CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY

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:

M.I.Yasir Arafat

1908031

Submitted to:

Arif Istiaque Rupom

LECTURER

DEPARTMENT OF ETE, CUET

April 21, 2024

Objective

- 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.

Design Process

General purpose register:

The General-Purpose Register (GPR) serves as a pivotal component in the architecture of digital systems, offering a versatile and high-speed storage solution for temporary data manipulation. Its design incorporates several key components, each contributing to its functionality and efficiency within the broader VLSI (Very Large Scale Integration) context. The design components of General Purpose register are:

- Storage Elements: The heart of the General-Purpose Register (GPR) comprises flip-flops or latches, storing binary data bits.
- Multiplexers: Multiplexers facilitate selective access to register bits, connecting the GPR to the data bus for efficient read and write operations.
- Decoder: Decoders interpret control signals, activating the appropriate multiplexers for data transfer.
- Clocking Mechanism: Clock signals synchronize read and write operations, ensuring proper timing and data integrity.
- Control Logic: Control logic orchestrates GPR behavior, generating signals to manage data flow and timing.

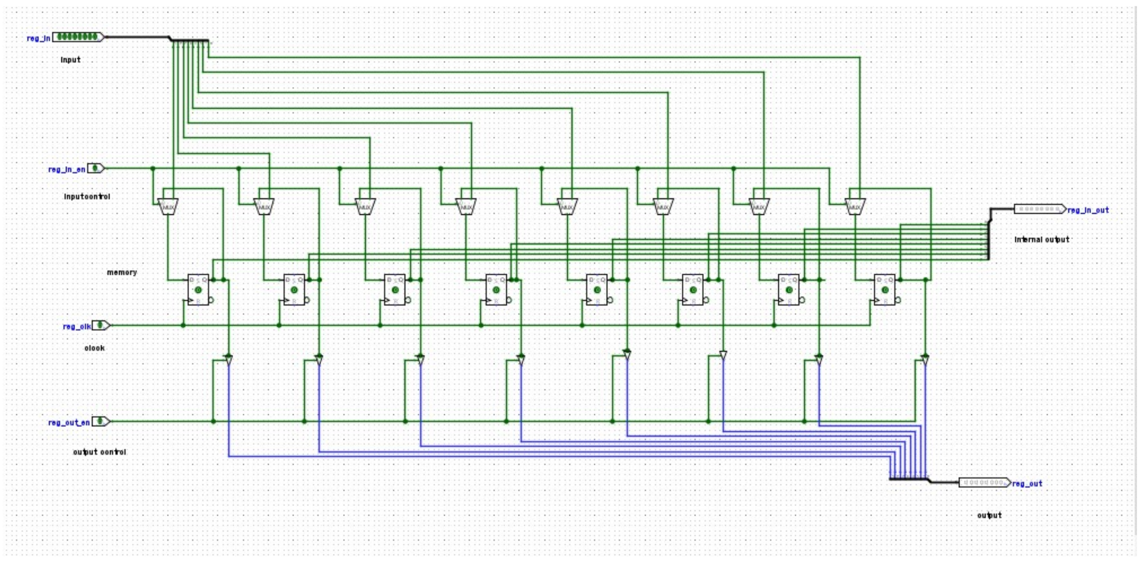

Figure 1: Schematic of general-purpose Register

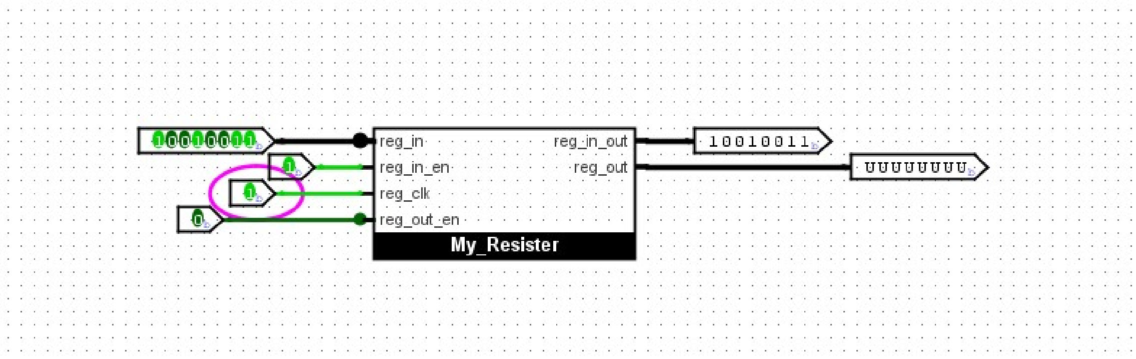

Figure 2: Testing the general-purpose Register

Figure 2 shows the testing of the general-purpose resister which is designed. It only takes input when the input resister enable (reg-i-en) is given followed by a clock pulse.

