Solutions for CSE140 - HW #3

Due Monday May 1st, 11:59PM

1 Grading Rubrics (20 points total)

Q2 (9 points total)
- Parts 1, 2, 3, 4, 6 = 1 point for correct answer
- Parts 5, 7 = 1 point for correct answer + 1 point for justification

Q3 (6 points total)
- Part 1.1 = 1 point for completion
- Part 1.2 = 1 point for completion + 1 point for correct answer + 1 point for approach (Shannon’s expansion/K-Map)
- Part 2.1 = 1 point for completion
- Part 2.2 = 1 point for correct answer

Q4 (5 points total)
- Parts 1, 2, 4 = 1 point for completion
- Parts 3, 5 = 1 point for correct answer

2 Universal Gates

Check if the set of gates or functions in the following list are universal. Justify your answer. Assume that constants 0 and 1 are available as inputs.

Note: For checking whether a set of gates is universal or not, we need to check if we can build the AND, OR and NOT functionality using the set.

2.1 \{XOR, XNOR\} : Not Universal

1. NOT : \( X \oplus 1 = X' \)
2. AND: AND gate cannot be built using \{XOR, XNOR\}
3. OR: OR gate cannot be built using \{XOR, XNOR\}

2.2 \{AND, OR\} : Not Universal

1. NOT : NOT gate cannot be built using \{AND, OR\}
2. AND: AND gate is given in the set
3. OR: OR gate is given in the set
2.3 \{f(a, b)\}, where \( f(a, b) = a + a'b' \): Universal

1. Simplify: \( f(a, b) = a + a'b' = a + a'b' + b' \) (Consensus Theorem) = \( a + b'(a' + 1) \) (Distributive law) = \( a + b' \) (Identity law)

2. NOT: \( f(0, b) = 0 + b' = b' \)

3. AND: \( f(a', b)' = (a' + b')' = a.b \) (DeMorgan’s Law)

4. OR: \( f(a, b') = a + (b')' = a + b \)

2.4 \{f(a, b)\}, where \( f(a, b) = ab' \): Universal

1. NOT: \( f(1, b) = 1.b' = b' \)

2. AND: \( f(a, b') = a.(b')' = a.b \)

3. OR: \( f(a', b)' = (a'.b')' = a + b \) (DeMorgan’s Law)

2.5 \{f(a, b, c)\}, where \( f(a, b, c) = a + ab + abc \): Not Universal

1. Simplify: \( f(a, b, c) = a + ab + abc = a(1 + b) + abc \) (Distributive Law) = \( a + abc \) (Identity law) = \( a(1 + bc) \) (Distributive Law) = \( a \) (Identity Law)

2. NOT: NOT gate cannot be built using \( \{f(a, b, c)\} \)

3. AND: AND gate cannot be built using \( \{f(a, b, c)\} \)

4. OR: OR gate cannot be built using \( \{f(a, b, c)\} \)

2.6 \{f(a, b, c)\}, where \( f(a, b, c) = ab + bc \): Not Universal

1. NOT: NOT gate cannot be built using \( \{f(a, b, c)\} \)

2. AND: \( f(a, b, 0) = a.b + b.0 = a.b \)

3. OR: \( f(a, 1, c) = a.1 + 1.c = a + c \)

2.7 \{f(a, b, c)\}, where \( f(a, b, c) = a'b + b'c + ac' \): Universal

1. NOT: \( f(a, 1, 1) = a'.1 + 1'.1 + a.1' = a' + 0.1 + a.0 = a' \)

2. AND: \( f(a, b, 1)' = (a'.b+b'+a.1)' = (a'.b+b'+a.0)' = (a'.b+b')' = (a'.b+b'+a') \) (Consensus Theorem) = \( a'.(b + 1) + b)' \) (Distributive Law) = \( a'+b' \) (Identity Law) = \( a.b \) (DeMorgan’s Law)

3. OR: \( f(a', b', 1) = (a').b' + (b')'1 + a'.1' = a.b' + b + a'.0 = a.b' + b = a.b' + b + a \) (Consensus Theorem) = \( a.(b' + 1) + b \) (Distributive Law) = \( a + b \) (Identity Law)
3 Other types of gates

