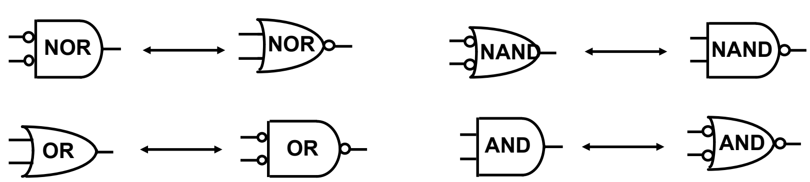
Canonical forms for Boolean logic

- Algebraic expressions to gates (lab 1)
- Canonical forms
- Incompletely specified functions
- Realizing two-level canonical forms

NAND, NOR, and de Morgan's theorem

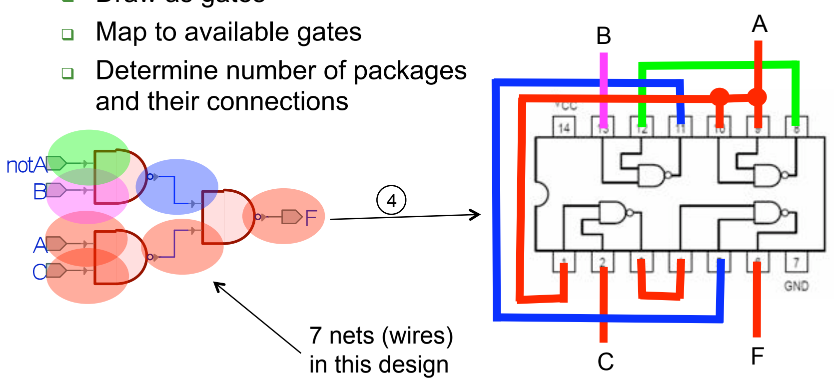
- de Morgan's
  - Standard form: A'B' = (A + B)' A' + B' = (AB)'
  - Inverted: A + B = (A'B')' (AB) = (A' + B')'

  - AND with complemented inputs = NOR
  - OR with complemented inputs = NAND
  - OR = NAND with complemented inputs
  - AND = NOR with complemented inputs

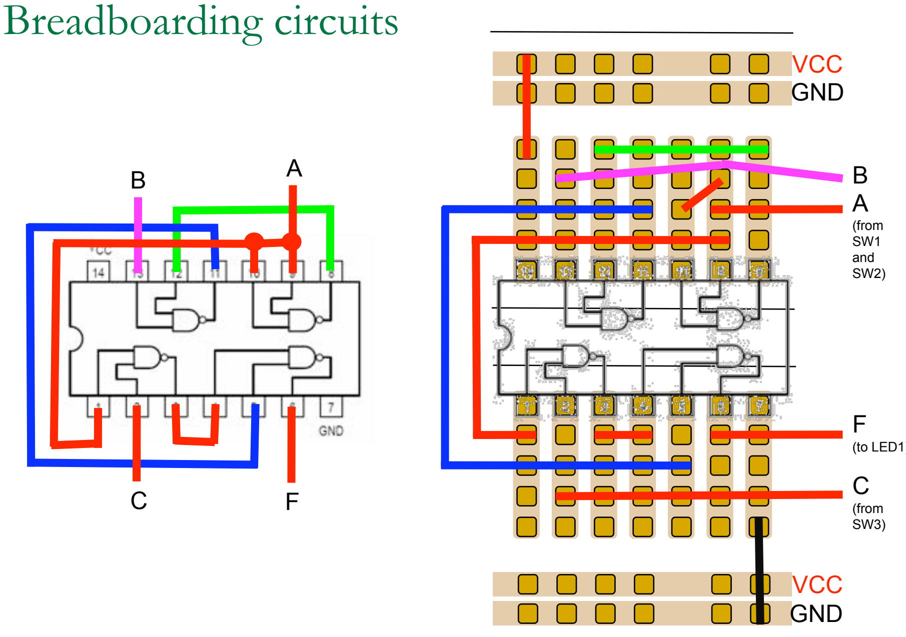


Mapping truth tables to logic gates

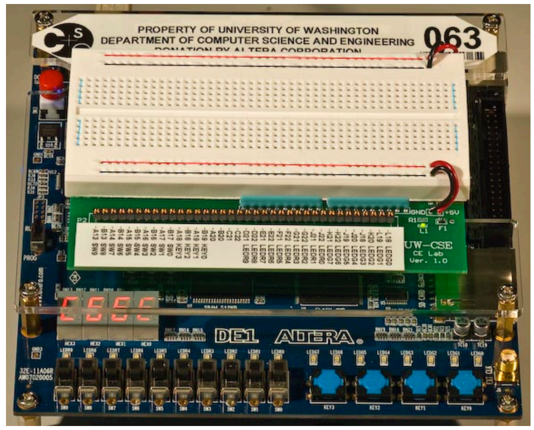
- Given a truth table:
  - Write the Boolean expression
  - Minimize the Boolean expression
  - Draw as gates
  - Map to available gates
  - Determine number of packages and their connections

