Introduction to Digital Logic

Lecture 6:

Logic Simplification

Negative Logic

Circuit Analysis

Warmup

- Consider F(w,x,y,z). Show the algebraic form of m4 and M4?
- Use Boolean algebra to find the minimal SOP expression for the function, $F = \Sigma_{a,b,s}(3,4,6,7)$
- Use Boolean algebra to find the minimal POS expression for the function, $G = \Pi_{x,y,z}(1,3,5,7)$

LOGIC SIMPLIFICATION

Design a Circuit

- Design a circuit to implement this truth table
- H = x'y'z' + x'y'z + xy'z'
- H = (x+y'+z)(x+y'+z')(x'+y+z')
  (x'+y'+z)(x'+y'+z')

X	Y	Z	H
0	0	0	**1**
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	**0**

2 & 3 Variable Theorems

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)

Logic Simplification Tips

- Apply DeMorgan's to get only literals
- Factor common subexpressions (use T8/T8'), then see if anything reduces or find something of the form of T9-T11 and apply those
  - Remember, single variables in T8-T11 can represent entire sub-expressions in an equation
  - It may be helpful to use T8/T8' to distribute and get the expression in SOP or POS form
- Replicate terms if needed

Use Your Theorems

- G = (A + D’ + C)(A+ C)

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)

- H = (ab)’c + bc’d + abc

Use Your Theorems

- J = a'bc + abc + abc'

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)

- K = ab'(c + bc'd + d'e)(a'cd)

Use Your Theorems

$$F = (\overline{X}+Y) + Z(Y+\overline{Z}) + X(\overline{Z}+Y)$$

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)

Simplification

$$F = (\overline{X}+Y) + Z(Y+\overline{Z}) + X(\overline{Z}+Y)$$

$$= \overline{X}+Y + ZY+Z\overline{Z} + X\overline{Z} +XY \qquad \textit{(T8 and T8')}$$

$$Z\overline{Z} \rightarrow 0$$

$$= \overline{X}+Y + YZ+ X\overline{Z} +XY \qquad \textit{(T5' and T1)}$$

$$= \overline{X}+Y(1 + Z + X) + X\overline{Z} \qquad \textit{(T8 and T2)}$$

$$1 + Z + X \rightarrow 1$$

$$= \overline{X}+Y + X\overline{Z} \qquad \textit{(T8 and T2)}$$

This doesn't look like it can be simplified further but it can.

$$= (\overline{X}+X)(\overline{X}+\overline{Z}) +Y \qquad \textit{(T8')}$$

$$= (\overline{X}+\overline{Z})+Y \qquad \textit{(T5 and T1')}$$

$$= \overline{X} + \overline{Z} +Y \qquad \textit{(Associativity)}$$

Deriving minterms or maxterms by expanding SOP/POS

EXPANSION

Expansion

W	X	Y	Z	F
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**0**
0	1	0	0	**1**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**1**
1	0	1	1	**1**
1	1	0	0	**0**
1	1	0	1	**0**
1	1	1	0	**0**
1	1	1	1	**0**

- F = w'x + wx'y
- Product term w'x =>
  - Checks for w,x = 0,1
  - What combinations does that cover?
  - w'xy'z' + w'xy'z + w'xyz' + w'xyz
  - 0100 + 0101 + 0110 + 0111
- Product term wx'y =>
  - Checks for w,x,y = 1,0,1
  - What combinations does that cover?
  - wx'yz' + wx'yz
  - 1010 + 1011

Expanding using T10

- We have said in an earlier problem:
  - "You will get a Good grade if you go to Class and do your Homework"
  - G = C•H
  - Does your grade depend on whether you like Pizza?
  - No, but we could introduce that as a variable:
  - G = C•H•P' + C•H•P
- Recall
  - T10: X = XY' + XY
  - T10': X = (X+Y')(X+Y)
- If a variable doesn't influence the function we can simply look for both combinations of that variable

Expanding Product/Sum Terms

- To add in **k** missing variables, take the current product/sum terms and expand it to 2^k terms…
  - Keep the current literals from the original
  - Add missing literals in all combinations of true/complemented form
- Examples (Add in w and y variables):
  - x'•z =>
    - w'x'y'z + w'x'yz + wx'y'z + wx'yz
  - x+z' =>
    - (w+x+y+z')(w+x+y'+z')(w'+x+y+z')(w'+x+y'+z')

