DESIGN PROJECT #2
LOGIC MINIMIZATION: EXCESS-3 TO 2-OF-5 CODE CONVERTER

Austin Guerrero
ECE 221 Digital Logic Design | Fall 2026 | October 8, 2026
Department of Electrical and Computer Engineering
Valparaiso University

ABSTRACT

In this design project, an Excess-3 to 2-of-5 code converter was designed twice: once for the fewest gates and once for the fewest integrated circuits (ICs). Minilog was used to minimize the five output functions four ways: in sum-of-products (SOP) and product-of-sums (POS) form, each in single-output and multiple-output mode. Choosing the cheaper form for each output produced a 22-gate design built only from AND, OR, and NOT gates, which still requires seven ICs. A second design used the don't-care inputs and four XOR gates to share logic between outputs and needed 19 gates on five ICs. Both circuits were built in OrCAD Capture and simulated in PSpice, and both produced the correct 2-of-5 code for all ten valid inputs.

♦ INTRODUCTION

The goal of this project is to practice logic minimization and see why it matters outside of a homework problem. Every gate in a design costs money, board space, and propagation delay, but gates are not sold one at a time; they come four or six to a chip. Minimizing gates and minimizing chips are therefore two different problems, and they do not always share an answer [1].

The circuit studied here converts an Excess-3 digit into the 2-of-5 code. Two implementations were developed from the same truth table. The first uses the fewest gates possible with 2- and 3-input AND gates, 2-input OR gates, and inverters. The second uses the fewest ICs with any of the eight 74-series chips allowed in the handout. Both designs were drawn by hand, entered into OrCAD Capture, and checked in PSpice across all sixteen input combinations.

♦ THEORY

Excess-3 and 2-of-5 Codes

Excess-3 represents a decimal digit by adding three and writing the sum in 4-bit binary, so digit 0 becomes 0011 and digit 9 becomes 1100. The input wxyz can only take the values 3 through 12. The remaining six combinations (0000, 0001, 0010, 1101, 1110, and 1111) never occur and were treated as don't cares.

The 2-of-5 code represents each digit with five bits, exactly two of which are 1. There are exactly ten ways to choose two positions out of five, one per digit. Unlike binary, the bit positions carry no weight; the code is simply an agreed-upon table. Its strength is error detection, since a single flipped bit leaves either one or three 1s, which can never be a valid code word.

Minimizing Gates

Each output, a through e, is its own function of the same four inputs, so the converter is really five small 4-input circuits that share one set of don't cares. Each function can be minimized in SOP form by grouping 1s or in POS form by grouping 0s. Because the 7432 is the only OR gate allowed in the minimum-gate design, every three-term OR costs two gates. A factored form such as

$$a = w'(x' + y' + z')$$

can therefore be cheaper than its expanded SOP equivalent, w′x′ + w′y′ + w′z′, even though both describe the same function.

Minimizing ICs

Counting chips changes the rules. A chip type costs a full package even if only one of its gates is used, so the goal becomes using as few chip types as possible and filling each package completely. Spare gates are free, and a NAND or XOR gate can stand in for an inverter. The don't cares help here as well. A function only has to be correct on the ten valid inputs, so an XOR can replace an OR wherever the two differ only on inputs that never occur:

$$x \oplus z = x + z \quad \textit{(unless } x = z = 1\textit{)}$$

♦ PROCEDURE

The specification and procedure are given in the Design Project #2 handout [1]. Hand drawings, schematics, waveforms, and the Minilog files are included in the Appendix.

I. The truth table was written for all sixteen input combinations, with the six unused inputs marked as don't cares (Table 1).

II. The table was entered into Minilog as a .tbl file and minimized four ways: SOP and POS, each in single-output and multiple-output mode (Fig. 3).

III. The cheaper form was chosen for each output, the gates and ICs were counted, and the minimum-gate design was drawn on paper (Fig. 1).

IV. The minimum-gate design was entered in OrCAD Capture as converter_gates using parts from the EVAL library, with input nets W, X, Y, and Z and output nets A through E (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.

VI. The equations were reworked for the fewest ICs using NAND, XOR, and inverter chips, drawn on paper (Fig. 2), and entered as converter_ic (Fig. 6).

VII. Both circuits were simulated in PSpice, and every valid time slot was checked against the truth table (Figs. 5 and 7).

♦ RESULTS

Table 1 is the complete truth table. Rows 3 through 12 come straight from the handout, and the six shaded rows are don't cares, marked X in every output column.

