On fault tolerant L-processors

By S. R. GROSS

1. Introduction and motivation

In sequential computers the number of real active elements is very small in contrast to the total number of elements. On the other hand the working speed of these elements has nearly reached the physical limitations (with respect to the today's technologies). Therefore a speed up is no more reasonable by improving the conventional (von Neumann type) architectures and a natural approach to increase the computing power seems to be the principle of parallel processing in computer architecture. Cellular spaces might be considered as models for such parallel computing devices. The implementation of cellular structures renders possible by the advent of LSI-technology, too.

Since cellular spaces consist of many cells which are not necessarily totally reliable, it is important to develop strategies for error correction and error detection. Up to now several articles concentrate on this topic, see e.g. [HANO 75], [NIKO 75], and [WRIG 76]. Here we want to investigate this topic in L-processors, which differ from a cellular space as follows:

- a state is a 2-tupel which consists of a visible (external) component and an invisible (internal) component, a so-called qualifier (i.e. the basic cell is a Mealy-type automaton),
- there are several local maps,
- the transition-functions are centralized in a special control unit and 'shared' by the single cells.

Thus, an L-processor may be considered to be an emulator of a cellular space. For details see Legendi [LEGE 76].

The techniques of error correction in L-processors are similar to those of NISHIO and KOBUCHI [NIKO 75] where the basic idea is that the work of each original cell is simulated by three cells. In this case we are able to correct single errors prior to each state transition by a majority decision, if we assume that the majority decision element is faultless.

2. Notations and basic definitions

Let $\mathbf{N}$ denote the natural numbers, $\mathbf{N}_0=\mathbf{N}\cup\{0\}$ and $\mathbf{Z}$ the set of integers. For $d\in\mathbf{N}$ the d-fold Cartesian product of $\mathbf{Z}$ is denoted by $\mathbf{Z}^d$, i.e. $\mathbf{x}\in\mathbf{Z}^d$ can be written as $\mathbf{x}=(x_1, x_2, \ldots, x_d)$ with $x_i\in\mathbf{Z}$ $(i=1, \ldots, d)$. Especially $\mathbf{0}=(0, 0, \ldots, 0)$. $P_i^{(d)}(x_1, \ldots, x_d):=x_i$ $(i=1, \ldots, d)$ denotes the i-th projection. If X is a set, the

cardinality of X is denoted by $|X|$ and the set of subsets of X by $\mathfrak{P}(X)$. $\emptyset$ denotes the empty set.

For $\mathbf{x}, \mathbf{y} \in \mathbf{Z}^d$ we define the operations $\mathbf{x} \pm \mathbf{y} := (x_1 \pm y_1, x_2 \pm y_2, \ldots, x_d \pm y_d)$. Let $X, Y \subseteq \mathbf{Z}^d$, $X \neq \emptyset$, $Y \neq \emptyset$. Then we define the sum and difference by $X \pm Y := \{\mathbf{x} \pm \mathbf{y} / \mathbf{x} \in X \wedge \mathbf{y} \in Y\}$. In the following we write $\mathbf{x} \pm Y$ instead of $\{\mathbf{x}\} \pm Y$. Let $\mathbf{x} \in \mathbf{Z}^d$ and $T = (t_{ij})_{i,j=1,\ldots,d}$ a matrix with $t_{ij} \in \mathbf{Z}$ $(i, j = 1, \ldots, d)$. Then we define the product

$$\mathbf{x} \cdot T := \left(\sum_{i=1}^{d} x_i \cdot t_{i1}, \ldots, \sum_{i=1}^{d} x_i \cdot t_{id} \right)$$

and for $X \subseteq \mathbf{Z}^d$, $X \neq \emptyset$

$$X \cdot T := \{\mathbf{x} \cdot T / \mathbf{x} \in X\}.$$

Definition 2.1. Let $d, n \in \mathbf{N}$. An n-tupel $N = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)$, $\mathbf{a}_i \in \mathbf{Z}^d$ $(i = 1, \ldots, n)$, $\mathbf{a}_i \neq \mathbf{a}_j$ for $i \neq j$ is called a d-dimensional neighbourhood index.

The (unordered) set $\tilde{N} = \{P_i^{(n)}(N) / i = 1, \ldots, n\}$ is called a neighbourhood template. In general we claim that there exists an i $(1 \leqq i \leqq n)$ such that $\mathbf{a}_i = \mathbf{0}$. In the following we use 'neighbourhood index' and 'neighbourhood template' synonymously.

Definition 2.2. Let $k \in \mathbf{N}_0$, $d \in \mathbf{N}$. The von Neumann neighbourhood template is defined by $H_k^{(d)} := \{\mathbf{x} / \mathbf{x} \in \mathbf{Z}^d \wedge |\mathbf{x}| \leqq k\}$, where $|\mathbf{x}| = \sum_{i=1}^{d} |x_i|$.

We write H_k instead of $H_k^{(d)}$ if there is no doubt of the dimension. $H_1 = \{(0, 0), (-1, 0), (0, 1), (1, 0), (0, -1)\}$ is the most frequently used von Neumann neighbourhood template.

