CSG399: Gems of Theoretical Computer Science. Lecture 8. Feb. 3, 2009.
Instructor: Emanuele Viola Scribe: Sathyaseelan Nethaji

Arithmetic in Log-Depth Circuits

In this lecture we show how small-depth circuit can implement various fundamental arithmetic operations.

1 Addition

Input: Two n-bit Integers $X, Y \in \{0,1\}^n$.
Output: $X + Y \in \{0,1\}^{n+1}$.

Theorem 1. *Addition is computable by polynomial-size circuits of unbounded fan-in and depth $O(1)$. In particular, addition is computable by fan-in 2 circuits of depth $O(\log n)$.*

Proof. The difficulty in proving the above theorem is that the computation of the carries appears sequential. Note however that if the carries $c_n, ..., c_1 \in \{0,1\}$ are given then each bit of $X + Y$ can be computed by circuits of size $O(1)$ (and hence depth $O(1)$). Specifically $(X + Y)_1 = X_1 + Y_1 + c_1$ where here "+" denotes bit XOR, and similarly for the other bits.

Our approach is to compute all the carries in parallel using *carry look-ahead.* Specifically we note that the i-th carry is 1 if and only if there is some less significant position $j < i$ where the carry is generated and it is propagated up to i. This can be written as

$$c_i = 1 \iff \bigvee_{j<i} \left(X_j = 1 \wedge Y_j = 1 \bigwedge_{k=j+1}^{i-1} (X_k = 1 \vee Y_k = 1) \right).$$

The above is an unbounded fan-in circuit of size poly(n) and depth $O(1)$. By the claim from last lecture, this can be implemented by a fan-in 2 circuit of depth $O(\log n)$. □

2 Iterated Addition

Input: n n-bit integers $x_1, ..., x_n \in \{0,1\}^n$.
Output: $\sum x_i$.

If we are able to compute iterated addition in depth $O(\log n)$, then Majority can also be computed in depth $O(\log n)$.

Theorem 2. *Iterated Addition is computable by fan-in 2 circuits of depth $O(\log n)$.*

Proof. We use the technique "2-out-of-3:" given 3 integers X, Y, Z, we compute 2 integers a, b such that

$$X + Y + Z = a + b,$$

where each bit of a, b is a function of one bit from X, one from Y, and one from Z, and thus can be computed by a circuit of constant size. If you can do this, then to compute iterated addition we construct a tree of logarithmic depth to reduce the original sum to a sum 2 terms, which we add as explained before.

Proof of trick: $X_i + Y_i + Z_i \leq 3$, so a_i will get the least significant bit, b_{i+1} will get the most significant one. Note that a_i is the XOR $X_i + Y_i + Z_i \in \{0, 1\}$, while b_{i+1} is the majority of X_i, Y_i, Z_i. □

3 Multiplication

Input: X, Y n-bit integers,
Output: $X \cdot Y$ $2n$-bit integer.

Theorem 3. *Multiplication is computable by fan-in 2 circuits of depth $O(\log n)$.*

Proof. "Shift and Add:" $X \cdot Y = \sum_i (X \cdot 2^i \cdot b_i)$. Each term $(X \cdot 2^i \cdot b_i)$ is easily computable in constant depth, since multiplication by 2^i is just a bit shift. Then we apply iterated addition. □

4 Division

Input: X n-bit integer,
Output: $1/X$ to within n bits of precision.
Note: if we can compute $1/X$, can compute Y/X as $Y \cdot 1/X$.

To divide, we are going to power.

Theorem 4 (Powering)**.** *Given X n-bit integer, we can compute X^n by fan-in 2 depth $O(\log n)$ circuits.*

Theorem 5 (Division)**.** *Given $X \geq 0$ n-bit integer, we can compute $1/X$ to within n bits of precision by fan-in 2 circuits of depth $O(\log n)$.*

Proof of Theorem 5 assuming Theorem 4. Given X, determine j such that $2^j \leq X < 2^{j+1}$, let $U := 1 - X/2^{j+1} \in (0, 1/2)$. Using iterated addition and multiplication, compute

$$2^{-(j+1)}(1 + U + U^2 + ... + U^n) = 2^{-(j+1)} \cdot \frac{1 - U^{n+1}}{1 - U}$$

$$= 2^{-(j+1)} \cdot \frac{1 - U^{n+1}}{X \cdot 2^{-(j+1)}} = \frac{1}{X} - \frac{U^{n+1}}{X} = \frac{1}{X} \pm 2^{-n}.$$

□

To power (Theorem 4) we use various tools from number theory.

5 Tools from number theory

Theorem 6 (Chinese Remainder Theorem)**.** *Let* $p_1, ..., p_l$ *be distinct primes and* $p' := \prod_i p_i$. $\mathbb{Z}_{p'}$ *is isomorphic to* $\mathbb{Z}_{p_1} \times \ldots \times \mathbb{Z}_{p_l}$.

The forward direction of the isomorphism is given by $x \in \mathbb{Z}_{p'} \to (x \text{ mod } p_1, x \text{ mod } p_2, ..., x \text{ mod } p_l) \in \mathbb{Z}_{p_1} \times ... \times \mathbb{Z}_{p_l}$.

For the converse direction, we claim that there exist integers $e_1, ..., e_l \leq \text{poly}(p')$ *such that* $(x \text{ mod } p_1, x \text{ mod } p_2, ..., x \text{ mod } p_l) \in \mathbb{Z}_{p_1} \times ... \times \mathbb{Z}_{p_l} \to x := \sum_{i=1}^{l} e_i \cdot (x \text{ mod } p_i)$.

Each integer e_i is 0 mod p_j for $j \neq i$, is 1 mod p_i, and can be found using the extended euclidean algorithm.

For example, $\mathbb{Z}_6 \simeq \mathbb{Z}_2 \times \mathbb{Z}_3$, and $2 + 3 = 5 \to (0, 2) + (1, 0) = (1, 2)$.

We recall the following celebrated result on the density of prime numbers, a weak version of which will be proved in the next lecture.

Theorem 7 (Prime number theorem)**.** $\lim_{n\to\infty}(\textit{Number of primes} \leq n)/(n/\log_e n) = 1$.

6 Powering

Input: $X \in \{0, 1\}^n$. Output: X^n.

Beginning of the proof of Theorem 4 that powering has fan-in 2 *circuits of depth* $O(\log n)$. Let $l := n^3$. We use the following algorithm:

1. Compute (X mod p_1, X mod $p_2, \ldots, X$ mod p_l),
2. Compute (X^n mod $p_1, \ldots, X^n$ mod p_l),
3. Compute X^n.

Correctness: Observe $X^n \leq 2^{n^2}$, thus the correctness follows from the Chinese remaindering theorem if $p' := \prod_{i=1}^{l} p_i \geq 2^{n^2}$, which follows immediately by our choice of l and the fact that each prime is at least 2.

In the next class we will show that the above algorithm can be implemented by log-depth circuits. □