Table 1: Excess-3 to 2-of-5 Truth Table

Row	w	x	y	z	a	b	c	d	e	Digit
0	0	0	0	0	X	X	X	X	X	unused
1	0	0	0	1	X	X	X	X	X	unused
2	0	0	1	0	X	X	X	X	X	unused
3	0	0	1	1	1	1	0	0	0	0
4	0	1	0	0	1	0	1	0	0	1
5	0	1	0	1	1	0	0	1	0	2
6	0	1	1	0	1	0	0	0	1	3
7	0	1	1	1	0	1	1	0	0	4
8	1	0	0	0	0	1	0	1	0	5
9	1	0	0	1	0	1	0	0	1	6
10	1	0	1	0	0	0	1	1	0	7
11	1	0	1	1	0	0	1	0	1	8
12	1	1	0	0	0	0	0	1	1	9
13	1	1	0	1	X	X	X	X	X	unused
14	1	1	1	0	X	X	X	X	X	unused
15	1	1	1	1	X	X	X	X	X	unused

Tables 2 and 3 list the Minilog results for all four modes. Minilog renamed the inputs A through D and the outputs V through Z in the order they appear in the input file, so every expression below has been translated back into the project's variable names; the raw output is reproduced in Fig. 3. Note that Minilog's X is output c, not input x. All twenty expressions were checked against Table 1 and are correct on every valid input.

Table 2: Minilog SOP Results

Output	Single-Output Mode	Multiple-Output Mode
a	w′y′ + w′z′ + w′x′	w′y′z + w′y′z′ + w′yz′ + w′x′
b	w′yz + x′y′	xyz + x′y′ + w′x′
c	w′y′z′ + wy + xyz	xyz + w′y′z′ + wy
d	w′y′z + wz′	w′y′z + wz′
e	w′yz′ + wx + wz	w′yz′ + wx + wz

Table 3: Minilog POS Results

Output	Single-Output Mode	Multiple-Output Mode
a	(x′ + y′ + z′)(w′)	(x′ + y′ + z′)(w′ + y)(w′ + y′)
b	(w + z)(x′ + y)(w′ + y′)	(w′ + x′)(w + y′ + z)(w + y + z′)(w′ + y′)(w + y + z)
c	(w + y′ + z)(w′ + y)(w + x)(y + z′)	(w′ + y)(w + y′ + z)(w + y + z′)(w + x)
d	(w + z)(w + y′)(x + z′)	(x′ + y′ + z′)(w′ + z′)(w + y′ + z)(w + y + z)(w + x)
e	(w + z′)(x + z)(w + y)	(x + z)(x′ + y′ + z′)(w + y + z′)(w + y + z)(w + x)

Table 4 totals the gates each Minilog result would need if built as written with the allowed parts. Each three-term OR counts as two 7432 gates, a product of four or five factors counts as two AND gates, and a product term shared between outputs is counted only once. All five designs also need the four inverters w′, x′, y′, and z′.

Table 4: Gate Count by Minimization Mode

Design	AND Gates	OR Gates	Inverters	Total Gates
SOP, single-output	13	8	4	25
SOP, multiple-output	10	10	4	24
POS, single-output	6	16	4	26
POS, multiple-output	9	14	4	27
Mixed: best form per output	**9**	**9**	**4**	**22**

Table 5 breaks the single-output results down by output, which shows where the mixed design gets its savings. Selecting the cheaper form for each output brings the total down to 22 gates, two fewer than the best result Minilog produced on its own.

Table 5: SOP and POS Gate Counts per Output (Single-Output Mode)

Output	SOP Form	Gates	POS Form	Gates	Selected
a	w′y′ + w′z′ + w′x′	5	w′(x′ + y′ + z′)	3	**POS**
b	w′yz + x′y′	3	(w + z)(x′ + y)(w′ + y′)	4	**SOP**
c	w′y′z′ + wy + xyz	5	(w + y′ + z)(w′ + y)(w + x)(y + z′)	7	**SOP**
d	w′y′z + wz′	3	(w + z)(w + y′)(x + z′)	4	**SOP**
e	w′yz′ + wx + wz	5	(w + z′)(x + z)(w + y)	4	**POS**
Total		21		22	**18 + 4 inverters = 22**

Table 6 lists the gates and chips needed for the selected minimum-gate design.

Table 6: Minimum-Gate Design, Gate and IC Count

Output	Equation	3-in AND	2-in AND	2-in OR	Gates
a	w′(x′ + y′ + z′)	0	1	2	3
b	w′yz + x′y′	1	1	1	3
c	w′y′z′ + wy + xyz	2	1	2	5
d	w′y′z + wz′	1	1	1	3
e	(w + z′)(x + z)(w + y)	1	0	3	4
—	Inverters w′, x′, y′, z′ (shared)				4
Total		**5**	**4**	**9**	**22**
ICs	7404 ×1, 7411 ×2, 7408 ×1, 7432 ×3	**2**	**1**	**3**	**6 ICs**

Table 7 gives the minimum-IC equations twice: once in Boolean form and once as nested gate functions that map directly onto the schematic. The signal $x \oplus z$ is built once and shared by outputs c and e.

