Direction Logic for Lift Controller

The first assignment was based on the decisions to make by the lift when it is at any floor and is pinged by people at different floors. We implemented the circuit through Xilinx ISE. Test cases were chosen for usual as well as corner cases. Here is our logic.

For all i=0,1,2,3

Let C_i = B_i AND (NOT F_i) where B_i and F_i are the standard input notations.

From here on, we will consider C_i instead of corresponding B_i whenever we refer to B_i.

Here C_i let us filter through some corner cases like when someone is at the ith floor but still presses the Bi button.

Go_upUsual = { (Ascending) AND (X) } OR { (Descending) AND (X) AND (NOT Y) }

Go_dnUsual = { (Descending) AND (Y) } OR { (Ascending) AND (Y) AND (NOT X) }

(These above two conditions handle the first priority condition as given in the instructions.)

Go_upHalted = NOT { (Ascending) AND (Descending) } AND { Above_up OR In_up OR (Above_dn AND (NOT Below_up)) }

Go_dnHalted = NOT { (Ascending) AND (Descending) } AND { (Below_up OR Below_dn OR In_dn) AND (NOT Go_upHalted) }

(The above two conditions take care of the cases when the lift is initially in the Halted state.)

Where X = (Above_up OR Above_dn OR In_up) & Y = (Below_up OR Below_dn OR In_dn)

The last case to consider is when no button is pressed.

For that consider :-

ALL = { OR (all input values)}

So, finally we have,

Go_up = (Go_upUsual OR Go_upHalted) AND (ALL)

Go_dn = (Go_dnUsual OR Go_dnHalted) AND (ALL)

Thus, the output values are 1 only if some button is pressed not corresponding to same floor (i.e. B_i for ith floor) and lift needs to go up or down according to the priority conditions given in the assignment.