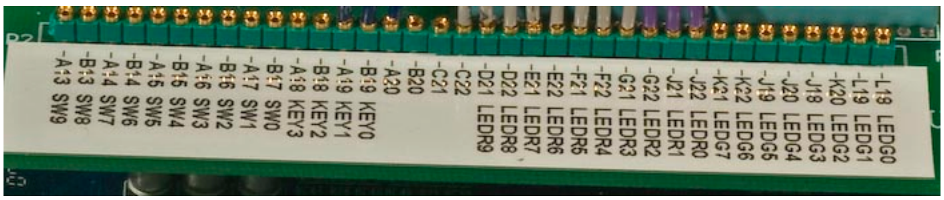


Breadboarding circuits



Lab 1 equipment





Random logic

- Too hard to figure out exactly what gates to use
  - map from logic to NAND/NOR networks
  - determine minimum number of packages
    - slight changes to logic function could decrease cost
- Changes too difficult to realize
  - need to rewire parts
  - may need new parts
  - design with spares (few extra inverters and gates on every board)
- Need higher levels of integration to keep costs down
  - cost directly related to number of devices and their pins

Regular logic

- Need to make design faster
- Need to make engineering changes easier to make
- Simpler for designers to understand and map to functionality
  - harder to think in terms of specific gates
  - easier to think in terms of larger multi-purpose blocks

Canonical forms

- Truth table is the unique signature of a Boolean function
- The same truth table can have many gate realizations
  - we've seen this already
  - depends on how good we are at Boolean simplification
- Canonical forms
  - standard forms for a Boolean expression
  - we all come up with the same expression

Sum-of-products canonical forms

- Also known as disjunctive normal form
- Also known as minterm expansion

F = *001 011 101 110 111*

F = A'B'C + A'BC + AB'C + ABC' + ABC

A	B	C	F	F'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

F' = A'B'C' + A'BC' + AB'C'

Sum-of-products canonical form (cont'd)

- Product term (or minterm)
  - ANDed product of literals – input combination for which output is true
  - each variable appears exactly once, true or inverted (but not both)

A	B	C	minterms	
0	0	0	A'B'C'	m0
0	0	1	A'B'C	m1
0	1	0	A'BC'	m2
0	1	1	A'BC	m3
1	0	0	AB'C'	m4
1	0	1	AB'C	m5
1	1	0	ABC'	m6
1	1	1	ABC	m7

short-hand notation for minterms of 3 variables

F in canonical form:

F(A, B, C) = Σm(1,3,5,6,7)
= m1 + m3 + m5 + m6 + m7
= A'B'C + A'BC + AB'C + ABC' + ABC

canonical form $\neq$ minimal form

F(A, B, C) = A'B'C + A'BC + AB'C + ABC + ABC'
= (A'B' + A'B + AB' + AB)C + ABC'
= ((A' + A)(B' + B))C + ABC'
= C + ABC'
= ABC' + C
= AB + C

Product-of-sums canonical form

- Also known as conjunctive normal form
- Also known as maxterm expansion

F = *000* *010* *100*

F = (A + B + C) (A + B′ + C) (A′ + B + C)

A	B	C	F	F′
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

F′ = (A + B + C′) (A + B′ + C′) (A′ + B + C′) (A′ + B′ + C) (A′ + B′ + C′)

Product-of-sums canonical form (cont’d)

- Sum term (or maxterm)
  - ORed sum of literals – input combination for which output is false
  - each variable appears exactly once, true or inverted (but not both)

A	B	C	maxterms	
0	0	0	A+B+C	M0
0	0	1	A+B+C′	M1
0	1	0	A+B′+C	M2
0	1	1	A+B′+C′	M3
1	0	0	A′+B+C	M4
1	0	1	A′+B+C′	M5
1	1	0	A′+B′+C	M6
1	1	1	A′+B′+C′	M7

short-hand notation for maxterms of 3 variables

F in canonical form:

F(A, B, C) = ΠM(0,2,4)
= M0 • M2 • M4
= (A + B + C) (A + B′ + C) (A′ + B + C)

canonical form ≠ minimal form

F(A, B, C) = (A + B + C) (A + B′ + C) (A′ + B + C)
= (A + B + C) (A + B′ + C)
(A + B + C) (A′ + B + C)
= (A + C) (B + C)

S-o-P, P-o-S, and de Morgan's theorem

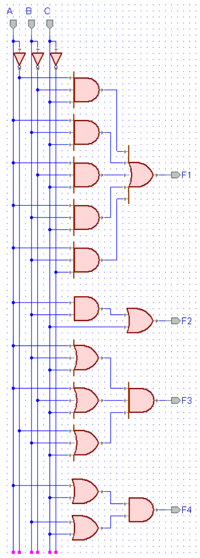
- Complement of function in sum-of-products form
  - F' = A'B'C' + A'BC' + AB'C'
- Complement again and apply de Morgan's and get the product-of-sums form
  - (F')' = (A'B'C' + A'BC' + AB'C')'
  - F = (A + B + C) (A + B' + C) (A' + B + C)