Table 7: Minimum-IC Design Equations

Output	Boolean Form	Gate Implementation
a	$w \oplus (xyz)'$	XOR(w, NAND(x, y, z))
b	$w'yz + x'y'$	NAND(NAND(NOT(w), y, z), NAND(NOT(x), NOT(y)))
c	$y \oplus w'(x \oplus z)$	XOR(NOT(y), NAND(NOT(w), XOR(x, z)))
d	$(w \oplus z)(yz)'$	NOT(NAND(XOR(w, z), NAND(y, z)))
e	$w(x \oplus z) + w'yz'$	NAND(NAND(w, XOR(x, z)), NAND(NOT(w), y, NOT(z)))

Table 8: Minimum-IC Design, Gate and IC Count

Chip	Function	Gates Used	Gates per Chip	ICs
7404	Inverter	5	6	1
7486	2-input XOR	4	4	1
7410	3-input NAND	3	3	1
7400	2-input NAND	7	4	2
Total		**19**		**5**

Table 9 compares both simulations against the truth table. Because every DigClock started high, the inputs counted down from 1111 to 0000, so the ten valid inputs appear between 1.5 µs and 6.5 µs with digit 9 first. Both designs produced the expected code in every valid slot. The annotated waveforms are shown in Figs. 5 and 7.

Table 9: Simulation Results for Valid Inputs

Time (µs)	wxyz	Digit	Expected abcde	converter_gates	converter_ic
1.5–2.0	1100	9	00011	00011 ✓	00011 ✓
2.0–2.5	1011	8	00101	00101 ✓	00101 ✓
2.5–3.0	1010	7	00110	00110 ✓	00110 ✓
3.0–3.5	1001	6	01001	01001 ✓	01001 ✓
3.5–4.0	1000	5	01010	01010 ✓	01010 ✓
4.0–4.5	0111	4	01100	01100 ✓	01100 ✓
4.5–5.0	0110	3	10001	10001 ✓	10001 ✓
5.0–5.5	0101	2	10010	10010 ✓	10010 ✓
5.5–6.0	0100	1	10100	10100 ✓	10100 ✓
6.0–6.5	0011	0	11000	11000 ✓	11000 ✓

Table 10: Summary of the Two Designs

Design	Gate Types	Gates	ICs	Spare Gates
Minimum gates	AND, OR, NOT	22	7	6
Minimum ICs	NAND, XOR, NOT	19	5	2

♦ DISCUSSION

Minimum-Gate Design

Minilog's SOP result was correct, but it was not the cheapest circuit. Output a is the clearest example. In SOP form, a = w′y′ + w′z′ + w′x′ needs three ANDs and two ORs. Factoring out w′ gives w′(x′ + y′ + z′), which needs two ORs and a single AND, saving two gates outright. Output e improved in POS form as well, from five gates to four, because its three sum terms feed one 3-input AND instead of a chain of ORs.

Outputs b, c, and d stayed in SOP form. Output c is a particularly poor fit for POS; it needs four sum terms, one of them three literals wide, and the 4-input product alone takes two AND gates.

Multiple-output mode did find terms worth sharing. In SOP form, xyz serves both b and c, and w′y′z, w′y′z′, and w′yz′ each serve output a along with d, c, and e. That trims the all-SOP design from 25 gates to 24. The mixed design has no use for it, though. Output a is far cheaper as w′(x′ + y′ + z′) than as a four-term sum of borrowed products, and once a is in POS form, no other output gains anything by sharing. In POS form, multiple-output mode actually made things worse at 27 gates; it stretched b, d, and e to five factors each so that factors could be shared, and the extra AND gates cost more than the sharing saved.

The result is 22 gates, which pack into seven ICs: two 7411s, one 7408, three 7432s, and one 7404. The nine OR gates are the expensive part, consuming three packages on their own.

Minimum-IC Design

The minimum-IC design was found in three steps. First, the minimum-gate equations were converted directly to NAND-NAND form. That version needed seven 3-input NANDs, eight 2-input NANDs, and five inverters, or six ICs. It saved one chip, but the packages were poorly filled.

Second, the don't cares were put to work with XOR gates. Output a is the simplest case. For every valid input with w = 1, the product xyz is 0, so (xyz)′ is 1, and XOR-ing it with w forces a to 0. When w = 0, the XOR passes (xyz)′ through unchanged, which is exactly a on inputs 3 through 6. Five gates became two.

Output c turned out to equal y on every valid input except 3, 4, and 6, where it equals y′. Those three inputs are exactly where w′(x ⊕ z) = 1, so c = y ⊕ w′(x ⊕ z). Output d is 1 wherever w ⊕ z = 1 except at inputs 3 and 7, and both of those have y = z = 1, so d = (w ⊕ z)(yz)′. Finally, on the valid inputs with w = 1, x ⊕ z matches x + z, because the two disagree only when x = z = 1, which happens only at the unused inputs 13 and 15. That let e reuse the x ⊕ z signal already built for c. Output b gained nothing from XOR and was left in NAND-NAND form.

Third, the gate counts were balanced against package sizes. The final design uses exactly four XORs (one full 7486), three 3-input NANDs (one full 7410), seven 2-input NANDs (two 7400s with one gate spare), and five inverters (one 7404 with one spare). A four-chip version would require removing an entire package and rebuilding its gates from spares, but only two spare gates remain, so five ICs was the lowest this approach could reach.

