DESIGN PROJECT #3
4-BIT RIPPLE ADDER/SUBTRACTOR

Austin Guerrero
ECE 221 Digital Logic Design | Fall 2026 | October 18, 2026
Department of Electrical and Computer Engineering
Valparaiso University

ABSTRACT

In this design project, a 4-bit 2's complement adder/subtractor was built by designing one cell and copying it four times. The cell, a full adder with an added Mode input, uses an XOR gate to pass B through unchanged when Mode = 0 and to invert it when Mode = 1. Karnaugh maps reduced the cell to six gates on three ICs: three XORs, two ANDs, and one OR. The cell was entered in OrCAD Capture, verified in PSpice across all sixteen input combinations, and converted into a hierarchical block. Four copies, chained through their carries, formed a ripple adder/subtractor that correctly computed 3 + 4, 7 − 3, and −5 − (−7) with no overflow.

♦ INTRODUCTION

The first two design projects built circuits one gate at a time. Real digital systems are rarely designed that way; they are assembled from smaller blocks that have already been tested, and the same block is often reused many times. This project applies that idea to arithmetic. A single one-bit cell is designed and verified, then copied four times to form a 4-bit circuit that can either add or subtract [1].

The cell is called a full adder plus. It behaves as an ordinary full adder when its Mode input is 0 and as a full adder with an inverted B input when Mode is 1. Chaining four of these cells, with Mode also feeding the carry-in of the least significant bit, produces a 2's complement adder/subtractor controlled by a single line. The cell was drawn by hand, entered into OrCAD Capture, and simulated in PSpice before being reused as a hierarchical block in the 4-bit design.

♦ THEORY

Subtraction in 2's Complement

In 2's complement, the negative of a number is found by inverting every bit and adding 1. Subtraction therefore needs no dedicated hardware, because A − B can be computed as an addition:

$$A - B = A + \bar{B} + 1$$

A 4-bit 2's complement number covers −8 through +7, and its most significant bit doubles as the sign bit.

The Full Adder Plus

An XOR gate works as a controlled inverter. With one input tied to Mode, it passes B through when Mode = 0 and outputs $\bar{B}$ when Mode = 1. This signal, called B′ in this report, replaces B at the input of an ordinary full adder:

$$B' = B \oplus Mode$$

Because Mode reaches the adder only through B′, the rest of the cell is a three-input function of A, B′, and Cin. Sum is 1 whenever an odd number of those inputs are 1, which is the definition of XOR, and Cout is 1 whenever two or more are 1:

$$Sum = A \oplus B' \oplus Cin$$

$$Cout = AB' + Cin(A \oplus B')$$

The first term generates a carry when A and B′ are both 1. The second propagates an incoming carry when exactly one of them is 1.

Ripple Carry and Overflow

In a ripple adder, the carry out of each bit becomes the carry in of the next. The extra 1 needed for subtraction comes from tying Mode to the carry-in of bit 0, so one control line both inverts B and completes its 2's complement. Overflow occurs only when two operands of the same sign produce a result of the opposite sign; a carry out of bit 3 is not, by itself, an overflow in signed arithmetic.

♦ PROCEDURE

The specification and procedure are given in the Design Project #3 handout [1]. Hand drawings, schematics, and waveforms are included in the Appendix.

I. The truth table was written for all sixteen combinations of Mode, A, B, and Cin, with B′ added as an intermediate column (Table 1, Fig. 1).

II. Three-variable K-maps in A, B′, and Cin were used to minimize Sum and Cout, and the carry equation was rewritten in generate/propagate form to share the A ⊕ B′ signal (Fig. 2).

III. The gate-level circuit was drawn by hand using a 7486, a 7408, and a 7432 (Table 2, Fig. 3).

IV. The circuit was entered in OrCAD Capture under the project fourbitaddsub, with input nets MODE, A, B, and C_in and output nets SUM and C_out (Fig. 4).

V. Four DigClock parts drove the inputs with on and off times of 4 µs, 2 µs, 1 µs, and 0.5 µs, so that an 8 µs simulation stepped through every input combination (Table 3, Fig. 5).

