Glitches

Delays (https://learn.digilentinc.com/list?tag=delays) Glitch (https://learn.digilentinc.com/list?tag=glitch) Basys2 (https://learn.digilentinc.com/list?tag=basys) Nexys (https://learn.digilentinc.com/list?tag=nexys)

Glitches

Propagation delays not only limit the speed at which a circuit can operate, they can also cause unexpected and unwanted transitions in outputs. These unwanted transitions, called *"glitches"*, result when an input signal changes state, provided the signal takes two or more paths through a circuit and one path has a longer delay than the other. The increased delay on one path can cause a glitch when the signal paths are recombined at an output gate. *Asymmetric path delays* commonly arise when an input signal drives an output through two or more paths, with one path containing an inverter and one not. Figure 1 below illustrates a glitch being formed by an inverter. Note the glitch (the 1-0-1 transition on Y) has the same duration as the delay in the inverter.

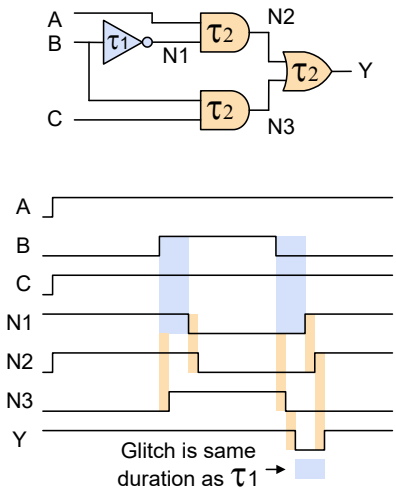

Figure 1. Glitch duration.

All logic gates add some delay to logic signals, with the amount of delay determined by their construction and output loading. In Fig. 1, the inverter is shown with a larger delay (identified by time T1) than the other gates (T2). This contrived example uses an over-long inverter delay to clearly show its role in creating an output glitch, but a glitch would appear no matter what the delay time. By carefully studying the timing diagram, it is clear how the inverter delay is related to the output glitch.

Glitches occur when an input is used in two product terms (or two sum terms for a POS equation), and inverted in one term but not in the other. This is illustrated in Fig. 2, the logic equation, and in the K-map in Fig. 3. In the K-map, two loops define a minimal logic expression. The $B \cdot C$ term is independent of A; that is, if B and C are both '1', then the output will be a '1' regardless of changes on A. Likewise, the term $A \cdot \overline{B}$ is independent of C, so that if A and B are '1' and '0', the output is a '1' regardless of how C might change. But note that if A is a '1' and C is a '1', the output should always be a '1' regardless of B, but no single term is driving the output independent of B. This is the situation that gives rise to the problem: two different product terms keep the output at a '1' when A and C are both '1'—one when B is a '1' ($B \cdot C$), and one when B is a '0' ($A \cdot \overline{B}$). Thus, as B changes, two different product terms must recombine at the output to keep the output high, and this is what gives rise to the glitch.

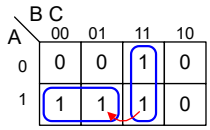

Figure 2. B as a coupled variable.

A circuit that can glitch can be identified by its schematic, K-map, or logic equation. In a schematic, an input that follows multiple paths to an output gate can create a glitch, if one path has an inverter and one does not. In a K-map, if loops are adjacent but not joined by an *"overlapping"* loop, then the adjacency not covered by a loop presents the opportunity for a glitch. For example, in the K-maps in Fig. 3 below, only K-map #1 results in a circuit that can glitch.

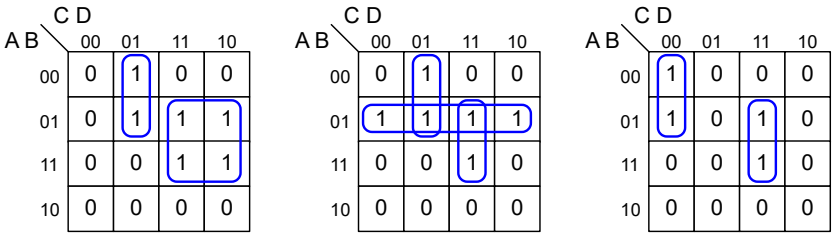

Figure 3. K-map 1: Possible glitch, K-map 2: No glitch, K-map 3: No glitch.

A glitch can be identified in a logic equation if two or more terms include the same logic signal, and the signal is inverted in one term but not in another. For this discussion, each pair of terms that contain a single variable that is inverted in one term but not the other are called "coupled terms", the inverted/non-inverted variable the "coupled variable", and the set of all other variables in both terms the "residue".

