EE224 Midsemester Test

February 22, 1100-1300 hrs

1. Consider a Boolean algebra Ω with operations $+$, $.$, identities 0, 1. For elements a, b of the Boolean algebra, define the operation $a \implies b$ to be $\overline{a} + b$. Recall that the operation $\oplus$ is defined as $a \oplus b = a.\overline{b} + \overline{a}.b$. Starting from the Axioms of Boolean algebra, show that each of the following expressions is equal to 1. (a, b, c, d, e are arbitrary elements of the Boolean algebra):

   (a) $(a \implies a)$. (1 mark)

   - $(a \implies a) = (\overline{a} + a) = 1$.

   (b) $((a \implies b) \implies (\overline{b} \implies \overline{a}))$. (1 mark)

   - The left-hand-side can be simplified as

$$\begin{aligned} LHS &= (\overline{a} + b) \implies (b + \overline{a}) \\ &= \overline{(\overline{a} + b)} + (b + \overline{a}) \\ &= (a.\overline{b} + b) + \overline{a} \\ &= (a + b) + \overline{a} \\ &= 1 + b \\ &= 1 \end{aligned}$$

   (c) $(a \implies b).(b \implies c) \implies (a \implies c)$. (1 mark)

- The left-hand-side can be simplified as

$$\begin{aligned}
LHS &= ((\overline{a}+b).(\overline{b}+c)) \implies (\overline{a}+c)\\
&= \overline{(\overline{a}+b).(\overline{b}+c)} + \overline{a} + c\\
&= (a.\overline{b} + b.\overline{c}) + \overline{a} + c\\
&= (a.\overline{b} + \overline{a}) + (b.\overline{c} + c)\\
&= (\overline{b} + \overline{a}) + (b + c)\\
&= (\overline{b} + b) + \overline{a} + c\\
&= 1 + \overline{a} + c\\
&= 1
\end{aligned}$$

(d) $(((a \implies b).(a \implies c)) \implies (a \implies (b.c)))$. (1 mark)

- Simplify the LHS (we have used $(a \implies a) = 1$ here):

$$\begin{aligned}
LHS &= ((\overline{a}+b).(\overline{a}+c)) \implies (\overline{a}+(b.c))\\
&= ((\overline{a}+b).(\overline{a}+c)) \implies (\overline{a}+(b.c))\\
&= ((\overline{a}+(b.c)) \implies (\overline{a}+(b.c))\\
&= 1
\end{aligned}$$

(e) $(((a \implies b).(b \implies a)) \implies \overline{a \oplus b})$. (1 mark)

- Simplify the LHS: we have used $(a \implies a) = 1$ here.

$$\begin{aligned}
LHS &= (((\overline{a}+b).(\overline{b}+a)) \implies \overline{a.\overline{b}+b.\overline{a}}\\
&= (((\overline{a}+b).(\overline{b}+a)) \implies (\overline{a}+b).(\overline{b}+a)\\
&= (((\overline{a}+b).(a+\overline{b})) \implies (\overline{a}+b).(a+\overline{b})\\
&= 1
\end{aligned}$$

2. Consider the following function f on 3 variables defined by the formula:

$$(x_1.\overline{x_2}) + (x_2.\overline{x_3}) + (x_3.\overline{x1})$$

Let g be a Boolean function defined by the formula $x_1 + x_2 + x_3$.

(a) Show that there exists a Boolean function h such that $f = g.h$. (2 marks)

- To write $f = g.h$, we must have $f \subset g$, that is whenever f evaluates to 1, g must evaluate to 1. It is easy to check that this is the case here.