VI. The DigClocks were replaced with hierarchical ports, and the schematic was renamed fulladdplus.

VII. Four fulladdplus blocks were placed in a new root schematic and connected through buses A[3..0], B[3..0], and SUM[3..0], with the carries chained from bit 0 to bit 3 and MODE wired to every block and to the carry-in of bit 0 (Fig. 6).

VIII. Two STIM4 sources and one STIM1 source applied the three test cases for 100 ms each, and the 300 ms simulation was checked against hand calculations (Table 4, Fig. 7).

♦ RESULTS

Table 1 is the complete truth table for the full adder plus. The unshaded rows (Mode = 0) form a standard full adder, and the shaded rows (Mode = 1) are the same table with B inverted. B′ is an internal signal, listed only to make each row easy to check.

Table 1: Full Adder Plus Truth Table

Mode	A	B	Cin	B′	Sum	Cout
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	1	0
0	0	1	1	1	0	1
0	1	0	0	0	1	0
0	1	0	1	0	0	1
0	1	1	0	1	0	1
0	1	1	1	1	1	1
1	0	0	0	1	1	0
1	0	0	1	1	0	1
1	0	1	0	0	0	0
1	0	1	1	0	1	0
1	1	0	0	1	0	1
1	1	0	1	1	1	1
1	1	1	0	0	1	0
1	1	1	1	0	0	1

Table 2 lists the six gates in the final cell and the chips they occupy. Each gate name matches the reference designator in the OrCAD schematic (Fig. 4).

Table 2: Full Adder Plus Gate List

Gate	Inputs	Output	IC / Reference
XOR 1	B, Mode	B′ = B ⊕ Mode	7486 / U1A
XOR 2	A, B′	P = A ⊕ B′	7486 / U1B
XOR 3	P, Cin	Sum	7486 / U1C
AND 1	A, B′	G = AB′	7408 / U2A
AND 2	P, Cin	T = Cin(A ⊕ B′)	7408 / U2B
OR 1	G, T	Cout	7432 / U3A

Table 3 gives the DigClock settings used to step the cell through all sixteen rows. Each period is twice the one before it, with Mode given the longest period as the handout specifies, so every 0.5 µs slot of the simulation corresponds to one row of Table 1.

Table 3: DigClock Settings for the Full Adder Plus

Input	ON / OFF Time	Period
C_in	0.5 µs	1 µs
B	1 µs	2 µs
A	2 µs	4 µs
MODE	4 µs	8 µs

Table 4 compares the 4-bit simulation with hand calculations. In each subtraction case, B is inverted inside every cell and MODE supplies the extra 1 at bit 0. All three results are correct, and none overflow.

Table 4: 4-Bit Adder/Subtractor Simulation Results

Window	Operation	A	B	MODE	Binary Operation	SUM	C_OUT	Overflow
0–100 ms	3 + 4	0011	0100	0	0011 + 0100	0111 (+7) ✓	0	No
100–200 ms	7 − 3	0111	0011	1	0111 + 1100 + 1	0100 (+4) ✓	1	No
200–300 ms	−5 − (−7)	1011	1001	1	1011 + 0110 + 1	0010 (+2) ✓	1	No

Table 5 counts the hardware in both circuits. The 4-bit count assumes the gates from all four blocks are packed together by type.

Table 5: Gate and IC Count

Design	XOR	AND	OR	Total Gates	ICs
Full adder plus (one cell)	3	2	1	6	3
4-bit, as drawn (one set of chips per block)	12	8	4	24	12
4-bit, packed by gate type	**12**	**8**	**4**	**24**	**6**

♦ DISCUSSION

Full Adder Plus Design

Keeping Mode out of the K-maps simplified the design. Since Mode reaches the adder only through B′, the cell splits into one XOR followed by a three-input full adder; a four-variable map would instead rebuild B ⊕ Mode inside four of its five carry terms.

