Synchronization of a bounded degree graph of cellular automata with non uniform delays in time $D\lfloor\log_m D\rfloor$

Serge Grigorieff
LIAFA, Université Paris 7 & CNRS
2, pl. Jussieu 75251 Paris Cedex 05 France
seg@liafa.jussieu.fr

January 12, 2006

Contents

Abstract

Jiang, 1989 [2, 3], proved a remarkable result: for every k, there exists a cellular automaton synchronizing every degree $\leq k$ connected graph with arbitrary symmetric communication delays. The synchronization time obtained by Jiang is $O(\Delta^3)$ where Δ is the maximum communication delay between two cells. Mazoyer, 1990 [6] proved an $O(D^2)$ synchronization time where D is the sum of the communication

delays of the degree $\leq k$ connected graph (together with an $O(D\log D)$ synchronization time in case the graph has only two cells).
In this paper, we prove (cf. Theorem 2.13) that, for any $m \geq 2$ one can synchronize in time $D\lfloor \log_m(D)\rfloor$ all lines of total communication delay $> m^9$ (shorter lines being synchronized in time $4D$). A result which extends to bounded degree connected graphs using Rosensthiel's technique [8, 9]. As shown by Vivien, 1994 [12], this result is already optimal for lines of two cells with arbitrary communication delay.
The method relies heavily on Jiang technique of circuit with revolving information.

1 Computations with Jiang circuits

1.1 Lines of ca's with communication delays

Letting n vary in $\mathbb{N}\setminus\{0\}$, we consider a line of n identical cellular automata (in short *ca*) $\mathcal{A} = (Q,\delta)$ where Q is the finite set of states and

$$\delta : (Q \cup \{\$\}) \times Q \times (Q \cup \{\$\}) \to Q$$

is the transition function and \$ denotes the constant virtual state of the virtual cells left to the leftmost cell and right to the rightmost cell.

A state e is said to be quiescent if $\delta(e,e,e) = \delta(\$,e,e) = \delta(e,e,\$)$

We shall suppose that there is a symmetric communication delay between adjacent cells $i, i+1$ where $1 \leq i < n$. This means that there exists strictly positive integers $(d_{i,i+1})_{i=1,\ldots n-1}$ such that the evolution of the line, i.e. its space-time diagram, obeys the rules stated in the following Definition.

Definition 1.1. The *space-time diagram* (in short *STD*) of a line of n copies of automaton $\mathcal{A}$ with strictly positive communication delays $(d_{i,i+1})_{i=1,\ldots n-1}$, in initial configuration $C \in Q^n$, is the function

$$STD_C : \{1,\ldots,n\} \times \mathbb{N} \to Q$$

such that, for every $k \in \{1,\ldots,n\}$ and $t \geq 1$,

$$\begin{array}{rcllll} STD_C(k,0) & = & C(k) & & & \\ STD_C(k,t) & = & \delta(& STD_C(& k-1 & , & t-d_{k-1,k} &), \\ & & & STD_C(& k & , & t-1 &), \\ & & & STD_C(& k+1 & , & t-d_{k,k+1} &) \quad) \end{array}$$

with the following conventions:

for any τ (≥ 0 or < 0)	replace	$STD_C(0,\tau)$, $STD_C(n+1,\tau)$	by \$
for $\tau < 0$	replace	$STD_C(1,\tau),\ldots,STD_C(n,\tau)$	by e

Remark 1.2. We follow the usual conventions for finite lines of n ca's: cells are numbered $1,\ldots,n$, but time steps are counted from 0, i.e. the initial configuration is that at time $t = 0$.

We shall use the following notation.

Notation 1.3. If $1 \leq i < j \leq n$, we denote by $D_{i,j}$ the total communication delay between cells i and j :

$$D_{i,j} = \sum_{i \leq k < j} d_{k,k+1}$$

This is the minimal time necessary for an information to be transmitted from cell i to cell j or from cell j to cell i.
We shall also write D in place of $D_{1,n}$ when n is clear from context.

1.2 Jiang circuits

Let's say that a cell P in a line of cells is rightward passive if it acts as a pure rightward transmission unit: it conveys rightwards the information it receives from its left neighbor without modifying it and with no delay.
Formally, this means that the definition of the transition function δ has to be modified as follows.

Definition 1.4 (Rightwards passive cells). Let Q be a finite set and $\langle Q \rangle$ be a copy of Q. Passive cells take states in $\langle Q \rangle$ and active cells take states in $\langle Q \rangle \times Q$ (*Intuition:* the first component is the information an active cell sends to a passive right neighbor). The transition function δ satisfies the following property when applied to a passive cell: for all $\langle p \rangle, \langle q \rangle \in \langle Q \rangle$, $s \in Q$ and $\xi \in \langle Q \rangle \cup (\langle Q \rangle \times Q)$,

$$\delta(\langle p \rangle, \langle q \rangle, \xi) \quad = \quad \delta((\langle p \rangle, s), \langle q \rangle, \xi) \quad = \quad \langle p \rangle$$

Note 1.5. 1. One can similarly define leftwards passive cells.
2. Observe that rightwards (resp. leftwards) passive cells break the possibility of leftwards (resp. rightwards) communication between two active cells separated by some rightwards (resp. leftwards) passive cells.
This can be avoided with the more complex notion of two-way passive cells which take states in $\langle Q \times Q \rangle$ whereas active cells take states in $\langle Q \times Q \rangle \times Q$ (*Intuition:* the two first components are the informations an active cell sends to a passive right or left neighbor). The transition function δ now satisfies the following property relative to passive cells: for all $\langle u, q \rangle, \langle r, s \rangle, \langle p, t \rangle \in \langle Q \times Q \rangle$ and $v, w \in Q$,

$$\begin{array}{rcllllll}
\langle p, q \rangle & = & \delta(& \langle u, q \rangle & , & \langle r, s \rangle & , & \langle p, t \rangle &) \\
& = & \delta(& (\langle u, q \rangle, v) & , & \langle r, s \rangle & , & \langle p, t \rangle &) \\
& = & \delta(& \langle u, q \rangle & , & \langle r, s \rangle & , & (\langle p, t \rangle, w) &) \\
& = & \delta(& (\langle u, q \rangle, v) & , & \langle r, s \rangle & , & (\langle p, t \rangle, w) &)
\end{array}$$

and the following property relative to active cells: for all $\langle u, q \rangle, \langle p, t \rangle \in \langle Q \times Q \rangle$ and $\alpha, \beta, \gamma \in \langle Q^2 \rangle \times Q$,

- $\delta(\langle u,q\rangle,\beta,\gamma)$ depends solely on q,β,γ, not on u,
- $\delta(\alpha,\beta,\langle p,t\rangle)$ depends solely on α,β,p, not on t,
- $\delta(\langle u,q\rangle,\beta,\langle p,t\rangle)$ depends solely on q,β,p, neither on u nor on t.

Following Jiang [2, 3], we can interpret a communication delay $d \geq 1$ between two cells A,B as $d-1$ fictive two-way passive cells $P_1,...,P_{d-1}$ lying between A and B, the communications delays between adjacent (active or passive) cells being 1 unit of time.
In this way, Jiang associates to lines with communication delays circuits consisting of active and passive cells with no communication delay.
In this paper, it will be sufficient to consider that, in such associated circuits, passive cells are rightwards passive.

Definition 1.6 (Jiang circuits). 1. To a line of n cells $C_1,...,C_n$ with communication delays $d_{1,2}$,...,$d_{n-1,n}$, we associate the closed circuit with $2\,n-2$ cells obtained by doubling cells $C_2,...,C_{n-1}$:

$$\begin{tikzcd}
C_2 \arrow[r, dash] \arrow[d, dash] & C_3 \arrow[r, dash] & \ldots \arrow[r, dash] & C_{n-2} \arrow[r, dash] & C_{n-1} \arrow[d, dash] \\
C_1 \arrow[d, dash] & & & & C_n \arrow[d, dash] \\
C_2 \arrow[r, dash] & C_3 \arrow[r, dash] & \ldots \arrow[r, dash] & C_{n-2} \arrow[r, dash] & C_{n-1}
\end{tikzcd}$$

Applying Jiang's idea of fictive passive cells, we get a circuit of a total of $2D_{1,n}$ cells:

- $2n-2$ ones are active,
- $(d_{1,2}-1)+...+(d_{n-1,n}-1)+(d_{n,n-1}-1)+...+(d_{2,1}-1) = 2D_{1,n}-(2n-2)$ ones are (fictive) rightwards passive associated to the communication delays between the $2n-2$ pairs of adjacent active cells.

2. Letting $C_2,...,C_n,...,C_2$ act as passive cells, we get a variant of the rightwards Jiang circuit which has one active cell C_1 and $2D_{1,n}-1$ rightwards passive cells. We shall call it the rightwards Jiang circuit (or simply Jiang circuit) of the given line with active cell C_1.

