Miscellaneous Expression Items (D)

The SOP form, POS form, and the Complement of a function

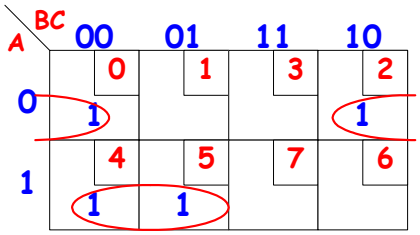
Let's take a break from expression simplification and take care of some miscellaneous design items. These topics did not seem to fit well anywhere else but they tend to complement the material that has already been discussed.

Reasons to choose SOP or POS results

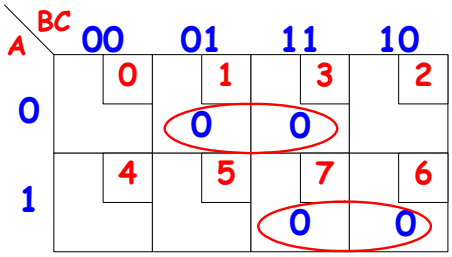
Quite often it is desirable to have an equation in **POS** form instead of an ***SOP*** form. We have already discussed the **POS** form but we haven't discussed why we would use one or the other. The **SOP equation works well with 'NAND' and 'AND' gates** while the **POS equation works well with NOR's and OR's**. You might also want to **use the POS** if you desire an **ACTIVE LOW output** instead of an **ACTIVE HIGH**.

The SOP and POS forms of a function from a K-map

K-maps can be used for other things other than equation minimization and expansion. Among other things, they can be used to easily determine the **POS** form of an **SOP** expression, and vice versa. The map to the right is a simple **K-map** showing the **SOP** of a function.



SOP of f(A,B,C)



POS of f(A,B,C)

Once the **SOP K-map** is known, it becomes an easy leap to the **POS** form. We can easily produce the **POS** form of the same expression by **placing 0's in all the cell's which didn't contain 1's in the original map**. Remember that this is **still representing the same truth table**. We are just using the rows which have outputs of '**0**' vice the rows which have outputs of '**1**'.

The complement (INVERSE) of POS and SOP expressions

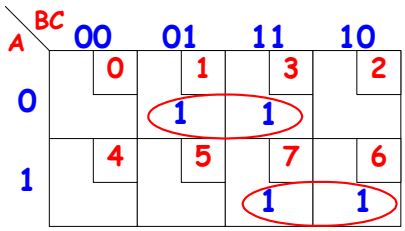
In order to understand how the **inverse** of an expression works, let's first look at the truth table represented by the **SOP** and **POS** expressions. The output column for the functions is shown in column **4**. As shown earlier,

- **if a SOP expression was desired, the 1's in that column would be mapped or,**
- **if a POS expression were desired, then the 0's in that column would be mapped.**

A	B	C	$f(A,B,C)$	$\bar{f}(A,B,C)$
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

Obviously, if we want to find the **inverse of a function** in a table; all that is necessary is to complete a column where the **1's and 0's have been inverted** as is demonstrated in column 5 of the table.

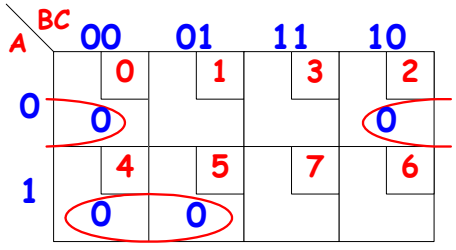
Since a **K-map** is nothing more than a graphical representation of a Truth Table, it should be possible to get the same result with a **K-map**. In order to find the complement (or **INVERSE**) of a **POS** expression, all that is done is take the **POS K-map** and replace all the **0**'s with **1**'s. It is still an **SOP** form but this time it is the **SOP** form of the **COMPLEMENT** of the function.



SOP of $\bar{f}(A,B,C)$

It follows that if it is desired to determine the **complement of the SOP expression**. All that is necessary is to **replace the 1's with 0's** and solve it as a **POS expression**.

