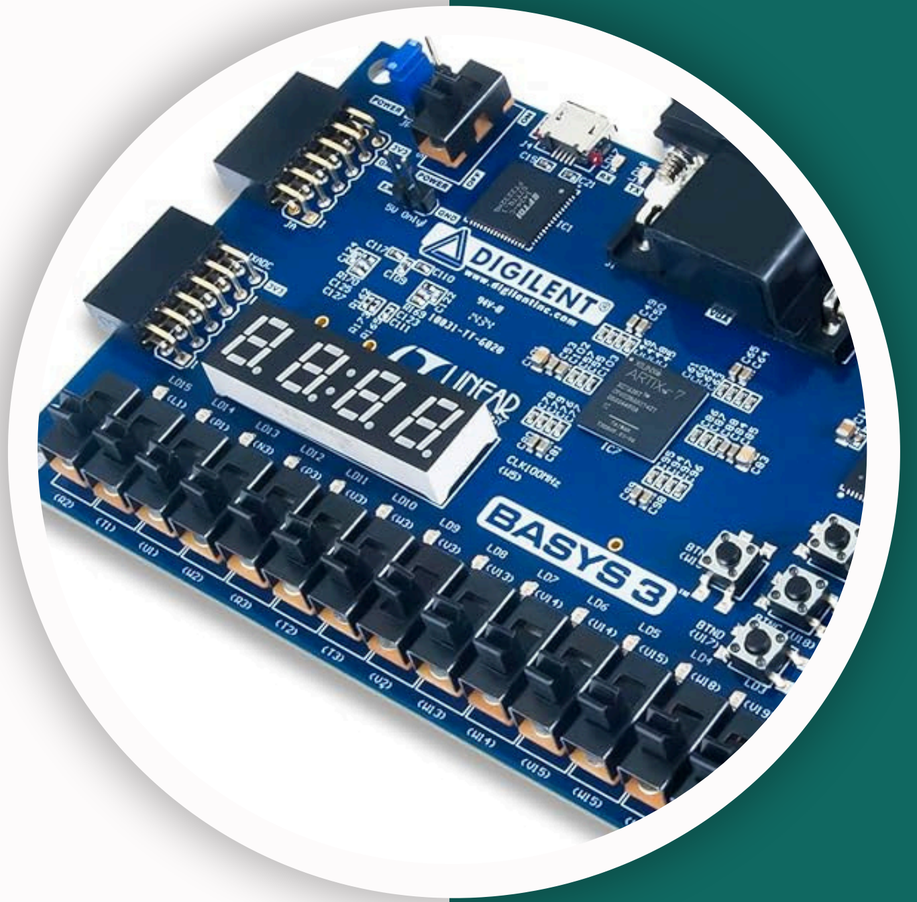
Lab 9 - 10

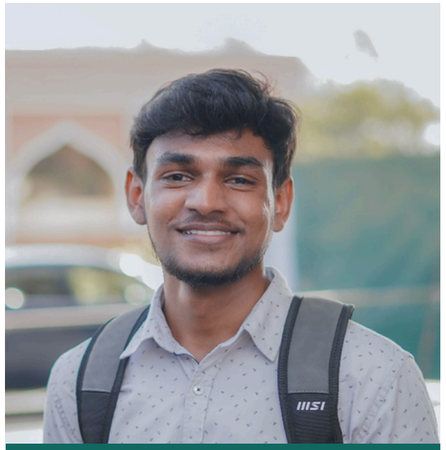
Nano Processor Design Competition

Group 24

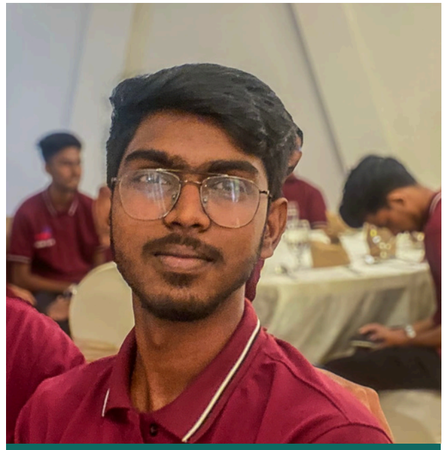


CS1050 - Computer Organization and Digital Design

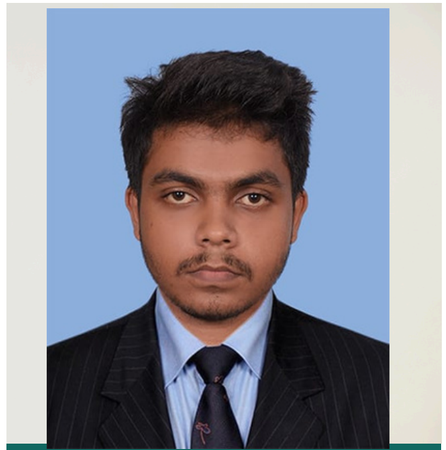
Our Team



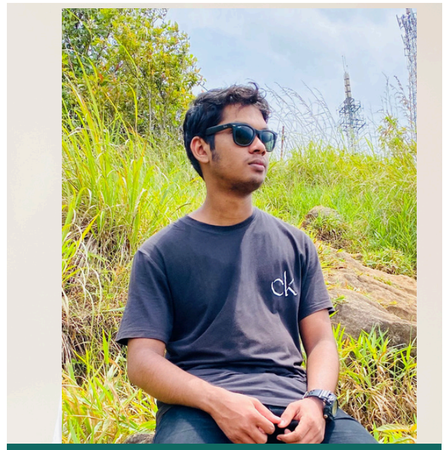
MADAELA
M.P.G.R.V.M
230381X



KESHARA
K.P.W.D
230334H



SOMARATHNA
M.D.A.M
230621K



SENANAYAKE
H.P.V.R
230595G

Introduction

The objective of this lab was to design a 4-bit Nano processor capable of executing basic instructions on a basys 3 board using components developed in previous labs.

4-Bit Nano Processor

Minimal Design

- The Minimal Design implements the core required instructions: ADD, SUB (via NEG), MOVI, and JZR, enabling essential arithmetic and control operations for basic processor functionality.

Extended Design

- The Extended Design adds support for AND, OR, and MUL operations, along with a 4-bit comparator, and includes enhancements such as tri-state buffer-based multiplexers and an improved ALU for expanded functionality.