The importance of rightwards Jiang circuits comes from the following observation. *A rightwards Jiang circuit with only one active cell, all other passive, can be used as a memory: it is able to convey a "revolving information" (namely a word of length at most that of the circuit) which moves rightwards and may be modified when (and only when) it passes through the active cell.*

1.3 Some computations on a rightwards Jiang circuit with only one active cell

Let's introduce notations for simple arithmetical operations.

Notation 1.7. Let $m, p \in \mathbb{N}$ be such that $2 \leq m$ and $1 \leq p$.

1. We denote $\log_m$ the base m logarithm function.

2. We denote $Base_m(x)$ the m-ary representation of $x \in \mathbb{N}$, i.e. $Base_m(x) = x_k...x_0$ where $x = \sum_{i=0,...,k} x_i m^i$ and $0 \leq x_i < m$, $x_k \neq 0$, $k = \lfloor \log_m x \rfloor$.

3. If u is a word on some alphabet included in $\mathbb{N}$, we denote $SUM(u)$ the sum of the digits of u. For any $p \geq 1$, this map is a retraction of the map

$$Unary_p : \mathbb{N} \to \{\lambda\} \cup \{1, ..., p\}\{p\}^*$$

(where λ denotes the empty word) such that

$$Unary_p(x) = \begin{cases} p...p & \text{(with } z \text{ times } p) \quad \text{if } x = zp \\ rp...p & \text{(with } z \text{ times } p) \quad \text{if } x = zp + r \ \wedge \ 1 \leq r < m \end{cases}$$

In particular, $Unary_1(x)$ is the usual unary representation of x.

Parts of the following proposition appear in Jiang [2, 3].

Proposition 1.8 (Computations on a Jiang circuit). 1. *For any one of the following sets $\mathcal{T}$, there exists a cellular automaton $\mathcal{A}$ such that, for any $N > 0$, in a rightwards Jiang circuit of $\mathcal{A}$ cells consisting of $N - 1$ passive cells and one active cell, the active cell enters a special "bip" state at all times in $\mathcal{T}$:*

i. $\mathcal{T} = \{t \in \mathbb{N} : t \equiv 0 \ mod \ N\}$

ii. $\mathcal{T} = \{t \in \mathbb{N} : t \equiv 0 \ mod \ N\lfloor \log_m N \rfloor\}$ *where $m \geq 2$ is fixed*

2. *Suppose m, p, c, j are fixed in $\mathbb{N}$ such that $2 \leq m$ and $1 \leq p, j$.*
Each line $(u, v, T, \mathcal{C})$ of Table 1 indicates that there exists a cellular automaton $\mathcal{A}$ such that, for any $N > 0$, for any words $u, v \in A^* = \{0, ..., A - 1\}^*$, in a rightwards Jiang circuit of $\mathcal{A}$ cells consisting of $N - 1$ passive cells and one active cell, there exists distinguished states q, r such that, if u, v satisfy conditions $\mathcal{C}$ and $u = x_k...x_0$ and $v = y_\ell...y_0$ then, in T steps, an initial configuration

$$\langle e\rangle^{N-k-1} \ \langle x_k\rangle...\langle x_1\rangle \ (\langle x_0\rangle, q)$$

(where the last cell is the active one) is transformed into the configuration

$$\langle e\rangle^{N-\ell-1} \ \langle y_\ell\rangle...\langle y_1\rangle \ (\langle y_0\rangle, r)$$

Before entering the long proof of Proposition 1.8 in the next subsection, let's state an easy but useful observation (to be used in Remark 2.14).

Proposition 1.9. *Let $f \in \mathbb{N}$ be some fixed constant ≥ 2. A rightwards Jiang circuit with one active cell and $N - 1$ rightwards passive ones can simulate a rightwards Jiang circuit with one active cell and $fN - 1$ rightwards passive ones.*

Proof. Consider f tracks on the cells and distribute a word $a_k...a_0$ with $k < fN$ as follows: for every $i = 1, ..., f$, the subword $a_{iN-1}...a_{(i-1)N}$ is written on track i. $\square$

Table 1

	initial word u	**transformed word** v	**time** T	**condition** $\mathcal{C}$ **on** u, v
i.	$a_k ... a_0$	$b a_k ... a_0$	N	$k + 2 \leq N$
ii.		$a_k ... a_0 b$	N	$k + 2 \leq N$
iii.	$0^\ell a_k ... a_0$	$a_k ... a_0$	$N(1 + \lceil \log_{m^j} N \rceil)$	$\ell + k + 1 \leq N \ \wedge \ (k = 0 \ \vee \ a_k \neq 0)$
iv.		$Base_m(x + y)$	N	$\log_m(x + y) < N$
v.	$Base_m(x)$	$Base_m(cx)$	N	$\log_m(cx) < N$
vi.	$Base_m(y)$	$0^i \ Base_m(x - y)$	N	$y \leq x \ \wedge \ i + \log_m(x - y) = \log_m x < N$
viia.	$Base_m(x_1)$	$Base_m(x - y)$	$N(1 + \lceil \log_{m^j} N \rceil)$	$y \leq x \ \wedge \ \log_m x < N$
viib.	...	$Base_m(x_1 + ... + x_s$	$N(1 + \lceil \log_{m^j} N \rceil)$	$y_1 + ... + y_s \leq x_1 + ... + x_t$
	$Base_m(x_s)$	$-y_1 - ... - y_t)$		$\wedge \ \log_m x_1 + ... + x_t < N$
viii.	$Base_m(y_1)$	$Base_m(xy)$	$N(2 + \lfloor \log_{m^j} N \rfloor)$	$\log_m(xy) < N \ \wedge \ \lfloor \log_m y \rfloor \leq \log_m N$
ix.	...	$Base_m(x \bmod y)$	$N(1 + \lfloor \log_{m^j} N \rfloor)$	$\log_m x < N \ \wedge \ 0 < y \leq x$
x.	$Base_m(y_t)$	$Unary_p(\lfloor x/y \rfloor)$	$N(1 + \lfloor \log_{m^j} N \rfloor)$	$\wedge \ \lfloor x/y \rfloor \leq \min(pN, \log_m N)$
xi.		$Unary_p(\lfloor \log_m x \rfloor)$	$N(1 + \lfloor \log_{m^j} N \rfloor)$	$\log_m x \leq pN$
xii.	empty word	$Unary_p(pN)$	N	
xiii.	$u \in \{0, ..., p\}^N$	$Base_m(SUM_p(u))$	$N(2 + \lfloor \log_{m^j} N \rfloor)$	$\log_m(SUM_p(u)) < N$
xiv.	$Base_m(x)$	$Unary_p(x)$	$N(1 + \lfloor \log_{m^j} N \rfloor)$	$x \leq pN \ \wedge \ \log_m(x) < \lfloor \log_m N \rfloor$
xv.	$Base_m(x)$ $Unary_p(y)$	$Base_m(xy)$	$N \max(1, \lfloor \log_{m^j} N \rfloor)$	$\log_m(xy) \leq N \ \wedge \ y \leq \lfloor \log_m N \rfloor$

(†) In case $N > m^{2j}$ then all times can be replaced by $N \lfloor \log_{m^j} N \rfloor$

1.4 Proof of Proposition 1.8

$\boxed{\text{Conditions } \mathcal{C}.}$ They insure that $|u|, |v| \leq N$ and that time T is enough. In case u or v is of the form $Base_m(z)$, we have $|Base_m(z)| = 1 + \lfloor \log_m(z) \rfloor$, so that $|Base_m(z)| \leq N \Leftrightarrow \lfloor \log_m(z) \rfloor \leq N - 1 \Leftrightarrow \log_m(z) < N$.

$\boxed{\text{1i.}}$ To bip the active cell at times $t \equiv 0 \bmod N$, simply let it send a rightwards signal at time 0. Such a signal will come back on the active cell at times multiples of N.

$\boxed{\text{1ii.}}$ To bip the active cell at time $N\lfloor \log_m N \rfloor$, we use the clock associated to point 1i and proceed in successive phases, each lasting N time steps. In phase 0, the active cell sends a rightwards signal at times $0, 1, ..., N-1$, i.e. it send signals at each time step up to the moment the first sent signal gets back. In phase $i+1$, the active cell kills $m-1$ out of m of the surviving signals, namely it iteratively kills the $m-1$ first encountered and let the m-th alive. It also checks if $< m$ signals are left alive. In case the check is positive, the active cell bips at the end of the phase.
After phase i, there remains only $\lfloor \frac{N}{m^i} \rfloor$ of the N signals created during phase 0. Now,

$$1 \leq \lfloor \frac{N}{m^i} \rfloor < m \Leftrightarrow m^i \leq N < m^{i+1} \Leftrightarrow i = \lfloor \log_m N \rfloor$$

Therefore, the active cell will bip at the end of phase $\lfloor \log_m N \rfloor$, i.e. at time $N\lfloor \log_m N \rfloor$.

$\boxed{\text{2i.}}$ When the active cell receives $\langle e \rangle$ for the first time, it sends b to its right passive neighbor at the next time step.