- Complement of function in product-of-sums form
  - F' = (A + B + C') (A + B' + C') (A' + B + C') (A' + B' + C) (A' + B' + C')
- Complement again and apply de Morgan's and get the sum-of-product form
  - (F')' = ((A + B + C')(A + B' + C')(A' + B + C')(A' + B' + C)(A' + B' + C'))'
  - F = A'B'C + A'BC + AB'C + ABC' + ABC

Mapping between canonical forms

- Minterm to maxterm conversion
  - use maxterms whose indices do not appear in minterm expansion
  - e.g., F(A,B,C) = Σm(1,3,5,6,7) = ΠM(0,2,4)
- Maxterm to minterm conversion
  - use minterms whose indices do not appear in maxterm expansion
  - e.g., F(A,B,C) = ΠM(0,2,4) = Σm(1,3,5,6,7)
- Minterm expansion of F to minterm expansion of F'
  - use minterms whose indices do not appear
  - e.g., F(A,B,C) = Σm(1,3,5,6,7) F'(A,B,C) = Σm(0,2,4)
- Maxterm expansion of F to maxterm expansion of F'
  - use maxterms whose indices do not appear
  - e.g., F(A,B,C) = ΠM(0,2,4) F'(A,B,C) = ΠM(1,3,5,6,7)

Four alternative two-level implementations of F = AB + C



canonical sum-of-products

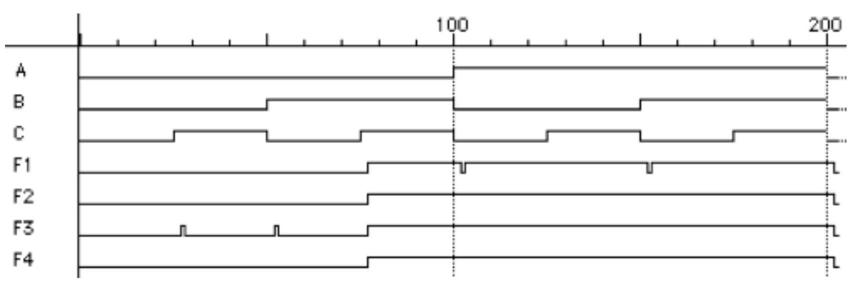
minimized sum-of-products

canonical product-of-sums

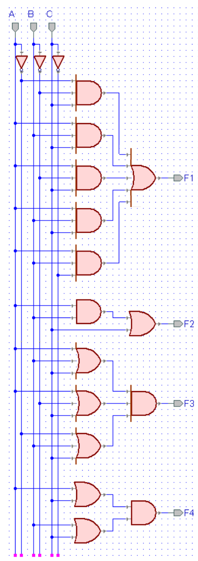
minimized product-of-sums

Waveforms for the four alternatives

- Waveforms are essentially identical
  - except for timing hazards (glitches)
  - delays almost identical (modeled as a delay per level, not type of gate or number of inputs to gate)



Four alternative two-level implementations of F = AB + C



	Transistors (NOT = 2)	Delay (approx) (NOT = 1)	Hazards
F1	*5 3-input NANDs* *1 5-input NAND* 5*6 + 1*10 = 40	*2 levels* 3^2 + 5^2 = 34	yes
F2	*2 2-input NANDs* 2*4 = 8	*2 levels* 2^2 + 2^2 = 8	no
F3	*4 3-input NANDs* 4*6 = 24	*2 levels* 3^2 + 3^2 = 18	yes
F4	*3 2-input NANDs* 3*4 = 12	*2 levels* 2^2 + 2^2 = 8	no

Incompleteley specified functions

- Example: binary coded decimal increment by 1
  - BCD digits encode the decimal digits 0 – 9 in the bit patterns 0000 – 1001

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

off-set of W

on-set of W

don't care (DC) set of W

these inputs patterns should never be encountered in practice – **"don't care"** about output values in these cases – might be useful in minimization

Notation for incompletely specified functions

- Don't cares and canonical forms
  - so far, only represented on-set
  - also represent don't-care-set
  - need two of the three sets (on-set, off-set, dc-set)
- Canonical representations of the BCD increment by 1 function:
  - Z = m0 + m2 + m4 + m6 + m8 + d10 + d11 + d12 + d13 + d14 + d15
  - Z = Σ [m(0,2,4,6,8) + d(10,11,12,13,14,15)]
  - Z = M1 • M3 • M5 • M7 • M9 • D10 • D11 • D12 • D13 • D14 • D15
  - Z = Π [M(1,3,5,7,9) • D(10,11,12,13,14,15)]