Acta Cybernetica, Tom. 2, Fasc. 4, Szeged, 1975.

A three-dimensional cellular space

(A challenge to Codd-ICRA)

By A. DOMÁN

Abstract

A three-dimensional cellular space is suggested. Three-dimensionality enables the designer to form structures of highly consize character. As an advantage to the traditional — generally two dimensional — cellular spaces it yields not only reduction in cells number but also, as a result of the paths, some increase in operation speed. By the way, crossover problem (so central in planar constructions) simply does not arise.

A next relevant feature of our space is the complete interchangeability of the inputs and unlike in Neumann-space the outputs too (full symmetry). As a consequence, full symmetry ensures a considerably simple possibility of hardware implementation by IC technology, MSI certainly suffices. In addition, along design symmetry properties of the cell can be well made use of.

A *signal* several means for signal propagation, gates, gating and storage elements have been defined. An attempt has been made to gain effectivity. Besides the undoubtedly advantageous features like hardware simplicity and software effectivity there are inevitably some disadvantages. A characteristic feature of cellular automata: self-reproduction have to be given up. An other feature, viz. growing automata and growing structures again, have to be sacrificed. These features, however, cannot be considered disadvantages as far as computational applications are concerned. A practical difficulty arises from the spatial characteristics. Along design traditional drawing have to be replaced, at least partly, by a sort of a model making technique.

Introduction

Cellular space is a homogeneous, clocked network of identical mutually connected automata where the next state of each automaton depends only on its own and on its neighbors' last states.

In the recent few years a number of papers have been published dealing with several types of cellular spaces. A representative collection of articles (until around the mid 60's) can be found in Burks [1]. Cellular automata, i.e. automata "planted" in cellular space, or configurations in cellular space have been first studied by von Neumann [12] motivated by brain researches and has its way back to the famous Hixon Symposium in 1948 (see Jeffress ed. 1951). Since Burks [1] a Conference on Biologically Motivated Automata represents the last development [9].

As for the motivations of most of the studies into cellular automata, it seems that cellular automata are considered as organisation *principles* or new types of computation *techniques*, rather than as a matter of hardware reality, or a true medium to be produced so as to act as a host medium for constructing and breeding growing and selfreproducing automata.

One exception to that approach comes from a Hungarian group aiming at the implementation and effectivization of cellular spaces (see Fáy [6], Fáy—Takács [7], Fáy [5], Takács [14] Dettai [3], Szőke [13], Fazekas [8], Kaszás [11], Huszár [10], Dettai [4]).

By the present paper, we would like to join this direction of research as far as the effectivization of cellular automata (or space) is concerned. Yet there is a difference, for all the last practically motivated papers (Dettai [4] excepted) are centering around the effectivization of Codd's cellular space while the one here is not. It turned out that at present a Codd-cell (even in its late version, called Codd-ICRA) is too complex to be implemented economically by integrated circuit technology. It's true, it could be done both theoretically as well as practically; it is only *economy* which is questionable in my opinion at least. At any rate, I feel I have found a version for cellular space which is competitive with respect to ICRA's performance and definitely much simpler than it is in its circuitry realization.

As for the preliminaries an attempt is made to be fairly self-contained. As for further backgrounds, underlying ideas and motivations, however, readers are better adviced to refer to Codd [2], and to the publications referred to above.

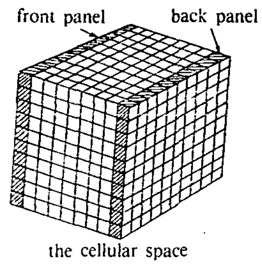


Fig. 1

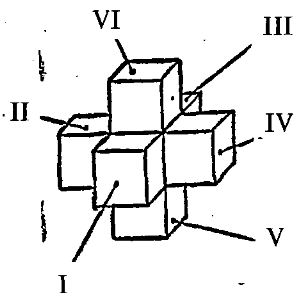


Fig. 2
The neighborhood template of the three-dimensional space with the names of neighbors.
(First (I) or front neighbor; second (II) or left neighbor; third (III) or back neighbor; fourth (IV) or right neighbor; fifth (V) or bottom neighbor; sixth (VI) or top neighbor.)