$\boxed{\text{2ii.}}$ At time 0, the active cell receives a_0 and puts it in its memory and sends b to its right passive neighbor. At time i, for $i = 1, ..., k$, the active cell has a_{i-1} in its memory and receives a_i, it sends a_{i-1} to its right neighbor and puts a_i in its memory. At time $k+1$ it receives nothing and sends to its right neighbor the contents a_k of its memory.

$\boxed{\text{2iii.}}$ The active cell looks for the tail of zeroes as follows. It proceeds through $\lfloor \log_{m^j} N \rfloor$ phases of N times steps (using the clocks of Point 1 with m^j in place of m). In phase 0, it colors the successive blocks of zeroes alternatively in m^{3j} colors $C_0, ..., C_{m^{3j}-1}$ and memorizes the color of the last block and also if there is one or more colored blocks.
If at the end of phase i the color of the last colored block is C_ε then in phase $i+1$, the active cell removes the colors of all blocks of color $\neq C_\varepsilon$ and recolors all successive color C_ε blocks of zeroes alternatively in the m^{3j} colors C_0,...,$C_{m^{3j}-1}$. It also memorizes the color of the last block and also if there is one or more colored blocks.
When there is only one colored block, this block is necessarily the last block of zeroes and, at the next phase, the active cell erases this block.

In order to save one phase, we slightly modify the above process. Instead of memorizing if there is one or more colored blocks, we memorize if there are at most $m^{3j}-1$ colored blocks and if so we memorize the number of colored blocks. Now, when we know there are at most $m^{3j}-1$ colored blocks and we also know their number, at the next phase we can erase the last colored block (and remove colors of the other blocks).
Nothing is done in subsequent phases, up to phase $\lfloor \log_{m^j} N \rfloor$. Their role is merely to have a total time of $N\ \lfloor \log_{m^j} N \rfloor$.
Observe that the set $\{z \in \mathbb{N} : z < \alpha\}$ is equal to $\{0, ..., \alpha - 1\}$ if $\alpha \in \mathbb{N}$ and to $\{0, ..., \lfloor \alpha \rfloor\}$ if $\alpha \notin \mathbb{N}$. Thus, the number of its elements is $\lceil \alpha \rceil - 1$.
Let n_i be the number of colored blocks at the end of phase i. Now, n_{i+1} is the number of elements of the set $\{x : 0 \le x < n_i \ \wedge \ x \equiv \varepsilon \bmod m^{3j}\}$, (for some $\varepsilon < m^{3j}$), a set in bijection with $\{y : ym^{3j} + \varepsilon < n_i\}$. Thus, $n_{i+1} \le \lceil (n_i - \varepsilon)/m^{3j} \rceil - 1 \le \lceil n_i/m^{3j} \rceil$. So that

$$n_i \le \lceil ... \lceil \lceil n_0/m^{3j} \rceil / m^{3j} \rceil ... / m^{3j} \rceil = \lceil n_0/m^{3ij} \rceil$$

(cf. Knuth [4], Ex.35 p.42). Now, n_0 is the number of blocks of zeroes in the word $0^\ell a_k ... a_0$, which is trivially $\le N$. Therefore $n_i \le \lceil N/m^{3ij} \rceil$. Thus, there are $< m^{3j}$ colored blocks at the end of phase i for some i such that $\lceil N/m^{3ij} \rceil < m^{3j}$. Now, $\lceil N/m^{3ij} \rceil < m^{3j} \Leftrightarrow N/m^{3ij} \le m^{3j} - 1 \Leftrightarrow m^{3ij} \ge N/(m^{3j}-1) \Leftrightarrow i \ge \log_{m^{3j}}(N/(m^{3j}-1)$. Since $\log_{m^{3j}}(N/(m^{3j}-1) < \log_{m^{3j}} N$, we see that the least such i is $\le \lceil \log_{m^{3j}} \rceil$. Thus, $1 + \lceil \log_{m^{3j}} N \rceil$ phases are enough (the $+1$ is the extra erasing phase).

$\boxed{\text{2iv, 2v, 2vi.}}$ Straightforward.

$\boxed{\text{2viia.}}$ Apply 2vi and 2iii. To keep time $N(1 + \lceil \log_m N \rceil)$ (rather than $N(2 + \lceil \log_m N \rceil)$), do simultaneously 2vi and the first phase of 2iii.

$\boxed{\text{2viib.}}$ Similar.

$\boxed{\text{2viii.}}$ The product xy is computed via the usual algorithm as a series of products of x by the successive digits $y_0, ..., y_k$ of y and a series of additions of the numbers $m^i(xy_i)$.
This is done through $\lfloor \log_m N \rfloor$ phases of N time steps and uses three tracks. Initially tracks 1 and 2 contain $Base_m(x)$ and $Base_m(y)$ and track 3 contains $Base_m(0)$. Track 2 is never modified. For $i < |Base_m(y)|$, at the end of phase i the contents of track 1 is shifted to the left so that the active cell will simultaneously read the rightmost (least significant) digit of $Base_m(x)$ and the $(i+1)$-th digit of $Base_m(y)$. This $(i+1)$-th digit is memorized by the active cell throughout phase $i+1$. During phase $i+1$ the product of this $(i+1)$-digit with x is computed and added to the contents of track 3. At the end of phase $|Base_m(y)|$, track 3 contains the wanted result $Base_m(xy)$. In all subsequent phases, the active cell does nothing. Since $|Base_m(y)| = 1 + \lfloor \log_m y \rfloor \le 1 + \lfloor \log_m N \rfloor$ (assumed condition),we see that $1 + \lfloor \log_m N \rfloor$

phases are enough.
Finally, group these phases by packs of j (the last pack may contain less than j phases). The number of packs is $\lceil (1+\lfloor \log_m N \rfloor)/j \rceil$ which is

$$\leq 1+\lfloor (1+\lfloor \log_m N \rfloor)/j \rfloor \leq 2+\lfloor \lfloor \log_m N \rfloor /j \rfloor = 2+\lfloor \log_m N/j \rfloor = 2+\lfloor \log_{m^j} N \rfloor$$

$\boxed{\text{2ix, 2x.}}$ We proceed through $\lfloor \log_m N \rfloor$ phases of N time steps and use four tracks. Tracks 1 and 2 contain x and y in base m. At the end of phase i, track 3 contains $Unary_p(i)$ and track 4 contains the base m representation of $x - iy$ with useless zeroes ahead so that it has the length of the base m representation of x. This is done using 2vi as long as $x - iy \geq y$ (which can be checked while writing down $x - iy$), i.e. up to $i = \lfloor x/y \rfloor$. In all subsequent phases, the active cell does nothing.
Clearly, at the end of this process, track 3 contains $Unary_p(\lfloor x/y \rfloor)$ and track 4 contains the base m representation of $x \bmod y$.
Condition $\lfloor x/y \rfloor \leq pN$ insures that there is no overflow. Condition $\lfloor x/y \rfloor \leq \log_m N$ insures that $\lfloor \log_m N \rfloor$ phases are enough.
Finally, grouping these phases by packs of j (as in the proof of 2viii) we get $1 + \lfloor \log_{m^j} N \rfloor$ phases.

$\boxed{\text{2xi.}}$ We proceed through $\lfloor \log_m N \rfloor$ phases of N time steps and use three tracks. Track 1 always contain x in base m. At the end of phase i, track 2 contains $Unary_p(i)$ and track 3 contains $Base_m(m^i) = 10^i$. This is done as long as $m^i \leq x$ (which can be checked while writing down m^{i-1}), i.e. up to $i = \lfloor \log_m(x) \rfloor$. Clearly, at the end of this process, track 2 contains $Unary_p(\lfloor \log_m(x) \rfloor)$.
Transformation $10^i \mapsto 10^{i+1}$ is done using 2ii. The passage from $Unary_p(i)$ to $Unary_p(i+1)$ is either $rp^n \mapsto (r+1)p^n$ with $1 \leq r < p$ or $p^n \mapsto 1p^n$. The first one is straightforward and the second one uses 2i.
Due to the comparison $m^i \leq x$, at least one phase is necessary. Which is no problem due to the below grouping phase.
Again, grouping these phases by packs of j (as in the proof of 2viii) we get $1 + \lfloor \log_{m^j} N \rfloor$ phases.

$\boxed{\text{2xii.}}$ Straightforward.

$\boxed{\text{2xiii.}}$ Let $x = SUM_p(u)$. We proceed through $\lfloor \log_m N \rfloor$ phases of N time steps and use two tracks. At time 0, track 1 contains u and track 2 is empty.
For every $j = 0, ..., \lfloor \log_m x \rfloor - 1$, at the start of phase j, the following properties will be true:

(α) Track 1 contains digits in $\{0, ..., p\}$, the sum of which is $\lfloor x/m^j \rfloor$.

(β) Track 2 contains the suffix of $Base_m(x)$ with length j, i.e. digits corresponding to $m^0, ..., m^{j-1}$.

(γ) If $j \geq 1$ then the active cell has memorized the digit of $Base_m(x)$ corresponding to m^j.