Ironically, the minimum-IC design also uses fewer gates than the minimum-gate design. This is not a contradiction; the XOR and NAND gates it depends on were not allowed in the gate design, and a single XOR quietly does the work of several ANDs and ORs.

Simulation Results

Both circuits produced the correct code in all ten valid time slots (Table 9). Outside that

window the two designs disagree. Between 0 and 0.5 µs, for example, the input is 1111; the gate design holds a at 0, while the IC design drives it to 1. This is expected, since each design assigned the don't cares differently, and neither input can occur in normal operation.

Each output changed a few tens of nanoseconds after its inputs, reflecting the delay through the gates along that path. Brief glitches also appear at a handful of transitions: on a at 4.0 µs in the gate design, where all four inputs flip at once (1000 to 0111), and on c and e near 5.0, 6.0, and 7.0 µs in the IC design. These are hazards caused by several inputs changing together and signals arriving through paths of different lengths. Each settles to the correct value long before the next input change, so they do not affect the results here, though they would matter if the outputs fed a clocked circuit.

Problems Encountered

The IC design did not work on the first attempt. In the first simulation, the Y input's net alias had been typed as W, which tied two DigClocks to the same net. Whenever the two clocks disagreed, PSpice could not resolve the conflict and marked the net as unknown, corrupting every output downstream. After the alias was renamed, a stray voltage marker was found displaying the inverter output w′ in place of Z, and the outputs driven through XOR gates read as undefined whenever they should have been high; both issues were corrected.

The last error was a miswired 3-input NAND in the w′yz′ term of output e, which left e stuck at 0 for digit 3 (input 0110). Placing markers on that gate's inputs found the fault quickly. A duplicate XOR(x, z) gate was also removed, since keeping it would have required a fifth XOR and a second 7486.

♦ CONCLUSION

Both converters produced the correct 2-of-5 code for every valid Excess-3 input, but minimizing gates and minimizing chips led to noticeably different circuits. The gate-minimized design reached 22 gates by choosing SOP or POS output by output, yet it still needs seven ICs, three of them just for OR gates. The IC-minimized design reached five ICs by leaning on the don't cares, trading ORs for XORs, and sharing one XOR between two outputs. The cheapest equation on paper is not always the cheapest circuit on the board; in the end, the packaging decides the bill.

♦ ACKNOWLEDGEMENTS

The objective, specification, truth table, and procedure were given and paraphrased from the ECE 221 Design Project #2 handout [1]. The minimized expressions in Tables 2 and 3 were generated with Minilog [2].

♦ REFERENCES

[1] Dept. of Electrical and Computer Engineering, *ECE 221: Digital Logic Design Laboratory & Design Projects User Manual*, Design Project #2. Valparaiso, IN, USA: Valparaiso University.

[2] W. M. J. de Valk, *MINILOG Logic Minimization Program*, ver. 4.61. Oss, the Netherlands, 1988–2006.