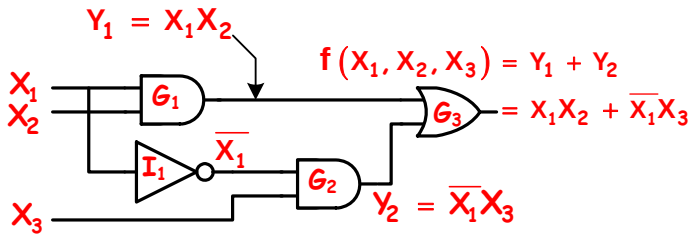
One might think that if you wanted to have all four of these equations, one would have to produce four **K-maps**. Not true. Notice that the 1st and the last **K-map** groups are the same while the middle two groups are also the same. All that is needed is to produce two **K-maps** and then you can carefully use them to produce the alternate equations by just imagining **1**'s or **0**'s in place of the actual contents of the cells.



POS of $\bar{f}(A,B,C)$

Static Timing Hazards

The following discussion requires that we temporarily think about **REAL GATES with REAL characteristics**. In this case it is necessary to include the **propagation delay** of gates into the thought process. The gates in the circuit below are **no longer IDEAL**. Each one has a **time delay** between when the input is applied and when the effect of that input is seen at the output. This '**Propagation Delay**" is different for each different type of gate and can be found in the data sheet for the device. It was discussed briefly in Chapter 2 but now the effect it has on the **timing** of a circuit is going to be examined.



The circuit above is going to be analyzed over a very small portion of its response. Specifically, the effect on the output when the input X_1 changes as shown here:

$$f(X_1, X_2, X_3) = f(0,1,1) \Rightarrow f(1,1,1) \Rightarrow f(0,1,1)$$

In other words, the circuit is going to be examined for the portion of its response when both X_2 and X_3 remain at a **logic '1'** and X_1 is allowed to change from a logic '**0**', up to a logic '**1**', and then back to a logic '**0**'.

X_1	X_2	X_3	Y_1 X_1X_2	$\overline{X_1}$	Y_2 $\overline{X_1}X_3$	$f(X_1, X_2, X_3)$ Y_1Y_2	X_1, X_2, X_3
0	1	1	0	1	1	1	0,1,1
1	1	1	1	0	0	1	1,1,1

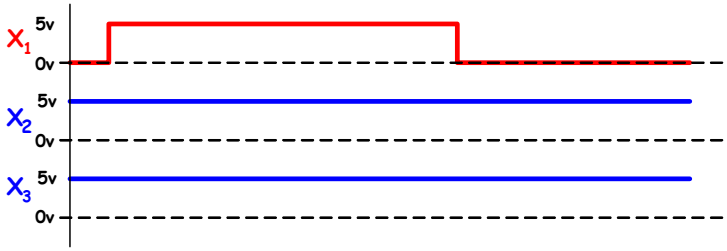
The figure above is a '**snapshot**' of a truth table for the portion that is going to be examined. One of the problems with truth tables is that they only examine the **IDEAL** reaction of a circuit to a set of inputs. **In order to see how a circuit truly reacts to an input change it becomes necessary to look at a circuit's TIMING DIAGRAM!**

Timing Diagrams

According to the truth table snapshot, **the output should always be 1 for the limited change of inputs being examined.** Let's see what happens when we take into account **pulse delay times**.

Now, let's examine the circuit by examining its **TIMING DIAGRAM**. Note that this circuit does not have a clock. This means that it is an **Asynchronous** circuit which is **more likely to have timing issues the Synchronous circuits**.

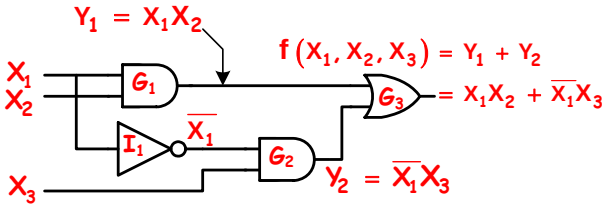
- Start out the **timing diagram** by showing the transitions (or non-transitions) of X_1, X_2, and X_3. Make sure to show the change in states large enough so that you can apply some make-believe propagation delays. (Naturally, a **logic analyzer** would be showing **REAL Propagation delays** but this is for example purposes).

