ECE 310
Project 2 Design

Ohm Patel

October 1, 2025

1 Introduction

The objective of this project is to design a synchronous digital system that captures four 8-bit inputs (A, B, C, D), computes

$$\text{Result} = (A + B) - (C + D),$$

and asserts a `valid` flag when the output is ready.

The design is implemented entirely with structural modules. The datapath is composed of registers, adders, a subtractor, and control logic. Additional helper modules such as multiplexers, demultiplexers, an encoder, and a countdown subtractor are used for operand routing and timing control.

2 Top-Level Block Diagram

Figure 1 shows the main block diagram of the system. Inputs are routed via a demultiplexer into operand registers, processed through adders and a subtractor, and finally captured into a result register. Control logic ensures the output is latched and the `valid` signal is asserted for exactly one cycle.

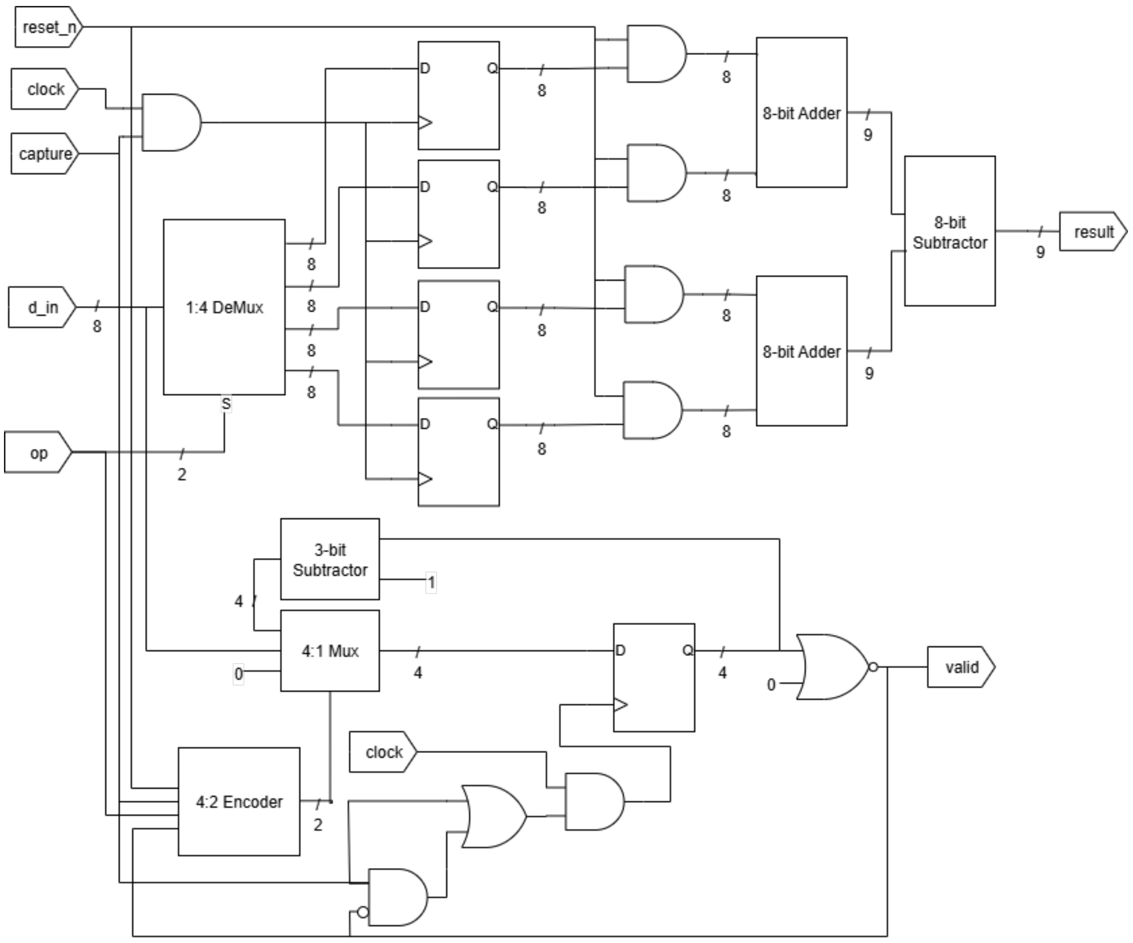

Figure 1: Top-level system block diagram.

3 Component Modules

3.1 D Flip-Flop

The D flip-flop (Figure 2) is the fundamental storage element. It is used to construct the 8-bit operand registers, the 9-bit result register, the valid flag register, and internal ready/issued flags in the controller logic.

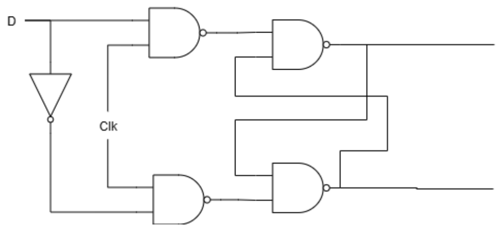

Figure 2: D flip-flop implementation.

3.2 Full Adder

The 1-bit full adder (Figure 3) is the building block of both the 8-bit and 9-bit adders. It takes three inputs (a, b, cin) and produces sum and carry-out.