APPENDIX

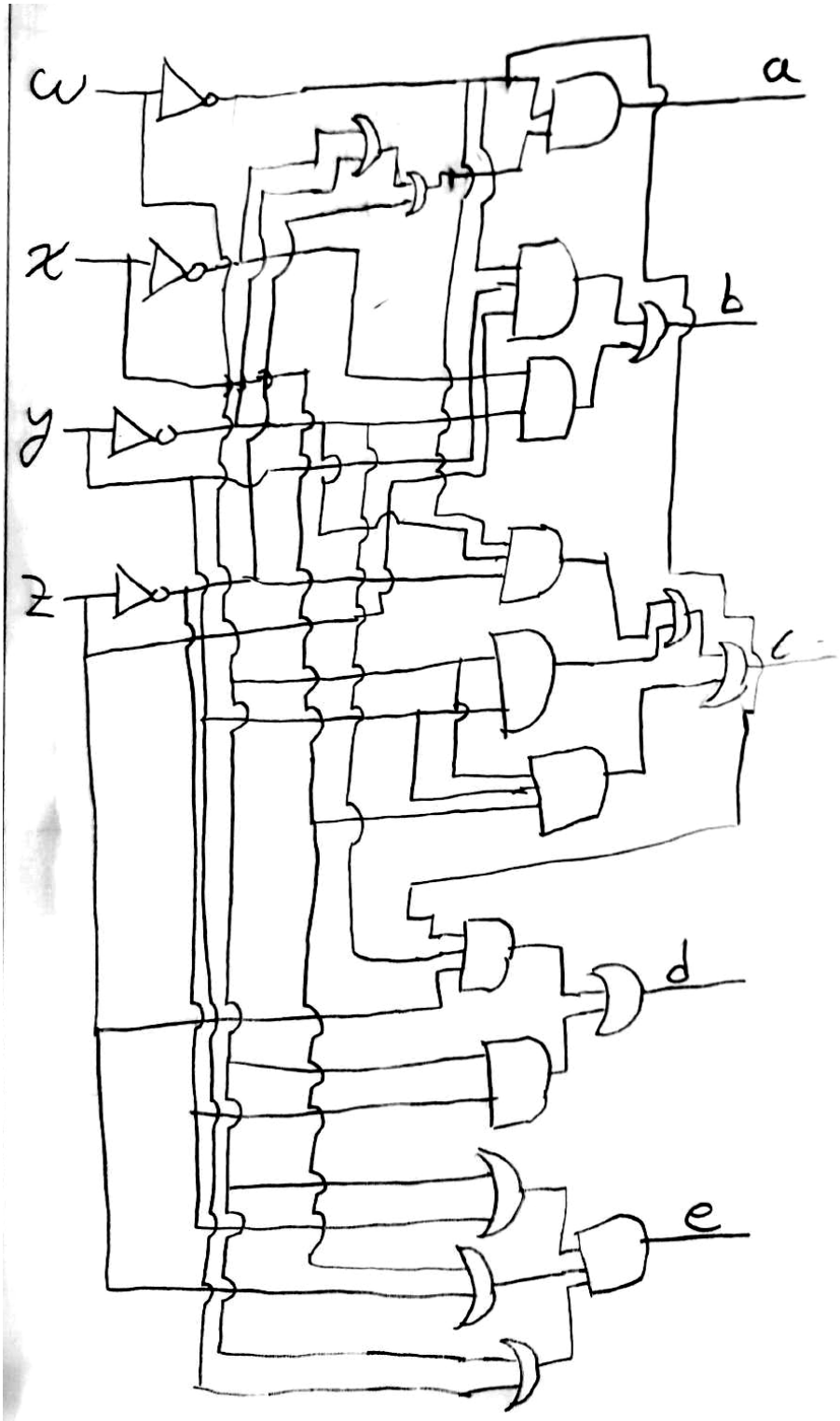


Fig 1. ***Hand-drawn minimum-gate implementation (AND, OR, NOT)***

APPENDIX

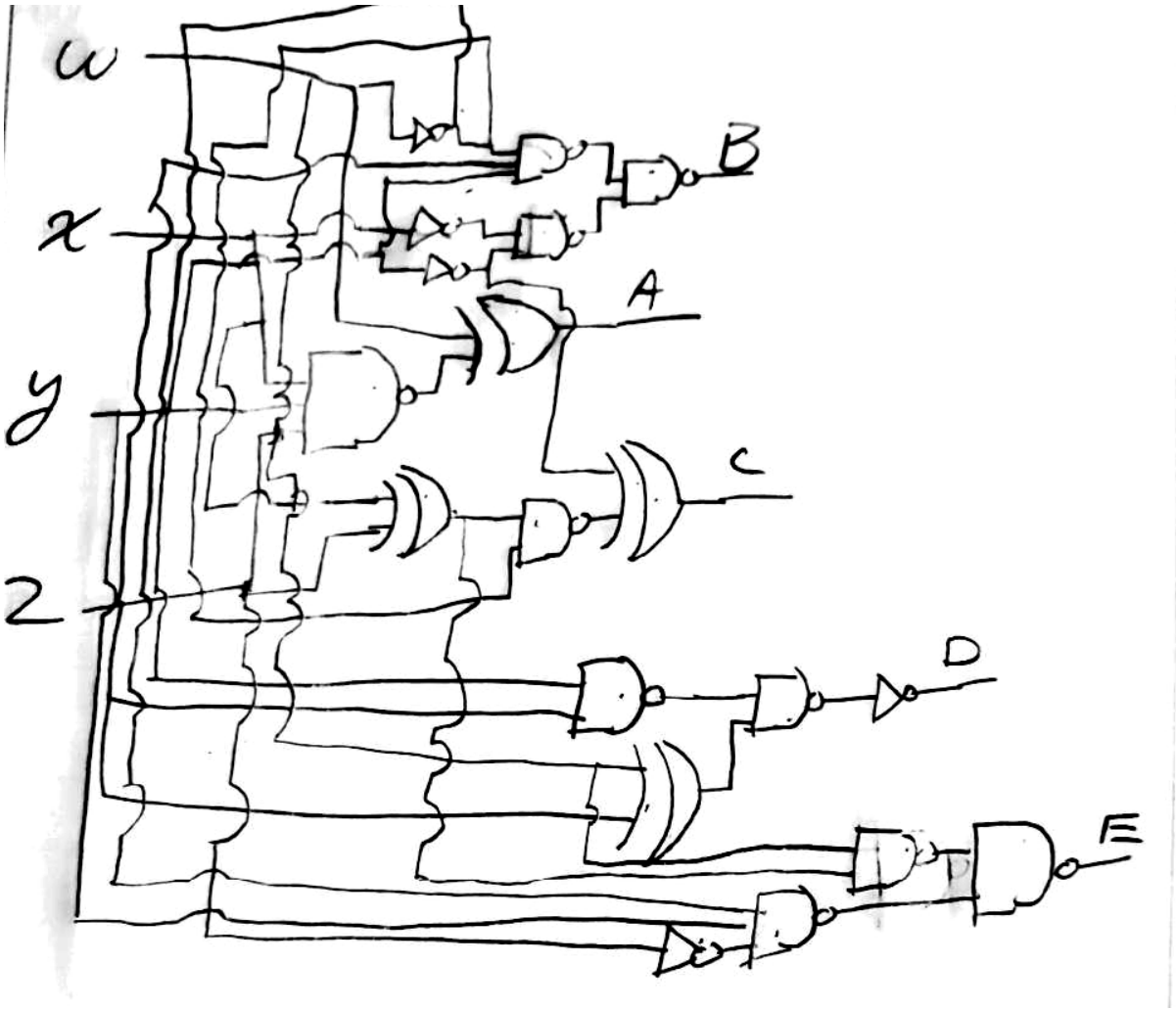


Fig 2. ***Hand-drawn minimum-IC implementation (NAND, XOR, NOT)***

APPENDIX

Function table read by Minilog

```
============
ABCD   VWXYZ
============
000- | -----
0010 | -----
0011 | 11000
0100 | 10100
0101 | 10010
0110 | 10001
0111 | 01100
1000 | 01010
1001 | 01001
1010 | 00110
1011 | 00101
1100 | 00011
1101 | -----
111- | -----
```

Minilog output: SOP, single-output mode

```
FOUND 13 ESSENTIAL PRODUCT TERMS  |  TOTAL LITERAL COUNT: 44
V = A'C' + A'D' + A'B'
W = A'CD + B'C'
X = A'C'D' + AC + BCD
Y = A'C'D + AD'
Z = A'CD' + AB + AD
```

Minilog output: SOP, multiple-output mode

```
FOUND 10 ESSENTIAL PRODUCT TERMS  |  TOTAL LITERAL COUNT: 39
V = A'C'D + A'C'D' + A'CD' + A'B'
