Spiral 1 / Unit 3

Minterm and Maxterms

Canonical Sums and Products

2- and 3-Variable Boolean Algebra Theorems

DeMorgan's Theorem

Function Synthesis use Canonical Sums/Products

Outcomes

- I know the difference between combinational and sequential logic and can name examples of each.
- I understand latency, throughput, and at least 1 technique to improve throughput
- I can identify when I need state vs. a purely combinational function
  - I can convert a simple word problem to a logic function (TT or canonical form) or state diagram
- I can use Karnaugh maps to synthesize combinational functions with several outputs
- I understand how a register with an enable functions & is built
- I can design a working state machine given a state diagram
- I can implement small logic functions with complex CMOS gates

SYNTHESIZING LOGIC FUNCTIONS

The Problem

- Given a logic function, how do we arrive at a circuit to implement this combinational function?

Primes between 0-7

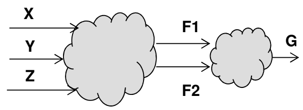
X	Y	Z	P
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

1's Count of Inputs

I3	I2	I1	C1	C0
0	0	0	0	0
0	0	1	0	1
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	1

Combining Functions

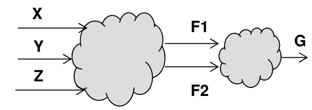
- Given intermediate functions F1 and F2, how could you use AND, OR, NOT to make G



X	Y	Z	F1	F2	G
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	1	1	0

Combining Functions

- Given intermediate functions F1 and F2, how could you use AND, OR, NOT to make G



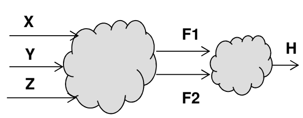
X	Y	Z	F1	F2	G
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	1	1	0

X	Y	Z	F1	F2	F1	F2'	_____
0	0	0	0	0	0	1	0
0	0	1	1	0	1	1	1
0	1	0	1	0	1	1	1
0	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1
1	0	1	1	0	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	0	0

G = _____

Combining Functions

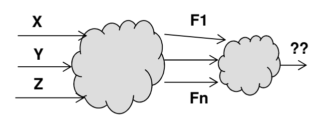
- Given intermediate functions F1 and F2, how could you use AND, OR, NOT to make H



X	Y	Z	F1	F2	H
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	0	1

Question

- Is there a set of functions (F1, F2, etc.) that would allow you to build ANY 3-variable function
  - Think simple, think many



X	Y	Z	F1	F2	Fn	?
0	0	0				?
0	0	1				?
0	1	0				?
0	1	1				?
1	0	0				?
1	0	1				?
1	1	0				?
1	1	1				?

X	Y	Z	m0	m1	m2	m3	m4	m5	m6	m7	?
0	0	0									?
0	0	1									?
0	1	0									?
0	1	1									?
1	0	0									?
1	0	1									?
1	1	0									?
1	1	1									?

OR together any combination of m_i's

Defining Minterms

- Remember these minterms are intermediate functions that we'll use to build larger functions
- Write the expression for each minterm of F(x,y,z)

						Minterms							
Row #		Minterm Expression	x	y	z	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	m_0		0	0	0	1	0	0	0	0	0	0	0
1	m_1		0	0	1	0	1	0	0	0	0	0	0
2	m_2		0	1	0	0	0	1	0	0	0	0	0
3	m_3		0	1	1	0	0	0	1	0	0	0	0
4	m_4		1	0	0	0	0	0	0	1	0	0	0
5	m_5		1	0	1	0	0	0	0	0	1	0	0
6	m_6		1	1	0	0	0	0	0	0	0	1	0
7	m_7		1	1	1	0	0	0	0	0	0	0	1

Applying Minterms to Synthesize a Function

- Each numbered minterm checks whether the inputs are equal to the corresponding combination. When the inputs are equal, the minterm will evaluate to 1 and thus the whole function will evaluate to 1.

x	y	z	P	
0	0	0	0	
0	0	1	0	use…
0	1	0	1	m_2
0	1	1	1	m_3
1	0	0	0	
1	0	1	1	m_5
1	1	0	0	
1	1	1	1	m_7