Expand this

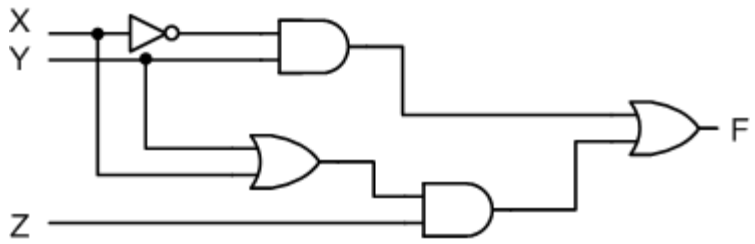
- What combinations of the inputs do the following functions cover (i.e. convert the equations to a canonical sum/product)
- y = as' + bs

- F= z' + w’xy

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

Analyze this

- What is this circuit doing?

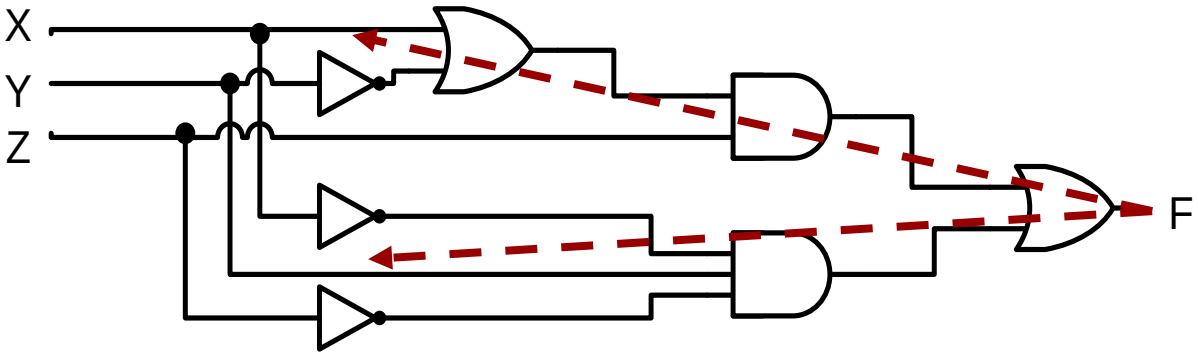


$$F = X'Y + Z(X+Y)$$

$$F_{SOP} = X'Y + XZ + YZ \qquad F_{POS} = (X'+Z)(Y+Z)(X+Y)$$

Circuit to Equations

- Start from the output and work your way back.
- Then convert equation to canonical sum or product



$$F = (X+Y')\bullet Z + X'\bullet Y\bullet Z'$$

$$F_{SOP} = XZ + Y'Z + X'YZ'$$

Expand terms:

$$= XZ(Y+Y') + Y'Z(X+X') + X'YZ'$$

$$= XYZ + XY'Z + XY'Z + X'Y'Z + X'YZ'$$

$$= XYZ + XY'Z + X'Y'Z + X'YZ'$$

$$= m_7 + m_5 + m_1 + m_2 = \Sigma_{XYZ}(1,2,5,7)$$

Converting to Canonical Sum

- 3 Steps:
  1. Use distributive property of T8 to convert to SOP form
  2. Then use T10 or T5 to add in missing variables (AND the term with the desired variable OR'ed with its complement)
  3. Redistribute using T8

w(x+yz)	**Original Expression**
wx+wyz	**T8**
wx•1•1 + w•1•yz	**T10**
wx(y+y')(z+z')+w(x+x')yz	**T10**
wxyz +wxyz' + wxy'z + wxy'z' + ~~wxyz~~ + wx'yz	**T8**
$m_{15} + m_{14} + m_{13} + m_{12} + m_{11}$	**Minterms in Canonical Sum**