The Sum map is a checkerboard, the signature of an XOR function, so Sum uses two cascaded XOR gates (Fig. 2). The Cout map gives the majority function AB′ + ACin + B′Cin, which would need three ANDs and a 3-input OR. Rewriting it as AB′ + Cin(A ⊕ B′) reuses the A ⊕ B′ signal from the Sum path and cuts the carry logic to two ANDs and one OR. The finished cell uses six gates on three ICs.

Full Adder Plus Simulation

The cell matched Table 1 in all sixteen 0.5 µs slots (Fig. 5). The inputs count down from 1111, so the subtraction half of the table appears first. The narrow glitches near 1, 3, 5, 6, and 7 µs occur where several inputs change at once and signals arrive through paths of different lengths; each settles within nanoseconds.

4-Bit Adder/Subtractor

All three test cases were correct (Table 4, Fig. 7). In the subtraction cases, MODE inverts B inside each cell and supplies the extra 1 at bit 0. C_OUT = 1 in those cases is not an overflow; both subtractions add operands of opposite sign, which can never overflow, and the carry simply means no borrow was needed.

As drawn, each block carries its own chips, for twelve ICs in total. Packed by gate type, the same 24 gates fit on six (Table 5), so the convenience of hierarchy costs chips unless the gates are redistributed.

♦ CONCLUSION

The 4-bit ripple adder/subtractor met the specification. One six-gate cell, verified against all sixteen of its input combinations, was copied four times and chained into a circuit that correctly computed 3 + 4, 7 − 3, and −5 − (−7). A single control line switched the system between addition and subtraction by inverting B in every cell and supplying the extra 1 at bit 0. Because the cell was proven on its own first, the 4-bit design only had to be checked for wiring, not logic. The ripple structure does make each bit wait for the carry from the one before it, but at four bits that delay is negligible; one well-tested block, used four times, did the whole job.

♦ ACKNOWLEDGEMENTS

The objective, specification, test cases, and procedure were given and paraphrased from the ECE 221 Design Project #3 handout [1].

