<u>LAB 4: TIMING HAZARDS</u>

This lab will explore the effects that nonideal time delays on gate outputs, the nature of the "glitches" that result in the output, methods for mitigating these hazards, and how/why these methods work, revisiting the particle accelerator case study in the previous lab, Lab 3. In particular, we will look at a couple of modules that allow key functions in the accelerator to either proceed when users want them to, and prevent them otherwise. Timing hazards pose unique issues in maintaining this control, and as such must be detected and resolved.

Going forward, *assume a delay of one unit of time for every gate.*

(As in past labs, please use Logisim-evolution or CircuitVerse to implement and simulate circuits.)

<u>Part 1: Beam Safety Module</u>

The particle accelerator operates by firing a beam of particles into a test area. Conditions in the detector are assessed, and the detector, in turn, provides a 4-bit input C (with C_3 as the MSB, and C_0 as the LSB). Based on these conditions, the Beam Safety Module assesses whether it is safe to activate the beam, returning an output S, which will be 0 if it is unsafe to activate, and 1 if it is, immediately activating the beam.

Given the nature of the Beam Safety Module's role, it is absolutely vital to ensure that the Beam Safety Module <u>does not</u> return 1 (however briefly) when it should return 0, as it will activate the beam when it is unsafe.

Before the BSM is improved, a total redesign has been ordered. The original design (Fig. 1) was designed at a time of greater scarcity, when the group only had NOR gates. ***Determine the function S(C) = S($C_3C_2C_1C_0$) as a POS. Then, design and implement an AND-OR circuit that realizes this function.***

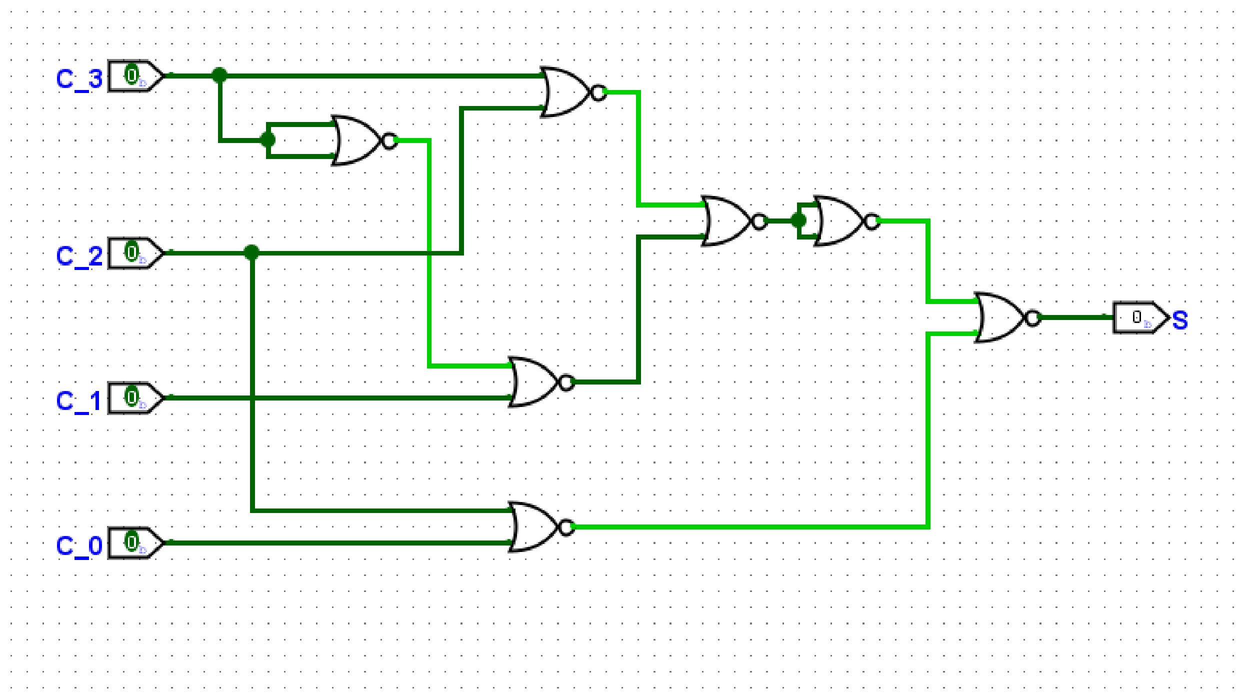

Figure 1. The original design of the BSM, implemented entirely using NOT gates.

With the Beam Safety Module redesign, you now have to contend with potential glitches. For this, you can make use of the POS version of the Consensus Theorem:

$$(A+B)(\overline{A}+C) = (A+B)(\overline{A}+C)(B+C) \quad \textbf{(1)}$$

First, test and determine any hazardous input transitions. For each hazardous transition, use the Consensus Theorem to determine a consensus term. Using these consensus terms, determine a hazard-free expression for S and modify your redesign of the BSM accordingly.

<u>Part 2: Data Writing Module</u>

The Data Writing Module (DWM) will come in two variations: "mini" version (or a mini DWM), and the regular version (just DWM). The "mini" version takes a 4-bit input from the detector, D (with D_3 as the MSB and D_0 as the LSB), representing the condition of the data being received. Based on these conditions, the mini DWM will output a value F will be 0 if the users want to "write" the data generated, and 0 if the data should be tossed.

Just like with the BSM, the DWM must *not* *glitch to 0 (however briefly) when the output should be 1, as this will result in the loss of data sought out by the user, and thus a lower quality analysis*.

F is naturally a function of the input D:

$F(D) \; = \; F(D_0,\, D_1,\, D_2,\, D_3) \; = (D_2' D_0') + (D_3 D_2 D_1') + (D_3 D_1 D_0)$ **(2)**

Design and implement an AND-OR circuit to realize the function F.

Does this implementation of the DWM have any timing hazards? If so, determine what input transitions cause the hazard, what hazardous pairs exist in the current SOP of F, and what consensus terms can be added to F from said hazardous pairs.

Afterward, modify your original AND-OR circuit for F to remove the hazards found.