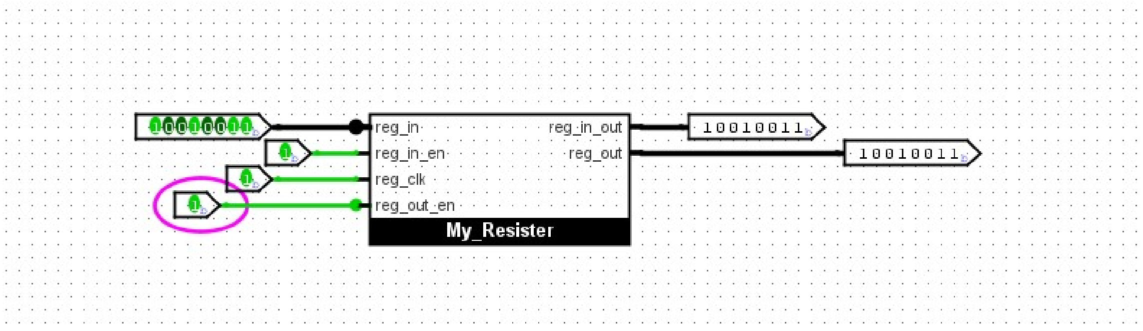

Figure 3: Example value saved in the Register

Figure 3 shows the output result of value saved in the register when register out enable (reg-out-en) is given.

Arithmetic Logic Unit:

The Arithmetic Logic Unit (ALU) is a fundamental component of the central processing unit (CPU) in a computer. It is responsible for executing arithmetic and logic operations on binary data, essential for carrying out computations and decision-making tasks. The design of an ALU involves several key components, each playing a vital role in its functionality and performance within the CPU architecture. The design components of ALU are:

- Arithmetic Operations: The ALU executes basic arithmetic functions like addition, subtraction, multiplication, and division using dedicated circuits.
- Logic Operations: It performs logical operations such as AND, OR, NOT, XOR, and shifts, applying Boolean logic rules to binary data.
- Registers: Registers store operands and intermediate results during computation, ensuring efficient data handling within the ALU.
- Multiplexers: Multiplexers enable the selection of operands and operation modes, providing flexibility in executing various instructions.
- Flags: Flags indicate specific conditions resulting from operations, aiding in conditional branching and decision-making.
- Data Paths: Data paths facilitate the movement of data between ALU components, ensuring efficient data flow during computation.

