CSE140 - HW #3

Due Monday May 1st, 11:59PM

1 Introduction

The purpose of this assignment is to get a better understanding of universal gates and basic memory elements of sequential circuits. For the first problem, we clarify the concept of universal set. For the second problem, we practice the operations of exclusive gates. In the third problem, we use BSV to verify the equalities.

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.

1. {XOR, XNOR}
2. {AND, OR}
3. $\{f(a,b)\}$, where $f(a,b) = a + a'b'$.
4. $\{f(a,b)\}$, where $f(a,b) = ab'$.
5. $\{f(a,b,c)\}$, where $f(a,b,c) = a + ab + abc$.
6. $\{f(a,b,c)\}$, where $f(a,b,c) = ab + bc$.
7. $\{f(a,b,c)\}$, where $f(a,b,c) = a'b + b'c + ac'$.

3 Other types of gates

I. Simplify the expression in a minimal sum of products form.

1. $f(a,b) = a \oplus b \oplus a'b' \oplus ab \oplus (a' + b) \oplus (a + b')$
2. $f(a,b,c) = a'b \oplus ab'c \oplus (a + b)c' \oplus (b + c) \oplus (ab + c)$

II. Prove or disprove the following equalities.

1. $a + b \oplus c = (a + b) \oplus (a + c)$.
2. $a \oplus (b + c) = a \oplus b + a \oplus c$

4 Results Verification

Below is a logic expression

$$f(a,b,c) = abc \oplus (a + b' + c) \oplus (b + c)b \oplus a'b \oplus (b + c') \tag{1}$$

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

$$f(a,b,c) = \sum m(1,2,5) \tag{2}$$

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.

```
function Bit#(1) oplus(Bit#(1) l, Bit#(1) r);
    return ___________________________;
endfunction
```

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

```
function Bit#(1) origin(Bit#(1) a,
                        Bit#(1) b,
                        Bit#(1) c);
    return ________________________________________;
endfunction
```

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

```
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(_________________________________________);
endfunction
```

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

```
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
```

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`.)