W = BCD + B'C' + A'B'
X = BCD + A'C'D' + AC
Y = A'C'D + AD'
Z = A'CD' + AB + AD
```

Minilog output: POS, single-output mode

```
FOUND 14 ESSENTIAL FACTORS IN PRODUCT OF SUMS MODE  |  TOTAL LITERAL COUNT: 44
V = (B'+C'+D').(A')
W = (A+D).(B'+C).(A'+C')
X = (A+C'+D).(A'+C).(A+B).(C+D')
Y = (A+D).(A+C').(B+D')
Z = (A+D').(B+D).(A+C)
```

Minilog output: POS, multiple-output mode

```
FOUND 10 ESSENTIAL FACTORS IN PRODUCT OF SUMS MODE  |  TOTAL LITERAL COUNT: 46
V = (B'+C'+D').(A'+C).(A'+C')
W = (A'+B').(A+C'+D).(A+C+D').(A'+C').(A+C+D)
X = (A'+C).(A+C'+D).(A+C+D').(A+B)
Y = (B'+C'+D').(A'+D').(A+C'+D).(A+C+D).(A+B)
Z = (B+D).(B'+C'+D').(A+C+D').(A+C+D).(A+B)
```

Variable key (from Minilog): A, B, C, D = w, x, y, z; V, W, X, Y, Z = a, b, c, d, e

Fig 3. ***Minilog function table and minimized output in all four modes (MINILOG ver. 4.61)***

