A Simple $\mathcal{NC}$ Problem

There is a lower bound do the amount of time any machine takes to execute one step.[1] Thus, Moore's Law, that computation speed increases expondentially, cannot hold forever. In fact, it's "already dead," according to some experts.

Eventually, to achieve higher speed, we must use parallel computation. $\mathcal{NC}$ (Nick's Class) is the the class of all languages which are accepted within polylogarithmic time using polynomially many parallel processors. That is, if $L \in \mathcal{NC}$, there is a constant k such that L is accepted (equivalently, decided) in $O(\log^k n)$ time by $O(n^k)$ processors.

Odd Number of 1's

We now describe a simple $\mathcal{NC}$ language. Let $\Sigma = \{0, 1\}$, and let $L = \{w \in \Sigma^* : \#_1(w)\%2 = 1\}$, that is, binary strings with an odd number of 1's.

We give an $\mathcal{NC}$ algorithm $\mathcal{A}$ which decides L in $O(\log n)$ time using $O(n \log n)$ processors, where n is the length of the input string w. If $x \in \Sigma^*$, define $F(x) = \#_1(x)\%2$, which we interpret as a Boolean function, which is true if and only if x has an odd number of 1's. The goal is to compute $F(w)$. There are $\Theta(n^2)$ substrings of w, but $\mathcal{A}$ computes F only for $O(n)$ selected substrings of w. A processor can read and write only finitely many bits at a step; at each step, each working processor reads $F(u)$ and $F(v)$ for adjacent substrings u and v, then computes and stores $F(uv) = (F(u) + (v))\%2$.

$\mathcal{A}$ is simpler to explain if the length of the input string is always a power of 2. We can insist on that by padding with 0's: for example if the input string is 0110100010 we let $w = 0110100010000000$.

Let $n = |w| = 2^m$. Let $\mathfrak{S}$ be the set of consisting of all subintervals obtained by breaking w into 2^i pieces each of length 2^{m-i}, for all $0 \le i \le m$. Thus $\mathfrak{S}$ consists of all subintervals of length 1, $n/2$ subintervals of length 2, $n/4$ subintervals of length 4, and so forth; these will include 2 subintervals of length $n/2$ and one of length n, namely w itself. The cardinality of $\mathfrak{S}$ is $2n - 1$. Each member of $\mathfrak{S}$ of length 2^i, for $i > 0$, except for the strings of length 1, is the concatenation of two members of $\mathfrak{S}$ of length 2^{i-1}. We let $u_{i,j}$ be the j^{th} member of $\mathfrak{S}$ of length 2^malyi; for any $0 \le i \le m$ and $1 \le j \le 2^{m-i}$. That is $u_{i,j}$ is the substring of w of length 2^i ending at the $(j \cdot 2^i)^{\text{th}}$ place of w. For example, if $w = 1001$,

$u_{0,1} = 1$
$u_{0,2} = 0$
$u_{0,3} = 0$
$u_{0,4} = 1$
$u_{1,1} = 10$
$u_{1,2} = 01$
$u_{2,1} = 1001$

Note that $u_{0,j} = w_j$, the j^{th} symbol of w, while $u_{i,j} = u_{i-1,2j-1}u_{i-1,2j}$ if $i > 0$.

[1]No physical object can be smaller than the Planck length, approximately 1.616×19^{-35}m, and no physical process can have duration less than Planck time, which is approximately 5.39×10^{-44} sec.

Algorithm $\mathcal{A}$

for($1 \le j \le n$) in parallel

$F(u_{0,j}) = w_j$;

for(int i = 1; i $\le$ m; i++) // sequential

for($1 \le j \le n/2^i$) in parallel

$F(u_{i,j}) = (F(u_{i-1,2j-1} + F(u_{i-1,2j})\%2$;

return $F(u_{m,1})$;

<table>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">1</td><td colspan="2">1</td><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">1</td><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">1</td><td colspan="2">0</td><td colspan="2">1</td><td colspan="2">1</td><td colspan="2">0</td><td colspan="2">1</td></tr>
<tr><td colspan="4">0</td><td colspan="4">0</td><td colspan="4">0</td><td colspan="4">1</td><td colspan="4">0</td><td colspan="4">1</td><td colspan="4">0</td><td colspan="4">1</td></tr>
<tr><td colspan="8">0</td><td colspan="8">1</td><td colspan="8">1</td><td colspan="8">1</td></tr>
<tr><td colspan="16">1</td><td colspan="16">0</td></tr>
<tr><td colspan="32">1</td></tr>
</table>

The first line is the string w, which has length $n = 32$.
The next line shows the values of $F(u_{0,j})$ for $1 \le j \le 32$.
The next line shows the values of $F(u_{1,j})$ for $1 \le j \le 16$.
The next line shows the values of $F(u_{2,j})$ for $1 \le j \le 8$.
The next line shows the values of $F(u_{3,j})$ for $1 \le j \le 4$.
The next line shows the values of $F(u_{4,j})$ for $1 \le j \le 2$.
The last line shows $F(u_{5,1}) = 1$, and thus $w \in L$.

$\mathcal{A}$ takes 6 steps and uses 32 processors for this example.