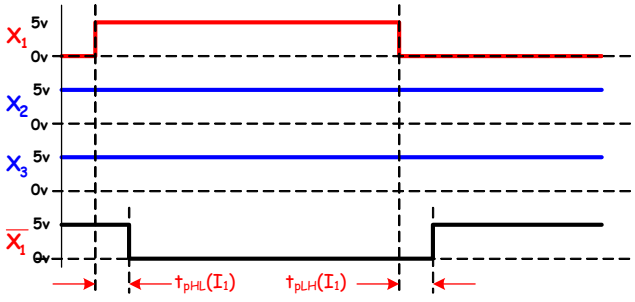


Note that as required by the specifications of this analysis, the only input which is changing state on the timing diagram is the X_1 line.

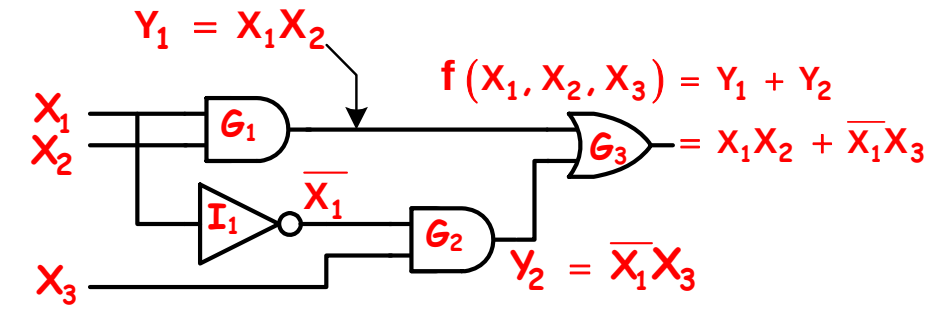
$$f(X_1, X_2, X_3) = f(0,1,1) \Rightarrow f(1,1,1) \Rightarrow f(0,1,1)$$

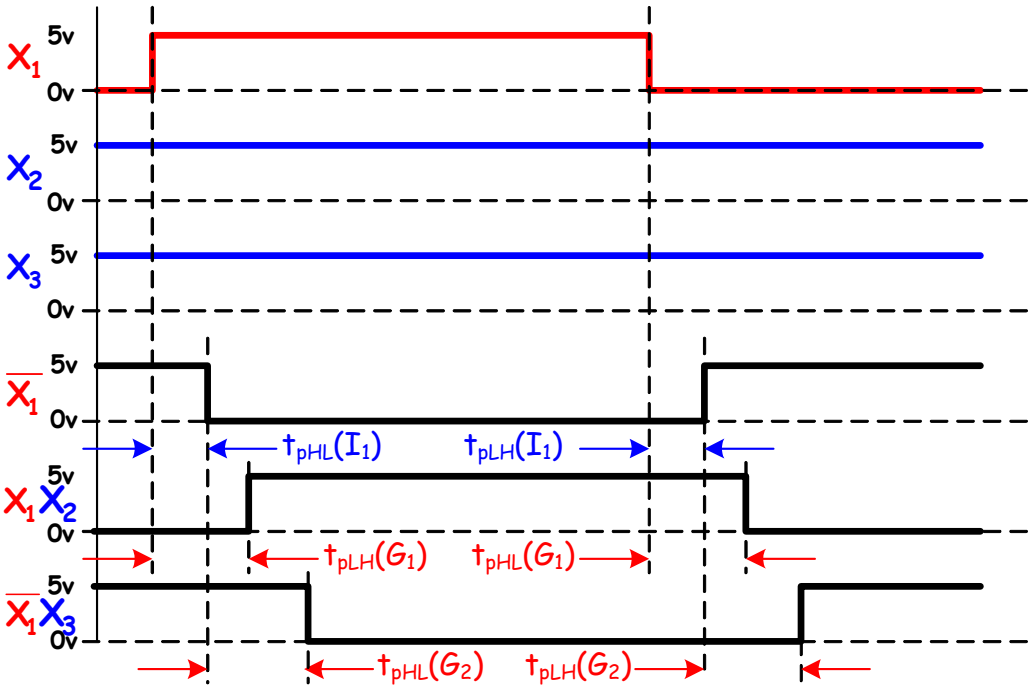
Next you need to take the inverter and the "**propagation delay**" into account. So a $\overline{X_1}$ trace is added to the diagram with some '**imaginary**' **propagation delay** applied to both transitions.