Minimal Design

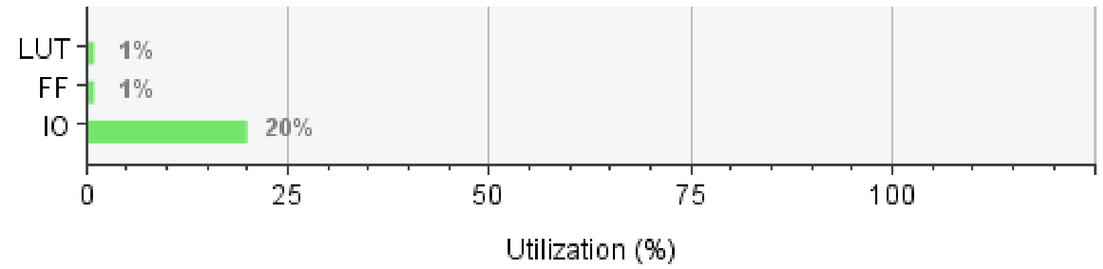
Features

- 4-bit signed computation using 2's complement
- Program Counter reset via a switch
- LEDs display Register R7 output
- 7-Segment Display shows Register R7 value
- Flag LEDs: Sign, Carry, Overflow, Zero

Resource Utilization

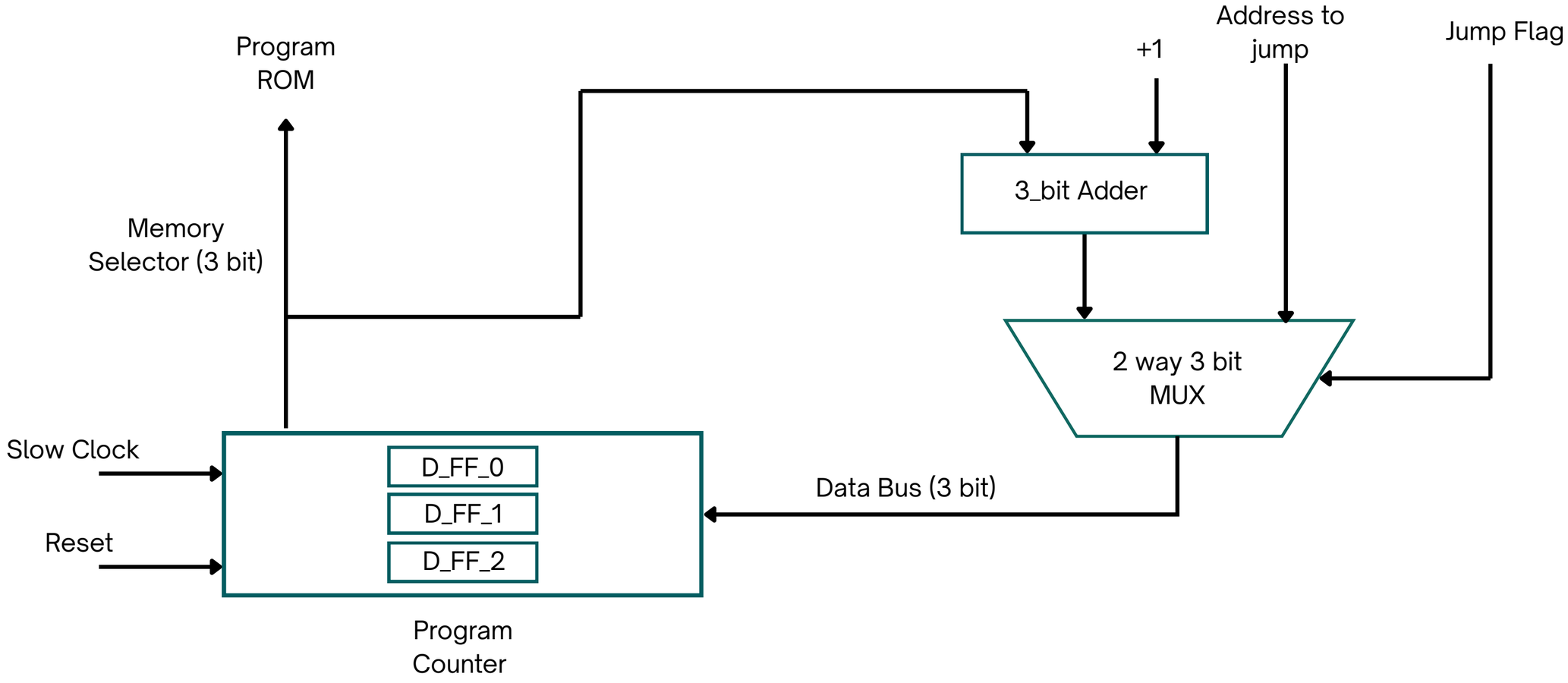
The analysis shows efficient utilization of FPGA resources, with LUTs and flip-flops consuming less than 1% and I/O resources approaching 20%, indicating potential optimization focus.

Resource	Utilization	Available	Utilization %
LUT	37	20800	0.18
FF	49	41600	0.12
IO	21	106	19.81



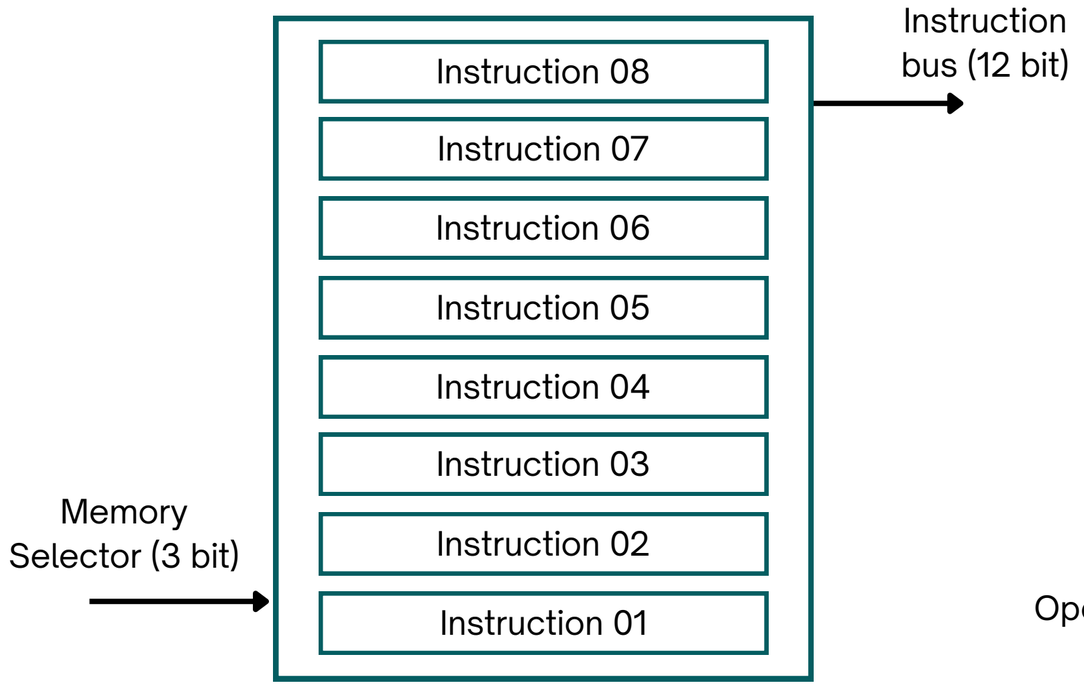
Program Counter

The program counter (PC) stores the memory address of the next instruction to be executed, dictating program flow, enabling jumps, and facilitating resets.



