The identification of transition arcs of Boolean functions is an important step in the simulation and design analysis of logic circuits. This identification allows the measurement of delay times in output state transitions given a change in a specific input. The automatization of this step is relevant for its potential to facilitate the study of circuits with truth tables with a large number of inputs and to speed up electrical characterization work in adjacent research. This work explores two algorithmic alternatives for recognizing these arcs. The two algorithms differ fundamentally in the way they deal with the data present in the circuit's truth table. In algorithm 1, the truth table is interpreted in the sense of its input and output columns. The value of each input is stored individually in a vector and will also be accessed individually later in the algorithm. The set of vectors referring to the inputs is saved in a large vector. Initially, all transition arc possibilities are created through a combinatorial loop, proportionally to the number of inputs. Next, these arcs are selected, excluding those that do not generate a transition in the output value or that have a change in more than one input. In algorithm 2, the truth table is interpreted from the perspective of the value of each line, taking advantage of the fact that each line corresponds to the value of an integer varying between 0 and 'n' for a table with 'n' entries. From each line, new input vectors are initially created by flipping each bit of the vector. This ensures that only 1 bit (corresponding to an input) is modified. After that, the transition arcs corresponding to the original and the new input vector are selected, excluding those that were already previously created and those that do not generate a change in the output value. Another significant difference between the two algorithms is the final format of the representation of the transition arcs found. Algorithm 1 returns a vector of tuples corresponding to the lines of the transition arc. Meanwhile, algorithm 2 returns a vector of vectors, where the numbers inside each vector correspond to the input vectors that form the transition arc when paired with the number of the vector's position. The next stage of the ongoing study is the in-depth analysis of the two developed algorithms. The objective is to measure its efficiency and functioning, comparing the results in different truth table situations. This measurement will be done using metrics such as execution time, algorithmic complexity, and memory usage. For this, the algorithms will be implemented using the Rust programming language.