3.1 Simplify the expression in a minimal sum of products form

3.1.1 \( f(a, b) = a \oplus b \oplus a'b' \oplus ab \oplus (a' + b) \oplus (a + b') \)

Simplify using Shannon’s expansion:
\[
f(a, b) = af(1, b) + a'f(0, b)
\]
\[
f(1, b) = 1 \oplus b \oplus 1'.b' \oplus 1.b \oplus (1' + b) \oplus (1 + b')
\]
\[
f(1, b) = 1 \oplus b \oplus 0.b' \oplus b \oplus (0 + b) \oplus 1
\]
\[
f(1, b) = 1 \oplus b \oplus 0 \oplus b \oplus b \oplus 1
\]
\[
f(1, b) = b
\]
\[
f(0, b) = 0 \oplus b \oplus 0'.b' \oplus 0.b \oplus (0' + b) \oplus (0 + b')
\]
\[
f(0, b) = 0 \oplus b \oplus 1.b' \oplus 0 \oplus (1 + b) \oplus b'
\]
\[
f(0, b) = 0 \oplus b \oplus b' \oplus 0 \oplus 1 \oplus b'
\]
\[
f(0, b) = b'
\]
\[
f(a, b) = ab + a'b'
\]

3.1.2 \( f(a, b, c) = a'b \oplus ab'c \oplus (a + b)c' \oplus (b + c) \oplus (ab + c) \)

Simplify using Shannon’s expansion:
\[
f(a, b, c) = af(1, b, c) + a'f(0, b, c)
\]
\[
f(1, b, c) = 1'b \oplus 1.b'c \oplus (1 + b)c' \oplus (b + c) \oplus (1.b + c)
\]
\[
f(1, b, c) = 0 \oplus b'c \oplus c' \oplus (b + c) \oplus (b + c)
\]
\[
f(1, b, c) = b'c \oplus c'
\]

Simplify using Shannon’s expansion:
\[
f(1, b, c) = bf(1, 1, c) + b'f(1, 0, c)
\]
\[
f(1, 1, c) = 1'.c \oplus c'
\]
\[
f(1, 1, c) = 0 \oplus c'
\]
\[
f(1, 1, c) = c'
\]
\[
f(1, 0, c) = 1.c \oplus c'
\]
\[
f(1, 0, c) = c \oplus c'
\]
\[
f(1, 0, c) = 1
\]
\[
f(1, b, c) = b'c + b' = bc' + b' + c' \quad (\text{Consensus Theorem})
\]
\[
f(1, b, c) = c'(b + 1) + b' = b' + c'
\]
\[
f(0, b, c) = 0'.b \oplus 0.b'c \oplus (0 + b)c' \oplus (b + c) \oplus (0.b + c)
\]
\[ f(0, b, c) = 1 \cdot b \oplus 0 \oplus bc' \oplus (b + c) \oplus c \]
\[ f(0, b, c) = b \oplus c \oplus bc' \oplus (b + c) \]

Simplify using Shannon’s expansion:
\[ f(0, b, c) = b f(0, 1, c) + b' f(0, 0, c) \]
\[ f(0, 1, c) = 1 \oplus c \oplus 1.c' \oplus (1 + c) \]
\[ f(0, 1, c) = 1 \oplus c \oplus c' \oplus 1 \]
\[ f(0, 1, c) = 1 \]
\[ f(0, 0, c) = 0 \oplus c \oplus 0.c' \oplus (0 + c) \]
\[ f(0, 0, c) = 0 \oplus c \oplus 0 \oplus c \]
\[ f(0, 0, c) = 0 \]
\[ f(0, c, b) = 1 \cdot a + b' \cdot 0 = b \]
\[ f(a, b, c) = a(b' + c') + a'b = ab' + ac' + a'b \]
\[ f(a, b, c) = ab' + ac' + a'b + bc' \text{ (Consensus Theorem)} \]
\[ f(a, b, c) = ab' + a'b + bc' \text{ (Consensus Theorem)} \]
\[ f(a, b, c) = ab' + a'b + bc' = ab' + ac' + a'b \]