Program ROM

The Program ROM permanently stores 8 machine code instructions, each 12 bits in length, which are fetched for execution.



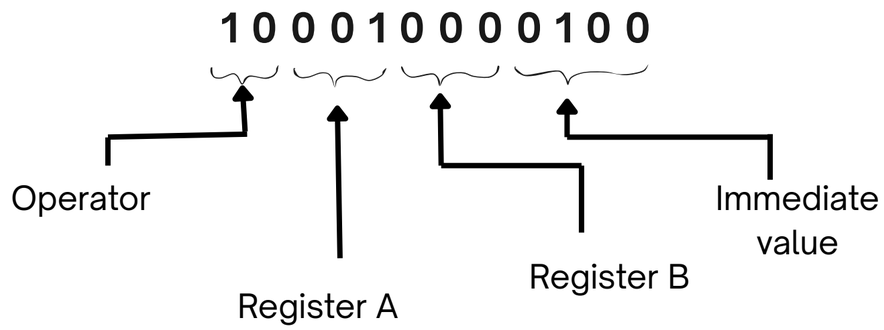
Instruction Set

MOVI R, d → Move immediate value
ADD Ra, Rb → Ra ← Ra + Rb
NEG R → 2's complement
JZR R, d → Jump if R == 0

Assembly Code

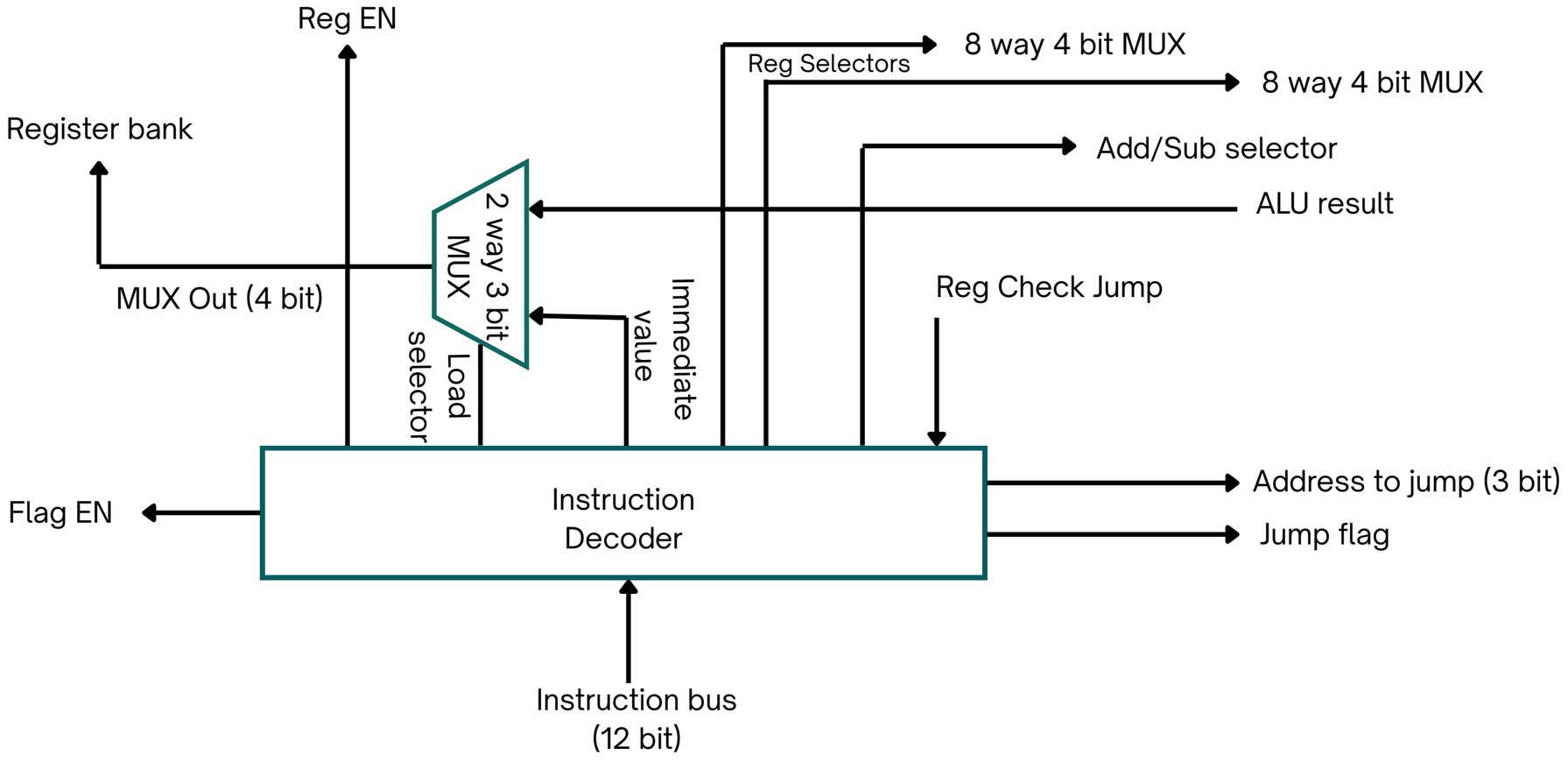
MOVI R1,4
MOVI R2,1
NEG R2
ADD R1,R2
JZR R1,7
ADD R7,R1
JZR R0,3
JZR R0,7