Preparations

It is well enough for our present purposes if the cellular space for developing and studying, is visualized as a rectangular block of identical cubes of finite number (see Fig. 1). This number is around one million in practice.

The *neighbors* of a cell in this space are the cells sharing common faces with the *center cell,* i.e. the cell in question. In other words, a next-neighborhood is accepted with six neighbors of each cell. The neighborhood *template* can be seen on Fig. 2.

In our six-neighbor space each cell has eight states. These are denoted by 0, 1, 2, ..., 7. Usually blank is used for 0.

We need a convention for telling the transition rules.

This again is explained through an example:

S cell state	N neighbors state	O next state
3	$\bar{3}$ 5 $\bar{7}$	6

This excerpt of the *transition table* reads as follows: *If* the present state of a cell is 3 *and* it has *no* neighbor in state 3 *and* at least one neighbor in state 5 *and* *no* neighbor in state 7 *then* this cell's next state will be 6.

Blank entry in column N means that the next state is completely independent from the neighbors.

According to this convention the complete *transition function* is contained in Table 1.

Table 1.
The transition function

S	N	O
0		0
1	4 6 $\bar{5}$	1
1	4 $\bar{6}$ 5	7
1	4 6 5	4
1	4 $\bar{6}$ $\bar{5}$	4
1	7	4
1	$\bar{4}$ $\bar{7}$	1
2		1
3	4 7	5
3	3 5	5
3	$\bar{3}$ 5 $\bar{7}$	6
3	7 $\bar{4}$	3
3	$\bar{7}$ $\bar{5}$	3
4		2
5	4 7	3
5	3 6	3
5	$\bar{3}$ $\bar{4}$	5
5	3 5	5
5	2 7	5
6	3	3
6	$\bar{3}$	6
7	2	2
7	$\bar{2}$	7

Needless to say, the transition function, just like that of Codd's, is a *partially defined* one. The function is so simple that its implementation by integrated circuit technology is quite a straightforward routine.

This convention of defining the transition function is legitimized by its peculiar logical structure.

A novel feature, compared with the usual cell spaces, is that our cells have *two modes of operation*. The transition function, defined by Table 1, refers to the first mode of operation called *ordinary mode* (OM). The second mode is called *shift mode* (SM). In this mode the whole cellular space acts as a collection of parallel coupled serial shift registers connecting the front panel to the back panel. In this mode any configuration of the front panel shifts forward during the mode. By this, one can transfer any configuration into the inside of the space. Of course, spatial figures (or structures) have to be sliced up and the slices have to put serially (step by step) in touch with the front panel's cells. Fig. 3 shows this "shift-in" procedure.

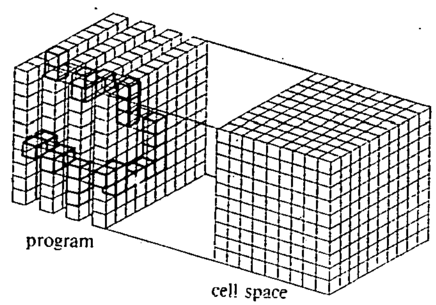


Fig. 3

The concept of the *empty* or *quiescent* state of the space is identical with the usual. In the space with each cell in state 0 and all the edge cells having no input, no change occurs.

The problem of *clearing or erasure* is solved by the shift mode. All we have to do is to switch to SM from OM and shift out everything to the back panel.

Basic structures and functions

Paths. As in Codd [2], a *path* can, in the simplest case, be defined as a linear row of neighboring cells all in state 1. Paths, then can have bends (corners), branches, loops etc. Typical paths are shown on Figures 4-5.