Converting to Canonical Product

- 3 Steps:
  1. Use distributive property T8' to convert to POS form
  2. Then use T10 or T5 to add in missing variables (OR the term with the desired variable AND'ed with its complement)
  3. Redistribute using T8'

w(x+yz)	**Original Expression**
w(x+y)(x+z)	**T8'**
(w+0+0+0)(x+y+0+0)(x+z+0+0)	**T1**
(w+xx'+yy'+zz')(x+y+ww'+zz')(x+z+ww'+yy')	**T5**
(w+x+y+z) (w+x+y+z') (w+x+y'+z) (w+x+y'+z') (w+x'+y+z) (w+x'+y+z') (w+x'+y'+z) (w+x'+y'+z') ~~(w+x+y+z)~~ (w'+x+y+z) ~~(w+x+y+z')~~ (w'+x+y+z') ~~(w+x+y+z)~~ ~~(w'+x+y+z)~~ ~~(w+x+y'+z)~~ (w'+x+y'+z)	**T8'**
$M_0 \cdot M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot M_5 \cdot M_6 \cdot M_7 \cdot M_8 \cdot M_9 \cdot M_{10}$	**Maxterms in Canonical Prod.**

OLD

Single Variable Theorem (T1)

X+0 = X (T1)

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

OR

Hold Y constant

X•1 = X (T1')

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

AND

Whenever a variable is OR'ed with 0, the output will be the same as the variable…

"0 OR Anything equals that anything"

Whenever a variable is AND'ed with 1, the output will be the same as the variable…

"1 AND Anything equals that anything"

Single Variable Theorem (T2)

X+1 = 1 (T2)

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

OR

Hold Y constant

X•0 = 0 (T2')

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

AND

Whenever a variable is OR'ed with 1, the output will be 1…

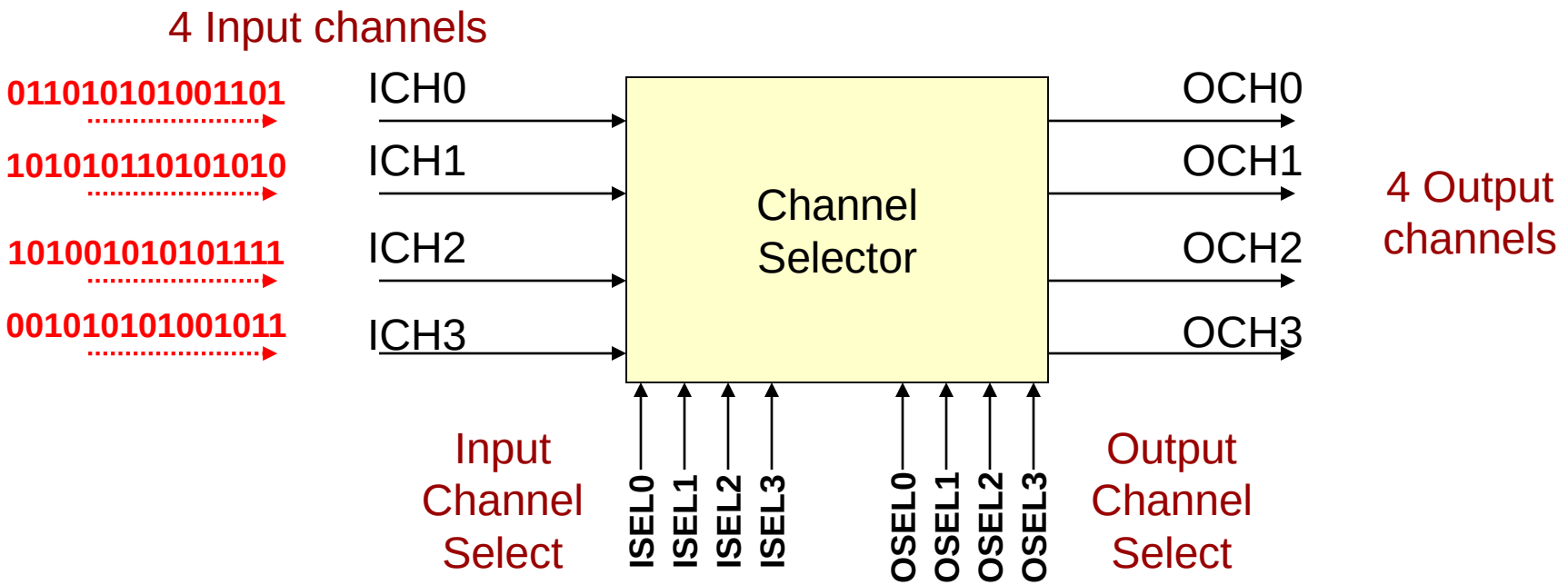
"1 OR anything equals 1"