During phase j, as long as the sum of digits on track 1 is not zero, the active cell does the following:

- If $j \geq 1$, using 2i, it prefixes its memorized digit (cf. (γ)) to the contents of track 2, so that this contents becomes the suffix of $Base_m(x)$ with length $j+1$. This insures the inductive step for (β).
- At any step of the phase it memorizes a current carry, always $< m$, and initially 0. It computes the sum S' of this current carry S with the digit it receives from its left passive neighbor on track 1. Clearly, $S' \leq S + p < m + p$.
  If $S' < m$ then it sends 0 on track 1 to its right passive neighbor (in other words, it erases the received digit) and memorizes S' as the new carry.
  Suppose $S' \geq m$, so that $1 \leq \lfloor S'/m \rfloor < 1 + \lfloor p/m \rfloor \leq p$. Then the active cell sends $\lfloor S'/m \rfloor$ on track 1 to its right passive neighbor and memorizes S' mod m as the new carry.

At the end of phase j, the sum of digits on track 1 has been divided by m. Using (α), we see that this sum is now $\lfloor \lfloor x/m^j \rfloor / m \rfloor = \lfloor x/m^{j+1} \rfloor$ (cf. Knuth [4], Ex.35 p.42). This insures the inductive step for (α).
Also, the current carry at the end of phase j is the associated remainder $\lfloor x/m^j \rfloor$ mod m, i.e. the digit of $Base_m(x)$ corresponding to m^{j+1}. This insures the inductive step for (γ).
When the sum of digits on track 1 is zero, i.e. for phases $j > 1 + \lfloor \log_m x \rfloor$, the active cell does nothing. Clearly, this process computes $Base_m(x)$ on track 2 and needs $1 + \lfloor \log_m x \rfloor \leq 1 + \lfloor \log_m (pN) \rfloor$ phases.
Finally, grouping phases by packs of j, we get $2 + \lfloor \log_m x \rfloor$ phases.

$\boxed{\text{2xiv.}}$ Let $1', 1'', ... p', p''$ be new symbols. Again, we proceed through $\lfloor \log_m N \rfloor$ phases of N time steps and use two tracks.
Initially, $Base_m(x) = x_k ... x_0$ is written on track 1 and track 2 is empty.
At the end of phase $i \leq k$ the following properties will be true:

(α) Digits $x_i, ..., x_0$ of $Base_m(x)$ are erased from track 1.

(β) The contents of track 2 is the word

$$Unary_p(x_0 + mx_1 + ... + m^i x_i)\ u$$

where u is a sequence of words $Unary_p(m^{i+1})$, alternatively written with digits $\{1', ..., p'\}$ and digits $\{1", ..., p"\}$, ended by a word also written with the primed or double primed digits such that the sum of its digits is $< m^i$.

To get these properties, the active cell acts as follows:

1. During phase 0, it deletes x_0 from track 1. Then it writes $Unary_p(x_0)$ (which is empty if $x_0 = 0$) on track 2. Left to that, it fills track 2 with a sequence of words $Unary_p(m)$ alternatively written with digits $\{1', ..., p'\}$ and digits $\{1", ..., p"\}$, ended by a prefix of $Unary_p(m)$ also written in the primed or double primed alphabet.

2. During phase $i < k$, it deletes x_i from track 1. It also considers on track 2 the word $Unary_p(x_0 + mx_1 + ... + m^{i-1}x_{i-1})$ followed by the x_i first words $Unary_p(m^i)$ alternately written with primed and double primed digits. Using an add and carry process, it computes their sum and writes down $Unary_p(x_0 + mx_1 + ... + m^i x_i)$.

3. To end phase i, the active cell considers successive blocks of m words $Unary_p(m^i)$ alternatively with primed and double primed digits. It computes the sum of each such block and writes down $Unary_p(m^{i+1})$. This last word is written down alternatively with primed and double primed digits.

4. During phase k (which the active cell recognizes since at the end of the previous phase, only one digit remains on track 1), the active cell follows point 2 but not point 3 and erases the prime and double prime digits which are now useless.
   In all subsequent phases, the active cell does nothing.

Clearly, this process puts $Unary_p(x)$ on track 2.
Condition $\log_m x \leq \lfloor \log_m N \rfloor$ insures that $\lfloor \log_m N \rfloor$ phases are enough.
Finally, grouping phases by packs of j, we get $1 + \lfloor \log_{m^j} x \rfloor$ phases.

$\boxed{\text{2xv.}}$ Interpret xy as the sum of y terms: $x + ... + x$ and apply 2iv. Since there are $y - 1$ additions, $\max(1, \lfloor \log_m y \rfloor - 1)$ phases are enough.
Finally, grouping phases by packs of j, we get $\max(1, \lfloor \log_{m^j} x \rfloor)$ phases.

$\boxed{\text{Case } N \geq m^{2j}.}$ In all points i to xv replace m^j by m^{3j}. Observe that $\lceil \log_{m^{3j}} N \rceil + 1 \leq \lfloor \log_{m^{3j}} N \rfloor + 2 = \lfloor (\log_{m^j} N)/3 \rfloor + 2 = \lfloor \lfloor \log_{m^j} N \rfloor /3 \rfloor + 2$
Now, $\lfloor x/3 \rfloor + 2 \leq x \;\Leftrightarrow\; x/3 < x - 1 \;\Leftrightarrow\; x > 3/2$. Thus, for $\lfloor \log_{m^j} N \rfloor \geq 2$, i.e. for $N \geq m^{2j}$, a total of $\lfloor \log_{m^j} N \rfloor$ phases is enough. $\square$

Remark 1.10. Even if $N < m^{2j}$, all times in Table 1 can also be replaced by $N \lfloor \log_{m^j} N \rfloor$ provided that the active cell knows (i.e. has coded in its state) the values modulo m^2 of the inputs (in case of inputs given in base m, this is the two rightmost digits). Indeed, it is easy to check that this allows to collapse two phases in each of the operations of Table 1.

2 Synchronization of a line in time $D\lfloor\log_m D\rfloor$

2.1 Strategy for synchronization

1. *The dichotomy process (§2.2): creating sublines L_ℓ and L_r (2.3).*
We adapt Minsky dichotomy process ([7], p.28-29 & 282-283) to split a line $L = \{1, ..., n\}$ with total communication delay D into two (left and right) sublines $L_\ell = \{1, ..., h\}$ and $L_r = \{h+1, ..., n\}$ with total communication delays $D_\ell = D_{1,h}$ and $D_r = D_{h+1,n}$. In general, due to the delays, there is no middle of the line, so that it is not possible to create sublines with equal total delays. We merely do for the best with L_ℓ and L_r by choosing h maximum such that $D_{1,h} \leq D_{h,n}$. Then an easy computation (cf. Proposition 2.1) shows that

$$(*) \qquad D_\ell < D/2 \leq D_\ell + d \quad , \quad D_r \leq D/2 < d + D_r$$

where d denotes the communication delay $d_{h,h+1}$ from cell h to $h+1$.

2. *Idle times and extended sublines (§2.5).*
Since the dichotomy process does not create sublines L_ℓ and L_r with equal delay, the sublines have to start their synchronization process after adequate idle times T_ℓ and T_r (cf. §2.8).
Now, sublines L_ℓ and L_r can be arbitrarily short: it may happen that most of the total delay lies between cells h and $h+1$. However, as shown by inequalities $(*)$ above, the extended sublines $L_\ell^+ = \{1, ..., h, h+1\}$ and $L_r^+ = \{h, h+1, ..., n\}$ have total delays at least $D/2$. Therefore, *we shall create Jiang circuits $\mathcal{J}(L_\ell^+)$ and $\mathcal{J}(L_r^+)$ on these extended sublines and use them to compute, code and consume the idle times* (cf. §2.4).
Important point: observe that this extension process does not give rise to an unbounded accumulation of signals for cells h and $h+1$ because none of these cells will be involved as the extra cell of any subsequent extended subline.

3. *Getting the values of D_ℓ, $D_r + d$ (resp. D_ℓ, $D_r + d$ and $2d$) on the rightwards Jiang circuit $\mathcal{J}(L_\ell^+)$ (resp. $\mathcal{J}(L_r^+)$), cf. §2.5.*
These are the values we shall need in order to compute the adequate idle times T_ℓ and T_r on circuits $\mathcal{J}(L_\ell^+)$ and $\mathcal{J}(L_r^+)$ using operations from Table 1 of Proposition 1.8. It will be sufficient to represent these parameters as unary words. inequalities $(*)$ above show that these unary words can be coded on at most two tracks of $\mathcal{J}(L_\ell^+)$ and $\mathcal{J}(L_r^+)$, i.e. such words can be longer that the circuit but shorter than twice the circuit. This is detailed in §2.5.