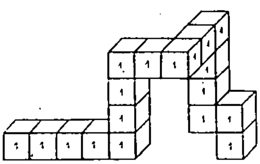


Fig. 4
A typical simple path

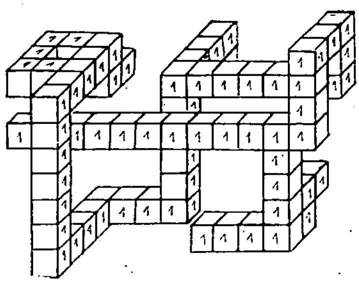


Fig. 5
A typical path

Sometimes it is possible — and preferable too — to resort to planar representation. If so we use the traditional means, for instance: cells are represented by squares, empty cell means cell in state 0, filled square means cell in state one, the other cell's states are denoted by numbers placed in the representing square. Unlike Codd we do *not* sheathe paths.

Signals. We have only one type of *signal* which is a pair of adjacent cells with state 2 and 4 respectively. So our signals are always of the form "24". (By contrast, Codd has four signals: 04, 05, 06 and 07.)

Signals propagate along paths with 4 heading and 2 tailing. So signal propagation is shown on the shot-figures:

```
1 1 2 4 1 1 1 1 1 1   at  t = 1
1 1 1 2 4 1 1 1 1 1       t = 2
1 1 1 1 2 4 1 1 1 1       t = 3
1 1 1 1 1 2 4 1 1 1       t = 4
1 1 1 1 1 1 2 4 1 1       t = 5
```

Propagating in opposite directions along the same path *signal collision* may occur. Signal collision results annihilation independently from the parity of the signal distance

Signal collision with odd parity:

```
            events                 shots
1 1 2 4 1 1 1 4 2 1 1 1 1          t
1 1 1 2 4 1 4 2 1 1 1 1 1          t+1
1 1 1 1 2 4 2 1 1 1 1 1 1          t+2
1 1 1 1 1 2 1 1 1 1 1 1 1          t+3
1 1 1 1 1 1 1 1 1 1 1 1 1          t+4
```

Signal collision with even parity:

```
1 1 2 4 1 1 1 1 4 2 1 1 1          t
1 1 1 2 4 1 1 4 2 1 1 1 1          t+1
1 1 1 1 2 4 4 2 1 1 1 1 1          t+2
1 1 1 1 1 2 2 1 1 1 1 1 1          t+3
1 1 1 1 1 1 1 1 1 1 1 1 1          t+4
```

Path branching:

```
        1                   1                   1
1 2 4 1 1 1 1       1 1 2 4 1 1 1       1 1 1 2 4 1 1
        1                   1                   1
        1                   1                   1
        t                  t+1                 t+2

              1                   4
        1 1 1 1 2 4 1       1 1 1 1 1 2 4
              4                   2
              1                   4
             t+3                 t+4
```

Signal collisions in crosspaths:

```
        1                 1                 1
        1                 1                 1
1 2 4 1 1 1 1     1 1 2 4 1 1 1     1 1 1 2 4 1 1
        1                 4                 2
        4                 2                 1
        t                t+1               t+2

        1                 4
        4                 2
1 1 1 1 2 4 1     1 1 1 1 1 2 4
        1                 1
        1                 1
       t+3               t+4
```

odd distance

```
      1               1               1
      1               1               4
1 2 4 1 1 1     1 1 2 4 1 1     1 1 1 2 4 1
      1               1               4
1 2 4 1         1 1 2 4         1 1 1 2
      t              t+1             t+2

      4               2
      2               1
1 1 1 1 2 4     1 1 1 1 1 2
      2               1
1 1 1 1         1 1 1 1
     t+3             t+4
```

even distance

Collision in fork:

```
1 1 2 4 1           1 1 1 2 4           1 1 1 1 2           1 1 1 1 1
        1 1 1 1             1 1 1 1             4 1 1 1             2 4 1 1
1 1 2 4 1           1 1 1 2 4           1 1 1 1 2           1 1 1 1 1
```

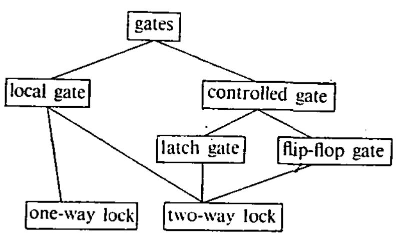


Fig. 6

Gates. In our construction gates play the key roles. Structurally one has four basic types of gates contrasted with Codd who had only one. (Whereas he had four signals while we manage with one.)