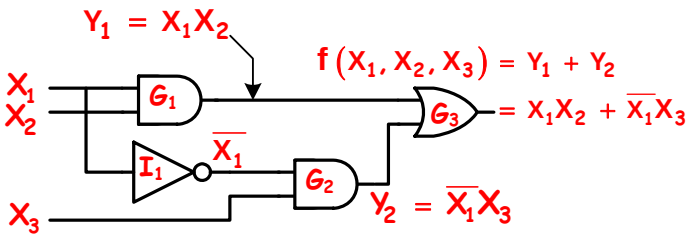
- Enough information is now shown in the diagram to create both the Y_1 and the Y_2 traces. Again, example **propagation delays** for both **AND** gates are going to be used. Since an **AND** gate will have a longer propagation delay then an **INVERTER**, the ones on the diagram will also be longer.



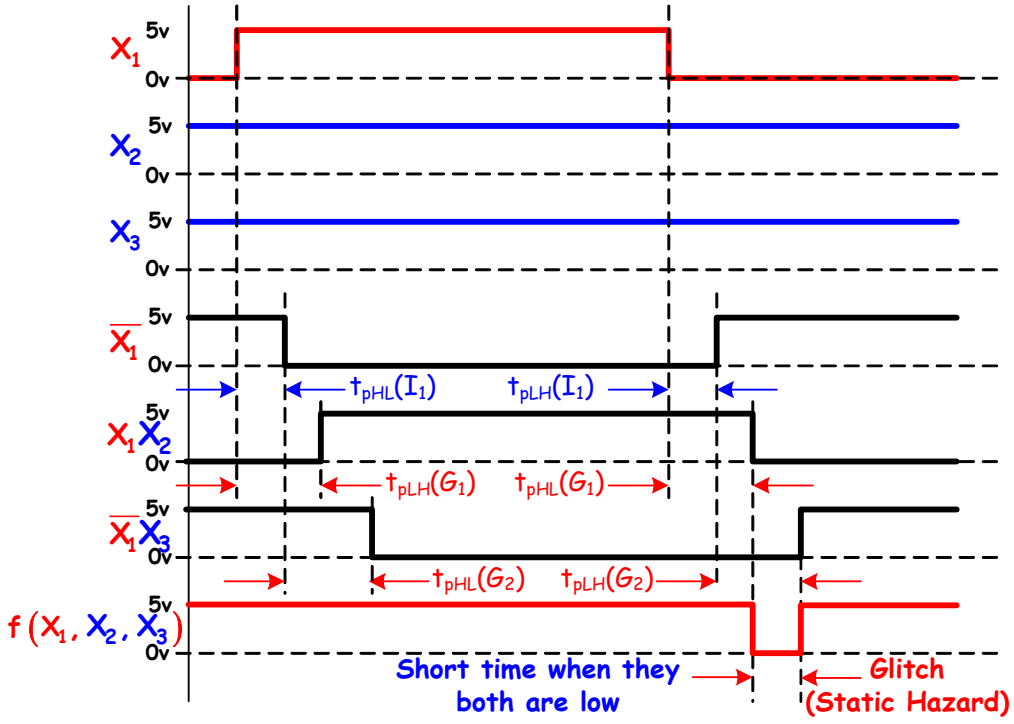


Note that the Y_2 waveform is referenced to the **inverted X** waveform while the Y_1 waveform is referenced to the **non-inverted X** waveform. Since they are both out of **AND** gates, the propagation delays are shown as the same length.

- And finally it is time to examine the output which is nothing else than an **OR** operation with Y_1 and Y_2.



NOTE: For this example the **OR gate** will be **considered to be ideal, i.e. no propagation delay**. This is to simplify the diagram so that you will not be confused by what you to see. In reality, if the **propagation delay** for the **OR gate** were used, the indicated **Glitch would shift to the right**.



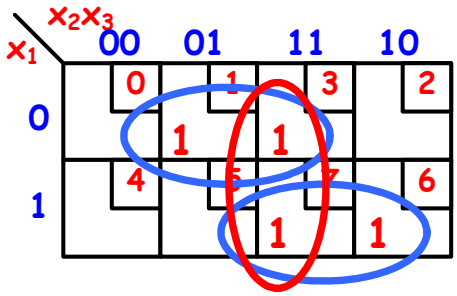
Note from the timing diagrams output that there is a period of time when the output goes low. The problem is that the truth table at the beginning of this problem showed that the output should stay high. This **GLITCH (timing hazard)** causes a period of time in which the output is **in ERROR**.

How do you go about solving this problem?