$P = m_2 + m_3 + m_5 + m_7$
$= x'yz' + x'yz + xy'z + xyz$

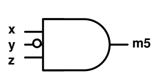
when x,y,z = {0,1,0} = 2 then
$P = 0'\bullet 1\bullet 0' + 0'\bullet 1\bullet 0 + 0\bullet 1'\bullet 0 + 0\bullet 1\bullet 0$
$= 1 + 0 + 0 + 0 = 1$

when x,y,z = {1,0,1} = 5 then
$P = 1'\bullet 0\bullet 1' + 1'\bullet 0\bullet 1 + 1\bullet 0'\bullet 1 + 1\bullet 0\bullet 1$
$= 0 + 0 + 1 + 0 = 1$

when x,y,z = {0,0,1} = 5 then
$P = 0'\bullet 0\bullet 1' + 0'\bullet 0\bullet 1 + 0\bullet 0'\bullet 1 + 0\bullet 0\bullet 1$
$= 0 + 0 + 0 + 0 = 0$

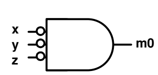
Checkers / Decoders

- The m_i functions on the previous slide are just AND gate checkers
  - That combination can be changed by adding inverters to the inputs
  - We can think of the AND gate as "checking" or "decoding" a specific combination and outputting a '1' when it matches.



AND gate decoding (checking for) combination 101

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



AND gate decoding (checking for) combination 000

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Minterms

- A minterm can be generated for every combination of inputs
- Each minterm is the AND'ing of variables that will evaluate to 1 for only that combination
- A minterm "checks" or "decodes" a specific input combination and outputs 1 when found

Minterm 3
$011 = x'\bullet y\bullet z = m_3$

Minterm 5
$101 = x\bullet y'\bullet z = m_5$

To make the minterm, complement the variables that equal 0 and leave the variables in their true form that equal 1.

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Using Decoders to Implement Functions

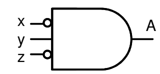
- Given an any logic function, it can be implemented with the superposition of decoders/checkers

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Using Decoders to Implement Functions

- Given an any logic function, it can be implemented with the superposition of decoders

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

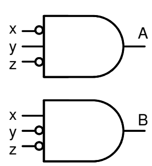


X	Y	Z	A
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Using Decoders to Implement Functions

- Given an any logic function, it can be implemented with the superposition of decoders

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

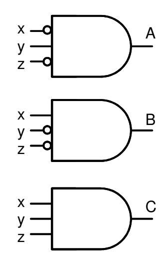


X	Y	Z	A	B
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	0	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

Using Decoders to Implement Functions

- Given an any logic function, it can be implemented with the superposition of decoders

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

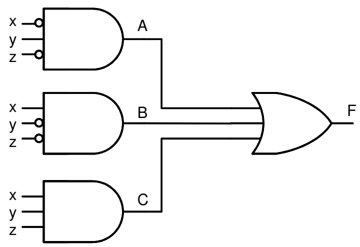


X	Y	Z	A	B	C
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	0	1

Using Decoders to Implement Functions

- Given an any logic function, it can be implemented with the superposition of decoders

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

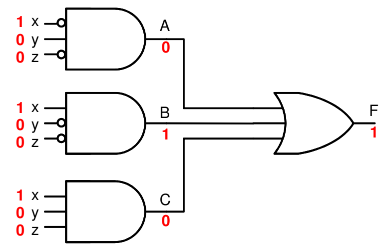


X	Y	Z	A	B	C	F
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	0	0	0
1	0	0	0	1	0	1
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	1	1

Using Decoders to Implement Functions

- Given an any logic function, it can be implemented with the superposition of decoders

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



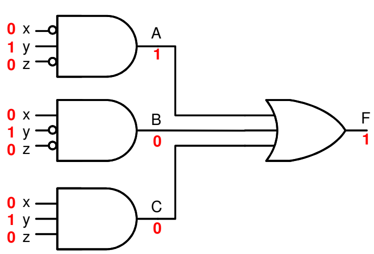
X	Y	Z	A	B	C	F
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	0	0	0
1	0	0	0	1	0	1
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	1	1

F(1,0,0) = 1

Using Decoders to Implement Functions

- Given an any logic function, it can be implemented with the superposition of decoders

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



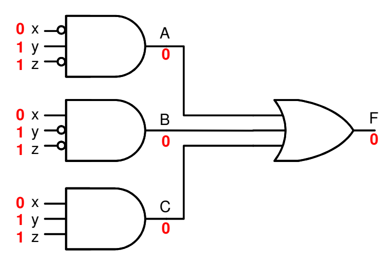
X	Y	Z	A	B	C	F
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	0	0	0
1	0	0	0	1	0	1
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	1	1