The four gate types are as follows (see Fig. 6). Local gates (being not controllable) are always permanent.

Local one-way locks consist of two cells placed beside the path to be gated, i.e. the *subordinate path:*

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1    subordinate path
          5 6
         /   \
slave cell     master cell
```

symbol: 

A signal propagating along the subordinate path will be annihilated by the local one-way lock if the signal first encounters the master cell then the slave cell. So, in contrast with Codd, no matter on which side of the path is the gate placed.

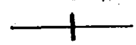
The function of the local one-way lock is:

```
1 1 1 1 1 4 2    1 1 1 1 4 2 1    1 1 1 1 2 1 1    1 1 1 1 1 1 1
    5 6              5 6              5 6              5 6
     t               t+1              t+2              t+3

2 4 1 1 1 1 1    1 2 7 1 1 1 1    1 1 2 4 1 1 1    1 1 1 2 4 1 1
    5 6              5 6              5 6              5 6
     t               t+1              t+2              t+3
```

Local two-way locks differ from the one-way locks that the slave cell is dropped:

```
1 1 1 1 1 1 1 1 1 1 1
          6
local two-way lock
```

symbol: 

Again, no matter whether it is on the right or left hand side of the path (from an approaching signal). A signal coming from any direction will be annihilated by the local two-way lock:

```
2 4 1 1 1 1 1    1 2 4 1 1 1 1    1 1 2 1 1 1 1    1 1 1 1 1 1 1
      6                6                6                6
      t               t+1              t+2              t+3

1 1 1 1 1 4 2    1 1 1 1 4 2 1    1 1 1 1 2 1 1    1 1 1 1 1 1 1
      6                6                6                6
      t               t+1              t+2              t+3
```

Controlled gates have access by paths either to the master cell or to the slave cell. If the control path leads to the master cell we got the *flip-flop type gate* (see Fig. 7).

```
                           1 1 1 1 1 1 1 1 1 1 1——subordinate path
slave cell (in state 5)——5 6——master cell (in state 6)
support head——————————7 5 1
support cell (in state 5)——1 1
                             1
                             1
                             1
          control path       1
```

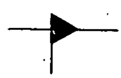
symbol: 

Fig. 7
Flip-flop gate

As it can be seen from Fig. 7 two additional cells are placed. The first is the first neighbor of the slave cell called *support cell,* the second is the second neighbor of the support cell called *support head.* Of course, other arrangements are equally possible as far as they are produced by a transformation permitted by the transition function. These other arrangements are considered to be isomorphic to the one on Fig. 7.

Owing to the support cell and support head control becomes possible through the control path.

Upon receiving a signal first the support cell goes into state 3 then in the next shot both the master and the slave cell change to 3:

```
1 1 1 1 1     1 1 1 1 1     1 1 1 1 1     1 1 1 1 1     1 1 1 1 1
  5 6           5 6           5 6           5 6           3 3
7 5 1         7 5 1         7 5 4         7 3 2         7 3 1
    1             4             2             1             1
    4             2             1             1             1
    2             1             1             1             1
    1             1             1             1             1
    1             1             1             1             1
```

Now the configuration of both the master and slave cells in state 3 are neutral making no effect on the signal passing the gate on the subordinate path:

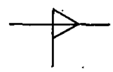
```
2 4 1 1 1 1 1 1     1 2 4 1 1 1 1 1     1 1 2 4 1 1 1 1
      3 3                 3 3                 3 3
    7 3 1               7 3 1               7 3 1
        1                   1                   1
        1                   1                   1
        1                   1                   1
        1                   1                   1
        1                   1                   1