3.2 Prove or disprove the following equalities.

3.2.1 \[ a + b \oplus c = (a + b) \oplus (a + c) \]

Provide a counter example. Let \( a = 1, b = 1, c = 0 \).
LHS = \( a + b \oplus c = 1 + 1 \oplus 0 = 1 + 1 = 1 \)
RHS = \((a + b) \oplus (a + c) = (1 + 1) \oplus (1 + 0) = 1 \oplus 1 = 0 \)
Therefore, LHS \( \neq \) RHS

3.2.2 \[ a \oplus (b + c) = a \oplus b + a \oplus c \]

Provide a counter example. Let \( a = 1, b = 1, c = 0 \).
LHS = \( a \oplus (b + c) = 1 \oplus (1 + 0) = 1 \oplus 1 = 0 \)
RHS = \((a \oplus b) + (a \oplus c) = (1 \oplus 1) + (1 \oplus 0) = 0 + 1 = 1 \)
Therefore, LHS \( \neq \) RHS
4 Results Verification

Below is a logic expression

\[ f(a, b, c) = abc \oplus (a + b' + c) \oplus (b + c)b \oplus a'b \oplus (b + c') \]  

(1)

Jim magically simplified this expression and the following is the results he get:

\[ f(a, b, c) = \sum m(1, 2, 5) \]  

(2)

To verify the correctness of Jim’s solution you are asked to write the Bluespec code.

i. Express the \( \oplus \) in the following bluespec function.

```plaintext
function Bit#(1) oplus(Bit#(1) l, Bit#(1) r);
    return l ^ r;
    OR
    return (l & ~r) | (~l & r);
endfunction
```

ii. Express the original expression in Bluespec in the following function.

```plaintext
function Bit#(1) origin(Bit#(1) a, Bit#(1) b, Bit#(1) c);
    return oplus((a & b & c), oplus((a | ~b | c), oplus(((b | c) & b), oplus(~a & b, (b | ~c)))));
endfunction
```

iii. Express Jim’s solution in Bluespec in the following function.

```plaintext
function Bit#(1) jims( Bit#(1) a, Bit#(1) b, Bit#(1) c);
    Bit#(3) t={a,b,c};
    UInt#(3) t_index = unpack(t);
    return pack((t_index == 1) || (t_index == 2) || (t_index == 5));
endfunction
```

iv. The following code that check’s the correctness of Jim’s result. Brief explain how the following code works.

```plaintext
package Tb;
import Checker::*;

(* synthesize *)
module mkTb();
    Reg#(Bit#(4)) t <- mkReg(0);
    Reg#(Bool) started <- mkReg(False);
```
rule start (!started);
    started <= True;
endrule
rule check(t < 8
    && started
    && origin(t[2], t[1], t[0])
    == jims(t[2], t[1], t[0]));
    t <= t + 1;
endrule
rule report_error ( t < 8
    && started
    && origin(t[2], t[1], t[0])
    != jims(t[2], t[1], t[0]));
    $display("mismatch when a=%d b=%d c=%d\n", t[2], t[1], t[0]);
    $display("origin =%d",origin(t[2], t[1], t[0]));
    $display("jims =%d",jims(t[2], t[1], t[0]));
    $finish;
endrule
rule finish(t == 8 && started);
    $finish;
endrule
endmodule
endpackage

Explanation: The code verifies the correctness of Jim’s solution against the original expression. The code iterates over all possible combination of the values of a, b and c. It then computes the result using both Jim’s solution and the original equation for every such combination and checks whether the results match. If there is a mismatch, then the code displays the values of a, b and c plus the results using both approaches for which the mismatch occurred.

v. Is the Jim’s solution correct? If yes, answer yes, if not, report which term in
the original expression is not covered by Jim’s simplified expression. (The codes are
available on ieng6 server with in the path /home/linux/ieng6/cs140s/public/hw3.)

Answer: No. Jim’s solution is not correct. The term m(6) = abc’ has not been
covered by Jim’s simplified expression.