F(0,1,0) = 1

Using Decoders to Implement Functions

- Given an any logic function, it can be implemented with the superposition of decoders

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



X	Y	Z	A	B	C	F
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	0	0	0
1	0	0	0	1	0	1
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	1	1

F(0,1,1) = 0

Minterm Definition

- **Minterm**: A product term where each input variable of a function appears as exactly one literal
  - Are the following minterms of f(x,y,z) =>
    - x'y'z
    - xyz
    - x'y
    - x+y+z

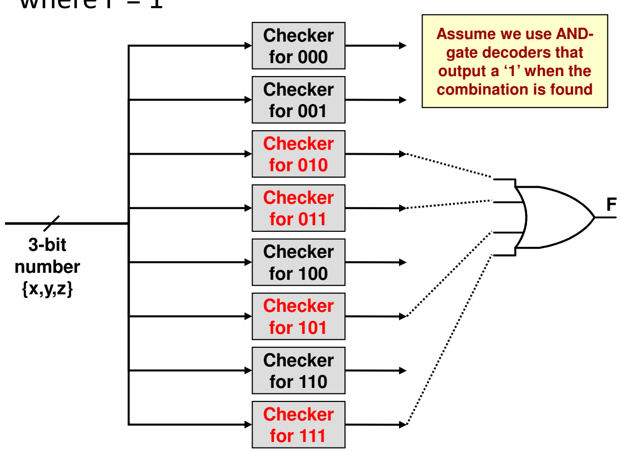
Minterms

- Consider F(A,B)
- One minterm per combination of the input variables
- Only *one* minterm can evaluate to 1 at any time

			m_0	m_1	m_2	m_3
A	B	F	A'•B'	A'•B	A•B'	A•B
0	0	0	1	0	0	0
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1

Finding Equations/Circuits

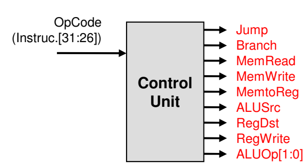
- Given a function and checkers (called decoders) for each combination, we just need to OR together the checkers where F = 1



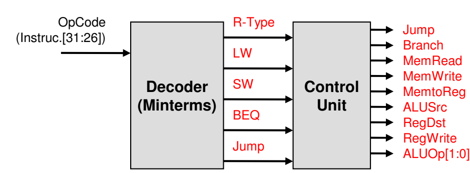
X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Control Signal Generation

- Other control signals are a function of the opcode
- We could write a full truth table or (because we are only implementing a small subset of instructions) simply decode the opcodes of the specific instructions we are implementing and use those intermediate signals to generate the actual control signals



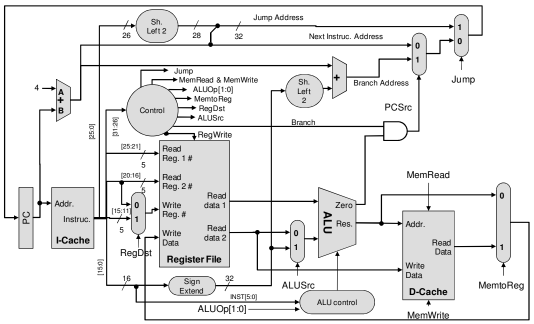
Could generate each control signal by writing a full truth table of the 6-bit opcode



Simpler for human to design if we decode the opcode and then use individual "instruction" signals to generate desired control signals

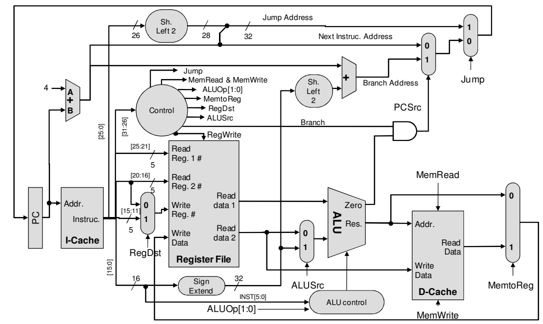
Control Signal Truth Table

OpCode [5:0]	R-Type	LW	SW	BEQ	J	Jump	Branch	Reg Dst	ALU Src	Memto-Reg	Reg Write	Mem Read	Mem Write	ALU Op[1]	ALU Op[0]
000000	1	0	0	0	0	0	0							1	0
100011	0	1	0	0	0	0	0							0	0
101011	0	0	1	0	0	0	0							0	0
000100	0	0	0	1	0	0	1							0	1
000010	0	0	0	0	1	1	0							X	X

