8th tutorial

1. Fast Multiplication. Derive an $\mathcal{O}(n \log n)$ algorithm for multiplying two n-bit numbers (using FFT).

---

2. AND OR NOT. Show that any Boolean k bit function (given by a table) can be expressed using the AND, OR, and NOT gates. (Bonus: show that it can be done by only using the NAND gate, which computes from two inputs the negation of the AND function. It suffices to show that other gates, e.g. AND OR NOT, can be expressed using NANDs.)
What is the depth of the circuit, and how many gates are needed?

3. Divisibility. Design a network of depth $\mathcal{O}(\log n)$ which decides whether an n-bit number given in binary is divisible by 5.
Hint: Assume you are working over the alphabet $\Sigma = 0, 1, 2, 3, 4$ (but the input is still given in binary).

4. Highest bit. Design a circuit of depth $\mathcal{O}(\log n)$ and of size $\mathcal{O}(n)$ which nulls all bits except the left-most (most significant) one, e.g. for input 0110 it returns output 0100.

5. Multiplication. We know how to add two n bit numbers using a circuit of depth $\mathcal{O}(\log n)$ and size $\mathcal{O}(n)$. Using this circuit design as small a circuit as possible which multiplies two n bit numbers on input. (Hint: consider what multiplication looks like in binary. Next, try to reduce, in constant parallel time, adding three numbers to adding two numbers.)

6. Complicated functions. Show that for no $c > 0$ is it true that every k-input boolean function is computable by a circuit with $\mathcal{O}(k^c)$ gates. Hint: how many boolean functions are there with k inputs? and how many circuits with $\mathcal{O}(k^c)$ gates?