APPENDIX

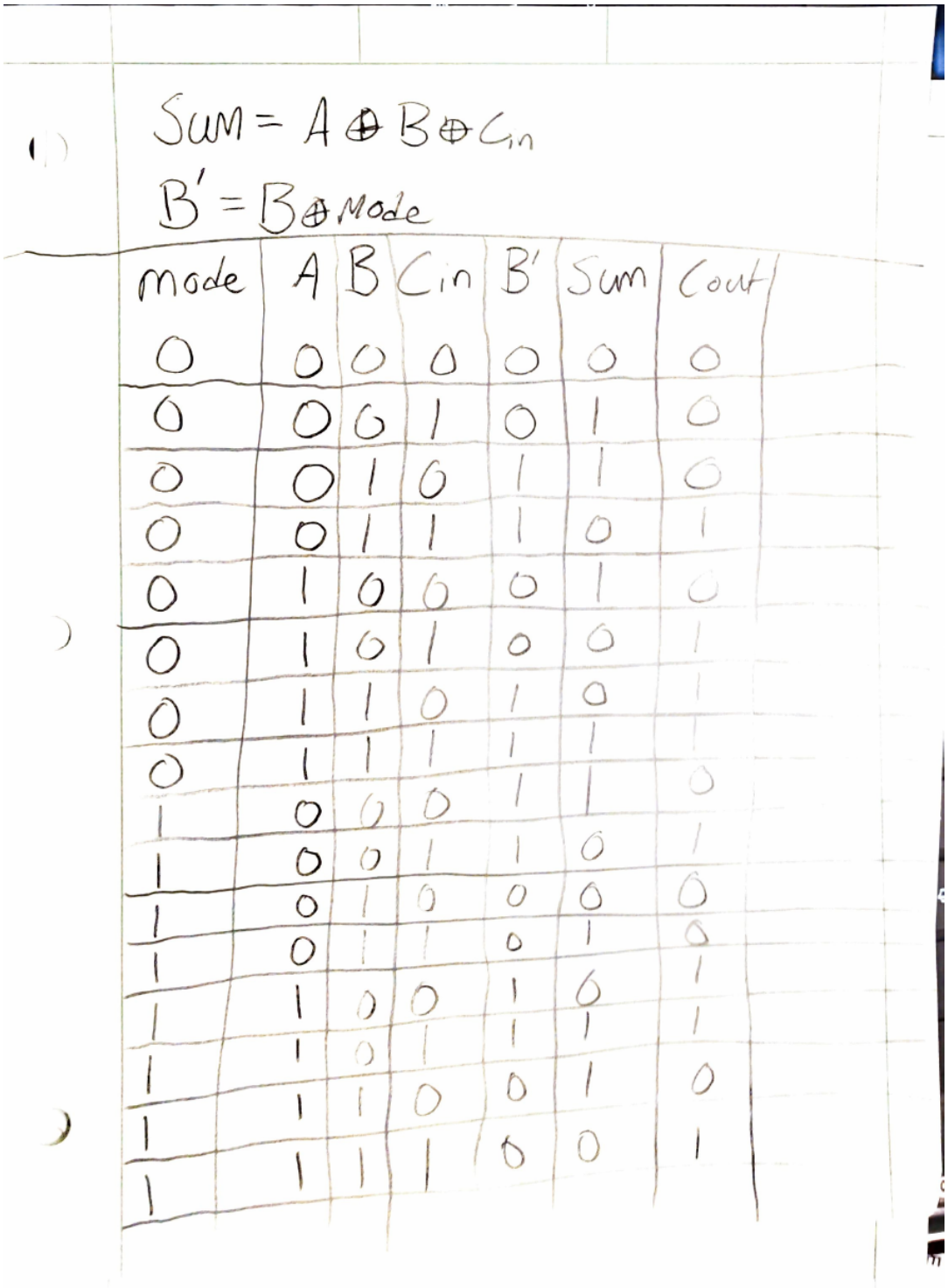


Fig 1. ***Handwritten truth table for the full adder plus***

APPENDIX

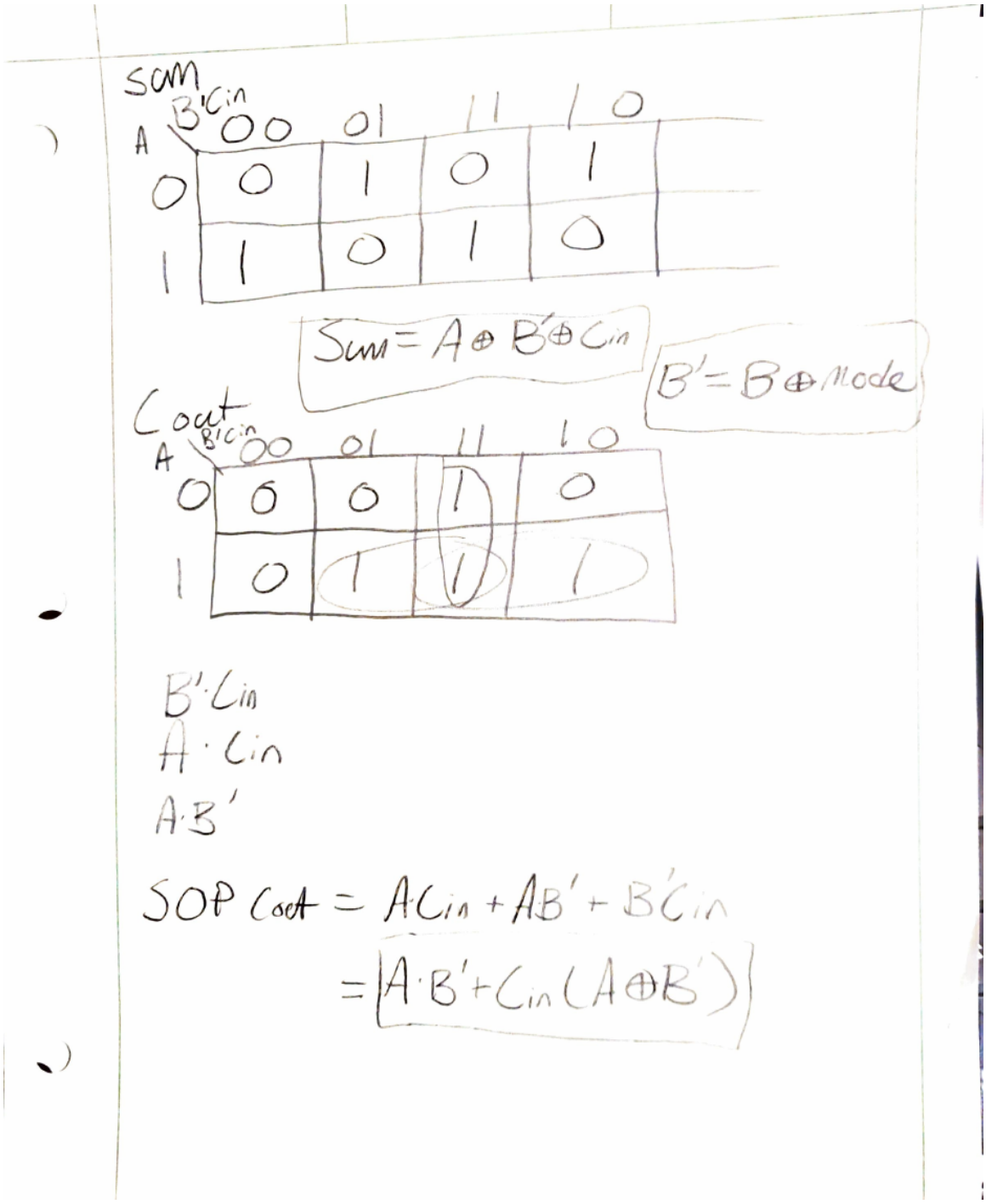