4. *Computing idle times on extended sublines (§2.8).*
We shall compute the idle times T_ℓ and T_r on the extended sublines $\{1, ..., h, h+1\}$ and $\{h, h+1, ..., n\}$. This is possible since

- Each of these two extended sublines has total communication delay $\geq D/2$, so that computations involving $D \log_m D$ are possible on the associated rightwards Jiang circuit (using binary representations).
- Point 2 insures that all needed information has been put on each of these extended sublines.

Of course, the time needed to compute the idle time has to be considered in the evaluation of the idle time.

5. *Consuming the idle times T_ℓ and T_r.*
The obvious way *to consume* an idle time (in order to delay some process), needs this idle time to be represented in unary. This will not be possible here because the idle times T_ℓ and T_r are of order $D \log_m D$ whereas the total communication delays of the extended sublines are of order D.
However, we shall use another trick which is based on the fact that the idle time T_ℓ (resp. T_r) will always satisfy $T_\ell < D \log_m D \ \leq 2(D_\ell + d) \log(2(D_\ell + d)) \leq (2(D_\ell + d))^2$ (resp. $T_r < (2(D_r + d))^2$).
In fact, given $T < N^2$ (with $N \geq 2$), let's write it as $T = \alpha N + \beta$ where $\alpha, \beta < N$. On a Jiang circuit with total delay N, we represent the idle time T by the unary representations of α, β written on two tracks of the circuit, with their last letter on the active cell (this is a unary representation of the two digits of $Base_N(T)$). Such a representation allows to consume the idle time T quite easily:

- If $\alpha \neq 0$ (i.e. if the active cell reads a digit 1 on the track containing α) then the active cell starts counting α many cycles (cf. point 1i of Proposition 1.8). At each cycle, the active cell erases the nearest remaining digit of $Unary(\alpha)$. The process stops when all digits of $Unary(\alpha)$ have been erased. This process consumes time αN.
- Then the active cell starts a new cycle reading $Unary(\beta)$ but halts as soon as this reading is over. This consumes time β.

6. *Short lines (§2.7).*
For short lines there is not enough time to compute the idle times as described in the above points. For such short lines, we shall halt the dichotomy process, compute another idle time and synchronize using a bounded counter.
Observe that we need to test whether the subline is short or not. This is time consuming and to be taken into consideration for the computation of the idle time.

2.2 The dichotomy process

The dichotomy process does not use Jiang circuits but classical signals as is the case for most synchronization solutions (cf. Mazoyer [5]). As in Mazoyer [6], we use Minsky's dichotomy process set up by the following three signals:

1. Signal $\mathcal{S}$ is launched by cell 1 (the general) at time 0 and goes rightwards at maximum speed. Thus, $\mathcal{S}$ reaches cell i at time $D_{1,i}$.
2. Signal $\widetilde{\mathcal{S}}$ is launched by cell n when receiving signal $\mathcal{S}$, i.e at time $D_{1,n}$. This signal goes leftwards at maximum speed. Thus, $\widetilde{\mathcal{S}}$ reaches cell i at time $D_{1,n} + D_{i,n}$.
3. Signal $\mathcal{Z}$ is launched by cell 1 at time 0 and goes rightwards through zigzags in the following way:

   $$1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow ... \rightarrow n-1 \rightarrow n \rightarrow n-1 \rightarrow n$$

   Let's say that $\mathcal{Z}$ reaches cell i when it does it for the second time, i.e. at time 3 $D_{1,i}$.

The wanted synchronization is obtained by iteratively splitting the line. The line of n cells $L = \{1, \ldots, n\}$ is split in two sublines:

$$\begin{array}{rcl} L_\ell & = & \{i : \widetilde{\mathcal{S}} \text{ reaches } i \text{ strictly later than } \mathcal{Z} \text{ does}\} \\ L_r & = & \{i : \widetilde{\mathcal{S}} \text{ reaches } i \text{ no later than } \mathcal{Z} \text{ does}\} \end{array}$$

Proposition 2.1. *1. Let $n \geq 2$. There exists h such that $1 \leq h < n$ and*

$$\begin{array}{rclcl} L_\ell & = & \{i : \widetilde{\mathcal{S}} \textit{ reaches } i \textit{ strictly later than } \mathcal{Z} \textit{ does}\} & = & \{1, \ldots, h\} \\ L_r & = & \{i : \widetilde{\mathcal{S}} \textit{ reaches } i \textit{ no later than } \mathcal{Z} \textit{ does}\} & = & \{h+1, \ldots, n\} \end{array}$$

2. The total communication delays $D_\ell = D_{1,h}$ and $D_r = D_{h+1,n}$ of L_ℓ and L_r satisfy

$$\begin{array}{rccccc} i. & D_\ell & < & D/2 & \leq & D_\ell + d \\ ii. & D_r & \leq & D/2 & < & d + D_r \end{array}$$

where d denotes the communication delay $d_{h,h+1}$ from cell h to $h+1$.

Proof. 1. Signals $\mathcal{Z}$ and $\widetilde{\mathcal{S}}$ reach cell i at respective times $3D_{1,i}$ and $D_{1,n} + D_{i,n}$. Thus,

$$\begin{array}{rcl} i \in L_\ell & \Leftrightarrow & 3D_{1,i} < D_{1,n} + D_{i,n} \\ & \Leftrightarrow & 3D_{1,i} < D_{1,i} + 2\ D_{i,n} \\ & \Leftrightarrow & D_{1,i} < D_{i,n} \\ i \in L_r & \Leftrightarrow & D_{1,i} \geq D_{i,n} \end{array}$$

Since $n > 1$ we have $D_{1,n} > 0$ (in fact $D_{1,n} = d_{1,2} + ... + d_{n-1,n} \geq n - 1$ since the $d_{i,i+1}$'s are ≥ 1), so that $0 = D_{1,1} < D_{1,n}$ and $D_{1,n} > D_{n,n} = 0$. Thus, $1 \in L_\ell$ and $n \in L_r$, which insures that the two sublines L_ℓ and L_r are non empty.
Clearly, L_ℓ and L_r are disjoint and respectively initial and final segments of

$\{1, ..., n\}$. Therefore there exists h satisfying point 1 of the Proposition.

2. Applying the above inequalities to $i = h$ and $i = h + 1$, we get

$$D_\ell < d + D_r \qquad\qquad D_\ell + d \geq D_r \tag{1}$$

Adding respectively D_ℓ and D_r to inequalities (1) and observing that $D = D_\ell + d + D_r$, we get

$$2D_\ell < D \ \ , \ \ D \geq 2D_r$$

Adding respectively $d + D_r$ and $D_\ell + d$ to inequalities (1), we get

$$D < 2(d + D_r) \ \ , \ \ 2(D_\ell + d) \geq D$$

which proves the inequalities of point 2 of the Proposition. □

2.3 Creating sublines $L_\ell = \{1, ...h\}$ and $L_r = \{h + 1, ...n\}$

Proposition 2.2. *One can distinguish cell h in time $D + D_r + d$ and cell $h + 1$ in both times $D + D_r + 2d$ and $\max(D + D_r, 3D_\ell + d)$.*

Note 2.3. Formally, the above Proposition means that there is an automaton $\mathcal{A}$ with distinguished states G, G_ℓ, G_r such that, given a line with total communication delay D, for every $n \geq 2$, a line of n cells (each one with automaton $\mathcal{A}$) in initial configuration Ge^{n-1}, evolves so that at the indicated time cell h (resp. $h + 1$) enters state G_ℓ (resp. G_r) for the first time.

Proof. 1. *Distinguishing cell h at time $D + D_r + d$.* When cell i receives the $\widetilde{\mathcal{S}}$ signal, i.e. at time $D + D_{i,n}$, it can decide whether it belongs to the subline L_r or to L_ℓ : if it has strictly previously received the $\mathcal{Z}$ signal then it belongs to L_ℓ, else it belongs to L_r.
Let cell i send a message to cell $i - 1$ at time $D + D_{i,n}$ telling whether it belongs to L_r or not.
Now, cell h is the sole one which belongs to L_ℓ with its right neighbor in L_r. Thus, cell h "understands" its status at time $D + D_{n,h} = D + D_r + d$.

2. *Distinguishing cell $h + 1$ at time $D + D_r + 2d$.* When cell h has been distinguished, it sends a special message to its right neighbor $h + 1$. Thus, cell $h + 1$ is distinguished when it receives this special message, i.e. at time $D + D_r + 2d$.

3. *Distinguishing cell $h + 1$ at time* $\max(D + D_r, 3D_\ell + d)$. When cell i receives the $\mathcal{Z}$ signal, i.e. at time $3D_{1,i}$, it can decide whether it belongs to the subline L_ℓ or to L_r : if it has not yet received the $\widetilde{\mathcal{S}}$ signal then it belongs to L_ℓ, else it belongs to L_r. Let then cell i send a message to cell $i + 1$ telling whether it belongs to L_ℓ or not.
Now, cell $h + 1$ is the sole one which belongs to L_r with its left neighbor in L_ℓ. The first information is obtained at time $D + D_r$ and the second one at time $3D_\ell + d$. Thus, at time $\max(D + D_r, 3D_\ell + d)$, cell $h + 1$ "understands" its status. □

Time $\max(D + D_r, 3D_\ell + d)$ is better than $D + D_r + 2d$.