In some applications, it may be desirable to remove the glitch so that the output remains steady when a coupled variable changes state. Note that in the solution to Problem 1, the glitch on Y is only possible if B and C are held high. This observation can be generalized: for a glitch to occur, a logic circuit must be *"sensitized"* to a coupled variable by driving all inputs to appropriate levels so that only the coupled variable can affect the output. In an SOP circuit, this means that all inputs other than the coupled input must be driven to a '1' so that they have no effect on the outputs of the first-level AND gates.

This observation leads directly to the method for removing a glitch from a logic circuit: combine all residue input signals in a new first-level logic gate (i.e., an AND gate for an SOP circuit), and add the new gate to the circuit. For example, in the equation $X = \overline{A} \cdot B + A \cdot C$, the coupled term is A, the residue signals would combine to form the term $B \cdot C$, and that term would be added to the circuit to form $X = \overline{A} \cdot B + A \cdot C + B \cdot C$. This is shown in the K-map—note that the original equation is minimal (blue loops), and that the glitch-free equation adds a redundant term (red loop).

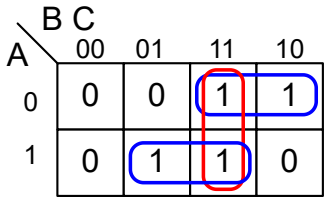

Figure 4. Residue input.

This is always the case—removing glitches requires a larger circuit with redundant logic. In practice, it is almost always preferable to design minimal circuits and deal with glitches in another manner (discussed in a later module). Perhaps the best lesson is to be aware that in general, whenever an input to a combinational circuit changes, glitches are possible (at least, until proven otherwise).

The loops for the original SOP equation in problem 1 did not overlap, and this is the hallmark of a potential glitch. When a loop for the redundant term is added, every loop overlaps with at least one other, and no glitches are possible.

If non-overlapping (or isolated) loops are located in non-adjacent K-map cells (see K-map #3 in Fig. 3 above), there are no coupled terms, no coupled variables, and it is not possible to add a loop (or loops) to cause all loops to overlap with at least one other. In such a case, no single input change can cause a glitch. In this type of circuit, two or more inputs might be directed to change state "at the same time", with the desired outcome of having the output remain at some stable state. For example, in the circuit Y = $\overline{A} \cdot \overline{C} \cdot \overline{D} + B \cdot C \cdot D$ from K-map #3 above, it might be desired that all inputs go from '0' to '1' simultaneously, and that in response, the output stay constantly at '1'. In practice, it is impossible to change all inputs simultaneously (at least, on a scale of picoseconds), and as a result, the output will show a glitch- like transition equal in duration to the time difference between input signal changes. Such unwanted transitions cannot be eliminated by adding redundant gates; rather, they must be dealt with by redefining the circuit or with sampling and pipelining (these topics are discussed in a later module). We will not deal further with unwanted output transitions resulting from multiple input changes here.

Most of the glitch discussion so far has focused on SOP circuits, but the same phenomenon is present in POS circuits as well. POS circuits suffer from glitches for the same reasons as SOP circuits (asymmetric path delays for an input arriving at multiple input gates). As you might expect, the conditions required are similar, but not identical to the SOP case.

These simple experiments demonstrate the basic effects of gate delays on digital circuits—namely, output glitches are possible in response to input transitions, provided the input passes through asymmetric circuit path delays in forming the output. In the more general case, any time an input passes through two different circuit branches, and those two branches are recombined at a "downstream" point in the circuit, timing problems like glitches are possible. Again, the lesson is to be aware that signals take time to propagate through logic circuits, and different circuit paths have different delays. And in certain cases, those differential delays can cause problems.

Important Ideas

- Unwanted transitions, called "glitches", result when an input signal changes state, provided the signal takes two or more paths through a circuit and one path has a longer delay than the other.
- Asymmetric path delays commonly arise when an input signal drives an output through two or more paths, with one path containing an inverter and one not.
- All logic gates add some delay to logic signals, with the amount of delay determined by their construction and output loading.
- A glitch can be identified in a logic equation if two or more terms include the same logic signal, and the signal is inverted in one term but not in another.
- Removing glitches requires a larger circuit with redundant logic. In practice, it is almost always preferable to design minimal circuits and deal with glitches in another manner.

ALSO ON **DIGILENT**

Piezoelectric Switch with Schmitt Trigger

7 years ago • 3 comments

For this project, we will be creating a switch that can turn an LED on and off by …

Getting Started with FPGA

7 years ago • 2 comments

This project sets up your FPGA board for use with Xilinx® Vivado™ and …

Active and P Sign Conver

7 years ago • 1 c

An explanation differences betv and passive sig