LTOETC (Less than or equal to Checker):

Sorting plays an important role in data processing and digital signal/image processing and there is a huge focus on the development and analysis of sorting algorithms. A major portion of digital system malfunctions is attributed to transient/intermittent faults, concurrent error detection techniques are used to ensure that erroneous outputs can be detected immediately during normal system operation. When a failure occurs in a processing element of a VLSI sorter, three types of errors may occur:

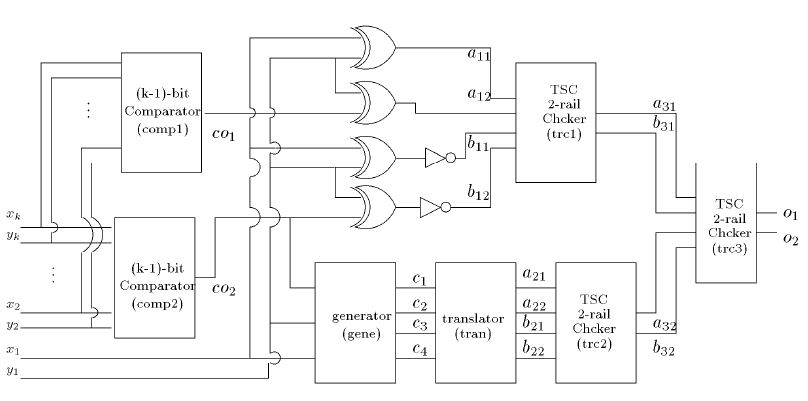
1. *Functional Errors:* The operands are incorrectly ordered, but the value of each operand is not changed.
2. *Data Errors:* One or more bits of operand(s) are changed. In multiple as well as single bit data errors, the correct order of the operands is maintained.
3. *Hybrid Errors:* A functional error as well as data error occurring simultaneously is a hybrid error.

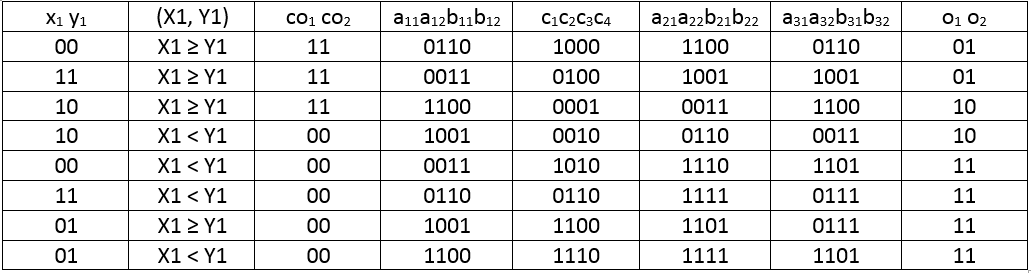
For example, there is a functional error if a sorted sequence (11, 9, 7, 5) is received as (11, 7, 9, 5). (11, 9, 7, 0) is a data error and (11, 6, 9, 5) is a hybrid error. There are two types of checkers to detect the above three types of errors. In addition, if such a checker is a self-testing checker, then it is capable of detecting fault(s) in the VLSI sorter as well as in itself during normal operation. LTOETC is the self-testing non increasing order checker. The advantage of this checker is that it can be used to check the sorter as well as itself during normal operations. The faults to be considered include all single stuck-at faults for primitive gates, which include AND, OR, inverter, exclusive-OR.

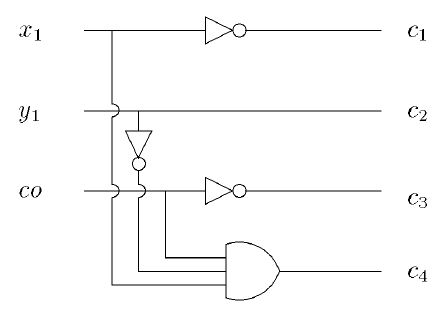
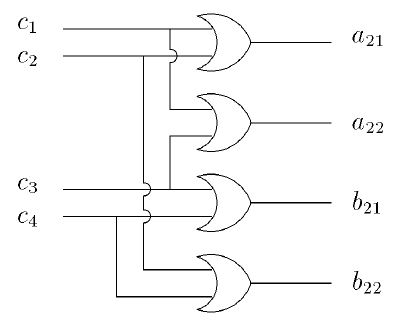
Consider a sequence N1, N2, N3, N4, N5 where each number is a non-negative number. The input code space includes all sequences that are in non increasing order and the input non-code space includes all sequences that are not in non increasing order. When an error occurs, all the numbers remain the same and it’s the sorting function that has not been implemented correctly which is in fact a functional error.

LTOETC consists of (k-1) bit comparators, inverters, exclusive-or gates, TSC checkers for two-part two-rail code, translator and 1-out-of-4 generator. If (x2,...,xk) >= (y2,...,yk) then output of the comparator co1 = 1 & co2 = 1 else co1 = 0 and co2 = 0.

In order to prove LTOETC is a self-testing checker, we need to prove that its code-disjoint and self-testing.







1. LTOETC is code-disjoint

Since inputs to LTOETC are sorted binary operands where (x1,x2,...xk)>= (y1,y2,...,yk) is always true during normal operation. Let X1 = (x2,....,xk) and Y1 = (y2,....,yk). The input code space includes:

1. x1 y1 = 00 and X1 ≥ Y1

2. x1 y1 = 11 and X1 ≥ Y1

3. x1 y1 = 10 and X1 ≥ Y1

4. x1 y1 = 10 and X1 < Y1

From the figures blah blah and the above table show the logic values of inputs and outputs of each functional block. Thus, LTOETC is code-disjoint.

2. LTOETC is self-testing circuit

a. Each functional block receives all necessary test vectors so that it is fully tested during normal operation.

* TSC checkers for the two-pair two-rail code receive 0011, 0110, 1001, 1100
* Translator receives 0001, 0010, 0100, 1000
* Each inverter receives 0 or 1
* Each XOR gate receives three vectors from the set {00, 01, 10, 11}.
* 3-input AND gate in the generator block receives 110, 101, 011, 111
* (k-1) bit comparators receive all possible input combinations.

b. When a fault in each functional block is excited, a non-code word will be generated at the outputs o1, o2.

* If a fault occurs in generator, translator, trc1, trc2, one of XOR gates and one of the inverters shown in Figure blah, then a test vector will generate 00 or 11 either at a31b31 or a32b32 (not both). Hence o1o2 become either 00 or 11.
* If a fault occurs in trc3, then a test vector will generate either 00 or 11.
* If a fault occurs in (k-1)-bit comparator, comp1, then a test vector will generate 00 or 11 only at a31b31. Hence, o1o2 become either 00 or 11.
* If a fault occurs in (k-1)-bit comparator, comp2, then a test vector will generate 00 or 11 only at a32b32. Hence, o1o2 become either 00 or 11.

From the above arguments, we can conclude that the proposed LTOETC is self-testing checker.

To check if non-increasing order checker is detecting functional errors correctly, consider five numbers N1, N2, N3, N4, N5 where N1 ≥ N2 ≥ N3 ≥ N4 ≥ N5. Here, N1, N2 will be fed as inputs to LTOETC L1; N2, N3 are fed to LTOETC L2; N3, N4 are fed to LTOETC L3; N4, N5 are fed as inputs to LTOETC L4. To make this circuit also self-checking, provide outputs to a multi-layer TSC.

If N1 < N2 or N2 < N3 or N3 < N4 or N4 < N5, output o1o2 of this circuit should be 11.

<<<<<<<<<<<<<draw a diagram here >>>>>>>>>>>>>>>>>