Proposition 2.4. $\max(D + D_r, 3D_\ell + d) \leq D + D_\ell + d$

Proof. Since $2D_\ell < D$, we get $3D_\ell + d < D + D_\ell + d$. Also, $D_r \leq D/2 \leq D_\ell + d$ hence $D + D_r \leq D + D_\ell + d$. $\square$

2.4 Creating Jiang circuits $\mathcal{J}(L_\ell^+)$ and $\mathcal{J}(L_r^+)$

Following Definition 1.6, we consider the following rightwards Jiang circuits associated to the line $L = \{1, ..., n\}$ with total communication delay D.

Definition 2.5. 1. $\mathcal{J}(L_\ell^+)$ is the rightwards Jiang circuit associated to the extended subline $\{1, ..., h, h + 1\}$ which has active cell h and $2(D_\ell + d) - 1$ rightwards passive cells. (of which h are real cells, the other ones being fictive).

2. $\mathcal{J}(L_r^+)$ is the rightwards Jiang circuit associated to the extended subline $\{h, h+1, ..., n\}$ and has active cell $h+1$ and $2(d+D_r)-1$ rightwards passive cells. (of which $n - h$ are real cells, the other ones being fictive).

The Jiang circuit $\mathcal{J}(L_\ell^+)$ (resp. $\mathcal{J}(L_\ell^+)$) is created as soon as cell h (resp. $h + 1$) is distinguished.
Though, as a cell in $\mathcal{J}(L_\ell^+)$, cell $h+1$ is a passive one, it is used as a normal (active) cell of line $\{1, ..., n\}$ to manage the Jiang circuit $\mathcal{J}(L_\ell^+)$ as is done in §2.5 below. Idem for cell h and circuit $\mathcal{J}(L_\ell^+)$.

2.5 Getting D_ℓ, D_r, d on circuits $\mathcal{J}({L_\ell}^+)$ and $\mathcal{J}({L_r}^+)$

Proposition 2.6. *Let x be D_ℓ (resp. D_r) or d. After cell h (resp. $h+1$) is distinguished, it takes $2x$ units of time to get the unary representation of $2x$ on both Jiang circuits $\mathcal{J}({L_\ell}^+)$ and $\mathcal{J}({L_r}^+)$. Such a representation is always shorter than twice the length of the circuit, hence can be coded on at most two tracks (if and when the first track is filled then the second track receives the remaining part). Thus,*

i. After cell h is distinguished, it takes $\max(2D_\ell, 2d, 2D_r + d)$ *units of time to put the unary representations of $2D_\ell$, $2D_r$ and $2d$ on circuit $\mathcal{J}({L_\ell}^+)$.*

ii. After cell $h + 1$ is distinguished, it takes $\max(2D_r, 2d, 2D_\ell + d)$ *units of time to put the unary representations of $2D_\ell$, $2D_r$ and $2d$ on circuit $\mathcal{J}({L_r}^+)$.*

Proof. 1. The assertion about the length of such representations compared to that of the circuit is an obvious corollary of inequalities of Proposition 2.1.

2. After cell h is distinguished, let it send a signal to cell 1 and another to cell $h+1$ and wait for their returns. While waiting, it writes the unary representations of $2D_\ell$ and $2d$ on (one or two tracks of) circuits $\mathcal{J}(L_\ell{}^+)$ and $\mathcal{J}(L_r^+)$.

3. After cell $h+1$ is distinguished, let it send a signal to cell n and another to cell h and wait for their returns. While waiting, it writes the unary representations of $2D_r$ and $2d$ on (one or two tracks of) circuits $\mathcal{J}(L_\ell{}^+)$ and $\mathcal{J}(L_r^+)$.

4. Condition i. Use 1 to get $2D_\ell$ and $2d$. To get $2D_r$ use 2 and observe that cell h can be distinguished d units of time after cell $h+1$. Idem with condition ii. □

2.6 Testing whether D_ℓ and D_r are short

As we shall treat in a special way short lines (or short sublines obtained through the recursive splitting), we have to test shortness.

Proposition 2.7. *Let $\xi > 0$ be fixed. Cell h (resp. $h+1$) can test in time $2D_\ell$ (resp. $2D_r$) whether $D_\ell \leq \xi$ or not (resp. $D_r \leq \xi$) or not.*

Proof. Simply send a signal from cell h to cell 1 which goes back to cell h at time $2D_\ell$. A counter up to 2ξ allows to compare D_ℓ and ξ. Idem with cell $h+1$ and D_r. □

2.7 Synchronization of short lines using counters

Short lines are those for which D is less than or equal to some fixed bound. They appear through the dichotomic splitting of the initial line.

Proposition 2.8. *Let $\xi > 0$ be fixed. All lines with total communication delay $D \leq \xi$ can be synchronized in time $4D$.*

Proof. We use counters up to $3D$.
1. *Cell i can know the communication delay $D_{1,i}$ at time $3D_{1,i}$.*
Induction on i. The initial case $i = 1$ is trivial.
Inductive step: from i to $i+1$. When i knows $D_{1,i}$ at time $3D_{1,i}$, it looks for the value of $d_{i,i+1}$. A signal to cell $i+1$ comes back to cell i at time $3D_{1,i} + 2d_{i,i+1}$, bringing to cell i the value of $d_{i,i+1}$. Then cell i sends to cell $i+1$ the value of $D_{1,i} + d_{i,i+1} = D_{1,i+1}$. Cell $i+1$ receives this value at time $(3D_{1,i} + 2d_{i,i+1}) + d_{i,i+1} = 3D_{1,i+1}$.

2. *Synchronization in time $4D$.*
When the last cell n receives the value of $D_{1,n} = D$, i.e. at time $3D$, it sends a leftwards signal. This signal is received by cell i at time $3D + D_{i,n}$. Then cell i counts up to $D_{1,i}$ (a value cell i knows since time $3D_{1,i}$) and fires. The firing time is $(3D + D_{i,n}) + D_{1,i} = 4D$. □

For the computation of idle times of short sublines, we shall need the following obvious analog of Proposition 2.6.

Proposition 2.9. *Let $\xi > 0$ be fixed. Let x be D_ℓ or d (resp. D_r or d). For all lines with total communication delay $D \leq \xi$, after cell h (resp. $h+1$) is distinguished, it takes $2x$ units of time to cell h (resp. $h+1$) to code the value of x in its state. Thus,*

i. After cell h is distinguished, it takes $\max(2D_\ell, 2d, 2D_r + d)$ units of time to code D_ℓ, D_r and d within the state of cell h.

ii. After cell $h+1$ is distinguished, it takes $\max(2D_r, 2d, 2D_\ell + d)$ units of time to code D_ℓ, D_r and d within the state of cell $h+1$.

Proof. Same proof as Proposition 2.6 : instead of writing on a Jiang circuit, let's cell h (resp. $h+1$) use a counter up to ξ. □

2.8 Idle times for synchronization

Let ω be some constant to be fixed later on (its actual value will be 9, cf. §2.9). In order to synchronize a line with total communication delay D in time

$$T(D) = \begin{cases} D\lfloor \log_m D \rfloor & \text{if } D > m^\omega \\ 4D & \text{if } D \leq m^\omega \end{cases}$$

we manage the splitting process as follows:

- If the initial line has total communication delay $D \leq m^\omega$ then there is no splitting process: the line is synchronized via counters (cf. Proposition 2.8).

- If the initial line has total communication delay $D > m^\omega$ then there is a recursive splitting process. This splitting process stops for sublines with total communication delay $\leq m^\omega$. Such sublines are not split and are synchronized via counters (cf. Proposition 2.8).

- Whatever be its length, the synchronization of a subline starts after an adequate idle time. As said in §2.1, point 3, the idle time for subline $L_\ell = \{1, \ldots, h\}$ will be computed on the rightwards Jiang circuit $\mathcal{J}(L_\ell^+)$ and that for $L_r = \{h+1, \ldots, n\}$ on $\mathcal{J}(L_r^+)$.

Let's look at the different components which have to enter in the idle time for L_ℓ (resp. L_r) when we split the line L.

C1. *First, there is the gross difference $T(D) - T(D_\ell)$ (resp. $T(D) - T(D_r)$).* Observe that $T(D) = D\lfloor \log_m D \rfloor$ since we split. However, $T(D_\ell)$ (resp. $T(D_r)$) depends on how D_ℓ (resp. D_r) compares to m^ω.
This gross difference has to be diminished of the following quantities.

C2. *The time needed to distinguish cell h (resp. $h+1$).* Applying Proposition 2.2, this time is $D + D_r + d$ (resp. $D + D_r + 2d$ or $\max(D + D_r, 3D_\ell + d)$).