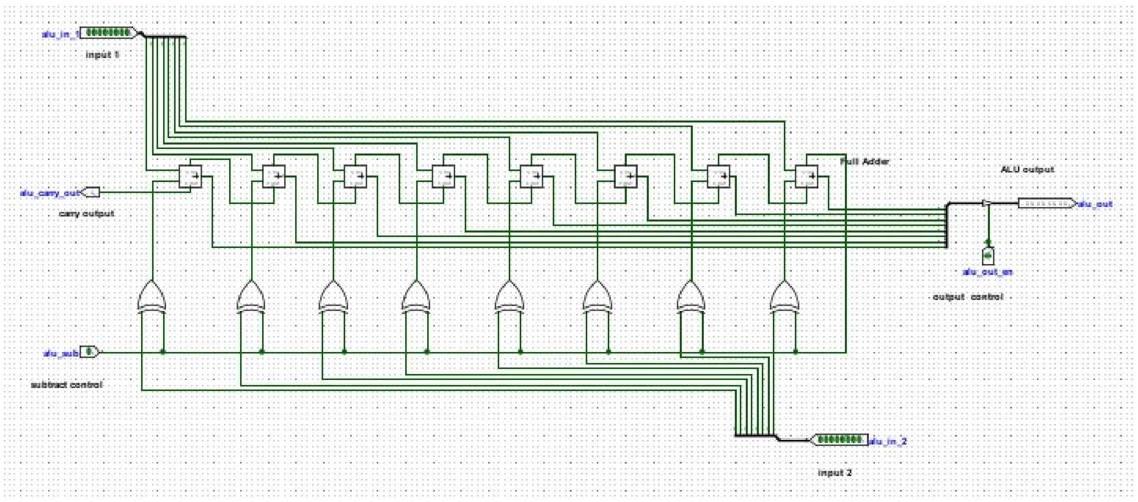

Figure 4: Schematic of ALU

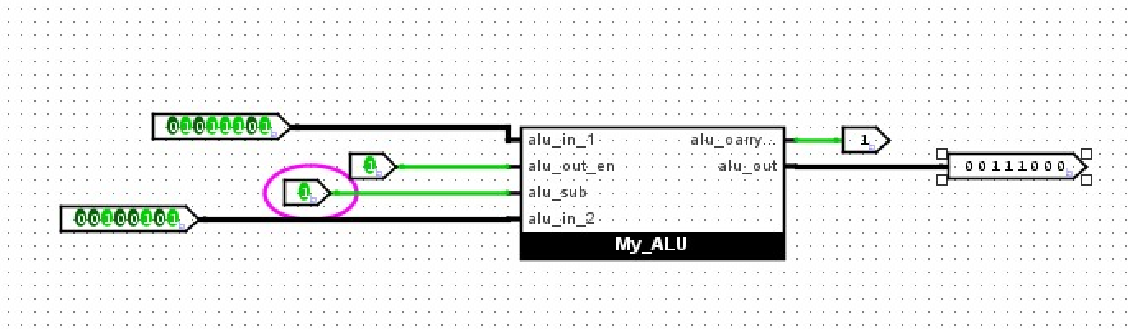

Figure 5: testing the ALU

Figure 5 shows the testing of the ALU which is designed. It only takes input when the inputs are enable (alu-in-1 , alu-in-2) is given followed by a clock pulse.

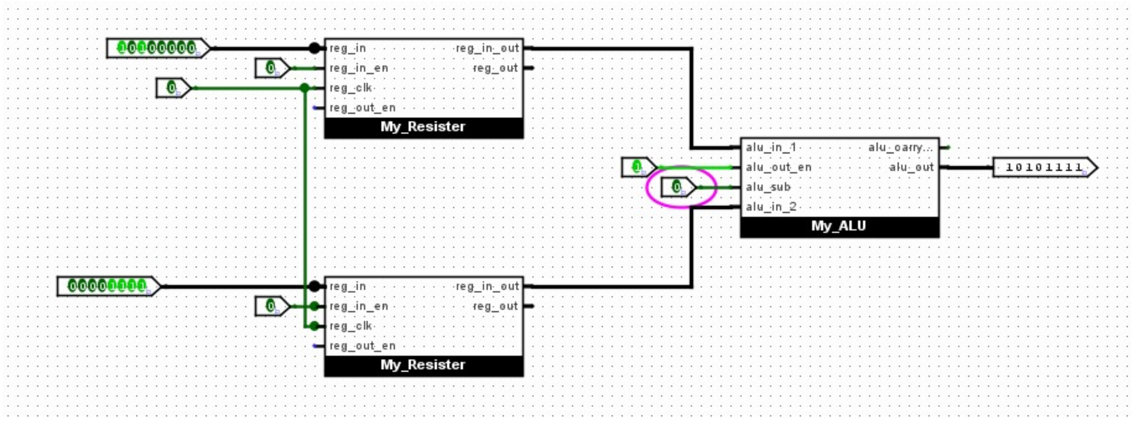

Figure 6: Example value showed in the ALU output

Figure 6 shows two 8-bit inputs are given to the ALU, the output is seen when alu-out-en is given. The output is added when alu-sub is 0 and subtracted when it is 1. This is verified that ALU is working properly in the manner.

Program Counter:

The Program Counter (PC) is a vital component within the central processing unit (CPU) of a computer system. It serves as a register that stores the memory address of the next instruction to be fetched and executed by the CPU. The design of a program counter involves several key components, each playing a crucial role in its functionality and operation within the CPU architecture.

- Storage Element: The Program Counter (PC) comprises a storage element, typically a flip-flop, holding the memory address of the next instruction to be executed.
- Incrementer: An incrementer circuit adds a fixed value to the current memory address, pointing to the location of the next instruction in memory during the instruction fetch phase.
- Control Logic: Control logic coordinates the operation of the program counter, generating signals for incrementing the PC and managing the instruction fetch cycle.
- Reset Mechanism: A reset mechanism initializes the PC's value at program start or in response to system events, ensuring it begins from a known initial address.