Fig 2. ***K-maps for Sum and Cout with the derived equations***

APPENDIX

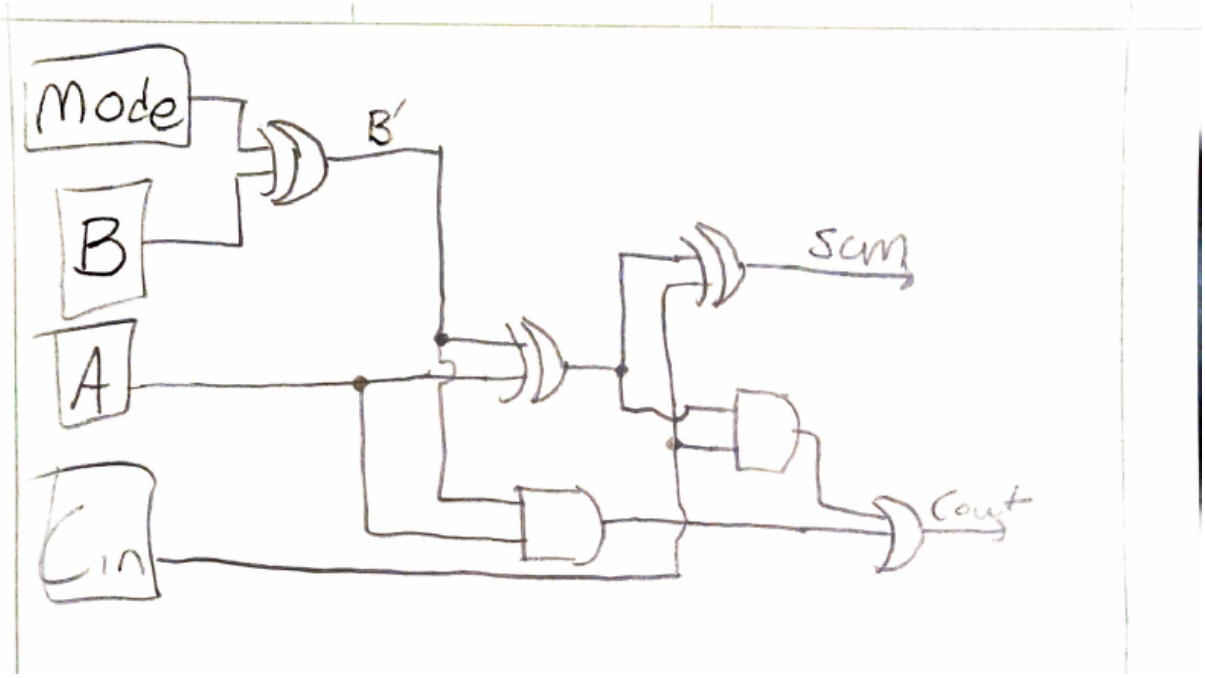