1 1 1 2 4 1 1 1     1 1 1 1 2 4 1 1     1 1 1 1 1 2 4 1
      3 3                 3 3                 3 3
    7 3 1               7 3 1               7 3 1
        1                   1                   1
        1                   1                   1
        1                   1                   1
        1                   1                   1
        1                   1                   1
```

On the other hand:

```
1 1 1 1 1 1 4 2     1 1 1 1 1 4 2 1     1 1 1 1 4 2 1 1
    3 3                 3 3                 3 3
  7 3 1               7 3 1               7 3 1
      1                   1                   1
      1                   1                   1
      1                   1                   1
      1                   1                   1
      1                   1                   1
```

```
1 1 1 4 2 1 1 1    1 1 4 2 1 1 1 1    1 4 2 1 1 1 1 1    4 2 1 1 1 1 1 1
    3 3                3 3                3 3                3 3
  7 3 1              7 3 1              7 3 1              7 3 1
      1                  1                  1                  1
      1                  1                  1                  1
      1                  1                  1                  1
      1                  1                  1                  1
      1                  1                  1                  1
```

symbol: 

Flip-flop gate with master and slave cells in state 3 is called to be in "off"-state while the other (with master cell in state 6 and slave in 5) is "on"-state. As a synonym a gate in "on"-state is also called *closed gate* and an off gate is called *open gate*. The transition from on to off is referred to as *opening* and from off to on as *closing*.

A new signal propagating along the control path is capable of closing an open flip-flop gate:

```
1 1 1 1 1 1      1 1 1 1 1 1      1 1 1 1 1 1
    3 3              3 3              3 3
  7 3 1            7 3 4            7 5 2
      4                2                1
      2                1                1
      1                1                1
      1                1                1
      1                1                1
- - - - - - - - - - - - - - - - - - - - - - - -
    1 1 1 1 1      1 1 1 1 1        1 1 1 1 1
      5 6            3 6              3 3
    7 3 2          7 3 1            7 3 1
        1              1                1
        1              1                1
        1              1                1
        1              1                1
        1              1                1
```

Latch gate differes structurally from the flip-flop gate that the control path is leading directly to the support cell rather than to the slave cell and passing the support (see Fig. 8).

```
                 1 1 1 1 1 1 1 1 1 1 1 1   subordinate path
slave cell ———————— 5 6 ——— master cell
support head ——— 7 5 ——— support cell
                   1
                   1
                   1
                   1
                   1
                   1 control path
```

Fig. 8
Latch gate in ON state

A latch gate, once closed, can never be opened. It has the latch property:

```
1 1 1 1 1 1   1 1 1 1 1 1   1 1 1 1 1 1   1 1 1 1 1 1   1 1 1 1 1 1
    5 6           5 6           5 6           5 6           5 6
  7 5           7 5           7 5           7 5           7 5
    1             7             2             1             1
    4             2             1             1             1
    2             1             1             1             1
    1             1             1             1             1
    1             1             1             1             1
    1             1             1             1             1
```

An open latch gate, however, can be closed:

```
1 1 1 1 1    1 1 1 1 1    1 1 1 1 1    1 1 1 1 1    1 1 1 1 1
    3 3          3 3          3 3          5 3          5 6
  7 3          7 3          7 5          7 5          7 5
    1            4            2            1            1
    4            2            1            1            1
    2            1            1            1            1
    1            1            1            1            1
    1            1            1            1            1
    1            1            1            1            1
```

A direct version of a controlled two-way lock is on Fig. 9.

```
    1 1 1 1 1 1 1 1 1 1
                      1
                      1
                      1 1 1 1 1 1 1 1 1 1   subordinate path
slave cell -------- 5 6 — master cell
support head ---- 7 5 1
                    | 1
support cell -------' 1
                      1
                      1
                      1   control path
```

Fig. 9

Controlled two-way lock (in ON state)

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1    subordinate path
        5 6   6 5
      7 5 1 1 1 5 7
            1
            1
            1
            1
            1   control path
```

Fig. 10

A controlled two-way lock as a combination of two controlled flip-flop gates

By combining gates of the above types one can produce new structures capable of performing new functions. For instance a controlled two-way lock can be seen on Fig. 10.

