Task 3: Issues at minimal cost implementation

Given a truth table, it was requested to see what happens when the truth table is implemented with the smallest quantity of logic gates. To perform this task, ICs that either have NOR or NAND logic gates will be used.

Low cost approach

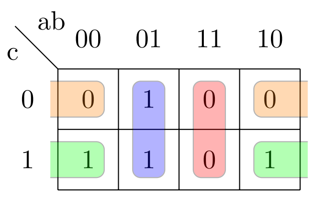

Figure 1: Karnaugh Map of the given truth table

We can get the minimal cost output expressed either in groups of maxterms (orange and red groups) or minterms (blue and green groups). Let Z be the output label, its reduced minterms expression is:

$$Z = (\bar{A}.B) + (C.\bar{B}) \tag{1}$$

Also, its reduced maxterms expression is:

$$Z = (\bar{A} + \bar{B}).(C + B) \tag{2}$$

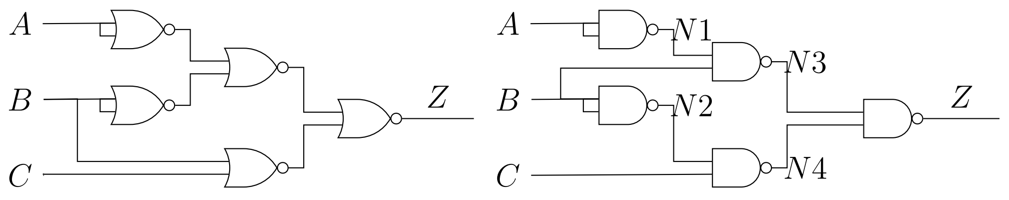

Figure 2: Implementation with NOR gates and NAND gates respectively

Note that the quantity of logic gates in both cases are the same.

Glitch analysis

Sometimes propagation delays cause unexpected and unwanted transition in the ouput. This issue is called 'glitch' and it's related to how the logical circuit is synthesized. In this section, the NAND implementation will be taken into account. If we take into accout the propagation time of the NAND logic gate (here we assume all propagation times are equal) and then observe transition of the states A,B,C from 011 to 001 respectively.

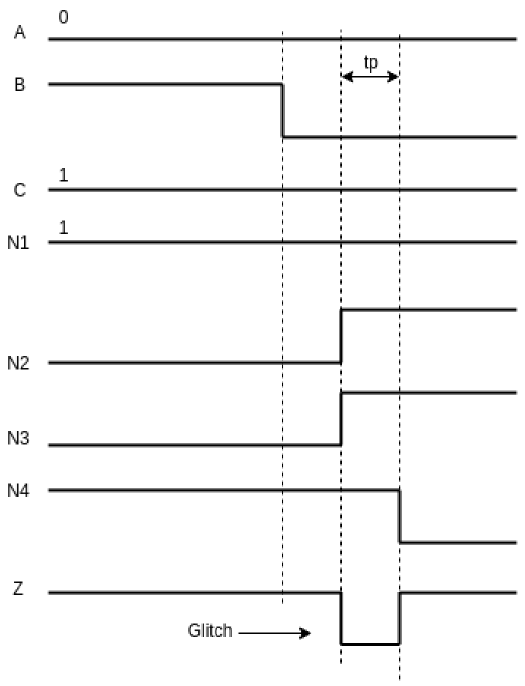

Figure 3: Propagation time analysis

Note that the interval of time tp in Figure 3 denotes the propagation time. The result might be at first surprising but in fact this effect was in someway expected. Take a look at the Karnaugh map in Figure 1. An input that follows multiple paths to an output can create a glitch if one path has an inverter (in this case implemented with a NAND logic gate) and one does not. This issue is called ”asymmetric path delay” and it can be seen in Figure 1 with the input B.

Conclusion

In order to remove glitches it is needed to add redundant groups in the Karnaugh Map so that disjoint groups overlap.

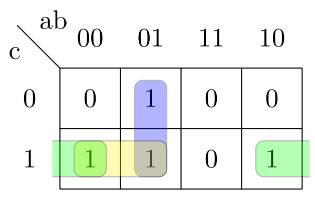

Figure 4: Improved karnaugh

In this case the output is given by:

$$Z = (\bar{A} + \bar{B}).(C + B) + (\bar{A}.C)$$

Note that the new term added to the expression is not dependent of B so regardless of the variation of B there won't be glitches in the output, because it will be fixed by this new term.
Finally to give some sense to the adding some redundant logic, when we make Karnaugh groups we take the non-changing states and the only variable that changes of state in the new group is B which is reasonable due to the fact that that's the variable causing the issue. Depending on the application it may be important or not to remove glitches. There might be more glitches, remember that this is just one particular case, where glitches are more likely to happen.