APPENDIX

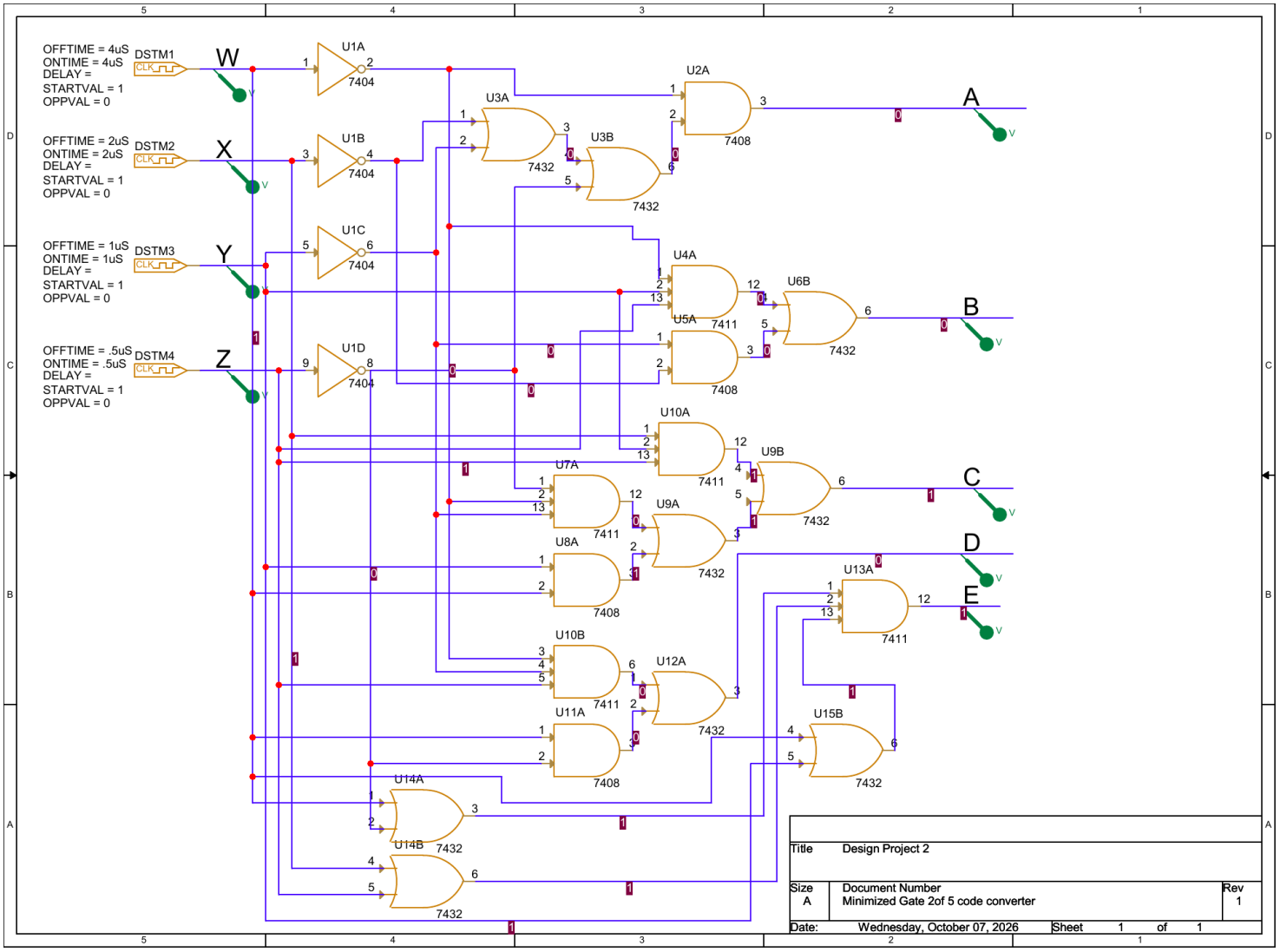


Fig 4. ***OrCAD schematic of the minimum-gate design (converter_gates)***

APPENDIX

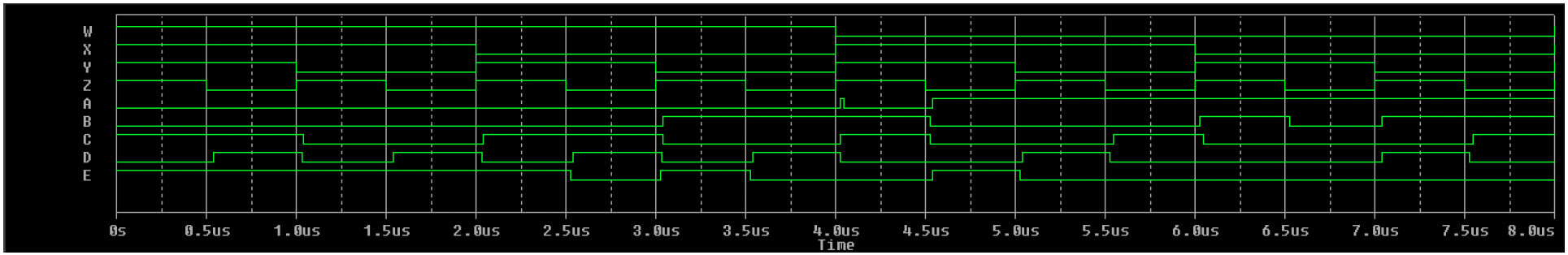


Fig 5. ***PSpice simulation of the minimum-gate design (time slots verified in Table 9)***