Also gates are suitable to control more than one path simultaneously. Some immediate versions can be seen on Figures 11—13. These intensively make use of the advantages due to the three-dimensionality of the space.

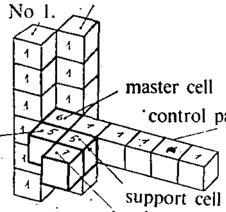


Fig. 11

Multifunctional usage of gates is utterly impossible in Codd's twodimensional space. We feel that this possibility in our space gives ample compensation for the difficulties arousing from the clumsy spatial representation techniques in design and demonstration.

```
                                  subordinate path No 2.
master cell ──────────\x ─────── (orthogonal to the drawing)
slave cell ────────── 5 6 x ──── subordinate path No 1.
support head ──── 7 5 1          (orthogonal to the drawing)
support cell ──────── 1
                        1
                        1
                        1
                        1
```

Fig. 12
Top view of a bi-functional controlled two-way lock (cf. Fig. 11)

```
master cells ──────────────\x──── subordinate paths
slave cell                x 6 x   (orthogonal to the drawing)
(serving two masters) ──── 5 6 x
support head ─────────── 7 5 1
support cell ____________/1
                          1
                          1
                          1
                          1
                          1
                          1  control path
```

Fig. 13
A multifunctional, threedimensional gate-arrangement for controlling four paths simultaneously in two-way lock mode

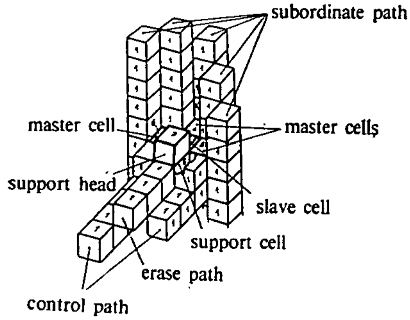


Fig. 14

The combination possibilities for the basic gate types are virtually indefinite. To close this paragraph let us see just one more (a truly spatial) multifunctional gate arrangement having two control paths and five subordinate paths (see Fig. 14).

A multifunctional double control oneway lock system. Support head is the bottom neighbor of the support cell. The system is capable of performing quite a sophisticated control function on five paths occupying only a volume less then ten cells.

Other structures and functions

Gating. Gates do not represent the only possibilities for performing gating functions. An other possibility is offered by signal collisions. OR-functions can be achieved by "forks". Signal 24 entering at point A or B or both simultaneously results a signal 24 at the output point C (see Fig. 15).

```
              5 6
A  1 1 1 1 1 1 1
                1 1 1 1 1 1 1  C
B  1 1 1 1 1 1 1
              5 6
```

in symbols:

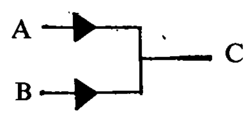


Fig. 15
Logical operation OR performed by a path fork rather than by a gate

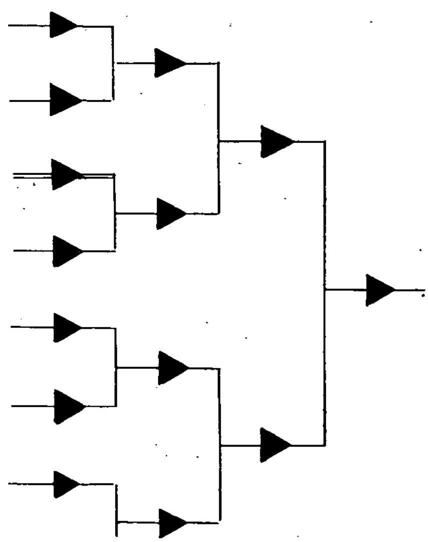


Fig. 16

An extension of this "gating by fork" can be seen on Fig. 16.

A *coincidence-* or AND-*gate* is somewhat more complex. Detailed shots of its operation can be seen on Fig. 17.

Into the control path a gate restoring loop is inserted by which the control signal opens the gate for 4 shots.