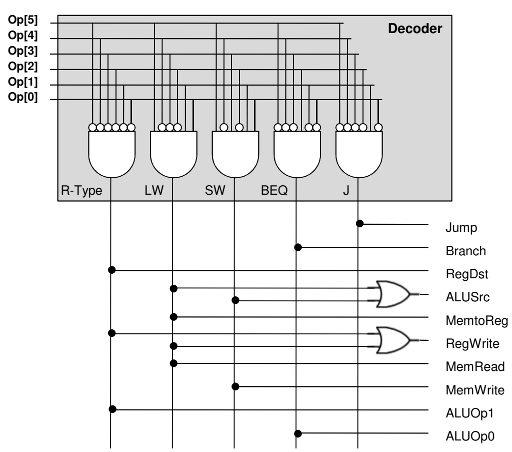


Control Signal Truth Table

OpCode [5:0]	R-Type	LW	SW	BEQ	J	Jump	Branch	Reg Dst	ALU Src	Memto-Reg	Reg Write	Mem Read	Mem Write	ALU Op[1]	ALU Op[0]
000000	1	0	0	0	0	0	0	1	0	0	1	0	0	1	0
100011	0	1	0	0	0	0	0	0	1	1	1	1	0	0	0
101011	0	0	1	0	0	0	0	X	1	X	0	0	1	0	0
000100	0	0	0	1	0	0	1	X	0	X	0	0	0	0	1
000010	0	0	0	0	1	1	0	X	X	X	0	0	0	X	X



Control Signal Logic

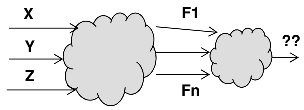


Using products of maxterms to implement a function

MAXTERMS

Question

- Is there a set of functions (F1, F2, etc.) that would allow you to build ANY 3-variable function
  - Think simple, think many



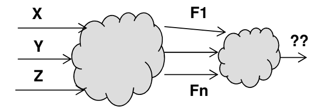
X	Y	Z	F1	F2	Fn	?
0	0	0				?
0	0	1				?
0	1	0				?
0	1	1				?
1	0	0				?
1	0	1				?
1	1	0				?
1	1	1				?

X	Y	Z	m0	m1	m2	m3	m4	m5	m6	m7	?
0	0	0	1	0	0	0	0	0	0	0	?
0	0	1	0	1	0	0	0	0	0	0	?
0	1	0	0	0	1	0	0	0	0	0	?
0	1	1	0	0	0	1	0	0	0	0	?
1	0	0	0	0	0	0	1	0	0	0	?
1	0	1	0	0	0	0	0	1	0	0	?
1	1	0	0	0	0	0	0	0	1	0	?
1	1	1	0	0	0	0	0	0	0	1	?

OR together any combination of m_i's

Question

- OR…this set of functions would also work.



X	Y	Z	M0	M1	M2	M3	M4	M5	M6	M7	?
0	0	0									?
0	0	1									?
0	1	0									?
0	1	1									?
1	0	0									?
1	0	1									?
1	1	0									?
1	1	1									?

G
1
0
1
0
1
1
0
1

AND together any combination of M_i's

G = M1 • M3 • M6

Defining Maxterms

- Remember these maxterms are intermediate functions that we'll use to build larger functions
- Write the expression for each Maxterm of F(x,y,z)

						Maxterms							
Row #		Maxterm	x	y	z	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	M_0		0	0	0	0	1	1	1	1	1	1	1
1	M_1		0	0	1	1	0	1	1	1	1	1	1
2	M_2		0	1	0	1	1	0	1	1	1	1	1
3	M_3		0	1	1	1	1	1	0	1	1	1	1
4	M_4		1	0	0	1	1	1	1	0	1	1	1
5	M_5		1	0	1	1	1	1	1	1	0	1	1
6	M_6		1	1	0	1	1	1	1	1	1	0	1
7	M_7		1	1	1	1	1	1	1	1	1	1	0

Applying Maxterms to Synthesize a Function

- Each numbered maxterm checks whether the inputs are equal to the corresponding combination. When the inputs are equal, the maxterm will evaluate to 0 and thus the whole function will evaluate to 0.

