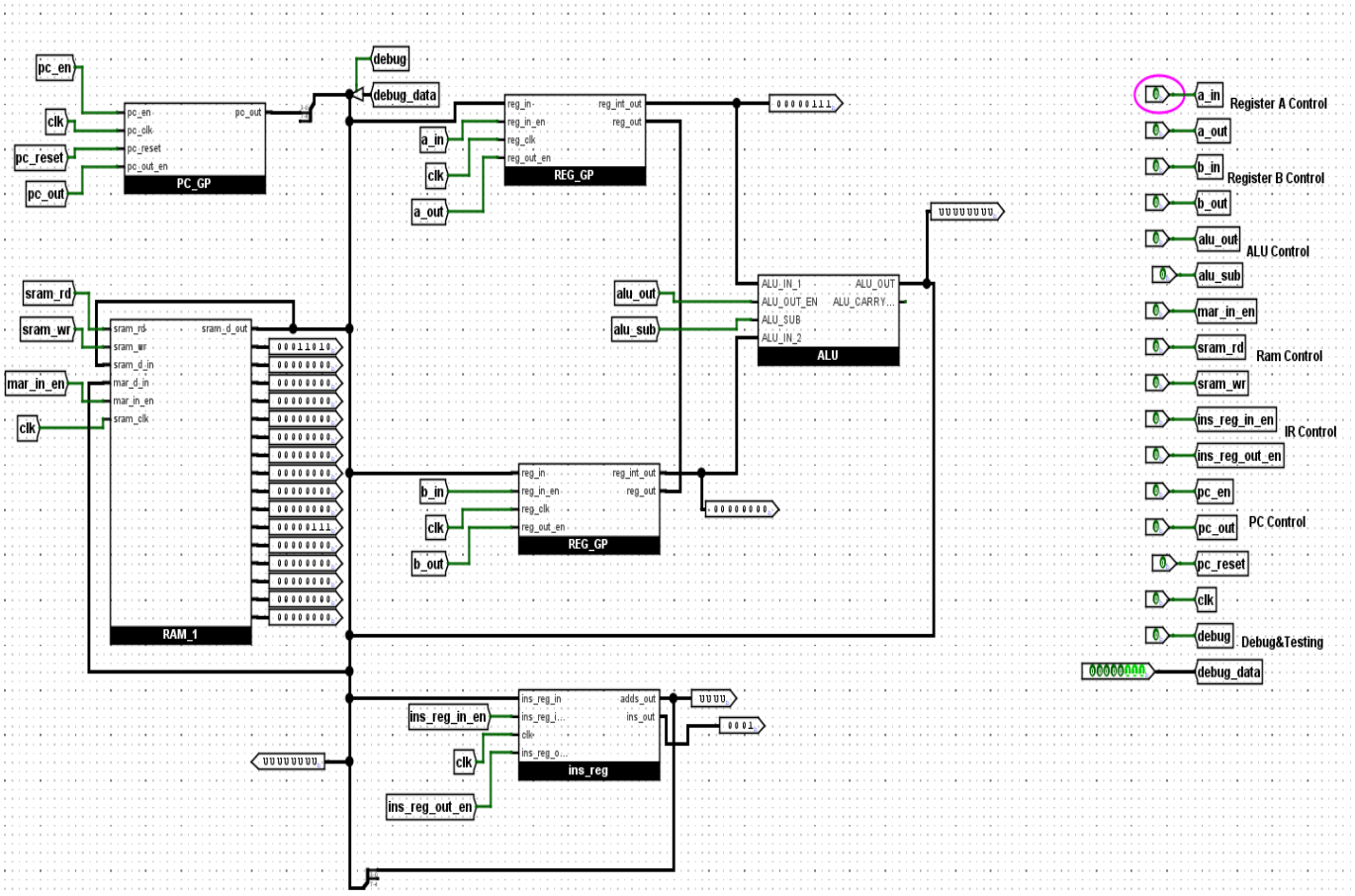

Figure 4.1.2: Testing the SAP Architecture during Load Data into Register A.