Machine Code



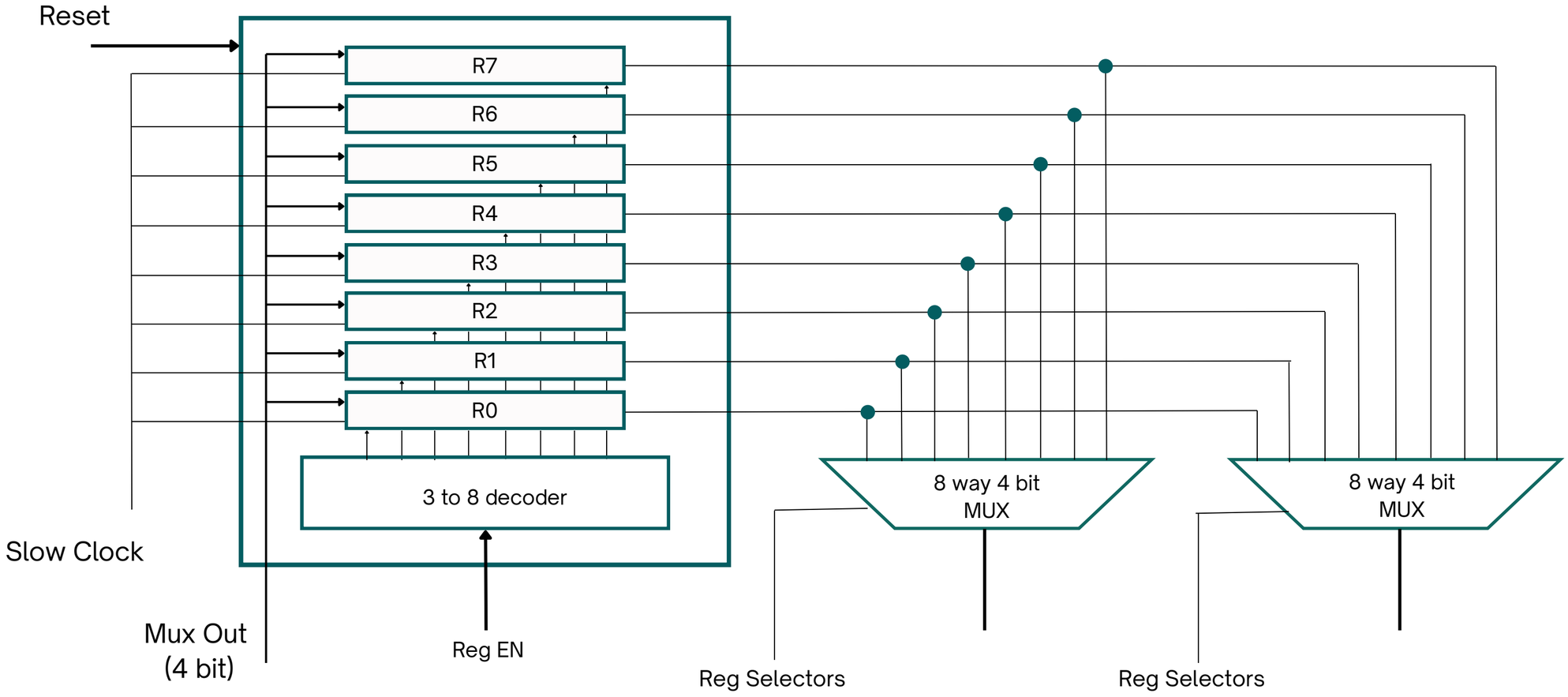
Instruction Decoder

The Instruction Decoder interprets the fetched machine code instruction to generate precise control signals that activate the necessary components within the nanoprocessor for execution.



Register Bank

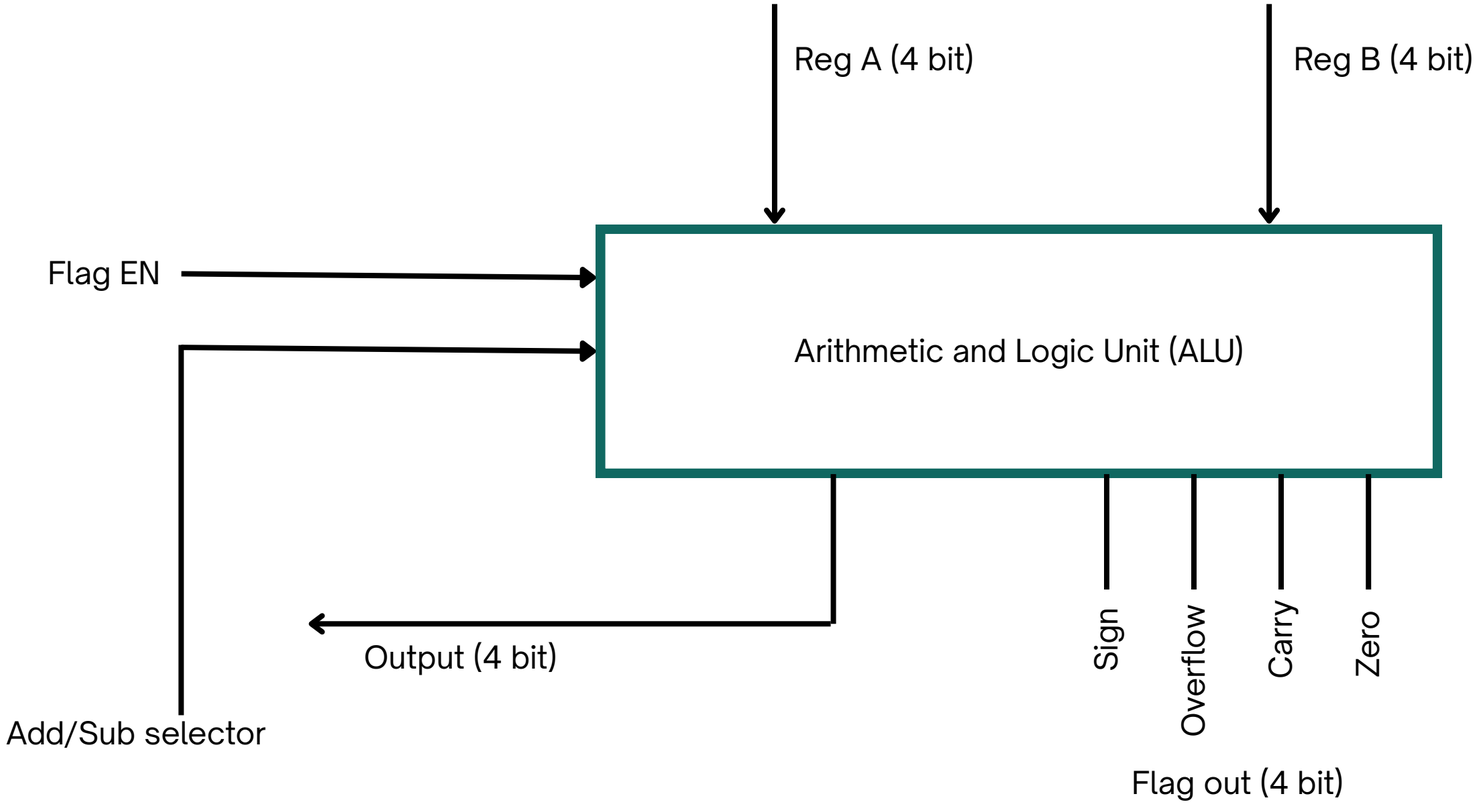
The Register Bank holds eight 4-bit registers (R0-R7), with R0 hardwired to all zeros, for data storage and quick access during processor operations.