Whenever a variable is AND'ed with 0, the output will be 0…

"0 AND anything equals 0"

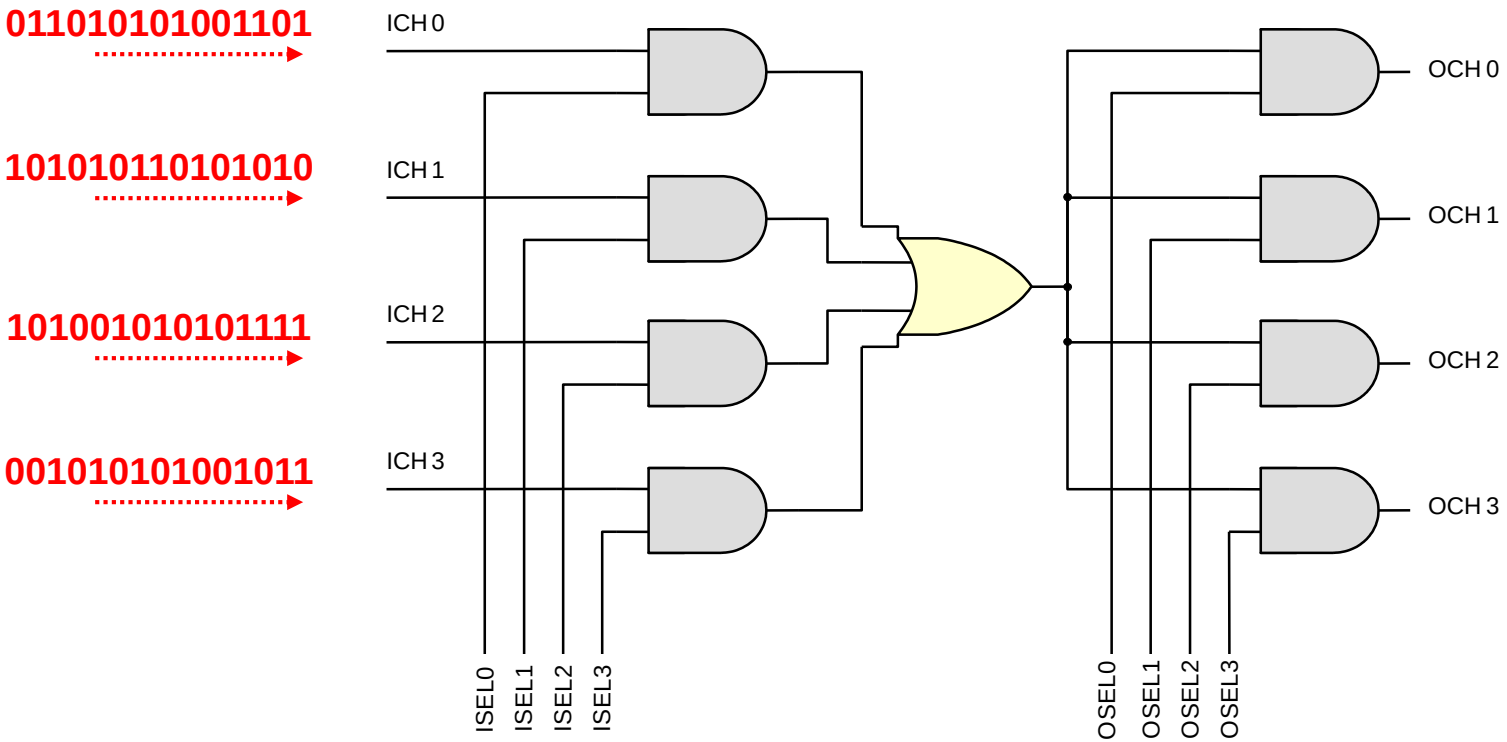
Application: Channel Selector

- Given 4 input, digital music/sound channels and 4 output channels
- Given individual "select" inputs that select 1 input channel to be routed to 1 output channel