4.2.2 Load Data into Register B: LDA12

To load register A with the memory contents of location 12, use the instruction with Opcode 0001 (load B) and operand 1100 (location 12). This instruction is stored in the first RAM location, and the value to be moved to register B is placed in memory location 12.

Programming Steps:

- Initiating the program by activating the debug pin and resetting the program counter. With the introduction of debug data 00000000, the bus was promptly loaded. Subsequently, activating ma_in_en and clock pulse, facilitated the loading of data onto the memory address register, followed by its deactivation. So,the intended operation to be executed at address 0000 0000.
- Setting the debug_data to 0001 1100, the subsequent action involved toggling the sram_wr pin and providing a clock pulse. Following this, the sram_wr pin was deactivated, allowing for the observation of the data stored in address 0000.
- To identity the location12 ,the debug_data was set to 0000 1100, the next step entailed toggling the mar_in_en pin and delivering a clock pulse. Upon completion of this action, the mar_in_en pin was promptly deactivated.
- Adjusting the debug_data to 0000 1000, the subsequent action involved toggling the sram_wr pin and providing a clock pulse. Following this, the sram_wr pin was deactivated, allowing for the observation of the data stored in address 1100. Finally, the debug pin was turned off, marking the completion of the programming sequence.

Fetch:

Before proceeding, the program counter was reset to 0000. During the fetch stage, which comprised three T-states, each followed by a clock pulse.

- **T1:** The pc_out and mar_in_en pins were toggled, and a clock pulse was provided, allowing the address of the next instruction to be executed to be sent from the PC to the MAR.
- **T2:** The sram_rd and ins_reg_in_en pins were toggled, and a clock pulse was delivered, ensuring that the instruction register (IR) was loaded with the opcode and operand.
- **T3:** the pc_en pin was toggled, and a clock pulse was provided, resulting in the incrementation of the counter's value to 0001.After each T-state, the control pins were promptly turned off.

Decode:

This state had no T states or clock pulse were provided as it was a pure combinational circuit.So,only combinational logic were executed here.

Excute:

- **T1:** The address to be fetched into the MAR was saved by toggling ins_reg_out_en and mar_in_en, followed by delivering a clock pulse. The subsequent deactivation of the control pins ensured the completion of this action.
- **T2:** The memory contents stored at address 1100, which were 1000, were read and saved in register A by toggling sram_rd and a_in, along with providing a clock pulse. The control pins were then turned off to conclude this stage.
- **T3:**T3 had remained unused for the LDA operation.