(b) Find the simplest possible Boolean function h (the one with a sum of products formula which has the fewest literals) such that $f = g.h$. (3 marks)

- We try to express f in terms of x_1, x_2, x_3 and $g = x_1 + x_2 + x_3$. The K-map is

```
    x1,x2  00 01 11 10
x3,g
00            d  d  d
01         d  1  1  1
11         1  1     1
10         d  d  d  d
```

We are looking to cover f by product terms that will contain g. These are

```
    x1,x2  00 01 11 10
x3,g
00
01         d  1  1  1
11
10
```

that is $g.\overline{x_3}$,

```
    x1,x2  00 01 11 10
x3,g
00
01         d  1
11         d  1
10
```

that is $g.\overline{x_1}$, and

```
    x1,x2  00 01 11 10
x3,g
00
01         d        1
11         1        1
10
```

that is $g.\overline{x_2}$. Thus, we can write

$$f = g.(\overline{x_1} + \overline{x_2} + \overline{x_3})$$

3. Using 2 to 1 multiplexors, implement the Boolean function with five input bits, defined to be 1 if and only if at least 3 of the input bits are 1. Try to use as few multiplexors as you can. (5 marks)

   - Use Shannon's expansion (By $f_j^i(...)$ we mean the function on j variables which evaluates to 1 if and only if at least i of its inputs are 1. Note that $f_j^0 = 1$ and $f_j^i = 0$ if $i > j$.)

$$\begin{aligned}
f_5^3(x_1,x_2,x_3,x_4,x_5) &= (x_1?\ f_4^2(x_2,x_3,x_4,x_5)\ :\ f_4^3(x_2,x_3,x_4,x_5)) \\
f_4^2(x_2,x_3,x_4,x_5) &= (x_2?\ f_3^1(x_3,x_4,x_5)\ :\ f_3^2(x_3,x_4,x_5)) \\
f_4^3(x_2,x_3,x_4,x_5) &= (x_2?\ f_3^2(x_3,x_4,x_5)\ :\ f_3^3(x_3,x_4,x_5)) \\
f_3^1(x_3,x_4,x_5) &= (x_3?\ 1\ :\ f_2^1(x_4,x_5)) \\
f_3^2(x_3,x_4,x_5) &= (x_3?\ f_2^1(x_4,x_5)\ :\ f_2^2(x_4,x_5)) \\
f_2^1(x_4,x_5) &= (x_4?\ 1\ :\ f_1^1(x_5)) \\
f_2^2(x_4,x_5) &= (x_4?\ f_1^1(x_5):\ 0) \\
f_1^1(x_5) &= (x_5?\ 1:\ 0)
\end{aligned}$$

   We can do the implementation with 8 mutliplexors.

4. Consider the following Mealy FSM: the input alphabet consists of input symbols $\{RST, U, D\}$ and the output alphabet is $\{TICK, TOCK\}$. Assume that at time instant 0, RST is applied at the input to put the machine into the initial state. Subsequent to the application of RST, suppose that at time instant k, the number of U's seen thus far (including the current input) is A and the number of D's seen thus far B, then the machine outputs a $TOCK$ at instant k if $A = B$ *modulo* 3, else it outputs a $TICK$ at instant k.

   (a) Identify a possible set of states and the next-state and output functions which implement the specified behaviour of the state machine. (2 marks)

      - Observe that the difference $A - B$ modulo 3 can take only 3 possible values. We introduce three states $S0, S1, S2$. Then the next-state and output functions can be written out as follows:

Present-state	Input-symbol	Next-state	Output
_	RST	S0	TICK
S0	U	S1	TICK
S0	D	S2	TICK
S1	U	S2	TICK
S1	D	S0	TOCK
S2	U	S0	TOCK
S2	D	S1	TICK

(b) Encode the set of states, input symbols, and output symbols using bits, and implement the next-state and output-functions using Karnaugh maps (that is, identify the simplest possible sum-of-product formulas for these functions). (3 marks)