Your paragraph text

Arithmetic and Logic Unit (ALU)

The ALU (Arithmetic Logic Unit) is responsible for executing operations such as ADD, SUB (via NEG), AND, OR, and MUL, based on control signals from the instruction decoder, and it outputs the result along with flags like Zero, Carry, Sign, and Overflow to guide program flow.



Overall Design

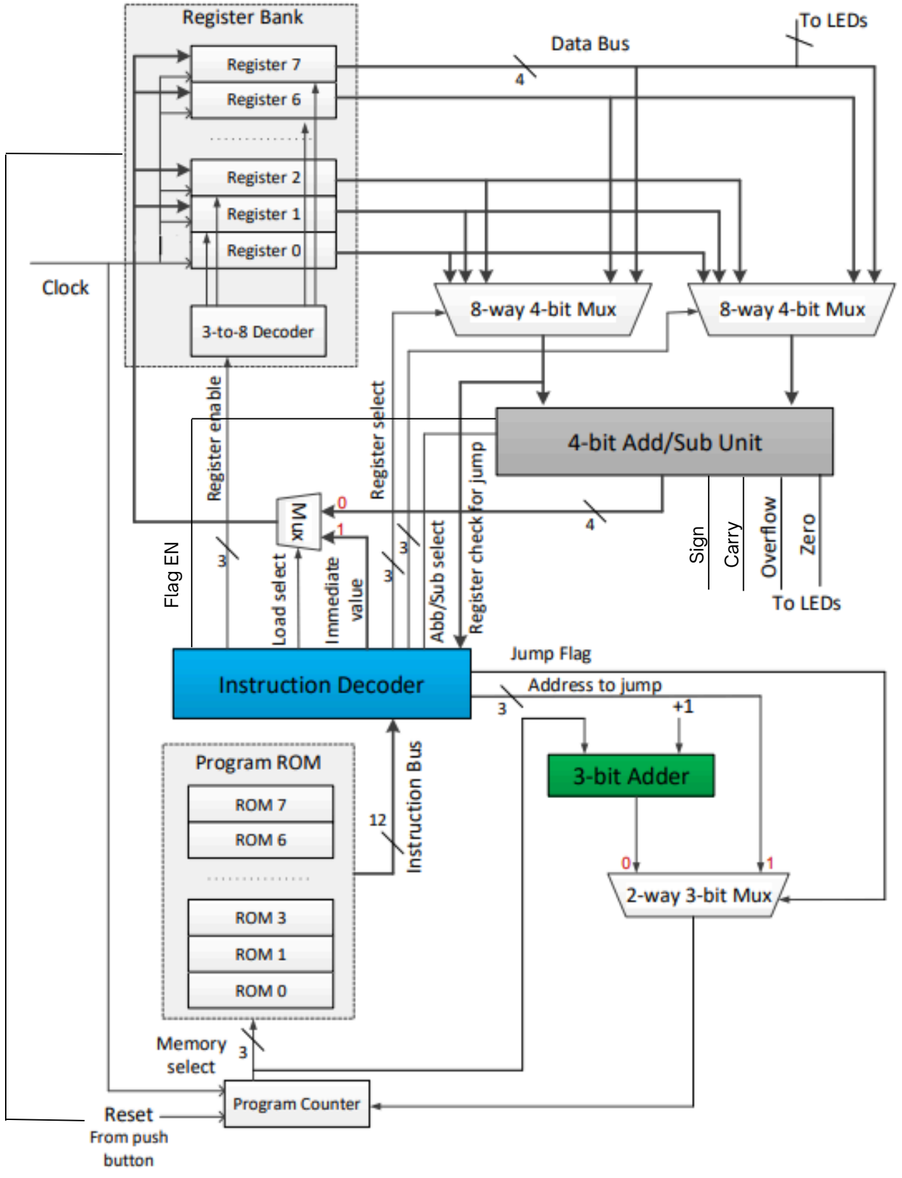


Figure 1 – High-level diagram of the nanoprocessor.

Extended Design

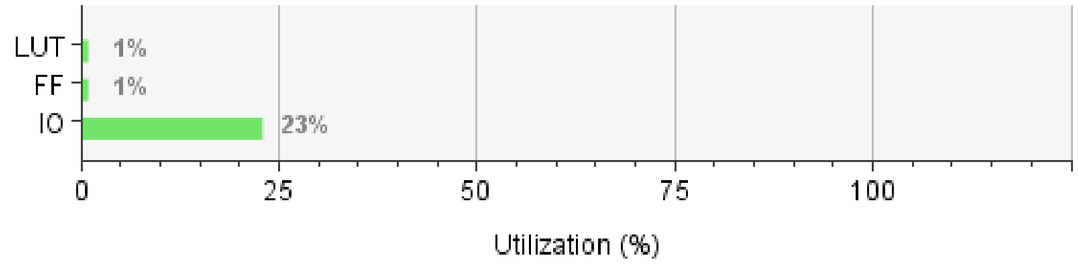
Features

- AND/NAND, OR/NOR, MUL support
- Comparator with output flags (Equal, Greater Than, Less Than)
- Tri-State Buffer-based MUXes
- Optimized ALU for multiple operations
- Extra LEDs for Comparator Flags

Resource Utilization

The analysis shows efficient utilization of FPGA resources, with LUTs and flip-flops consuming less than 1% and I/O resources consume 23%, indicating potential optimization focus.

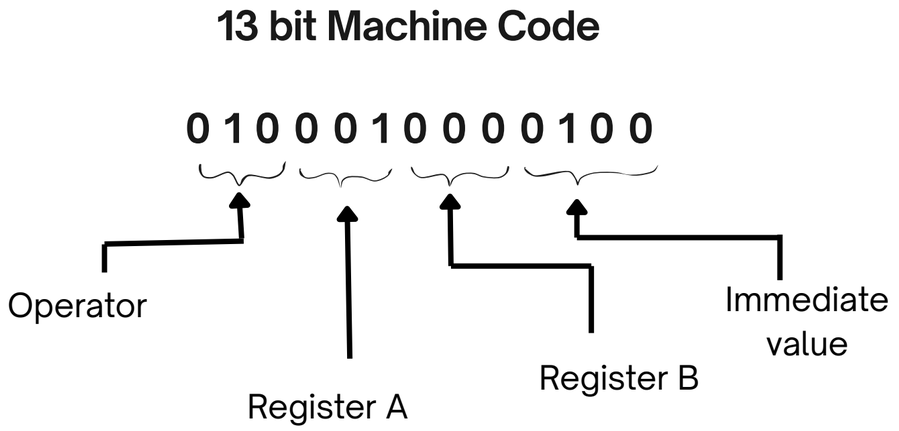
Resource	Utilization	Available	Utilization %
LUT	89	20800	0.43
FF	45	41600	0.11
IO	24	106	22.64



Instruction Decoder

The extended Instruction Decoder processes 13-bit instructions, enabling compatibility with 7 operations by accommodating three additional instructions: MUL, AND (NAND), and OR (NOR).