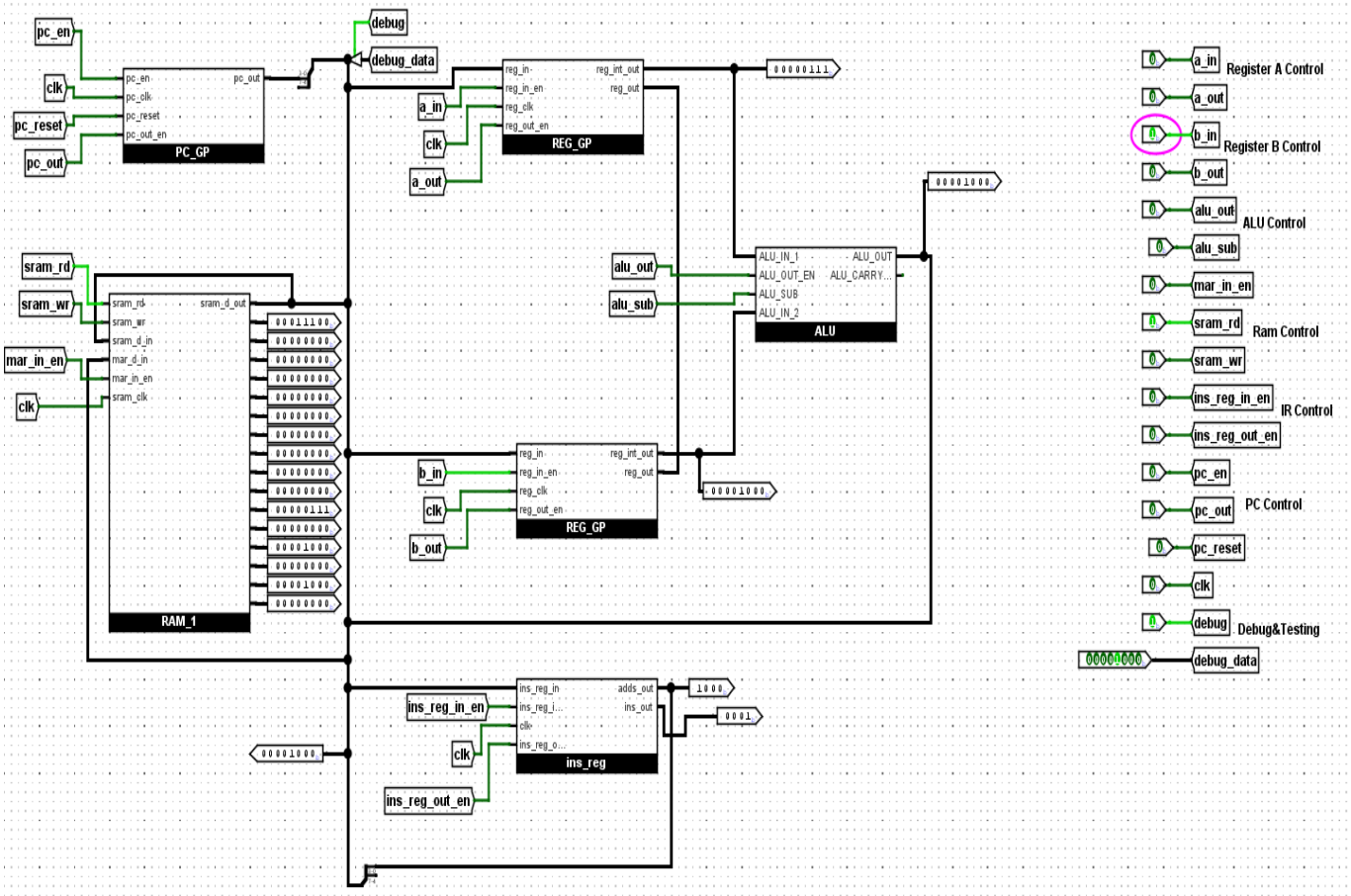

Figure 4.1.3: Testing the SAP Architecture during Load Data into Register B.

4.2.3 Load ALU Result into RAM: LDA07

The debug_data had been set to 0000 0111, and mar_in_en was toggled, followed by a clock pulse, selecting the 7th (0111) location in RAM. After this, mar_in_en was turned off, and the debug pin was deactivated. Subsequently, sram_wr and alu_out were toggled, and a clock pulse was provided. Finally, alu_out and sram_wr were turned off, successfully saving the result of ALU(00001111) in the 7th location(00000111) of RAM.

