CS684: Embedded Systems
Spring 2020
Assignment 3.1 - LUSTRE version 6

Aaron John Sabu: 170070050

February 23, 2020

1 Problem Statement

1. Describe in English the output produced by the following Lustre V6 node:

```
node use_boolred(t: bool^5) returns (res: bool);
let
    res = boolred<<1,2,5>>(t);
tel
```

2. Explain the output for the following heptagon node for the first 7 cycles in response to the input given below:

i	1	2	1	-1	3	0	-1
c	true	false	true	true	true	false	false

```
node even_times(i : int; c: bool) returns (o: int)
let
    automaton
        state EVEN
            do o = 100 -> i+1
            unless c continue ODD
        state ODD
            do o = 203 -> -2 * i
            unless c then EVEN
    end
tel
```

3. Compute the output for X=[0,1,1,0,1,1,1,0] and Y=[1,1,0,1,0,1,1,1] given at the 0^{th} cycle. Using this Bar node, define a counter which counts in binary modulo 64:

```
node Foo(cin, x, y: bool) returns (cout, z: bool);
let
    z = cin xor x xor y;
    cout = if cin then x or y else x and y;
tel
node Bar(X: bool^8; Y: bool^8) returns (over: bool; Z: bool^8);
let
    (over, Z) = fillred<<Foo, 8>>(false, X, Y);
tel
```

2 Solution

Given below is my approach to the problem.

2.1 Part 1

It denotes a combinational node with profile **bool^5** $\rightarrow$ **bool**, *res* is **true** if and only if *at least 1* and *at most 2* elements are **true** in *t*.

2.2 Part 2

This problem can be solved by considering two flows: o_E and o_O; the former represents the flow o in EVEN state and the latter represents the flow o in ODD state. The output flow o is a MUXed output of these two flows, controlled by the state which is further controlled by c

o_E	100	2	3	2	0	4	1	0
o_O	203	-2	-4	-2	2	-6	0	2
c		T	F	T	T	T	F	F
State	E	O	O	E	O	E	E	E
o		-2	-4	2	2	4	1	0

Hence, the output is $o = [-2, -4, 2, 2, 4, 1, 0]$

2.3 Part 3

Foo represents a full adder where *cin* is the input carry from a previous adder, x and y are two inputs to be added, *cout* is the output carry and z is the output sum from the adder. *Bar* is a generic 8-bit adder developed from *Foo*.

$$z[i] = cin[i] + x[i] + y[i]$$
$$cin[i] = cout[i-1]$$

As a result of this observation, we can compute the output of *Bar* as follows:

cin	0	0	0	0	0	0	0	0
x	0	1	1	0	1	1	1	0
y	1	1	0	1	0	1	1	1
cout	1	1	1	1	1	1	1	0
z	0	1	0	0	0	1	0	1

Hence, the output is $Z = [0, 1, 0, 0, 0, 1, 0, 1]$ and $over = 1$ for $X = [0, 1, 1, 0, 1, 1, 1, 0]$ and $Y = [1, 1, 0, 1, 0, 1, 1, 1]$

In order to construct an 8-bit counter from an 8-bit adder, we need to supply the output of the adder back as the input and the other input should be 1.

```
node Mod64Counter (CLK, RST: bool) returns (Y: bool^8);
var over: bool;
let
    (over, Y) = (0, [0,0,0,0,0,0,0,0]) ->
            if RST then  (0, [0,0,0,0,0,0,0,0])
            else Bar(pre(Y), [0,0,0,0,0,0,0,1]);
tel
```