The main culprit that causes this issue is the propagation delay due to the inverter. This issue can be addressed by adding in a gate to the circuit. But where would it be added?

In order to formalize the process of finding out the best location a **K-map** the function needs to be created. The output expression for the circuit is:

$$f(X_1, X_2, X_3) = X_1X_2 + \overline{X_1}X_3$$

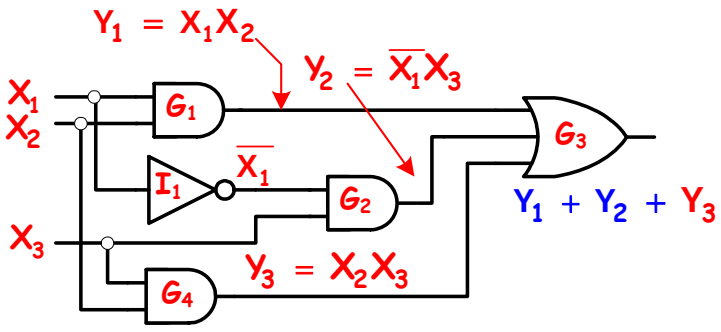


The problem here is that there are two adjacent min-terms **(3 and 7)** which are not connected (grouped). So, it is possibly (but not necessarily) a '**Static Timing Hazard**' in the circuit.

This possible problem can be fixed by breaking our "**smallest number of groups**" rule and adding in an additional group as shown. The new equation is:

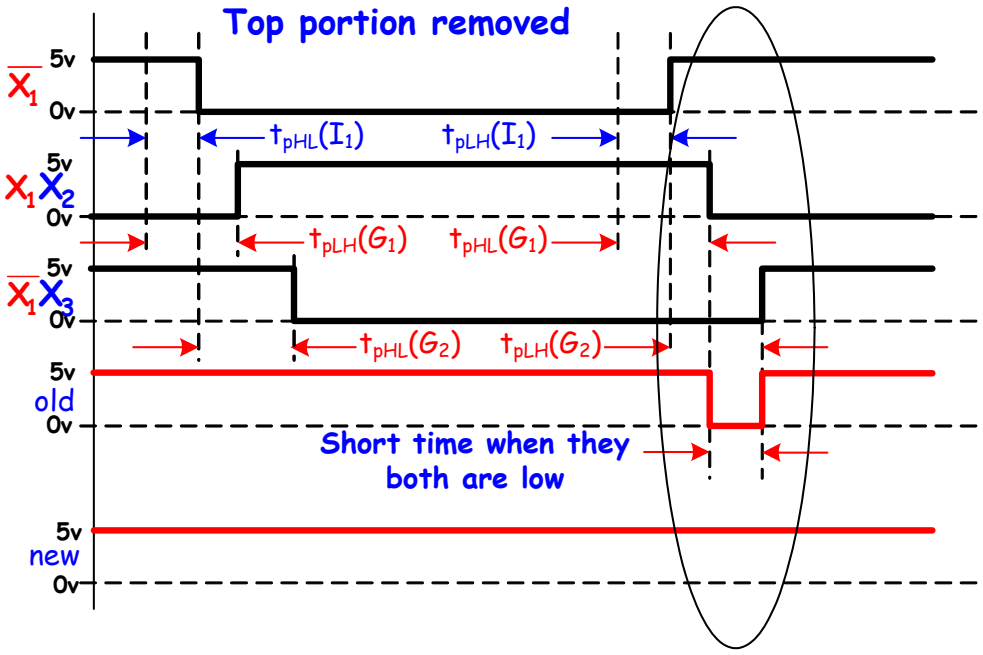
$$f(X_1, X_2, X_3) = X_1X_2 + \overline{X_1}X_3 \boxed{+X_2X_3}$$

The corrected circuit now is shown below.



The new timing diagram is shown on the next page.

The timing diagram for the corrected circuit is:



Note that the output of the three input OR gate now represents the way the truth table indicates the output should look like. Mission accomplished!

Question: Why not just always do this?

Answer: **Because we don't always have an entire circuit which is not synchronous. If we synchronized each gate by feeding in a "Read" pulse to each gate, we wouldn't have a static hazard problem except at high frequencies.**

Static Hazard Example:

Let's take a look at a 4 bit timing hazard.

The timing hazard is corrected by the two cell group 14 and 15.