CS2100: Computer Organisation

Tutorial #6: Boolean Algebra, Logic Gates and Simplification

(Week 8: 9 – 13 March 2020)

Answers

Tutorial Questions:

1. Without referring to the book or lecture slides, name the essential theorems A, B, C and D used in the following derivation:

$$\begin{aligned}
F(j, k, m, p) &= k' \cdot (j' \cdot p \cdot (j' + m'))' + (p + k' + j)' \\
&= k' \cdot (j + p' + j \cdot m) + p' \cdot k \cdot j' && \text{... [A; involution]} \\
&= j \cdot k' + k' \cdot p' + j \cdot k' \cdot m + p' \cdot k \cdot j' && \text{... [distributive; commutative]} \\
&= j \cdot k' + p' \cdot (k' + k \cdot j') + j \cdot k' \cdot m && \text{... [associative; commutative; distributive]} \\
&= j \cdot k' + k' \cdot p' + j' \cdot p' + j \cdot k' \cdot m && \text{... [B; distributive; commutative]} \\
&= j \cdot k' + k' \cdot p' + j' \cdot p' && \text{... [associative; C]} \\
&= j \cdot k' + j' \cdot p' && \text{... [D]}
\end{aligned}$$

Note: In writing out terms, you should write the literals in the order of significance, especially in your final answer. For instance, for the above Boolean function $F(j, k, m, p)$, you should write the final answer as $j \cdot k' + j' \cdot p'$ and not $k' \cdot j + j' \cdot p'$ or $j \cdot k' + p' \cdot j'$.

Answers:

A = DeMorgan's Theorem

B = Absorption Theorem 2: $a + a' \cdot b = a + b$ ➔ $k' + k \cdot j' = k' + j'$

C = Absorption Theorem 1: $a + a \cdot b = a$ ➔ $j \cdot k' + j \cdot k' \cdot m = j \cdot k'$

D = Consensus Theorem: $a \cdot b + a' \cdot c + b \cdot c = a \cdot b + a' \cdot c$

➔ $j \cdot k' + k' \cdot p' + j' \cdot p' = j \cdot k' + j' \cdot p'$

2. Using Boolean algebra, simplify each of the following expressions into simplified **sum-of-products (SOP) expressions**. Indicate the law/theorem used for each step.

   (a) $F(x,y,z) = (x + y\cdot z')\cdot(y' + y) + x'\cdot(y\cdot z' + y)$

   (b) $G(p,q,r,s) = \Pi M(5, 9, 13)$

   Tip: For (b), it is simpler to start with the given expression, rather than to expand it into sum-of-products/sum-of-minterms expression first.

Answers:

Note: There might be more than one way of derivation.

(a) $(x + y\cdot z')\cdot\mathbf{(y' + y)} + x'\cdot\mathbf{(y\cdot z' + y)}$

$= (x + y\cdot z') \cdot \mathbf{1} + x'\cdot y$ [complement; absorption 1]

$= \mathbf{x} + y\cdot z' + \mathbf{x'\cdot y}$ [identity]

$= x + \mathbf{y} + \mathbf{y\cdot z'}$ [absorption 2]

$= \mathbf{x + y}$ [absorption 1]

(b) $G(p,q,r,s) = \Pi M(5, 9, 13)$

$= \mathbf{(p + q' + r + s')} \cdot (p' + q + r + s') \cdot \mathbf{(p' + q' + r + s')}$

$= ((q' + r + s') + (\mathbf{p\cdot p'})) \cdot (p' + q + r + s')$ [distributive]

$= (\mathbf{(q' + r + s') + 0}) \cdot (p' + q + r + s')$ [complement]

$= (q' + \mathbf{r + s'}) \cdot (p' + q + \mathbf{r + s'})$ [identity]

$= (\mathbf{q' \cdot (p' + q)}) + (r + s')$ [distributive]

$= \mathbf{p'\cdot q' + r + s'}$ [absorption 2]

3. Design a **divide-by-3 circuit** as follows: the input is a 4-bit unsigned binary number *ABCD*, and the output is a 3-bit unsigned binary number *XYZ* which is the quotient of *ABCD* / 3. For example, if *ABCD* = 1100 (or 12 in decimal), then *XYZ* = 100 (or 4 in decimal); if *ABCD* = 0111 (or 7 in decimal), then *XYZ* = 010 (or 2 in decimal).

   (a) Draw the truth table and try to obtain the **simplified SOP expressions** of *X*, *Y*, and *Z* just from observation. (It is quite easy to obtain the simplified SOP expressions for *X* and *Y* just from observing the truth table, but harder for *Z*. The K-map technique is probably useful here, but we'll do that in the next question.)

   (b) Verify your answer for *X* by first writing out its **sum-of-minterms expressions** and then simplifying the expression from there using Boolean algebra. Write out the law/theorem you use at each step.

   (c) From the simplified SOP expressions, implement *X*, *Y*, and *Z* using (i) **2-level AND-OR circuits**, and (ii) 2-**level NAND-only circuits**. Assume that primed literals are not available. (Always assume that prime literals are not available unless otherwise stated.)