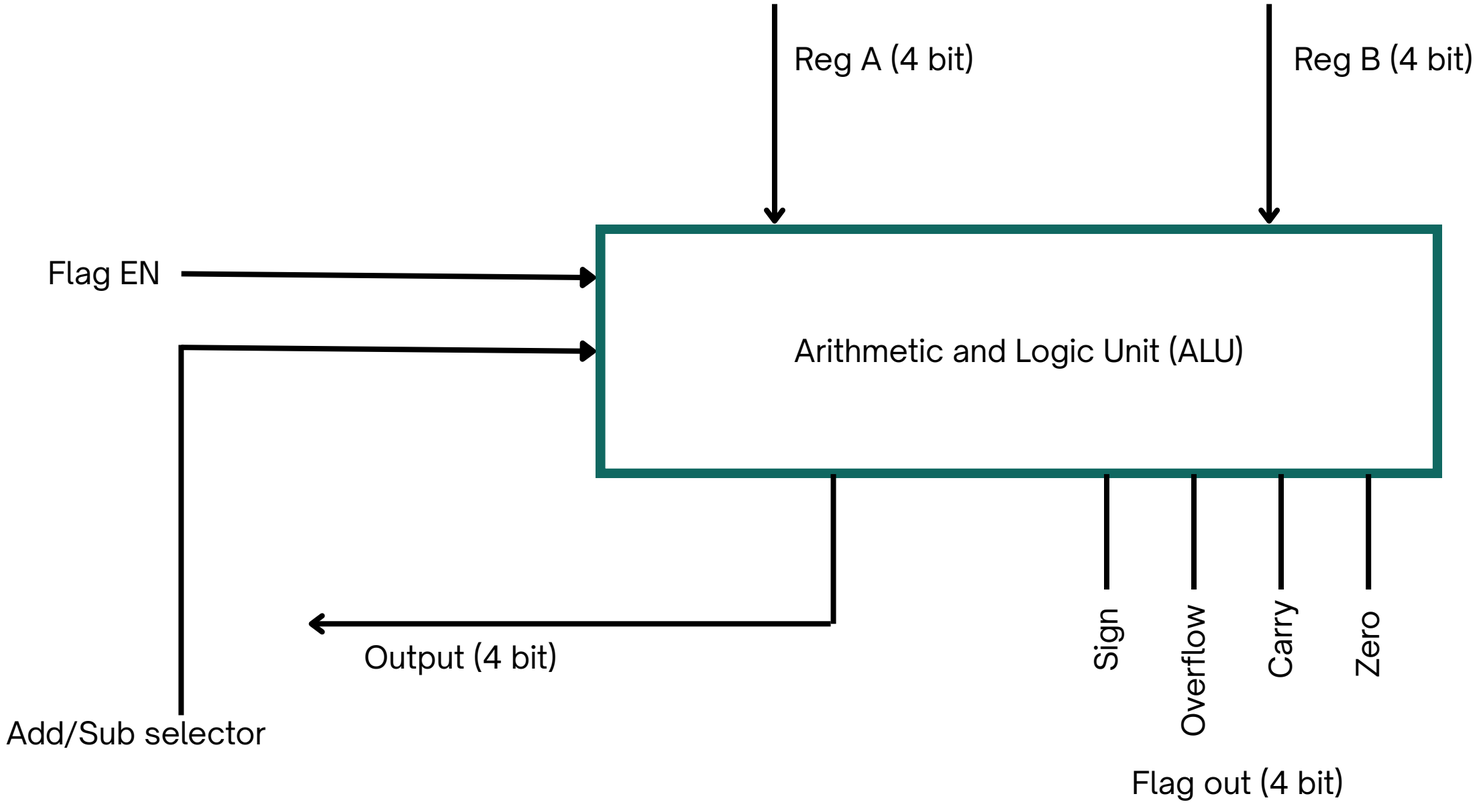
```
elsif Operator = "100" then --AND
    Comp_EN <= '1';
    Reg_Sele1 <= Instruction_Bus(9 downto 7);
    Reg_Sele2 <= Instruction_Bus(6 downto 4);
    Reg_EN <= Instruction_Bus(9 downto 7);
    Func <= "000";

elsif Operator = "101" then --OR
    Comp_EN <= '1';
    Reg_Sele1 <= Instruction_Bus(9 downto 7);
    Reg_Sele2 <= Instruction_Bus(6 downto 4);
    Reg_EN <= Instruction_Bus(9 downto 7);
    Func <= "001";

elsif Operator = "110" then --into
    Comp_EN <= '1';
    Reg_Sele1 <= Instruction_Bus(9 downto 7);
    Reg_Sele2 <= Instruction_Bus(6 downto 4);
    Reg_EN <= Instruction_Bus(9 downto 7);
    Func <= "011";
```