x	y	z	P	use…
0	0	0	0	M_0
0	0	1	0	M_1
0	1	0	1	
0	1	1	1	
1	0	0	0	M_4
1	0	1	1	
1	1	0	0	M_6
1	1	1	1	

$$P = M_0 \bullet M_1 \bullet M_4 \bullet M_6$$
$$= (x+y+z) \bullet (x+y+z') \bullet (x'+y+z) \bullet (x'+y'+z)$$

when x,y,z = {0,0,1} = 1 then

$$P = (0+0+1) \bullet \mathbf{(0+0+1')} \bullet (0'+0+1) \bullet (0'+0'+1)$$
$$= 1 \bullet \mathbf{0} \bullet 1 \bullet 1 = \mathbf{0}$$

when x,y,z = {1,1,0} = 6 then

$$P = (1+1+0) \bullet (1+1+0') \bullet (1'+1+0) \bullet \mathbf{(1'+1'+0)}$$
$$= 1 \bullet 1 \bullet 1 \bullet \mathbf{0} = \mathbf{0}$$

when x,y,z = {1,1,1} = 7 then

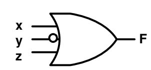
$$P = (1+1+1) \bullet (1+1+1') \bullet (1'+1+1) \bullet (1'+1'+1)$$
$$= 1 \bullet 1 \bullet 1 \bullet 1 = 1$$

Maxterm Definition

- **Maxterm**: A sum term where each input variable of a function appears exactly once in that term (either in its true or complemented form)
  - f(x,y,z) =>
    - x'+y'+z
    - x+y+z
    - y+z'
    - x'y'z'

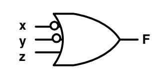
Checkers / Decoders

- An OR gate only outputs '0' for 1 combination
  - That combination can be changed by adding inverters to the inputs
  - We can think of the OR gate as "checking" or "decoding" a specific combination and outputting a '0' when it matches.



OR gate decoding (checking for) combination 010

X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

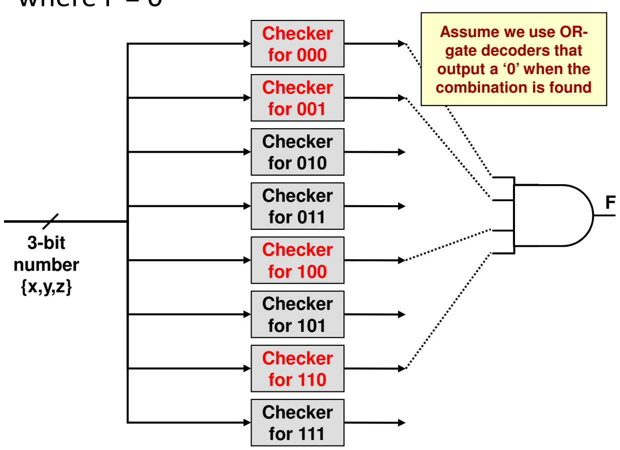


OR gate decoding (checking for) combination 110

X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Finding Equations/Circuits

- Given a function and checkers (called decoders) for each combination, we just need to AND together the checkers where F = 0



X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

LOGIC FUNCTION NOTATION

Canonical Sums

- We ______ together all the minterms where F = 1
  - (Σ = SUM or OR of all the minterms)

$F = m_2 + m_3 + m_5 + m_7$

Canonical Sum:

$F = \Sigma_{xyz}(2,3,5,7)$

List the minterms where F is 1.

	X	Y	Z	F
m_0	0	0	0	0
m_1	0	0	1	0
m_2	0	1	0	1
m_3	0	1	1	1
m_4	1	0	0	0
m_5	1	0	1	1
m_6	1	1	0	0
m_7	1	1	1	1

Canonical Products

- We _____ together all the maxterms where F = 0

$F = M_0 \bullet M_1 \bullet M_4 \bullet M_6$

Canonical Product:

$F = \Pi_{xyz}(0,1,4,6)$

List the maxterms where F is 0.