APPENDIX

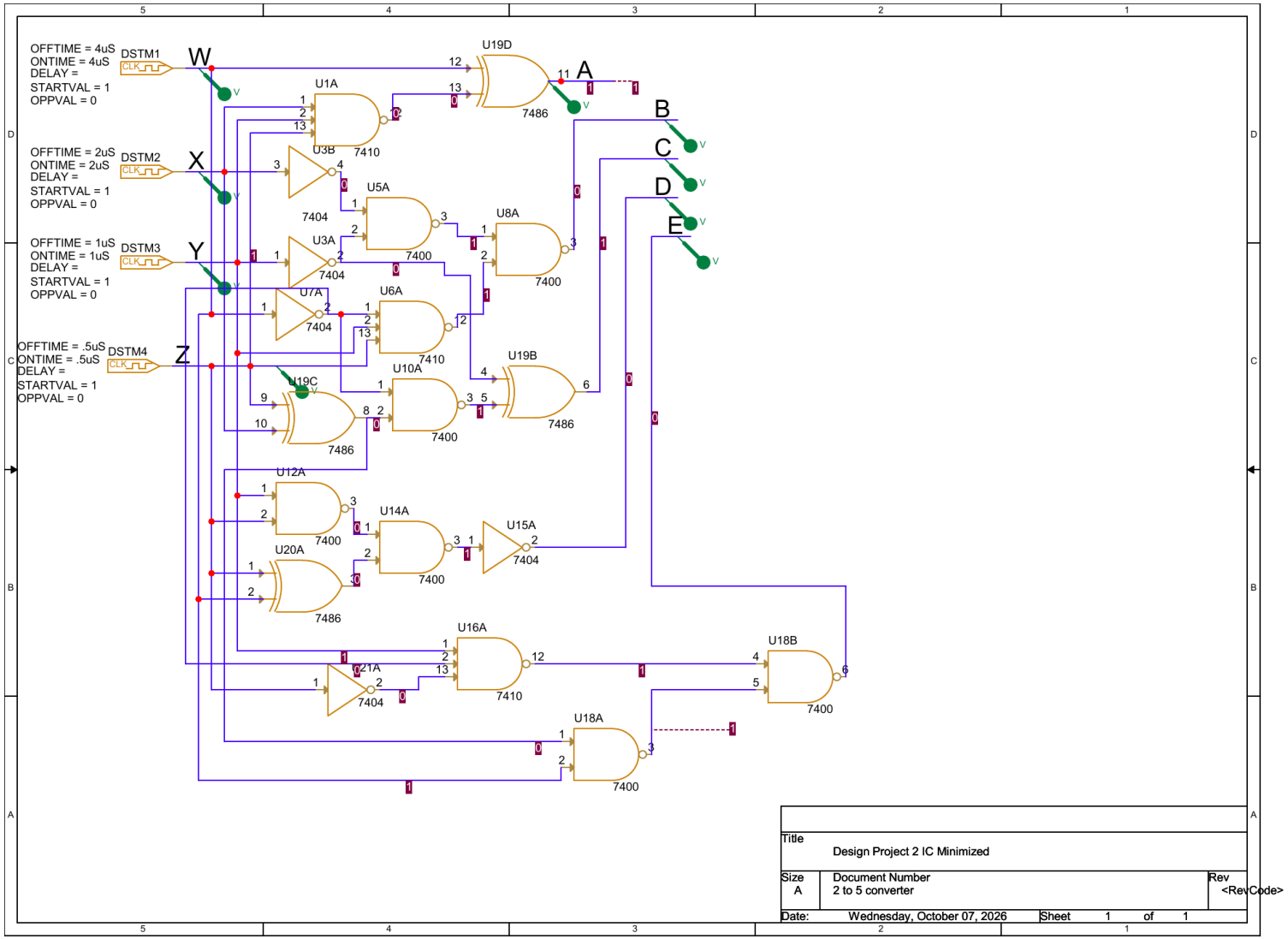


Fig 6. ***OrCAD schematic of the minimum-IC design (converter_ic)***

APPENDIX

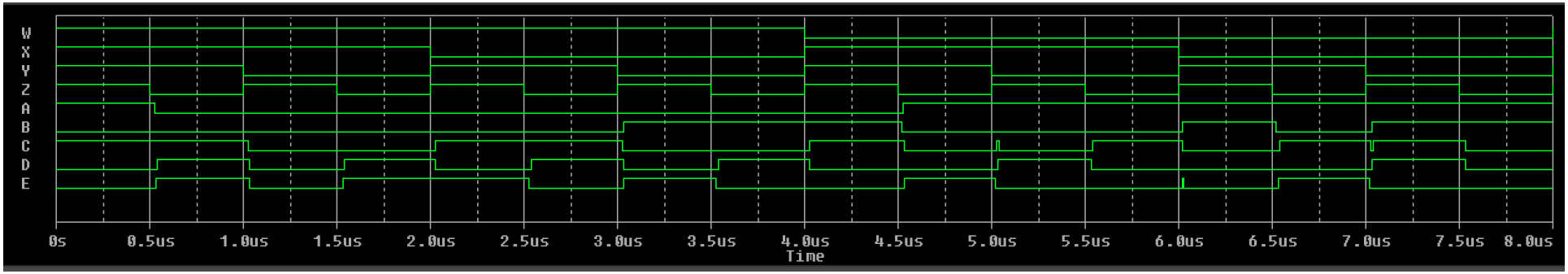


Fig 7. ***PSpice simulation of the minimum-IC design (time slots verified in Table 9)***

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

Design Project #2:
LOGIC MINIMIZATION

Name: Austin Guerrero

Honor Code Pledge:

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

_____

Signature: _____Austin Guerrero_____