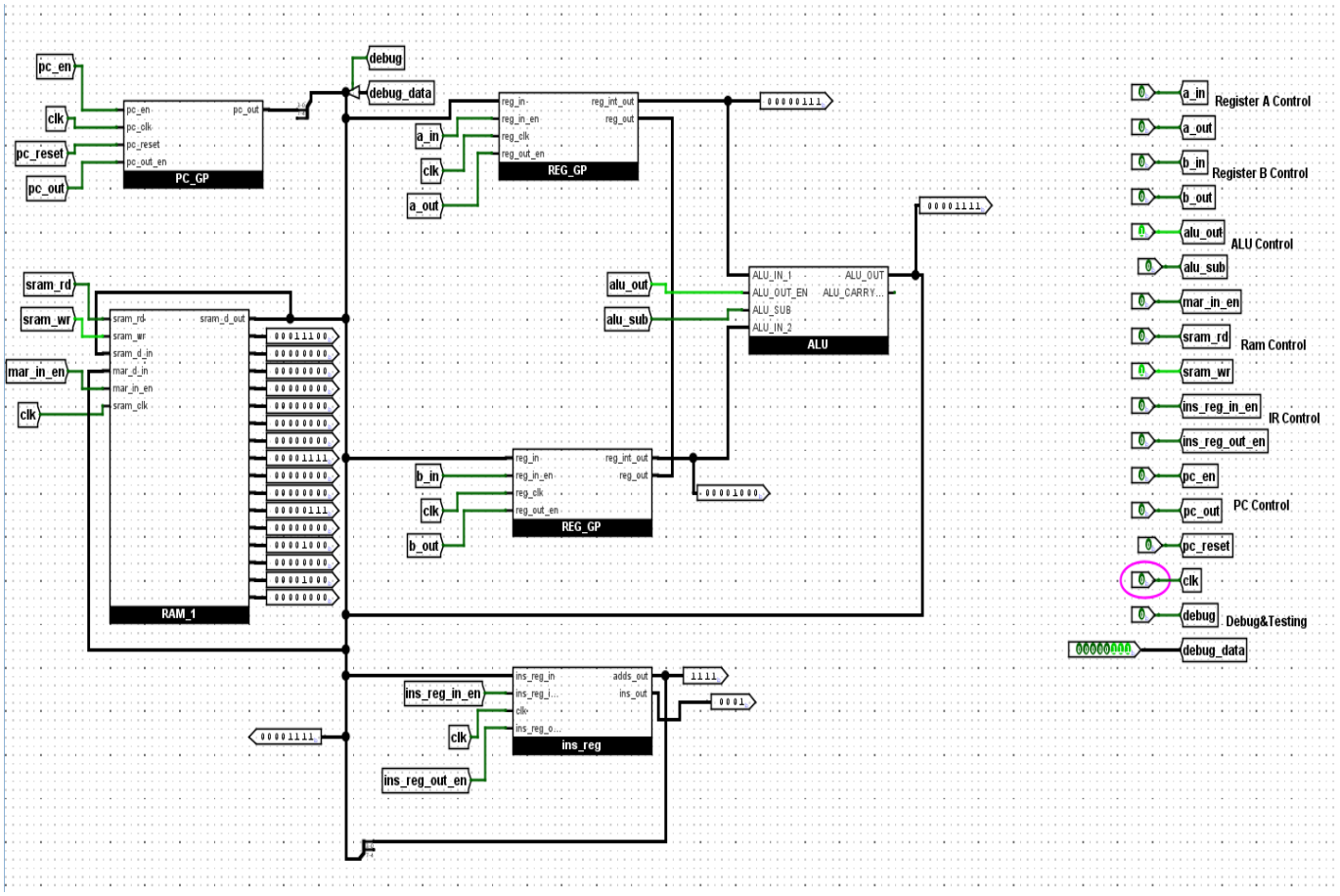

Figure 4.1.4: Testing the SAP Architecture during Load ALU result into RAM.

5 Discussion:

- In this experiment, the schematic modeling and implementation of the modified SAP architecture had been carried out in Logisim .
- Testing procedures had been provided to verify proper operation of the components in the SAP-1 architecture.
- Decoder, SRAM cell, RAM , Instruction Register and CPU were designed to carry out the final layout of the SAP architecture.
- The 16 output lines of the 4x16 decoder had been used to access the 16 different addresses in RAM.
- Furthermore,RAM was used to store data and IR was used to temporarily holds the currently being executed instruction.
- Various parts of the processor were connected through the help of the central BUS.
- Output and input were adjusted with required number of bits.
- The control pins of all the subcircuits were tunneled to a unified location for the ease of manipulations.
- Moreover,The control unit controlled the control BUS to manipulate the various control pins to ensure proper operation.
- All components had same clock pulse for better synchronization and proceeding for next steps previous control pins were turned off.
- 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 the BUS in a microprocessor system, addressing and RAM, and the program execution cycle of the SAP-1 architecture through the utilization of Logisim-evolution software.

References

[1] wikipedia, "Instruction Cycle,[Online].
Available: https://en.wikipedia.org/wiki/Instruction_cycle.

[2] FutureLearn, "Fetch-Decode-Execute Cycle," [Online].
Available: https://www.futurelearn.com/info/courses/how-computers-work/0/steps/49284.