Exercise 6: Containment

Task 1: Containing Choice

The goal in this exercise is to prove Lemma 6.5.

a) Show the equivalence stated in the lemma.

b) Construct a $k$-bit $\text{MUX}_M$ implementation out of two $(k-1)$-bit $\text{MUX}_M$ implementations and a CMUX. (Hint: To show correctness, make a case distinction on the $k$th control bit, which is fed to the CMUX.)

c) What is the size of the resulting $\text{MUX}_M$ implementation when applying the construction from b) recursively?

Solution

a) Suppose $\text{MUX}_M(x, s) = b \in \{0, 1\}$, or, equivalently, that for all $x \preceq y \in \{0, 1\}^k$ and $s \preceq s' \in \{0, 1\}^k$, we have that $y_{s'} = \text{MUX}(y, s') = b$. It follows that $x_{s'} = b$ for each $s \preceq s' \in \{0, 1\}^k$, as otherwise $z \in \{0, 1\}^k$ given by

$$z_k := \begin{cases} 1 - b & \text{if } s = s' \\ y_s & \text{else} \end{cases}$$

satisfied that $x \preceq z$ and $\text{MUX}(z, s') = 1 - b$. As this holds for any $s \preceq s' \in \{0, 1\}^k$, this shows that

$$\forall s \preceq s' \in \{0, 1\}^k: x_{s'} = b.$$

For the reverse implication, assume that the latter is true. Thus, for all $x \preceq y \in \{0, 1\}^k$ and $s \preceq s' \in \{0, 1\}^k$, $\text{MUX}(y, s') = b$. This is equivalent to $\text{MUX}_M(x, s) = b \in \{0, 1\}$.

b) For $s \in \{0, 1, M\}^k$, denote by $s_1$ the most significant bit of $s$ and by $s_{2..k}$ the remaining bits of $s$. Take two copies of a circuit implementing a $(k-1)$-bit $\text{MUX}_M$ and fix inputs $x \in \{0, 1, M\}^{2k}$, $s \in \{0, 1, M\}^k$. To the first one, feed the $2^k - 1$ inputs $x_{s'}$ with $s'_1 = 0$ and the control bits $s_{2..k}$. Similarly, the second copy receives inputs $x_{s'}$ with $s'_1 = 1$ and control bits $s_{2..k}$. The outputs of these circuits are fed as inputs into a CMUX, whose control bit is given by $s_1$. We claim that this results in an implementation of a $k$-bit $\text{MUX}_M$, whose output is given by the output of the CMUX.

To show this, we use the equivalence from a) and make a case distinction. If $s_1 = 0$, then $s'_1 = 0$ for any $s \preceq s'$. Hence, $\text{MUX}_M(x, s) = b \in \{0, 1\}$ if and only if for all $s_{2..k} \preceq s'_{2..k} \in \{0, 1\}^{k-1}$ we have that $x_{0s'_{2..k}} = b$. This is equivalent to the first $(k-1)$-bit $\text{MUX}_M$ having output $b$, which due to $s_1 = 0$ is equivalent to the output of the CMUX being 0. Arguing analogously, we see that the implementation is also correct for $s_1 = 1$.

It remains to consider the case that $s_1 = M$. Thus, the equivalence from a) shows that $\text{MUX}_M(x, s) = b \in \{0, 1\}$ if and only if we have that for all $s_{2..k} \preceq s'_{2..k} \in \{0, 1\}^{k-1}$ both that $x_{0s'_{2..k}} = b$ and that $x_{1s'_{2..k}} = b$. This is equivalent to both $(k-1)$-bit $\text{MUX}_M$ implementations having output $b$. Accordingly, the selectable inputs of the CMUX are both $b$, and its output is $b$ despite the select bit being $M$.

c) The size of a $k$-bit $\text{MUX}_M$ implementation following this construction is twice the size of a $(k-1)$-bit $\text{MUX}_M$ plus that of a CMUX. Summing over all levels of the construction, we thus get $\sum_{k=0}^{k-1} 2^k = 2^k - 1$ times the size of a CMUX. The size of a CMUX is constant, so we end up with size $O(2^k)$.
**Task 2: Copy and Conquer**

