Problem

Define the function $majority_n : \{0,1\}^n \longrightarrow \{0,1\}$ as

$$majority_n(x_1, \ldots, x_n) = \begin{cases} 0 & \sum x_i < n/2\,; \\ 1 & \sum x_i \geq n/2\,. \end{cases}$$

Thus, the $majority_n$ function returns the majority vote of the inputs. Show that $majority_n$ can be computed with:

a. $O(n^2)$ size circuits.

b. $O(n \log n)$ size circuits. (Hint: Recursively divide the number of inputs in half and use the result of Problem 9.23.)

Step-by-step solution

Step 1 of 2

a) Let the number of inputs taken is n. A **bubble-sort** can be implemented as a circuit. It is used tocompare two bits and after comparing, reordering them if necessary is rather easy. The inputs can be called as x_1, x_2 and the outputs can be called as y_1, y_2. A sub-circuit can be written which accomplishes this as $y_1 = OR(x_1, x_2)$ and $y_2 = AND(x_1, x_2)$.**This circuit contains a size of two**.

• Now, the action of the bubble-sort algorithm can be mimicked on an array. It can be implemented one step at position to be the n input, n-output sub-circuit that passes through all the inputs taken as $< k \; and \geq k+1$ are unchanged.

• Now, the compare-swap sub-circuit, which is described above, on $< k \; and \geq k+1st$ input can be used to generate the kth and $k+1st$ output. This still has size two. Now, **a pass** can be implemented as the serial concatenation of steps for each of $k = 1, 2, \ldots, n-1$, which has a size $(n-1)*2$.

• A bubble-sort can be Proceed to implement as the serial concatenation of n passes. Therefore, this gives a size $\mathbf{n(n-1)*2 = O(n^2)}$.

Therefore, it can be said that $\mathbf{majority_n}$ can be computed in $O(n^2)$ size circuits.

Comment

Step 2 of 2

b) Let the **number of inputs taken** is n. A **Merge-sort** can be implemented as a circuit. It is used to compare two bits after recursively dividing the given inputs in to half. The total time taken here (to divide the inputs into equal halves iteratively)is $\log n$.

• Finally at the last, the inputs can be called as x_1, x_2 and the outputs can be called as y_1.

• Now,**the action of the merge-sort algorithm** can be mimicked on an array. It can be implemented one step at position to be the n input, $n/2$**-output sub-circuit**.

• Now, **a pass** can be implemented as the serial concatenation of steps, which has a size $n \log n$. Therefore, this gives a size $n * \log n = O(n \log n)$.

Therefore, it can be said that $\mathbf{majority_n}$ can be computed in $O(n \log n)$ size circuits.

Comment