Answers:

(a)

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

$X = A \cdot B$

$Y = A \cdot B' + A' \cdot B \cdot C$

$Z = A' \cdot B \cdot C' + B' \cdot C \cdot D + A \cdot C \cdot D + A \cdot B' \cdot D + A \cdot B' \cdot C$

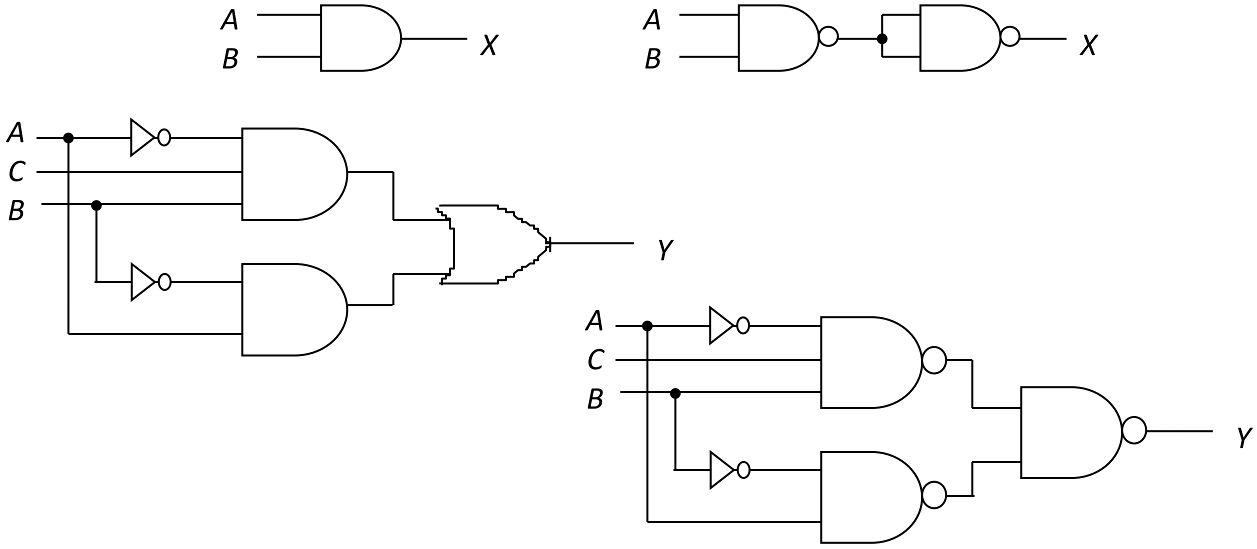
(b) Simplifying from sum-of-minterms expressions:

$$
\begin{aligned}
X = \Sigma m(12, 13, 14, 15) \quad &= \mathbf{A \cdot B} \cdot C' \cdot D' + \mathbf{A \cdot B} \cdot C' \cdot D + \mathbf{A \cdot B} \cdot C \cdot D' + \mathbf{A \cdot B} \cdot C \cdot D \\
&= A \cdot B \cdot (\mathbf{C' \cdot D' + C' \cdot D} + \mathbf{C \cdot D' + C \cdot D}) && \text{[distributive]} \\
&= A \cdot B \cdot (\ C' \cdot (\mathbf{D' + D}) + C \cdot (\mathbf{D' + D})\ ) && \text{[distributive]} \\
&= A \cdot B \cdot (C' \cdot \mathbf{1} + C \cdot \mathbf{1}) && \text{[complement]} \\
&= A \cdot B \cdot (\mathbf{C' + C}) && \text{[identity]} \\
&= A \cdot B \cdot \mathbf{1} && \text{[complement]} \\
&= \boldsymbol{A \cdot B} && \text{[identity]}
\end{aligned}
$$

You may also use the 'binary' notation (this is used in Quine-McCluskey method).

$$
\begin{aligned}
X = \Sigma m(12, 13, 14, 15) \quad &= \underline{1100 + 1101} + \underline{1110 + 1111} \\
&= 110\text{-} + 111\text{-} \\
&= 11\text{--} \\
&= \boldsymbol{A \cdot B}
\end{aligned}
$$

(c)



Circuits for *Z* not shown.

4. A 3-bit code *PQR* is designed to represent the 6 digits (0 – 5) in a base-6 system as follows:

Digit	3-bit code *PQR*
0	001
1	011
2	010
3	101
4	100
5	110

Design a logic circuit (you need not draw it) that takes in this 3-bit code *PQR* and generates two outputs:

- Output *Y* is 1 if the code *PQR* represents a prime number; otherwise, it is 0.
- Output *V* is 1 if the code *PQR* is valid; otherwise, it is 0.

Complete the truth table below. Using K-maps, obtain the simplified SOP expressions for *Y* and *V*. Give all alternative answers, if any. (Questions to settle before you can proceed: What are prime numbers? How do you handle invalid codes?)

P	*Q*	*R*	*Y*	*V*
0	0	0	X	0
0	0	1	0	1
0	1	0	1	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	X	0

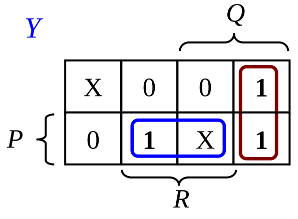
I deliberately left the table empty, to see how students would fill in the input columns. It is better to fill in the input columns in binary sequence, because then it makes filling the K-map easier. Point this out to the students.
This question also tests them on the use of don't-cares.

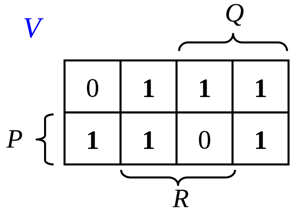
Answers:

Truth table: see above.

***Y* = *P*·*R* + *Q*·*R*'**

***V* = *P*·*Q*' + *P*'·*R* + *Q*·*R*'** or
P*·*R*' + *Q*'·*R* + *P*'·*Q


Y
Q
X 0 0 1
P 0 1 X 1
R


V
Q
0 1 1 1
P 1 1 0 1
R