	X	Y	Z	F
M_0	0	0	0	0
M_1	0	0	1	0
M_2	0	1	0	1
M_3	0	1	1	1
M_4	1	0	0	0
M_5	1	0	1	1
M_6	1	1	0	0
M_7	1	1	1	1

Canonical Form Practice

- $G = \Sigma_{XYZ}(\qquad\qquad) = \prod_{XYZ}(\qquad\qquad)$

- $B = \Sigma_{X,Y,Z}(5,6,7)$

- $F =$

X	Y	Z	G
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

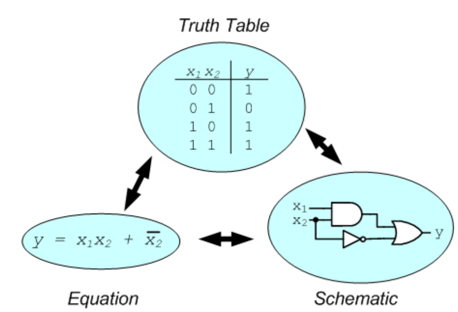
X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

P. 60 and 61 in the Lecture Notes

Logic Functions

- A logic function maps input combinations to an output value ('1' or '0')
- 3 possible representations of a function
  - Equation
  - Schematic
  - Truth Table
- Can convert between representations
- Truth table is only unique representation*

* Canonical Sums/Products (minterm/maxterm) representation provides a standard equation/schematic form that is unique per function



Unique Representations

- Canonical => Same functions will have same representations
- Truth Tables along with Canonical Sums and Products specify a function *uniquely*
- Equations/circuit schematics are NOT inherently canonical

Truth Table

x	y	z	P
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Canonical Sum

$$P = \sum_{x,y,z}(2,3,5,7)$$

ON-Set of P (minterms)

Yields AND-OR circuit

Canonical Product

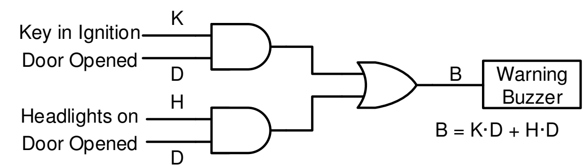
$$P = \prod_{x,y,z}(0,1,4,6)$$

OFF-Set of P (maxterms)

Yields OR-AND circuit

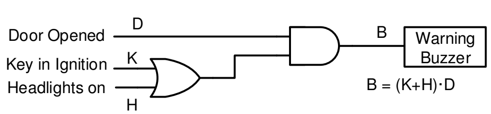
Example: Automobile Buzzer

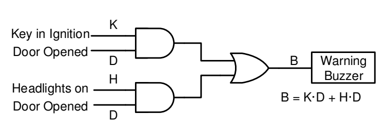
- Consider an automobile warning **B**uzzer that sounds if you leave the **K**ey in the ignition and the **D**oor is open OR the **H**eadlights are on and the **D**oor is open.
- We can easily derive an equation and implementation: B = KD + HD



Example: Automobile Buzzer

- But we see that we can alter this equation…
  - From B = KD + HD
  - To B = D(K+H)
    - Buzzer sounds if the Door is open and ***either*** the Key is in the Ignition or the Headlights are on
- Which is better?
- What is the canonical minterm/maxterm representation?





Example Form

- Given a function, ***B(D,K,H)*** we can define the minterm functions (which serve as intermediate functions) and then generate the overall function from the minterms
  - B = Σ
  - B = Π

Row	D	K	H	Minterm	Designation	Maxterm	Designation	B
0	0	0	0	D'•K'•H'	m_0	D+K+H	M_0	0
1	0	0	1	D'•K'•H	m_1	D+K+H'	M_1	0
2	0	1	0	D'•K•H'	m_2	D+K'+H	M_2	0
3	0	1	1	D'•K•H	m_3	D+K'+H'	M_3	0
4	1	0	0	D•K'•H'	m_4	D'+K+H	M_4	0
5	1	0	1	D•K'•H	m_5	D'+K+H'	M_5	1
6	1	1	0	D•K•H'	m_6	D'+K'+H	M_6	1
7	1	1	1	D•K•H	m_7	D'+K'+H'	M_7	1

2 & 3 Variable Theorems

T6	X+Y = Y+X	**T6'**	X•Y = Y•X	Commutativity
T7	(X+Y)+Z = X+(Y+Z)	**T7'**	(X•Y)•Z = X•(Y•Z)	Associativity
T8	XY+XZ = X(Y+Z)	**T8'**	(X+Y)(X+Z) = X+YZ	Distribution & Factoring
T9	X + XY = X	**T9'**	X(X+Y) = X	Covering
T10	XY + XY' = X	**T10'**	(X+Y)(X+Y') = X	Combining
T11	XY + X'Z + YZ = XY + X'Z	**T11'**	(X+Y)(X'+Z)(Y+Z) = (X+Y)(X'+Z)	Consensus
DM	(X+Y)' = X'•Y'	**DM'**	(X•Y)'=X'+Y'	DeMorgan's