C3. *The time needed to test whether $D_\ell > m^\omega$ (resp. $D_r > m^\omega$) or not and whether $D_\ell + d > m^{2j}/2$ or not (resp. $D_r + d > m^{2j}/2$).* The first test is needed for the value of $T(D_\ell)$ (resp. T_r) : if true then $T(D_\ell) = D_\ell \lfloor \log_m D_\ell \rfloor$ else $T(D_\ell) = 4D_\ell$. The second test is needed in point C5 below. Proposition 2.7 insures that these tests can be done in time $2(D_\ell + d)$ (resp. $2(D_r + d)$).

C4. *The time needed to get $2D_\ell, 2D_r, 2d$ on the Jiang circuit $\mathcal{J}(L_\ell^+)$ (resp. $\mathcal{J}(L_r^+)$) or to code them within the state of cell h (resp. $h+1$) in case $2(D_\ell + d) \le m^{2j}$ (resp. $2(D_r + d) \le m^{2j}$).* Applying Propositions 2.6 and 2.9, this time is that to distinguish cell h (resp. $h+1$) augmented of these values.
This can be done in parallel with point C3. The total time of C3 and C4 is therefore $\max(2(D_\ell + d), 2D_r)$ (resp. $\max(2(D_r + d), 2D_\ell)$).

C5. *The time needed for the arithmetic computations in order to compute the idle time.* Here there is a "fixed point problem" since the computation time of the idle time is itself a component of the idle time (entering in subtraction of the gross difference). As we shall use Proposition 1.8, in particular property (†) from Table 1, we shall fix some constant j, the exact value of which will be determined later on (it will be 67, cf. Proposition 2.12). This is to simplify the expression of the computation times of the diverse operations in case $2(D_\ell + d) > m^{2j}$ (resp. $2(D_r + d) > m^{2j}$). When $2(D_\ell + d) \le m^{2j}$ (resp. $2(D_r + d) \le m^{2j}$), all arithmetic computations can be done in one step since all parameters are coded in the state of cell h (resp. $h+1$).

We can now get the expressions of the idle times $idle_\ell$ and $idle_r$.

Proposition 2.10. *Let j, ω be some fixed constants.*

1. If $D_\ell \le m^\omega$ (i.e. L_ℓ is short) then

$$idle_\ell = D\lfloor \log_m D \rfloor - 4D_\ell - (D + D_r + d) - \max(2(D_\ell + d), 2D_r)$$

2. If $m^\omega < D_\ell$ and $2(D_\ell + d) \le m^{2j}$, then

$$\begin{aligned} idle_\ell = {} & D\lfloor \log_m D \rfloor - D_\ell \lfloor \log_m D_\ell \rfloor - (D + D_r + d) \\ & - \max(2(D_\ell + d), 2D_r) \end{aligned}$$

3. If $m^\omega < D_\ell$ and $2(D_\ell + d) > m^{2j}$ then

$$\begin{aligned} idle_\ell = {} & D\lfloor \log_m D \rfloor - D_\ell \lfloor \log_m D_\ell \rfloor - (D + D_r + d) \\ & - \max(2(D_\ell + d), 2D_r) - 6(D_\ell + d)\lfloor \log_{m^j} 2(D_\ell + d) \rfloor \end{aligned}$$

4. If $D_r \le m^\omega$ (i.e. L_r is short) then

$$\begin{aligned} idle_r &= D\lfloor \log_m D \rfloor - 4D_r - \max(D + D_r, 3D_\ell + d) \\ &\quad - \max(2(D_r + d), 2D_\ell) \end{aligned}$$

5. *If $m^\omega < D_r$ and $2(D_r + d) \leq m^{2j}$, then*

$$\begin{aligned} idle_r &= D\lfloor \log_m D \rfloor - D_r\lfloor \log_m D_r \rfloor \\ &\quad - \max(D + D_r, 3D_\ell + d) - \max(2(D_r + d), 2D_\ell) \end{aligned}$$

6. *If $m^\omega < D_r$ and $2(D_r + d) > m^{2j}$ then*

$$\begin{aligned} idle_r &= D\lfloor \log_m D \rfloor - D_r\lfloor \log_m D_r \rfloor - \max(D + D_r, 3D_\ell + d) \\ &\quad - \max(2(D_r + d), 2D_\ell) - 6(D_r + d)\lfloor \log_{m^j} 2(D_r + d) \rfloor \end{aligned}$$

Proof. 1 & 2 & 4 & 5. The computation time of the idle time (point C5 in the above analysis) is null. In fact, the state of cell h (resp. $h+1$) codes all parameters entering the idle time so that $idle_\ell$ (resp. $idle_r$) is known from the state of cell h (resp. $h+1$). This uses the ability of an automaton to compute from parameters bounded by some fixed constant. The value of $idle_\ell$ (resp. $idle_r$) is thus given by points C1 to C4 in the above analysis.

3 & 6. As explained in §2.1, points 3 and 4, the idle time $idle_\ell$ (resp. $idle_r$) is to be computed on the Jiang circuit $\mathcal{J}(L_\ell^+)$ (resp. $\mathcal{J}(L_r^+)$) and is to be given as two unary representations of the digits of its representation in base the length of $\mathcal{J}(L_\ell^+)$ (resp. $\mathcal{J}(L_r^+)$), i.e. base $2(D_l + d)$ (resp. $2(D_r + d)$). The computation starts when all parameters are written on $\mathcal{J}(L_\ell^+)$ (resp. $\mathcal{J}(L_r^+)$), i.e. after time for C2, C3 and C4 in the above analysis. It is done as follows:

i. Compute in parallel

   (a) the base m representations of D, D_ℓ, D_r, d and $2(D_\ell + d)$ (resp. $2(D_r + d)$),

   (b) the unary representations of $\lfloor \log_m D \rfloor$ and $\lfloor \log_m D_\ell \rfloor$ (resp. $\lfloor \log_m D_r \rfloor$),

   (c) the unary representations of $\lfloor \log_{m^j}(D_\ell + d) \rfloor$ (resp. $\lfloor \log_{m^j}(D_r + d) \rfloor$).

ii. While achieving i(a) (i.e. as digits of increasing significance are obtained), determine (in parallel) the largest elements out of the sets $\{D + D_r, 3D_\ell + d\}$, $\{2D_\ell, 2d, 2D_r + d\}$ and $\{2D_r, 2d, 2D_\ell + d\}$.

iii. Compute in parallel the base m representations of the products
   • $D\lfloor \log_m D \rfloor$ and $D_\ell\lfloor \log_m D_\ell \rfloor$ (resp. $D_r\lfloor \log_m D_r \rfloor$),
   • $(D_\ell + d)\lfloor \log_{m^j}(D_\ell + d) \rfloor$ (resp. $(D_r + d)\lfloor \log_{m^j}(D_r + d) \rfloor$).

iv. While achieving iii (i.e. as digits of increasing significance are obtained), compute the linear combination leading to $idle_\ell$ (resp. $idle_r$) as seen from Proposition 2.10.

v. Lastly, compute the unary representations of $idle_\ell \bmod 2(D_\ell+d)$ (resp. $idle_r \bmod 2(D_r+d)$) and $\lfloor idle_\ell/2(D_\ell+d)\rfloor$ (resp. $\lfloor idle_r/2(D_r+d)\rfloor$). Observe that $idle_\ell \leq D\log_m D \leq 2(D_\ell+d)\log_m 2(D_\ell+d)$. Thus, the unary representation of $\lfloor idle_\ell/2(D_\ell+d)\rfloor$ has length $\log_m 2(D_\ell+d) \leq 2(D_\ell+d)$ hence can be written on circuit $\mathcal{J}_\ell^+$. Idem for $idle_r$ and circuit $\mathcal{J}_r^+$.

Computations i, iii and v above are successively done via respective operations 2xiii+2xi, 2xv and 2ix+2x of Table 1 (cf. Proposition 1.8). Condition $2(D_\ell+d) > m^{2j}$ (resp. $2(D_r+d) > m^{2j}$)) insure that property (†) of Table 1 can be applied. Thus, the total computation time for i, iii and v above is $6(D_\ell+d)$ (resp. $6(D_r+d)$). □

Remark 2.11. Observe that when $2D_\ell, 2D_r, 2d$ are put in unary on a Jiang circuit, the active cell can memorize the values of $D_\ell \bmod m^2$, $D_r \bmod m^2$, $d \bmod m^2$. Also, at the end of the computation of points ii and iii of the above proof, the active cell can memorize the values modulo m^2 of the results. This allows to use Remark 1.10 and to avoid point 2 (resp. 4) in the above Proposition 2.10 and consider that point 3 (resp. 6) applies whenever $D_\ell < m^\omega$ (resp. $D_r < m^\omega$).

2.9 Idle times are non negative

The above computations of $idle_\ell$ and $idle_r$ are purely algebraic. It remains to check that these idle times are non negative!

Proposition 2.12. *If $\omega \geq 9$ and $j \geq 67$ then $idle_\ell$ and $idle_r$ are positive in all cases.*

Proof. Recall that
- $\lfloor \log_{m^j} x\rfloor = \lfloor(\log_m x)/j\rfloor = \lfloor\lfloor\log_m x\rfloor/j\rfloor \leq \lfloor \log_m x\rfloor/j$,
- $D_\ell < D/2$ and $D_r \leq D/2$ (cf. Proposition 2.1).