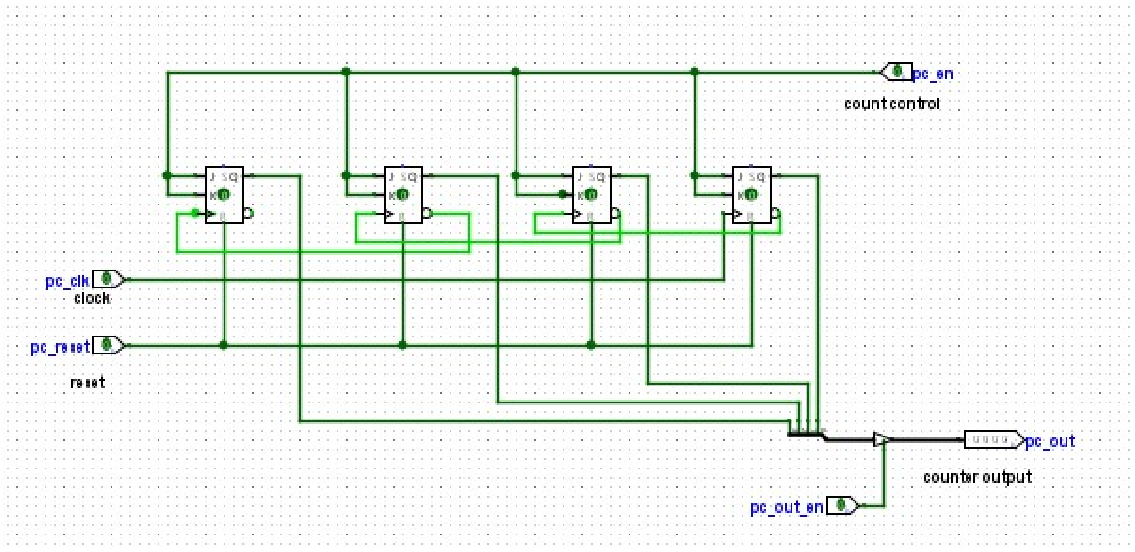

Figure 7: Schematic of PC

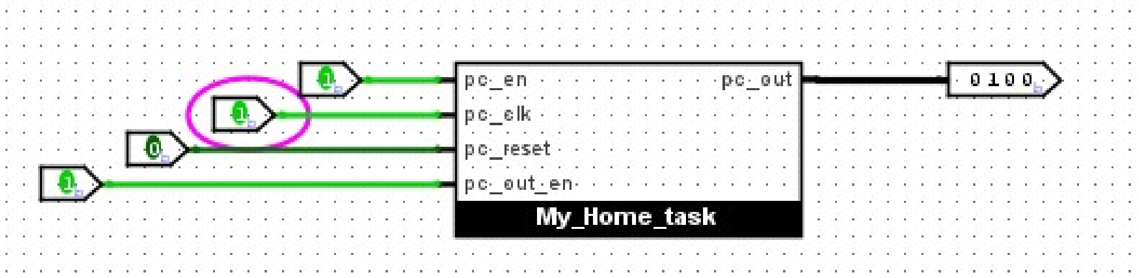

Figure 8: Testing the PC

Figure 8 shows the testing of the PC which is designed. It only takes input when the enables and clock pulse are given. every time the clock pulse changed its count increments by one. The output is displayed when the pc-out-en is given.

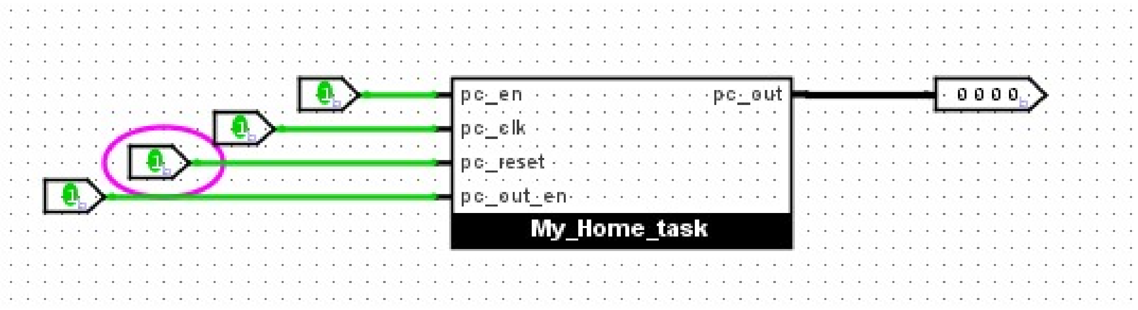

Figure 9: testing the PC with reset function

Figure 9 shows when pc-reset is given output is reset to the initial value O. This verified that the PC is working properly in the manner.

Discussion

The design considerations for key CPU components the General-Purpose Register (GPR), Arithmetic Logic Unit (ALU), and Program Counter (PC) are crucial for optimizing performance, efficiency, and functionality in computing systems. For GPRs, balancing capacity, access time, and power consumption is paramount, influencing register size and number to meet performance requirements while minimizing energy usage. ALU design focuses on supporting diverse operations with precision and maximizing throughput through pipelining techniques, all while maintaining power efficiency through careful circuit design and management. Similarly, PC design must efficiently handle both sequential and branch instructions, incorporate error detection and recovery mechanisms, and accommodate varying memory configurations to ensure reliable program execution and data integrity. By carefully considering these design factors, CPU architects can create systems that meet the demanding requirements of modern computing applications while achieving optimal performance and efficiency.