Fig 3. ***Hand-drawn gate-level circuit of the full adder plus***

APPENDIX

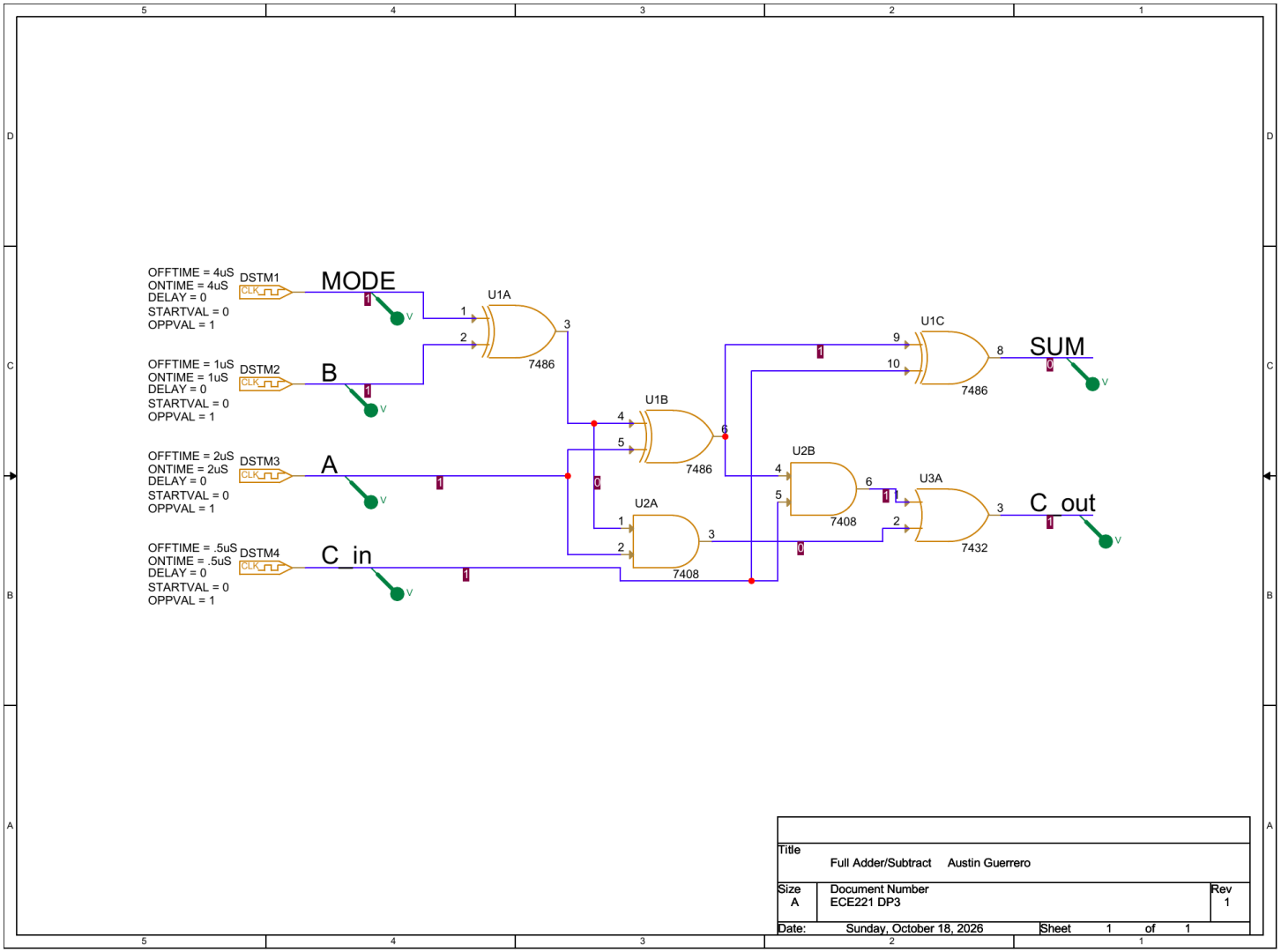


Fig 4. ***OrCAD schematic of the full adder plus (fulladdplus)***

