Deriving logic functions from truth tables

- Given a truth table, we might want to derive a logic function (the function is easier to manipulate).
- Two simple ways to derive a logic function — one method will yield a *Sum-Of-Products* (SOP) representation for the logic function while the other method will yield a *Product-Of-Sums* (POS) representation for the logic function.

Minterms

- Consider a truth table for an n-input function with 2^n rows.
- For *each* row, create an **AND** of *all* inputs according to the following rule:
  - If the input variable has value 1 in the current row, then include the variable uncomplemented (the positive literal);
  - If the input variable has value 0 in the current row, then include the variable complemented (the negative literal).
- The result of applying the above rule yields the so-called "minterm" for each row of the truth table.
  - An n-input function has 2^n minterms (one for each row) and all are different.

Minterms

- Example for a 3-input function...

x	y	z	minterm	
0	0	0	$!x!y!z$	$= m_0$
0	0	1	$!x!yz$	$= m_1$
0	1	0	$!xy!z$	$= m_2$
0	1	1	$!xyz$	$= m_3$
1	0	0	$x!y!z$	$= m_4$
1	0	1	$x!yz$	$= m_5$
1	1	0	$xy!z$	$= m_6$
1	1	1	xyz	$= m_7$

- Minterms are denoted by the lower-case letter "m_i" to represent the minterm for the i-th row of the truth table.
- Property of minterms: The minterm evaluates to 1 if and only if the corresponding input pattern appears, otherwise it evaluates to 0.

Canonical Sum-Of-Products

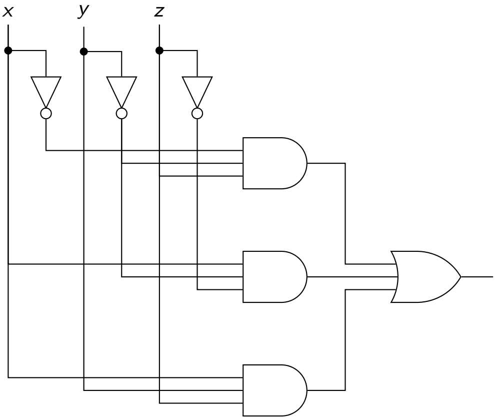
- Given the truth table for a logic function, we can **ALWAYS** write down a logic expression by taking the **OR** of the **MINTERMS** for which the function is 1.
  - The resulting expression if "canonical" (i.e., unique) and is called the Canonical Sum-Of-Products (SOP) or Sum-Of-Minterms representation for the function.
- Example...

x	y	z	f	
0	0	0	0	
0	0	1	1	$!x!yz = m_1$
0	1	0	0	
0	1	1	0	
1	0	0	1	$x!y!z = m_4$
1	0	1	0	
1	1	0	0	
1	1	1	1	$xyz = m_7$

  We can write $f(x, y, z) = m_1 + m_4 + m_7 =!x!yz + x!y!z + xyz$.
- Shortcut notation exists...
  $f(x, y, z) = m_1 + m_4 + m_7 = \underbrace{\sum(1, 4, 7)}_{\text{minterms required for } f}.$

SOP circuit implementations

- When implemented as an SOP, a function f always has the same "structure"; a "plane" of **NOT**, followed by a "plane" of **AND** followed by a single **OR**.



$f = !x!yz + x!y!z + xyz$

- Note that SOP implementations of f composed of minterms are not cheap!

Maxterms

- Consider a truth table for an n-input function with 2^n rows.
- For *each* row, create an **OR** of *all* inputs according to the following rule:
  - If the input variable has value 0 in the current row, then include the variable uncomplemented (the positive literal);
  - If the input variable has value 1 in the current row, then include the variable complemented (the negative literal).
- The result of applying the above rule yields the so-called "maxterm" for each row of the truth table.
  - An n-input function has 2^n maxterms (one for each row) and all are different.

Maxterms

- Example for a 3-input function...

x	y	z	minterm	
0	0	0	$x + y + z$	$= M_0$
0	0	1	$x + y + !z$	$= M_1$
0	1	0	$x + !y + z$	$= M_2$
0	1	1	$x + !y + !z$	$= M_3$
1	0	0	$!x + y + z$	$= M_4$
1	0	1	$!x + y + !z$	$= M_5$
1	1	0	$!x + !y + z$	$= M_6$
1	1	1	$!x + !y + !z$	$= M_7$

- Maxterms are denoted by the upper-case letter "M_i" to represent the maxterm for the i-th row of the truth table.
- Property of maxterms: The maxterm evaluates to 0 if and only if the corresponding input pattern appears, otherwise it evaluates to 1.

Canonical Product-Of-Sums

- Given the truth table for a logic function, we can **ALWAYS** write down a logic expression by taking the **AND** of the **MAXTERMS** for which the function is 0.
  - The resulting expression if "canonical" (i.e., unique) and is called the Canonical Product-Of-Sums (POS) or Product-Of-Maxterms representation for the function.
- Example...