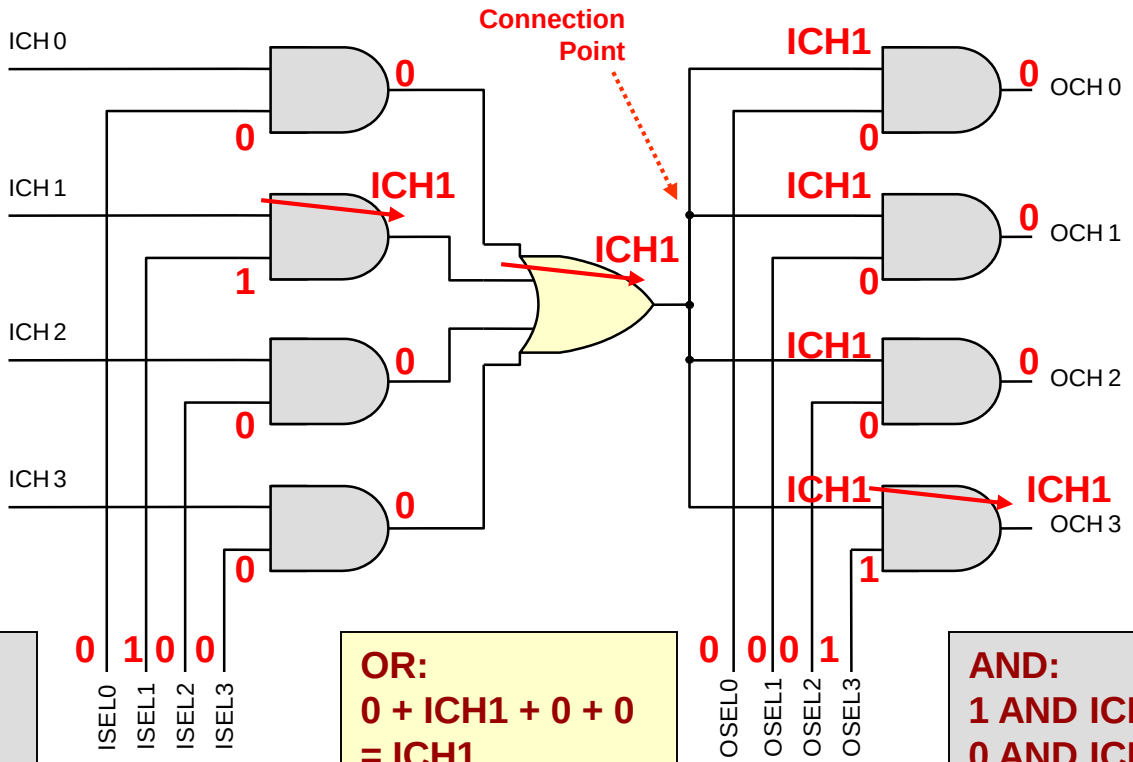
Application: Steering Logic

- 4-input music channels (ICHx)
  - Select one input channel (use ISELx inputs)
  - Route to one output channel (use OSELx inputs)



Application: Steering Logic

- 1st Level of AND gates act as barriers only passing 1 channel
- OR gates combines 3 streams of 0's with the 1 channel that got passed (i.e. ICH1)
- 2nd Level of AND gates passes the channel to only the selected output