During that period the open gate permits the subordinate signal to pass, otherwise not. Thus the arrangement produces a *time slot* for the subordinate signal to pass. The minimal time slot is 4 shots.

In drawing this *interference gate* will be denoted this way: 

The number in the symbol indicates the time slot. Blank means minimal slot i.e. shots.

```
      1 1 1
      1   1
1 2 4 1 1 1 5 7
          6 5
1 2   1 1 1
  4   1   1 1 1
  1 1 1
```

$t = 1$

```
      1 1 1
      1   1
1 1 2 7 1 1 5 7
          6 5
1 1   1 1 1
  2   1   1 1 1
  4 1 1
```

$t = 2$

```
      1 1 1
      4   1
1 1 1 2 4 1 5 7
          6 5
1 1   1 1 1
  1   1   1 1 1
  2 4 1
```

$t = 3$

```
      4 1 1
      2   1
1 1 1 1 2 4 5 7
          6 5
1 1   1 1 1
  1   1   1 1 1 1
```

$t = 4$

```
  1 2 4
      2 4 1
      1   1
1 1 1 1 1 2 3 7
          6 5
1 1   1 1 1
  1   4   1 1 1
  1 1 2
```

$t = 5$

```
      1 2 7
      1   1
1 1 1 1 1 1 3 7
          6 3
1 1   4 1 1
  1   2   1 1 1 1
  1 1 1
```

$t = 6$

```
      1 1 2
      1   4
1 1 1 1 1 1 3 7
          3 3
  1   2 4 1
  1   1   1 1 1
  1 1 1
```

$t = 7$

```
      1 1 1
      1   2
1 1 1 1 1 4 3 7
          3 3
1 1   1 2 4
  1   1   1 1 1
  1 1 1
```

$t = 8$

```
      1 1 1
      1   1
1 1 1 1 1 2 5 7
          3 3
1 1   1 1 2
  1   1   4 1 1
```

$t = 9$

```
  1 1 1
      1 1 1
      1   1
1 1 1 1 1 1 5 7
          3 5
1 1   1 1 1
  1   1   2 4 1
  1 1 1
```

$t = 10$

```
      1 1 1
      1   1
1 1 1 1 1 1 5 7
          6 5
1 1   1 1 1
  1   1   1 2 4
  1 1 1
```

$t = 11$

By a suitable delay between signals leaving points A and B respectively one can manage that they meet again right at an Interference Gate I_i ($i=1, 2, \ldots, 7$). In this case this gate will be turned off (if previously it was ON) since signal can reach gates only through the open time slots.

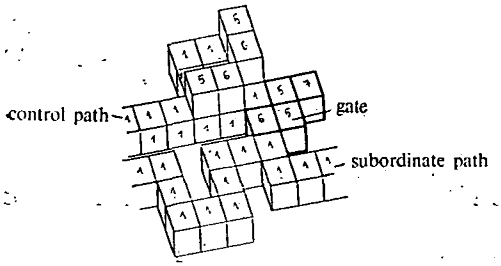


Fig. 17

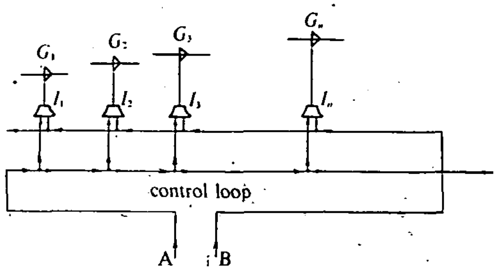


Fig. 18
The interference principle

Storage. Making use of the possibilities given by the spatial structure of gates one can easily form a *storage element* with read-write-erase capabilities (see Figures 19, 20). Fig. 19 shows a version with a one-way lock.