$\boxed{\textit{Case } D_\ell \leq m^\omega.}$

$$\begin{aligned}
idle_\ell &= D\lfloor\log_m D\rfloor - 4D_\ell - (D+D_r+d) - \max(2(D_\ell+d), 2D_r)\\
&= D\lfloor\log_m D\rfloor - D - \max(6D_\ell+D_r+3d, 4D_\ell+3D_r+d)\\
&= D\lfloor\log_m D\rfloor - D - \max((D_\ell+D_r+d)+2(D_\ell+d)+3D_\ell,\\
&\qquad\qquad (D_\ell+D_r+d)+2(D_\ell+D_r)+D_\ell)\\
&= D\lfloor\log_m D\rfloor - D - (D_\ell+D_r+d) - 2(D_\ell+d) - 3D_\ell\\
&\geq D\lfloor\log_m D\rfloor - 5.5D\\
&> 0 \qquad \text{if } \omega \geq 6 \text{ (since } D > m^\omega \text{ implies } \lfloor\log_m D\rfloor \geq \lfloor\omega\rfloor)
\end{aligned}$$

$\boxed{\textit{Case } D_r \le m^\omega.}$ We apply inequality from Proposition 2.4 to reduce to the analog of the previous case.

$$\begin{aligned}
idle_r &= D\lfloor \log_m D\rfloor - 4D_r - \max(D+D_r, 3D_\ell + d)\\
&\quad - \max(2(D_r+d), 2D_\ell)\\
&\ge D\lfloor \log_m D\rfloor - 4D_r - (D + D_\ell + d) - \max(2(D_r+d), 2D_\ell)\\
&> 0 \qquad \text{if } \omega \ge 6
\end{aligned}$$

$\boxed{\textit{Case } D_\ell > m^\omega.}$ In view of the expressions of $idle_\ell$ in points 2, 3 of Proposition 2.10, it suffices to consider the subcase $2(D_\ell + d) > m^{2j}$.

$$\begin{aligned}
idle_\ell &= D\lfloor \log_m D\rfloor - D_\ell\lfloor \log_m D_\ell\rfloor - (D + D_r + d)\\
&\quad - \max(2(D_\ell+d), 2D_r) - 6(D_\ell + d)\lfloor \log_{m^j} 2(D_\ell + d)\rfloor\\
&\ge D\lfloor \log_m D\rfloor - (D/2)\lfloor \log_m D\rfloor - (6D/j)(\lfloor \log_m D\rfloor + 1)\\
&\quad -(D + D_r + d) - \max(2(D_\ell+d), 2D_r)\\
&= D\lfloor \log_m D\rfloor(\frac{1}{2} - \frac{6}{j}) - D(1 + (6/j))\\
&\quad - \max(2D_\ell + D_r + 3d, 3D_r + d)\\
&\ge D\lfloor \log_m D\rfloor(\frac{1}{2} - \frac{6}{j}) - D(1 + (6/j))\\
&\quad - \max(D + (D_\ell + d) + d, (D_r + d) + 2D_r)\\
&= D\lfloor \log_m D\rfloor(\frac{1}{2} - \frac{6}{j}) - D(4 + (6/j))\\
&= (\frac{1}{2} - \frac{6}{j})D\ (\lfloor \log_m D\rfloor - 2(4j+6)/(j-6))\\
&= (\frac{1}{2} - \frac{6}{j})D\ (\lfloor \log_m D\rfloor - 8 - (60/(j-6)))\\
&> 0 \qquad \text{if } \omega \ge 9 \text{ and } j > 66
\end{aligned}$$

(since $D > m^\omega \ \Rightarrow\ \lfloor \log_m D\rfloor \ge \lfloor \omega\rfloor \ge 9$ and $j > 66\ \Rightarrow\ 60/(j-6) < 1$).

$\boxed{\textit{Case } D_r > m^\omega.}$ Again, it suffices to consider the subcase $2(D_r + d) > m^{2j}$. As already seen, $\max(D + D_r, 3D_\ell + d) \le D + D_\ell + d$. Thus,

$$\begin{aligned}
idle_r &= D\lfloor \log_m D\rfloor - D_r\lfloor \log_m D_r\rfloor - \max(D + D_r, 3D_\ell + d)\\
&\quad - \max(2(D_r+d), 2D_\ell) - 6(D_r+d)\lfloor \log_{m^j} 2(D_r + d)\rfloor\\
&\ge D\lfloor \log_m D\rfloor - D_r\lfloor \log_m D_r\rfloor - (D + D_\ell + d)\\
&\quad - \max(2(D_r+d), 2D_\ell) - 6(D_r+d)\lfloor \log_{m^j} 2(D_r + d)\rfloor\\
&> 0 \qquad \text{if } \omega \ge 9 \text{ and } j > 66
\end{aligned}$$

□

2.10 Main theorem for the line

§2.1–2.9 prove the following theorem.

Theorem 2.13. *Let $m \geq 2$. There exists a cellular automaton synchronizing every line of cells with arbitrary symmetric communication delays in time*

$$\begin{cases} D\lfloor \log_m D \rfloor & \text{if } D > m^9 \\ 4D & \text{if } D \leq m^9 \end{cases}$$

where D is the total communication delay of the line.

Remark 2.14. 1. Vivien [11, 12]) has shown that $O(D\log(D))$ is a lower bound for synchronizing lines of two cells with arbitrary symmetric communication delays. Thus, Thm.2.13 is optimal.

2. Nevertheless, one can achieve linear time synchronization under particular restrictive conditions, namely (cf. Umeo, 1994 [10], and Yunes [13]),

- a bound on the number of pairs of cells with communication delay > 1,
- and inequalities stating that around a pair of cells with communication delay $d > 1$ there at least d pairs with communication delay 1.

3 Synchronization of graphs of degree $\leq d$

We shall consider graphs with degree $\leq d$ and such that the edges connected to some vertex are labelled in $\{1, ..., d\}$.
Let's call total communication delay D of a graph the sum of all communication delays between pairs of cells constituting the edges of the graph.

Use a depth-first search of the graph as in Rosenstiehl [8, 9] to cover the graph by a line which visits each cell twice. The total communication delay of this line is equal to twice the total communication delay of the graph.
Under the above hypothesis on the graph, there is an automaton which is able to set up this line. Finally, synchronization of the line is a synchronization of the graph.
From Theorem 2.13 we deduce the following result.

Theorem 3.1. *Fix some constant $d \geq 1$ and let $m \geq 2$. There exists a cellular automaton synchronizing every graph of cells with degree $\leq d$ and arbitrary symmetric communication delays in time*

$$\begin{cases} 2D\lfloor \log_m 2D \rfloor & \text{if } 2D > m^9 \\ 8D & \text{if } 2D \leq m^9 \end{cases}$$

where D is the total communication delay of the graph.

Acknowledgements. I am very grateful to the referee for his corrections and remarks.

References

[1] Christian Choffrut, editor. *Automata networks.* Lecture Notes in Computer Science, vol.50, Springer-Verlag, 1986.

[2] Tao Jiang. The synchronization of non uniform networks of finite automata. In *FOCS 89*, pages 376–381, 1989.

[3] Tao Jiang. The synchronization of non uniform networks of finite automata. *Information and Computation*, 97(2):234–261, 1992.

[4] Donald E. Knuth. *The art of computer programming, volume I: Fundamental algorithms.* Addison-Wesley, 1968.

[5] Jacques Mazoyer. An overview of the firing squad synchronization problem. In [1], pages 82–94, 1986.

[6] Jacques Mazoyer. Synchronization of a line of finite automata with non uniform delays. Unpublished, 1990.

[7] Marvin Minsky. *Finite and infinite machines.* Prentice-Hall, 1967.

[8] Pierre Rosenstiehl. Existence d'automates capables de s'accorder bien qu'arbitrairement connectés et nombreux. *International Computation Center Bull.*, 5:245–261, 1966.

[9] P. Rosenstiehl and J.R. Fiksel and A. Holliger. Intelligent graphs: networks of finite automata capable of solving graph problems. In R.C. Read, editor, *Graph theory and computing*, pages 219–265, Academic Press, 1972.

[10] Hiroshi Umeo. A fault-tolerant scheme for optimum-time firing squad synchronization. In G.R. Joubert and F.J. Peters and D. Trystram and D.J. Evans, editors, *Parallel computing: Trends and Applications*, pages 223–230, Elsevier Science, 1994.

[11] Hélène Vivien. A quasi-optimal time for synchronizing two interacting finite automata. *Journal of Algebra and Computation*, 6(2):261–267, 1996.

[12] Hélène Vivien. *Cellular automata: a geometrical approach.* Book to appear.

[13] Jean-Baptiste Yunés. *Fault tolerant solutions to the firing squad synchronization problem in linear cellular automata.* To appear.