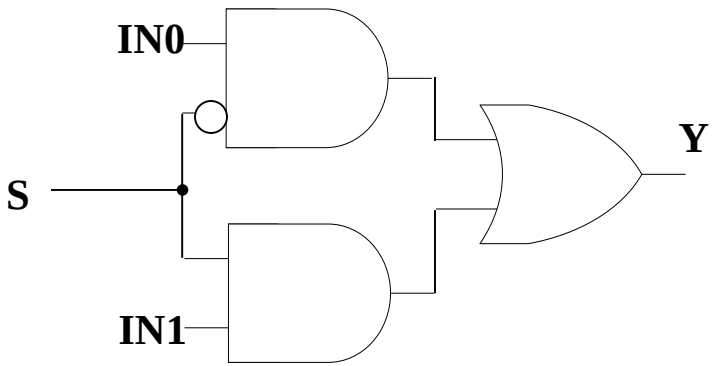
AND:
1 AND ICHx = ICHx
0 AND ICHx = 0

OR:
0 + ICH1 + 0 + 0 = ICH1

AND:
1 AND ICH1 = ICH1
0 AND ICH1 = 0

Your Turn

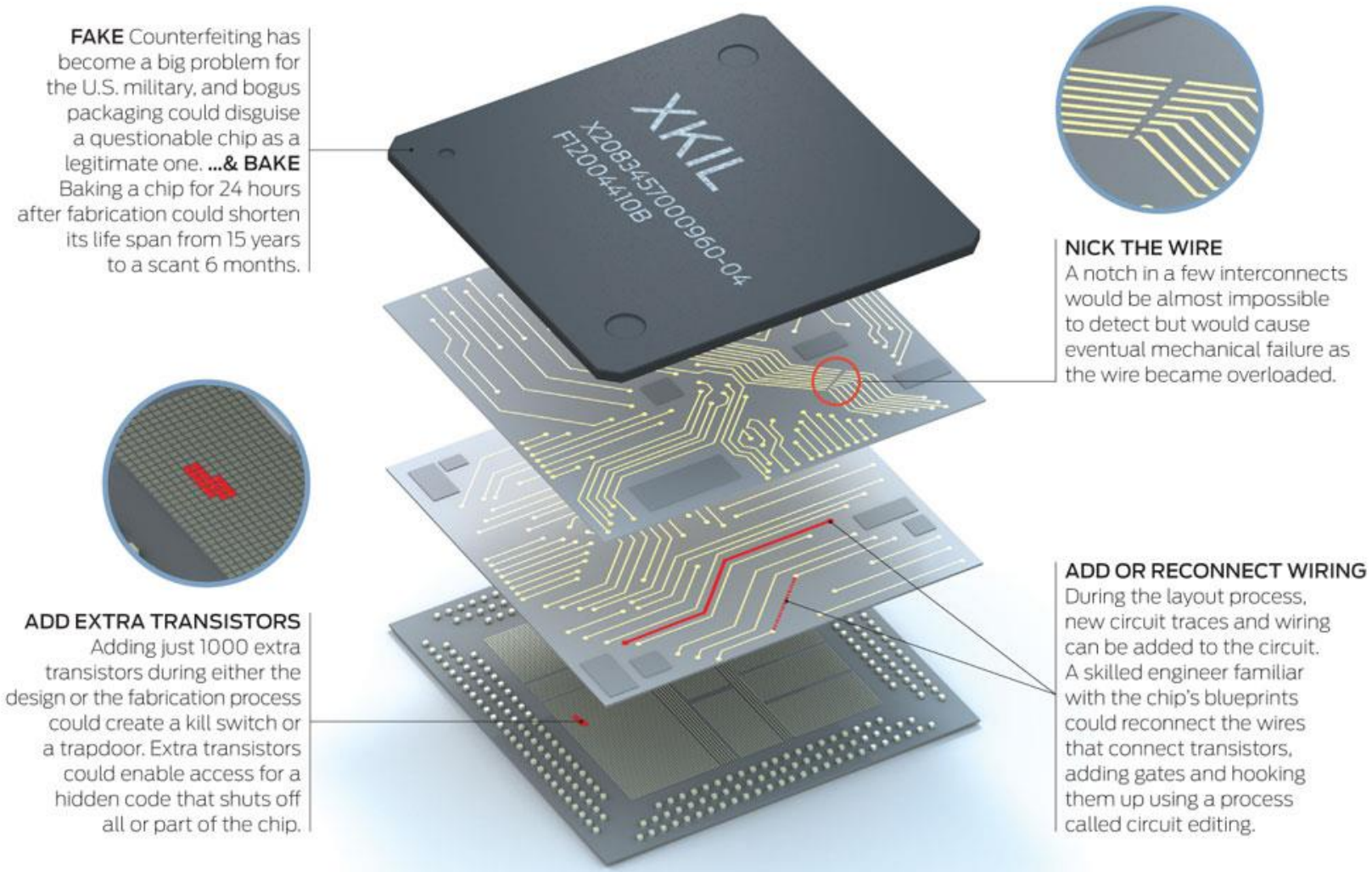
- Build a circuit that takes 3 inputs: S, IN0, IN1 and outputs a single bit Y.
- It's functions should be:
  - If S = 0, Y = IN0 (IN0 passes to Y)
  - If S = 1, Y = IN1 (IN1 passes to Y)