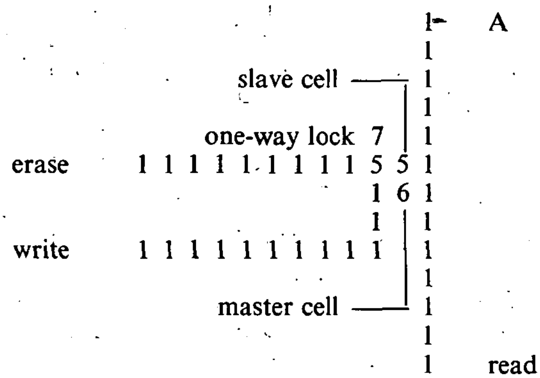


Fig. 19
A storage element using a one-way lock with read-write-erase capabilities

By path "write" one can control the master cell thereby gating the signal propagating along path "read". If the gate is ON then no signal can pass it. This state, say *empty state*, of the storage element is read out by the *lack* of the read signal. A read-signal appearing *beyond* the gate, at point A, indicates the *off state* of the storage element.

As it can be seen from Fig. 19 path "erase" is just a control path of a latch gate while, at the same time, from the "write"-path the same gate is a flip-flop gate. This flip-flop path is used for writing-in purposes (i.e. for writing in both states nought and one). On the other hand, the other path, i.e. the latch path can be eminently used for performing the erase-function of a storage element. On Fig. 20 a two-way lock version of the latter storage element is shown.

```
                              1   read
                              1
                              1
                              1
                        7     1
erase   1 1 1 1 1 1 1 1 5 5 6 1
                        1 6   1
write   1 1 1 1 1 1 1 1       1
                              1
                              1
                              1
                              1
                              1   read
```

Fig. 20
Storage element using two-way lock with two accesses for reading (cf. Fig. 19)

Conclusions

It has been demonstrated that quite effective components and functions can be gained in a cellular space containing unquestionably simple cells to implement. The number of ways of combination possibilities for the main components, such as several types of paths, gates and storage elements is virtually indefinite.

Of course, these elementary components themselves throw meager light on the true design advantages and disadvantages. Further R & D is necessary to be able to make decision in this respect.

COMPUTING INSTITUTE
MINISTRY OF LABOUR
H—1089 BUDAPEST, HUNGARY

References

[1] BURKS, A. W., *Essays on cellular automata,* University of Illinois Press, Urbana Illinois, 1968.
[2] CODD, E. F., *Cellular automata,* Academic Press, New York and London, 1968.
[3] DETTAI, I., A version of von Neumann's cell, *KGM ISzSzI Technical Report* Budapest, 1974.
[4] DETTAI, I., Effectivity problems and suggestions concerning Codd-ICRA cellular space, *KGM ISzSzI Technical Report* Budapest, 1974.
[5] FÁY, G., Circuitry realization of Codd's cellular automaton's cell, *Conference Paper* Székesfehérvár, 1973.
[6] FÁY, G., A lecture on DSL ALPHA and Codd-ICRA, *Conference Lecture* Vienna, 1974.
[7] FÁY, G. & D. V. TAKÁCS, Galois perceptron cell assemblies in cellular space, *Conference paper* Paris, 1974.
[8] FAZEKAS, B., Design for a "RETINA", *KGM ISzSzI Technical Report* Budapest, 1974.
[9] HERMANN, G. T., Conference on Biologically Motivated Automata held at MITRE Virginia June, 1974.
[10] HUSZÁR, A., Testing equipment for automaton cell circuitry, *KGM ISzSzI Technical Report* Budapest, 1974.
[11] KASZÁS, O., Character generation in Codd-ICRA, *KGM ISzSzI Technical Report* Budapest, 1974.

[12] von Neumann, J., *The theory of self-reproducing automata*, ed. A. W. Burks, University of Illinois Press, Urbana Illinois, 1966.
[13] Szőke, M., Codd-ICRA: Regularization of Codd's transition function and its implementability, *KGM ISzSzI Technical Report* Budapest, 1974.
[14] Takács, D. V., A bootstrap for Codd's cellular space, *Conference Paper* Székesfehérvár, 1973.
[15] Takács, D. V., Galois connection and DPL ALPHA system, *Conference Paper* Paris, May. 1974.

(Received Oct. 29, 1974)