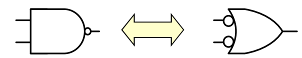
DeMorgan's Theorem

- Inverting output of an AND gate = inverting the inputs of an OR gate
- Inverting output of an OR gate = inverting the inputs of an AND gate

A function's inverse is equivalent to inverting all the inputs and changing AND to OR and vice versa

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

$\overline{A \bullet B}$

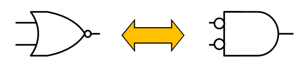


$\overline{A}+\overline{B}$

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

$\overline{A+B}$

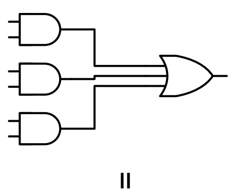


$\overline{A} \bullet \overline{B}$

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

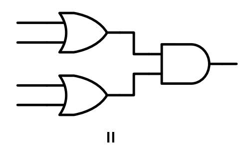
AND-OR / NAND-NAND

- Canonical Sums yield
  - AND-OR Implementation
  - NAND-NAND Implementation



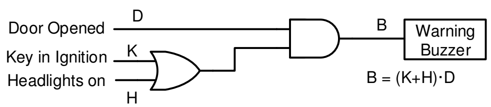
OR-AND / NOR-NOR

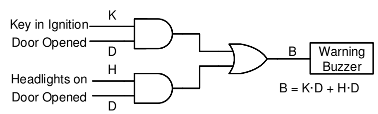
- Canonical Products yield
  - OR-AND Implementation
  - NOR-NOR Implementation



Example: Automobile Buzzer

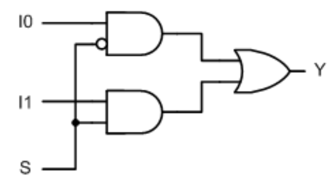
- Convert each implementation to use either just NOR or just NAND gates + inverters





Convert to NAND-NAND

- Convert the 2-to-1 mux below to use just NAND or NOR gates?



Logic Synthesis

- Describe the function
  - Usually with a truth table
- Find the sum of products or product of sums expression
  - Fewer 1's in the output => use canonical sum
  - Fewer 0's in the output => use canonical product
- Use Boolean Algebra (T8-T11) to find a simplified expression

Exercise 1

- Synthesize this function in two ways
  - First use the canonical sum
  - Then use the canonical product

T8	XY+XZ = X(Y+Z)	**T8'**	(X+Y)(X+Z) = X+YZ
T9	X + XY = X	**T9'**	X(X+Y) = X
T10	XY + XY' = X	**T10'**	(X+Y)(X+Y') = X
T11	XY + X'Z + YZ = XY + X'Z	**T11'**	(X+Y)(X'+Z)(Y+Z) = (X+Y)(X'+Z)

Primes between 0-7

X	Y	Z	P
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Exercise 2

- Synthesize this function in two ways
  - First use the canonical sum
  - Then use the canonical product

T8	XY+XZ = X(Y+Z)	T8'	(X+Y)(X+Z) = X+YZ
T9	X + XY = X	T9'	X(X+Y) = X
T10	XY + XY' = X	T10'	(X+Y)(X+Y') = X
T11	XY + X'Z + YZ = XY + X'Z	T11'	(X+Y)(X'+Z)(Y+Z) = (X+Y)(X'+Z)

I3	I2	I1	M1	M0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Encode the highest input ID (ie. 3, 2, or 1) that is ON (=1)

Exercise 3

- Synthesize this function in two ways
  - First use the canonical sum
  - Then use the canonical product

T8	XY+XZ = X(Y+Z)	T8'	(X+Y)(X+Z) = X+YZ
T9	X + XY = X	T9'	X(X+Y) = X
T10	XY + XY' = X	T10'	(X+Y)(X+Y') = X
T11	XY + X'Z + YZ = XY + X'Z	T11'	(X+Y)(X'+Z)(Y+Z) = (X+Y)(X'+Z)

1's Count of Inputs

I3	I2	I1	C1	C0
0	0	0	0	0
0	0	1	0	1
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	1