APPENDIX

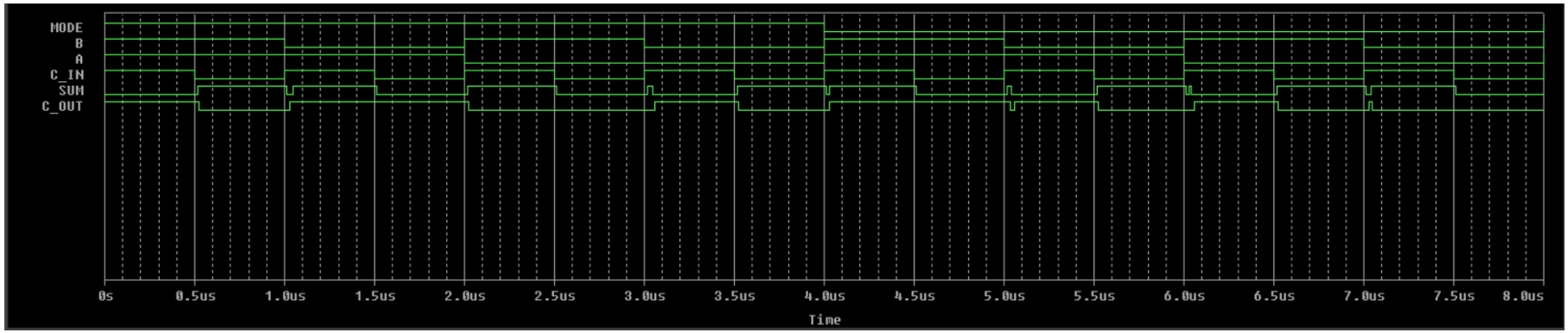


Fig 5. ***PSpice simulation of the full adder plus (all 16 input combinations)***