Your paragraph text

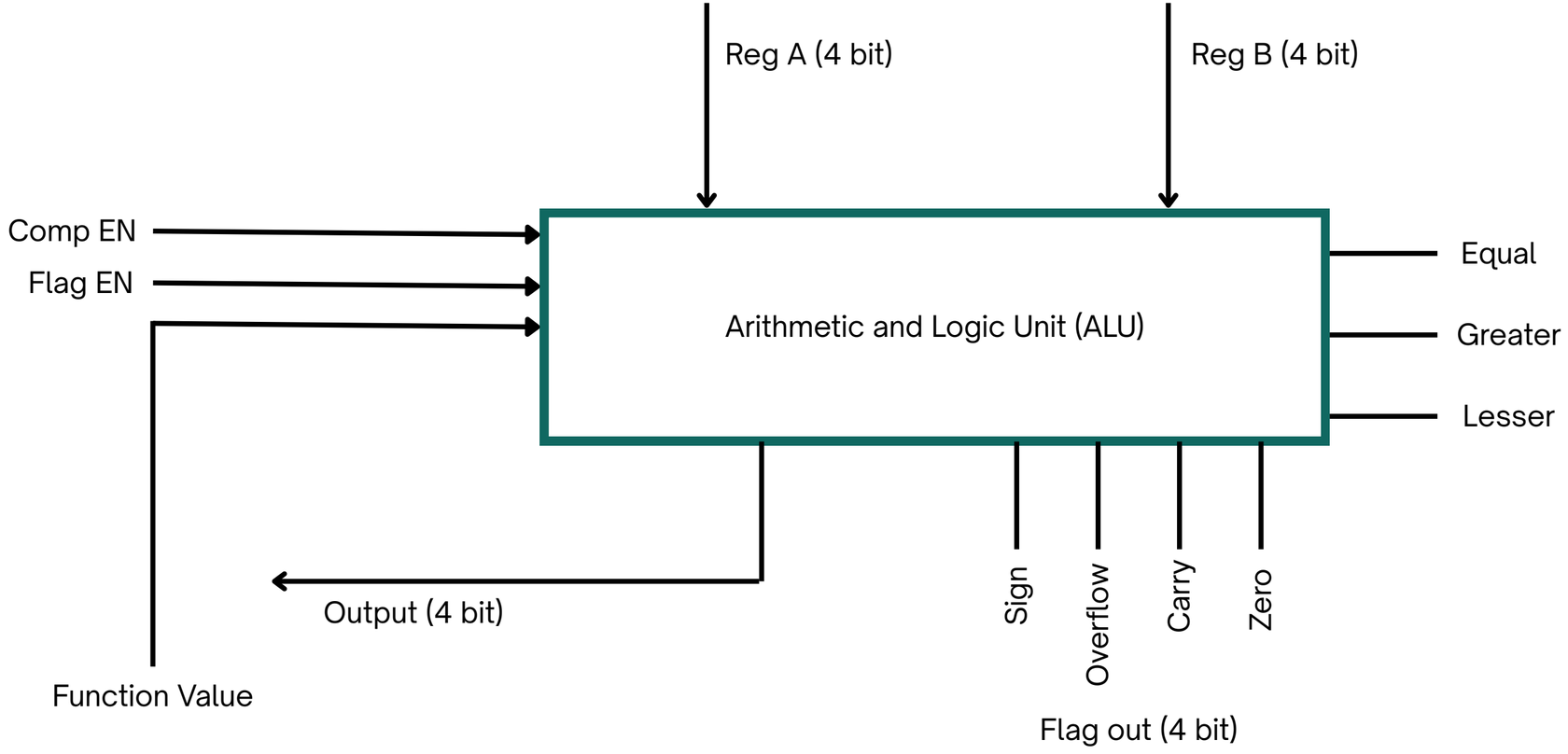
Arithmetic and Logic Unit (ALU) - Minimal

The ALU (Arithmetic Logic Unit) is responsible for executing operations such as ADD, SUB (via NEG), AND, OR, and MUL, based on control signals from the instruction decoder, and it outputs the result along with flags like Zero, Carry, Sign, and Overflow to guide program flow.

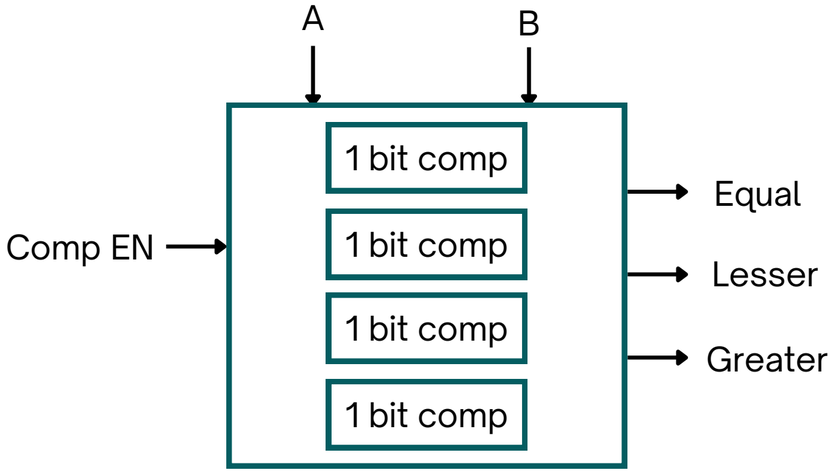


Arithmetic and Logic Unit (ALU) - Extended

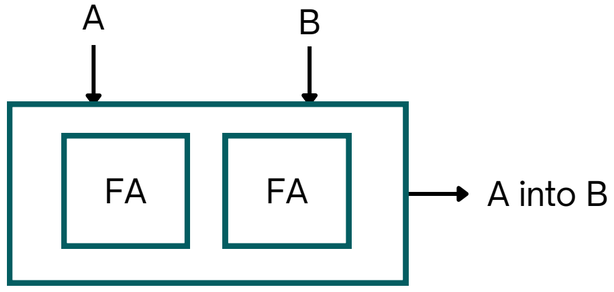
The extended ALU performs arithmetic (ADD, SUB), includes a comparator unit outputting equal, lesser, and greater flags, supports bitwise AND/NAND and OR/NOR, and integrates a 2-bit multiplier.



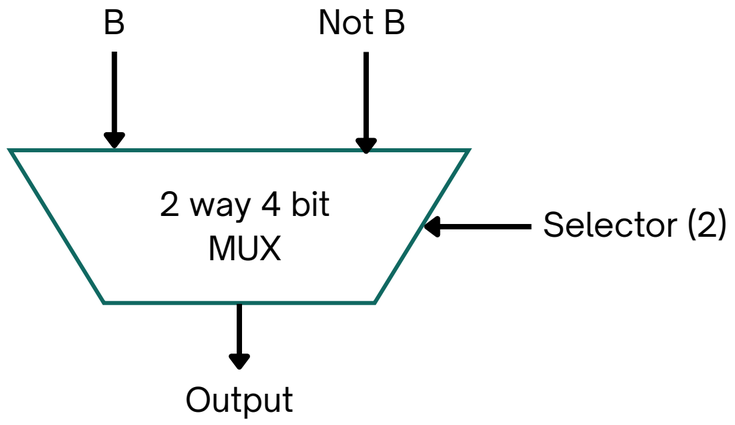
Arithmetic and Logic Unit (ALU) - (contd.)



