Retake of Quiz 2: Universality

Collaboration: This quiz is open note but individual. You may use any resources that were provided to you by the course, or that you had recorded as part of your own notes prior to when you first viewed this quiz. You may not discuss this quiz with your cohort-mates, class-mates, tutor, friends, family, or anyone else except the course staff (who will answer clarification questions only).

Problem 1: Comparing Computing Models

In the original Quiz 2 we defined the function $\texttt{COMP}_n$. Following this definition, we would have that $\texttt{COMP}_1(a, b) \equiv a \wedge \neg b$.

Using this definition of $\texttt{COMP}_1$, determine whether the gate set $\{\texttt{COMP}_1, \texttt{NOT}\}$ is equivalent to $\{\texttt{AND}, \texttt{OR}, \texttt{NOT}\}$. Support your answer with a proof.

Problem 2: Counting Gates

We will define the function $\texttt{PALI}_n : \{0,1\}^{2n} \to \{0,1\}$ such that $\texttt{PALI}_n(a_{2n-1}, \ldots a_0)$ returns 1 provided that the string $a_{2n-1} \ldots a_0$ is a palindrome. A palindrome is a string that is the same forwards as backwards, so $\texttt{PALI}_n(a_{2n-1}, \ldots a_0)$ returns 1 provided that the string $a_{2n-1} \ldots a_0$ is the same as the string $a_0 \ldots a_{2n-1}$.

Here is an example of a (sugary) way that we could implement $\texttt{PALI}_3$.

```
def PALI3(a5, a4, a3, a2, a1, a0):
    firstlast_diff = XOR(a_5, a_0)
    firstlast_same = NOT(firstlast_diff)
    mid_pali = PALI2(a4, a3, a2, a1)
    return IF(mid_pali, firstlast_same, 0)
```

Use induction to show that for every $n \in \mathbb{N}$ we can implement the function $\texttt{PALI}_n$ using no more than $8n$ NAND gates. (*Hints:* You can use without proof that XOR and IF can each be done using no more than 4 NAND gates. Generalizing the given implementation of $\texttt{PALI}_3$ will not work, you'll need to provide an equivalent but more efficient implementation to get to $\leq 8n$ gates used.)