- Use a one-hot code: Use 3 bits q_0, q_1, q_2 to code the states so that $S0$ is coded as 100, $S1$ as 010 and $S2$ as 001. Use two variables $reset$ and x to code the input symbols so that RST is coded as 10, U as 01 and D as 00. Use a single variable y to code the output so that $TICK$ is coded as 0 and $TOCK$ as 1. The next-state equations are then

$$\begin{aligned} nq_0 &= reset + q_1.\overline{x} + q_2.x \\ nq_1 &= \overline{reset}.(q_0.x + q_2.\overline{x}) \\ nq_2 &= \overline{reset}.(q_1.x + q_0.\overline{x}) \end{aligned}$$

  The output equation is

$$y = \overline{reset}.(q_1.\overline{x} + q_2.x)$$

5. Each of the following statements is either true or false. In each case, decide whether the statement is true or false, and give a justification/proof for your claim.

(a) There exists a Boolean algebra with seven elements. (1 mark)

- False. The number of elements in a finite Boolean algebra must be a power of 2 (because it is equivalent to as set algebra).

(b) Given just multiplexors, one can implement any Boolean function. (1 mark)

- True. Using multiplexors (and constants), we can implement AND and NOT gates, hence every function.

(c) Given just gates which implement the $\implies$ operator introduced in Question 1, one can implement any Boolean function. (1 mark)

- True. You can implement a NOT gate using $(x \implies 0)$, and an OR gate using $\overline{x} \implies y$.

(d) Let f be a Boolean function on n variables $x_1, x_2, \ldots x_n$. Recall Shannon's expansion, $f = x_1.f_{x_1} + \overline{x_1}.f_{\overline{x_1}}$: then, f is 0 at all points if and only if $f_{x_1} + f_{\overline{x_1}}$ is zero at all points. (1 mark)

- True. If f is zero everywhere, so will f_{x_1} and $f_{\overline{x_1}}$. Conversely if the co-factors are 0 everywhere, so will f.

(e) The set of subsets of a finite set is a Boolean algebra. (1 mark)

- True. + is set union, . is set intersection, complement of an element is its set complement, 0 is the empty subset and 1 is the finite universe set.

6. Show that if we have logic gates that implement the $\oplus$ operator and gates that implement the . operator, then using the constant 1 and these gates, we can implement any Boolean function. (2 marks) Find an implementation of the formula $x_1 + x_2 + x_3 + x_4$ using only the constant 1, and the $\oplus$, . operations. (3 marks)

   - It is easy to see that $1 \oplus x = \overline{x}$. Thus, $1 \oplus (a.b)$ implements a NAND gate and thus we can implement all Boolean functions. It is then easy to write $x_1 + x_2 + x_3 + x_4$ in terms of NANDs and finish the job. Going a bit further: we see that $a + b = a \oplus b \oplus a.b$. Also the $\oplus$ operation is associative and . distributes over $\oplus$. Thus we can write $x_1 + x_2 + x_3 + x_4$ as

$$\begin{aligned} &= (x_1 + x_2) + (x_3 + x_4) \\ &= (x_1 + x_2) \oplus (x_3 + x_4) \oplus (x_1 + x_2).(x_3 + x_4) \end{aligned}$$

     and

$$\begin{aligned} (x_1 + x_2) &= (x_1 \oplus x_2 \oplus x_1.x_2) \\ (x_3 + x_4) &= (x_3 \oplus x_4 \oplus x_3.x_4) \end{aligned}$$

Putting it all together $x_1 + x_2 + x_3 + x_4$ can be expressed as

$$\begin{aligned} x_1 \oplus x_2 \oplus x_3 \oplus x_4 \\ \oplus x_1.x_2 \oplus x_1.x_3 \oplus x_1.x_4 \oplus x_2.x_3 \oplus x_2.x_4 \oplus x_3.x_4 \\ \oplus x_1.x_2.x_3 \oplus x_2.x_3.x_4 \oplus x_3.x_4.x_1 \\ \oplus x_1.x_2.x_3.x_4 \end{aligned}$$

This is called the algebraic normal form representation of $x_1 + x_2 + x_3 + x_4$ and is also a useful counting formula.