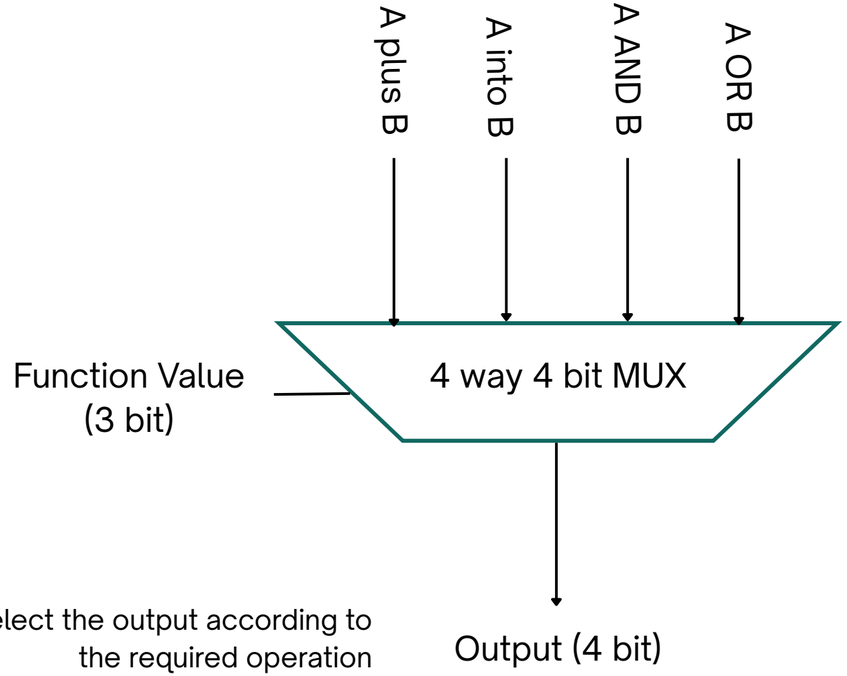
4 bit comparator



2 bit Multiplier



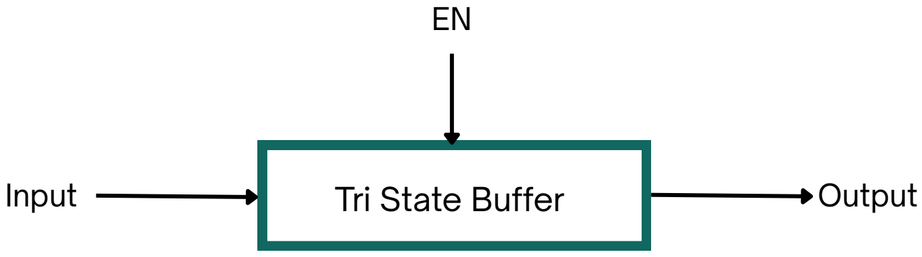
Use for NAND and NOR operations

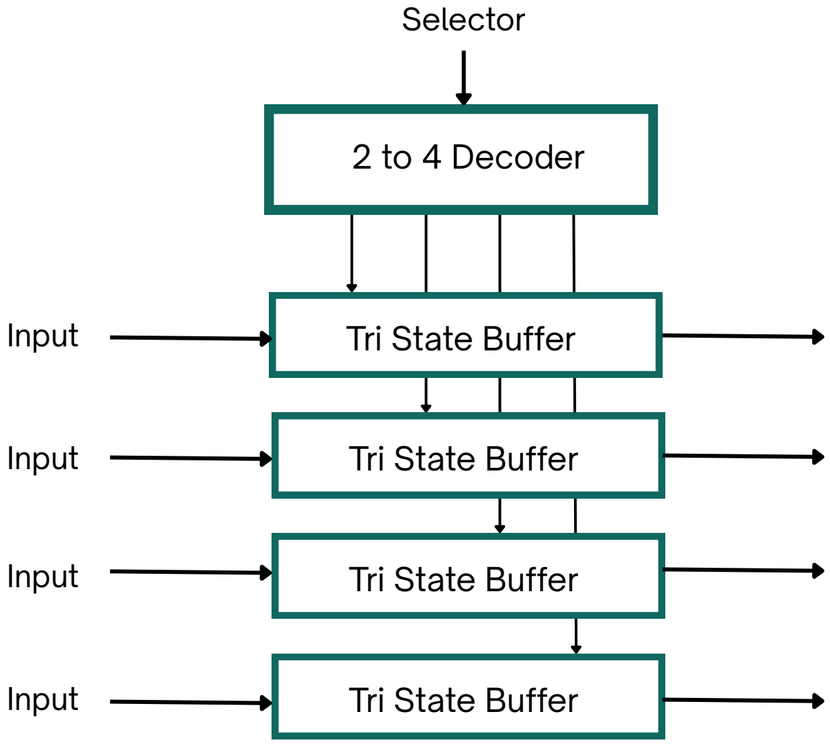


Use to select the output according to the required operation

Mux with Tri State Buffers

A multiplexer (MUX) implemented with tri-state buffers uses enable signals to selectively connect one of multiple inputs to a single output by enabling only one buffer at a time, allowing other outputs to be in a high-impedance state.





Outputs the input from the enabled buffer

Ex: 4 way 4 bit MUX

Problems Faced

Load Function via Switches

Instruction Width Limitation

Reset Mechanism Using Pushbutton

Contributions

Member	Contributions
MADAELA M.P.G.R.V.M. 230381X	• Designed and implemented Minimal Nanoprocessor and Extended Nanoprocessor top modules. • Port mapping of Minimal and Extended processor components. • Developed testbenches and timing diagrams for both processor versions. • Contributed to VHDL code for integration and functional validation. • Coordinated group tasks and contributed to final report compilation
KESHARA K.P.W.D. 230334H	• Designed and implemented the Instruction Decoder, Program Counter, and Program ROM for both processor versions. • Created corresponding testbenches and timing diagrams. • Assisted in developing machine code and instruction encoding for both versions. • Worked on constraint files (Basys3.xdc) and ensured board compatibility.
SENANAYAKE H.P.V.R. 230595G	• Developed and tested the ALU, including additional operations (AND, OR, MUL) in the extended version. • Implemented and validated the Comparator Unit (EQ, GT, LT flags). • Handled add/sub unit, 4-bit adder, and negation logic. • Worked on extended ALU testbenches and diagrams.
SOMARATHNA M.D.A.M. 230621K	• Designed and implemented MUXes (2-way/4-way/8-way) and Tri-State Buffers. • Built and tested Register Bank, 4-bit Registers, and related components. • Developed 3 to 8 Decoder, LUT for 7-Segment Display, and Slow Clock Generator. • Responsible for final report formatting and proofreading.

Conclusion

We successfully designed and tested both Minimal and Extended 4-bit Nanoprocessors on the BASYS 3 FPGA, demonstrating essential functionality with core instructions and enhanced capabilities through added operations (AND, OR, MUL), a comparator, and tri-state MUXes. This project significantly strengthened our understanding of processor architecture, VHDL, and FPGA system integration, preparing us for future digital design challenges.

Thank You!