Masking registers are registers that have somewhat predictable behavior when storing a metastable bit. A mask-0 register \( R \) has the following behavior. Like an ordinary register, if \( R \) stores a bit \( b \in \{0,1\} \), then every time the value of \( R \) is read, it will return \( b \). If the bit stored in \( R \) is \( M \), then every sequence of accesses to \( R \) will return a sequence of values of the form 00\(
\cdots\)01\(
\cdots\). In particular, every sequence of accesses to \( R \) will return a sequence of values containing at most a single \( M \).

a) Let \( f : \{0,1\}^n \rightarrow \{0,1\} \) be a function, and suppose \( x \in \{0,1,M\}^n \) satisfies \( f_M(x) \neq M \). Let \( C \) be an arbitrary (not necessarily metastability containing!) circuit implementing \( f \). Suppose the individual bits of \( x \) are stored in mask-0 registers, and let \( x^{(1)}, x^{(2)}, \ldots, x^{(2n+1)} \) denote the values of \( x \) read by a sequence of accesses to the registers storing \( x \). Finally, for each \( i \in \{1,2,\ldots,2n+1\} \), define \( y_i = C(x^{(i)}) \). Show that the value \( f_M(x) \) can be inferred from \( y_1, y_2, \ldots, y_{2n+1} \).

b) Come up with a small circuit that sorts its \( n \) inputs according to the total order \( 0 \leq M \leq 1 \). That is, devise a circuit \( C \) with \( n \) inputs and \( n \) outputs such that if \( y = C(x) \) then we have \( y_1 \leq y_2 \leq \cdots \leq y_n \), where \( y \) has the same number of 0s, 1s, and \( Ms \) as \( x \). (Hint: Figure out a solution sorting two values and then plug it into a binary sorting network to get the general circuit. You don’t have to (re)invent sorting networks, you may just point to a reference.)

c) Combine a) and b) to derive a circuit implementing \( f_M \) from any (non-containing) circuit implementing \( f \)! Your solution should only be by a factor of \( n^{O(1)} \) larger than to the non-containing solution.

**Solution**

a) As at most a single \( M \) is read from each masking register, we have that at least \( n+1 \) of the copies are stable. For each of these \( n+1 \) copies, by definition of \( f_M \) the circuit implementing \( f \) will compute output \( f_M(x) \).

b) Given two inputs \( x_1 \) and \( x_2 \), in terms of the order \( 0 \leq M \leq 1 \), we simply have that \( \text{OR}_M(x_1, x_2) = \max\{x_1, x_2\} \) and \( \text{AND}_M(x_1, x_2) = \min\{x_1, x_2\} \). Using an OR and an AND gate as comparator in a sorting network, we get a circuit of size \( O(n \log n) \) sorting \( n \) inputs.

c) We make \( 2n+1 \) copies of the input and feed each copy into a copy of the circuit \( C \) implementing \( f \), as in a). If \( f_M(x) = b \in \{0,1\} \), from a) we know that at least \( n+1 \) of the outputs are \( b \). In particular, after sorting the \( (n+1)_{th} \) value is \( b \). Using b), we can output the \( (n+1)_{th} \) value of the sorted sequence. In total, we have spent \( O(n|C| + n \log n) \subset O(n^{O(1)}|C|) \) gates.

**Task 3*: Clocked Circuits**

a) How would a model for clocked circuits based on the same worst-case assumptions look like? (Hint: Reading up on it is fine.)

b) Standard registers, when being read, will output \( M \) if they’re internally metastable and 0 or 1, respectively, when they’re stable. Show that they add no power in terms of the functions that can be computed! (Hint: Unroll the circuit, i.e., perform the multi-round computation in a single round with a larger circuit.)
c) In Task 2, you saw that masking registers allow for more efficient metastability-containing circuits. Show that they are also computationally more powerful, i.e., they can compute functions that cannot be computed with masking registers! (Hint: You already used this in Task 2!)