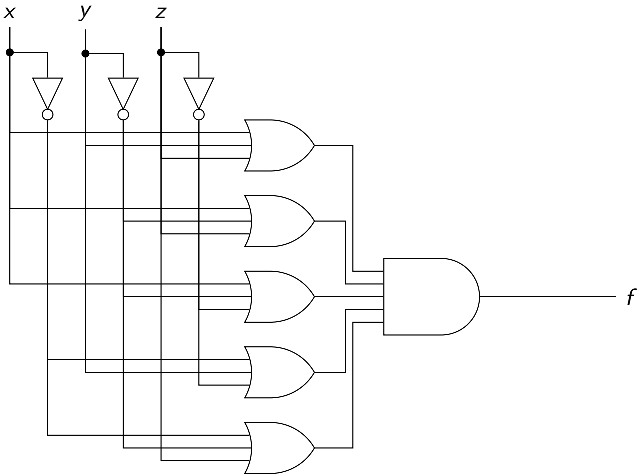
x	y	z	f	
0	0	0	0	$x + y + z = M_0$
0	0	1	1	
0	1	0	0	$x + !y + z = M_2$
0	1	1	0	$x + !y + !z = M_3$
1	0	0	1	
1	0	1	0	$!x + y + !z = M_5$
1	1	0	0	$!x + !y + z = M_6$
1	1	1	1	

We can write $f(x, y, z) = M_0 M_2 M_4 M_5 M_6 =$
$(x + y + z)(x + !y + z)(x + !y + !z)(!x + y + !z)(!x + !y + z)$

- Shortcut notation exists...
$f(x, y, z) = M_0 M_2 M_4 M_5 M_6 = \underbrace{\Pi(0, 2, 4, 5, 6)}_{\text{maxterms required for } f}$.

POS circuit implementations

- When implemented as an POS, a function f always has the same "structure"; a "plane" of **NOT**, followed by a "plane" of **OR** followed by a single **AND**.



$$f = (x + y + z)(x+!y + z)(x+!y+!z)(!x + y+!z)(!x+!y + z)$$

- Note that POS implementations of f composed of maxterms are not cheap!

Conversion and equality of representations

- Minterms and maxterms are "*duals*" of each other; $m_i = !M_i$ (and $M_i = !m_i$).
- You can always convert from one canonical represention to the other — We can convert from minterms to maxterms by changing $\sum$ to Π and list those terms missing from the original list.
- The reverse (maxterms to minterms) works the same way.
  - Convert the canonical SOP $f(x, y, z) = \sum(1, 4, 7)$ to canonical POS...

    Solution is $f(x, y, z) = \underbrace{\Pi}_{\text{changed from } \sum \text{ to } \Pi} \overbrace{(0, 2, 3, 4, 5, 6)}^{\text{terms not in minterm list}}$.

  - $$\begin{aligned} f &= \textstyle\sum(1, 4, 7) \\ &= m_1 + m_4 + m_7 \\ &= !(!f) \\ &= !(m_0 + m_2 + m_3 + m_5 + m_6) \\ &= !m_0 !m_2 !m_3 !m_5 !m_6 \\ &= M_0 M_2 M_3 M_5 M_6 \\ &= \Pi(0, 2, 4, 5, 6) \end{aligned}$$

Additional comments

- SOP and POS representations of a logic function f are often referred to as the so-called "*two-level*" implementations of f.
  - The is because if we ignore the inverters (e.g., because we assume that input variables are available either complemented or un-complemented), f is the output of two levels of logic gates (AND the OR for SOP; OR then AND for POS).

Standard Sum-Of-Product (SOP) implementations

- A function f implemented as a canonical SOP using minterms is by no means minimal.
- Let any AND of input literals be called a **product term**.
- We can then express any function f as a Sum-Of-Products where, instead of using minterms, we use product terms.
  - The resulting SOP is typically simpler and of lower cost than the canonical SOP.
- This simplification can, for example, be obtained by Boolean algebra.

Standard Sum-Of-Product (SOP) implementations

- Example...

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

  We can write $f(x, y, z) = !xyz + x!yz + xy!z + xyz$.

- However, Boolean algebra can be used to show that $f(x, y, z) = xz + zy + yz$ which is simpler and composed of 3 product terms (none of which is a minterm).
- Note: A minterm is a product term, but a product term is not necessarily a minterm!

Standard Product-Of-Sum (POS) implementations

- A function f implemented as a canonical POS using maxterms is by no means minimal.
- Let any OR of input literals be called a **sum term**.
- We can then express any function f as a Product-Of-Sums where, instead of using maxterms, we use sum terms.
  - The resulting POS is typically simpler and of lower cost than the canonical POS.
- This simplification can, for example, be obtained by Boolean algebra.

Standard Product-Of-Sum (POS) implementations

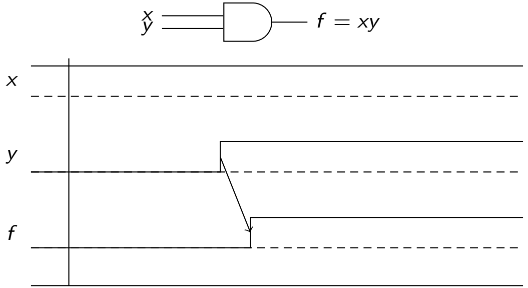
- Example...

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

  We can write
  $f(x, y, z) = (x + y + z)(x+!y + z)(x+!y+!z)(!x + y+!z)(!x+!y + z)$.
- However, Boolean algebra can be used to show that $f(x, y, z) = (x + z)(x+!y)(!y + z)(!x + y+!z)$ which is simpler and composed of 4 sum terms (one of which IS a maxterm).
- Note: A maxterm is a sum term, but a sum term is not necessarily a maxterm!

Propagation delay through gates

- I want to mention it now so that there is no confusion...
- Theoretically, there is no "time" involved with working with logic functions.
- However, when we actually *make* a circuit using gates (which, in turn, are made from transistors) "time" is involved.
  - *Outputs do not change instantaneously!*
- When an input signal changes, there is a slight delay after which the output will change.
- The amount of delay is called "gate delay" or "propagation delay".



- Notice the small amount of delay between the time that y changes from $0 \to 1$ and the time that $f = xy$ changes from $0 \to 1$.