Application: TRUST Program

- The situation…
  - U.S. military consumes 1% of all chips worldwide
  - More and more chips are designed in the U.S. but fabricated in other countries (Taiwan, Korea, etc.)
  - Chip designs often use some IP (=Intellectual Property = pre-designed circuit components from another vendor)
- The problem: Sabotage in the form of…
  - Malicious changes to a design before fabrication
  - Introduce flaws during the fabrication process
  - Add additional logic that can be triggered at some later point in time
- Solution: DARPA's Trust in IC's program

Application: TRUST Program



Solutions

- Design Changes
  - Simulation
    - Describe the circuit to a computer, indicate what input combinations to plug in and let software CAD tools tell us what the output will be
    - Problem: 2^n combinations…too many
  - Formal Verification
    - Given two circuits, use theorems and other methods to formally prove that two circuits are equivalent or not
- Fabrication Errors
  - Reverse Engineer a circuit: shave off one layer of a chip at a time, taking images of each layer and let a computer reconstruct a model of the circuit; then compare it to original design
  - Other more advanced techniques (X-ray imaging, etc.)

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

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

rows where P=1

Yields SOP equation, AND-OR circuit

Canonical Product

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

rows where P=0

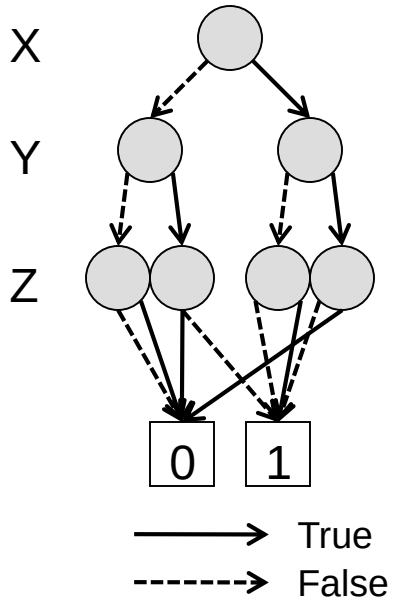
Yields POS equation, OR-AND circuit

Binary Decision Diagram

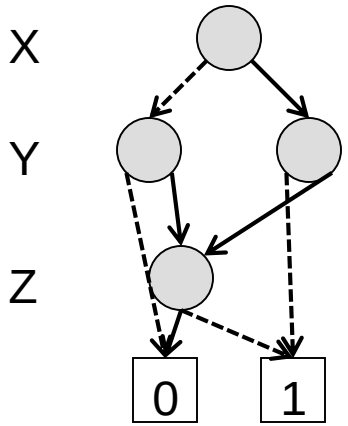
- Graph (binary tree) representation of logic function
- Vertex = Variable/Decision
- Edge = Variable value (T / F)

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	**1**
1	1	0	**0**
1	1	1	**1**

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



BDD for F



Canonical Representations

- As long as variable mapping between 2 functions is known then the following can be used as a unique/canonical representation [i.e. will be the same for the 2 functions]
- Truth Tables
  - show output for all possible combinations
  - n inputs => 2^n rows, only good for ≤ 5 var's.
- Canonical Sums
  - indicates where a function is 1 by listing the input combinations (rows) where it equals 1
- Canonical Products
  - indicates where a function is 0 by listing the input combinations (rows) where it equals 0
- Binary Decision Diagrams
  - Can be compact & are helpful to represent for computer tools

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

? = ?

$$Q = \sum_{a,b,c}(2,3,5,7)$$

Yes, If x=a, y=b, z=c