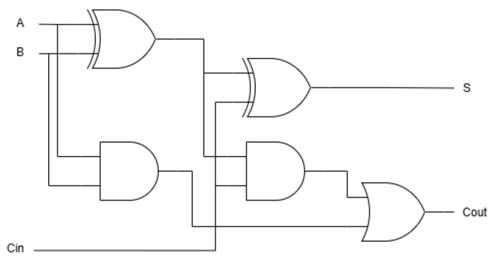

Figure 3: Full adder logic.

3.3 8-bit Adder

An 8-bit ripple-carry adder (Figure 4) is built by cascading eight full adders. It computes partial sums such as $A + B$ and $C + D$. The carry-out bit extends the result to 9 bits.

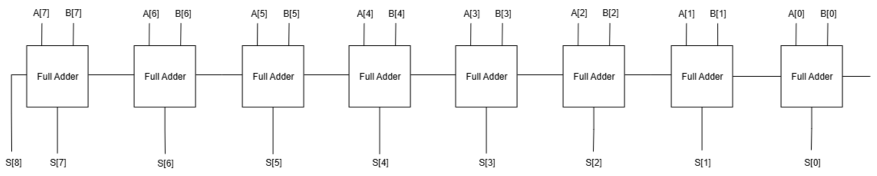

Figure 4: 8-bit ripple-carry adder constructed from full adders.

3.4 9-bit Subtractor

The 9-bit subtractor (Figure 5) is implemented using two's complement addition:

$$A - B = A + (\sim B) + 1.$$

It is used to compute the final result $(A + B) - (C + D)$.

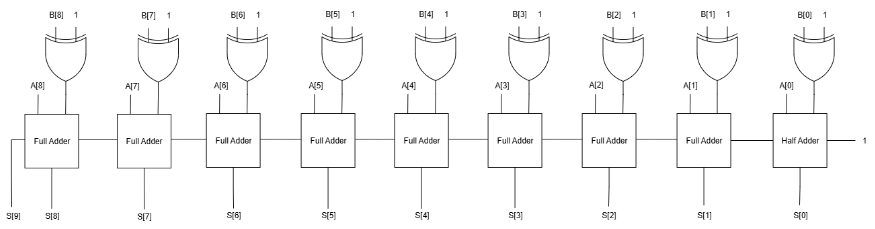
B[8] 1
B[7] 1
B[6] 1
B[5] 1
B[4] 1
B[3] 1
B[2] 1
B[1] 1
B[0] 1
A[8] A[7] A[6] A[5] A[4] A[3] A[2] A[1] A[0]
Full Adder
Half Adder
1
S[9] S[8] S[7] S[6] S[5] S[4] S[3] S[2] S[1] S[0]

Figure 5: 9-bit subtractor implementation.

3.5 1:4 Demultiplexer and 4:1 Multiplexer

The 1:4 demultiplexer routes incoming 8-bit data to the correct operand register based on the `op` select lines. The 4:1 multiplexer is used in the control logic to select countdown values and manage operand readiness. See Figures 6 and 7.

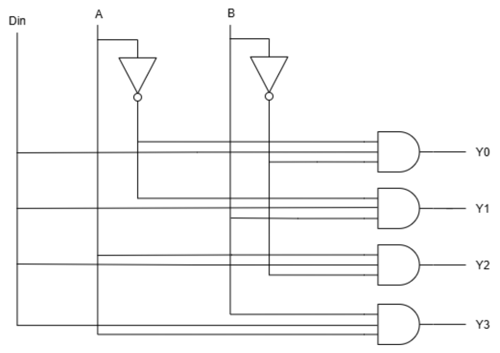

Figure 6: 1:4 demultiplexer for operand loading.

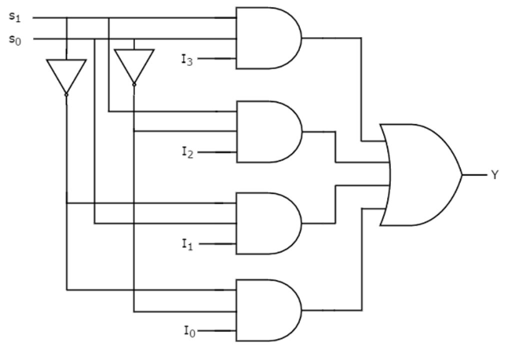

Figure 7: 4:1 multiplexer.

3.6 4:2 Encoder

The 4:2 encoder (Figure 8) converts operand select signals into binary form for efficient control logic routing.

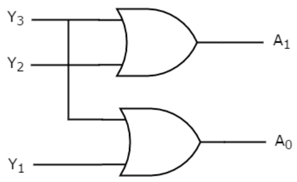

Figure 8: 4:2 encoder.

4 Control Logic and Valid Flag

The control logic uses operand-ready bits and a countdown mechanism to issue a single-cycle signal when all operands are loaded. The subtractor result is latched into the result register on this cycle.

A D flip-flop delays the compute signal by one cycle, producing a one-cycle `valid` flag in the cycle after the result is captured. This guarantees that the output is stable when `valid` is high.

5 Conclusion

This design implements the required computation $(A+B)-(C+D)$ using structural building blocks. Each module was built from lower-level primitives (gates, full adders, flip-flops), ensuring the design adheres to project specifications. The datapath and control logic operate synchronously, producing a correct result and asserting the `valid` signal for one cycle after computation.

This report was compiled using LaTeX .