APPENDIX

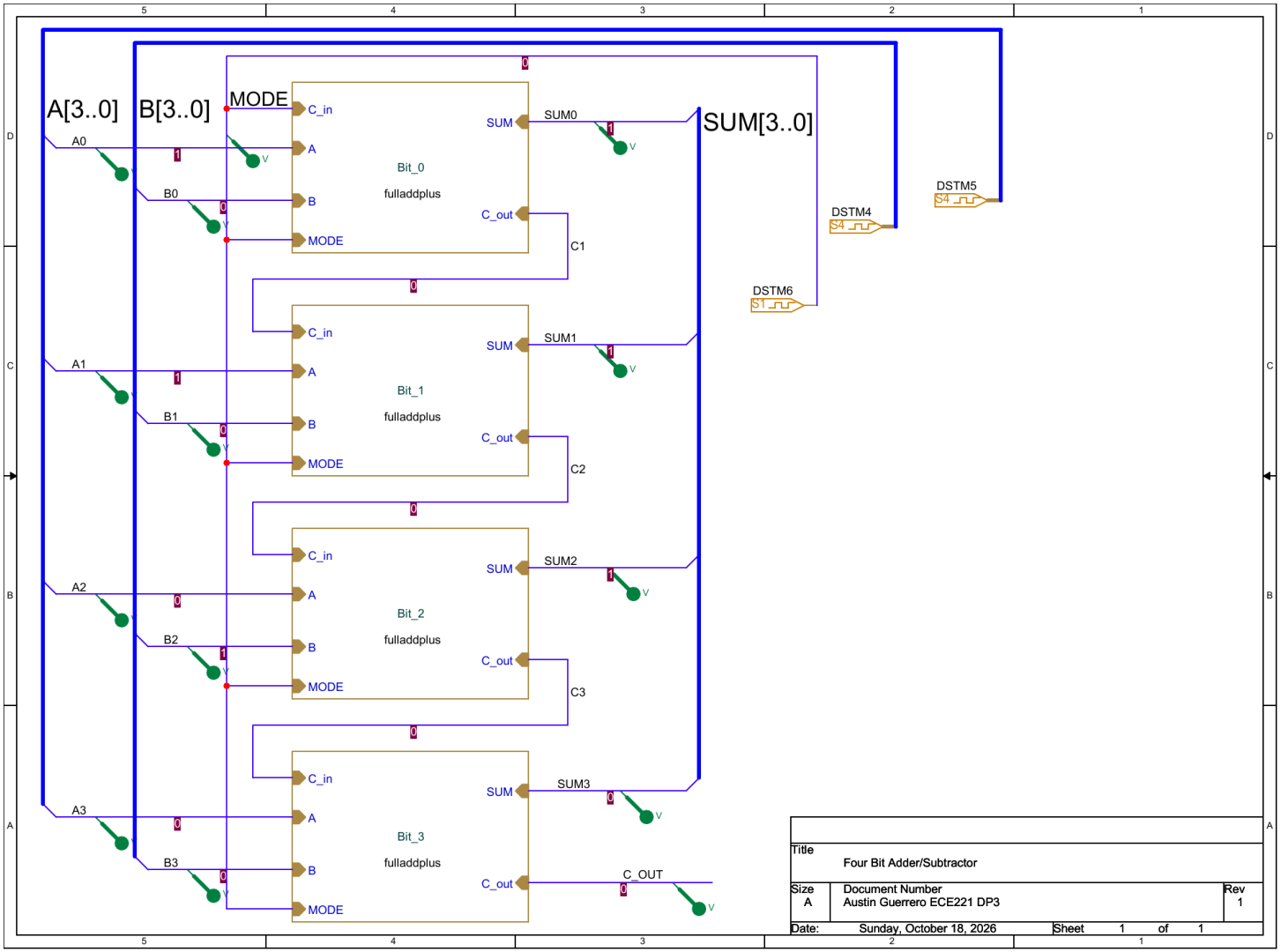


Fig 6. ***OrCAD schematic of the 4-bit ripple adder/subtractor (fourbitaddsub)***

APPENDIX

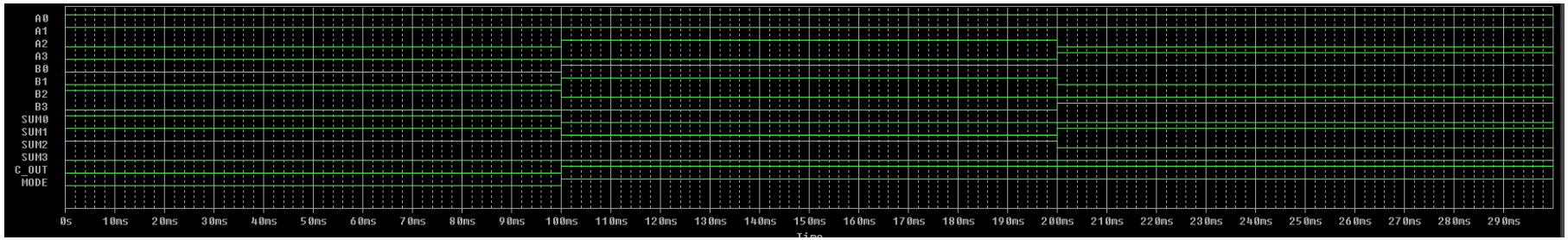


Fig 7. ***PSpice simulation of the 4-bit adder/subtractor (results in Table 4)***

Valparaiso University
Department of Electrical and Computer Engineering
ECE 221: Digital Logic Design

Design Project #3
4-Bit Adder/Subtractor

Name: Austin Guerrero

Honor Code Pledge:

"I have neither given or received nor tolerated others' use of unauthorized aid"

Signature: Austin Guerrero