Definition 2.3. Let A, Q be finite nonempty sets and $\tilde{N} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ a d-dimensional neighbourhood template. $f: A^n \times Q \to A \times Q$ is called a local transition function.

Definition 2.4. $\mathfrak{A} = (A, Q, d, N, \{f^{(1)}, \ldots, f^{(l)}\})$ is called an L-processor, if

1) A is a finite nonempty set of (external) states,
2) Q is a finite nonempty set of (internal) states, in the following called qualifiers,
3) d is the dimension of the space,
4) $N = (\mathbf{a}_1, \ldots, \mathbf{a}_n)$ is a d-dimensional neighbourhood index,
5) $f^{(j)}: A^n \times Q \to A \times Q$ $(j = 1, \ldots, l)$ are local transition functions,
6) There exists a $(z_0, q_0) \in A \times Q$, called the quiescent state of the L-processor, with $f^{(j)}(z_0, \ldots, z_0, q_0) := (z_0, q_0)$ for $j = 1, \ldots, l$.

Definition 2.5. Let $\mathfrak{A}$ be an L-processor. A function $c: \mathbf{Z}^d \to A$ is called an external configuration or simply configuration. A function $q: \mathbf{Z}^d \to Q$ is called an internal configuration. The set of total (internal and external) configurations is denoted by C.

Definition 2.6. $F: C \to C$ is called a global transition function (induced by f), if for all $\mathbf{x} \in \mathbf{Z}^d$ and for all total configurations $(c, q) \in C$ the following holds: $Fc(\mathbf{x}), q(\mathbf{x})) = f(c(\mathbf{x} + \mathbf{a}_1), \ldots, c(\mathbf{x} + \mathbf{a}_n), q(\mathbf{x}))$.

Definition 2.7. The support of a total configuration is defined by $\sup(c, q) = \{\mathbf{x}/\mathbf{x}\in\mathbf{Z}^d \wedge (c(\mathbf{x}), q(\mathbf{x})) \neq (z_0, q_0)\}$ with (z_0, q_0) the quiescent state. The set of all total configurations with finite support is denoted by $\bar{C}$.

If we use L-processors for computations we want a reliable system although the individual cell may be unreliable. A cell is said to misoperate if its next external state differs from the expected one (by application of the local transition function to the states of its neighbours). The reason for such a misoperation may be the permanent breakdown of a cell, an occasional failure caused by noise or something else. Therefore an L-processor is called a real L-processor if some cells may misoperate.

In order to make analysis practicable, we restrict the occurrence of misoperations in the following way:

Definition 2.8. Let K be a finite connected subset of $\mathbf{Z}^d$ containing the origin $\mathbf{0}$. A real L-processor is said to misoperate with K-separated errors if at most one cell of each area $\mathbf{x}+K$, $\mathbf{x}\in\mathbf{Z}^d$, misoperates at each state transition.

'Connected' means connected with respect to the underlying neighbourhood template $\tilde{N}$. In the following we denote this as K-separated error condition (K-se condition).

Definition 2.9. Let $\mathfrak{A}_1=(A_1\times Q_1, 2, H_1, \{f_1^{(1)}, \ldots, f_1^{(k)}\})$ be an L-processor, $\mathfrak{A}_2=(A_2\times Q_2, 2, N_2, \{f_2^{(1)}, \ldots, f_2^{(l)}\})$ a real L-processor and $k_1, k_2\in\mathbf{N}$. We say $\mathfrak{A}_2$ simulates $\mathfrak{A}_1$ in k_2/k_1-time if and only if there exist functions G and H such that the diagram in Fig. 1 is commutative. The index K indicates that the K-se condition holds.

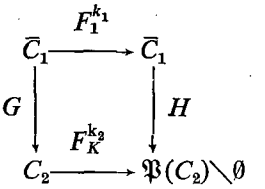

Fig. 1
Simulation of L-processors

The 'realistic' global transition function F_K maps C_2 into the set $\mathfrak{P}(C_2)\setminus\emptyset$ of subsets of C_2, because a configuration may have more successors in consequence of the real behavior.

3. Error correction in realtime with an 21-element neighbourhood template

Now we want to design a real L-processor $\mathfrak{A}_2=(A_2\times Q_2, 2, N_2, \{f_2^{(1)}, \ldots, f_2^{(l)}\})$ which simulates a given L-processor $\mathfrak{A}_1=(A_1\times Q_1, 2, H_1, \{f_1^{(1)}, \ldots, f_1^{(k)}\})$ in realtime, i.e. $k_1=k_2$ (see Definition 2.9).

The most elementary trick of error correction might be to implement each cell three times and to take the majority result. Therefore we must spread out the

2-dimensional space of integers by a transformation matrix $T=\begin{pmatrix}3 & 0\\ 0 & 1\end{pmatrix}$. A coding unit $M=\{(0,0),(1,0),(2,0)\}$ fills up the gaps. It is easy to show that T and M guarantee a unique and total cover of $\mathbf{Z}^2$. G maps a cell and its neighbours into the following region, see Fig. 2.

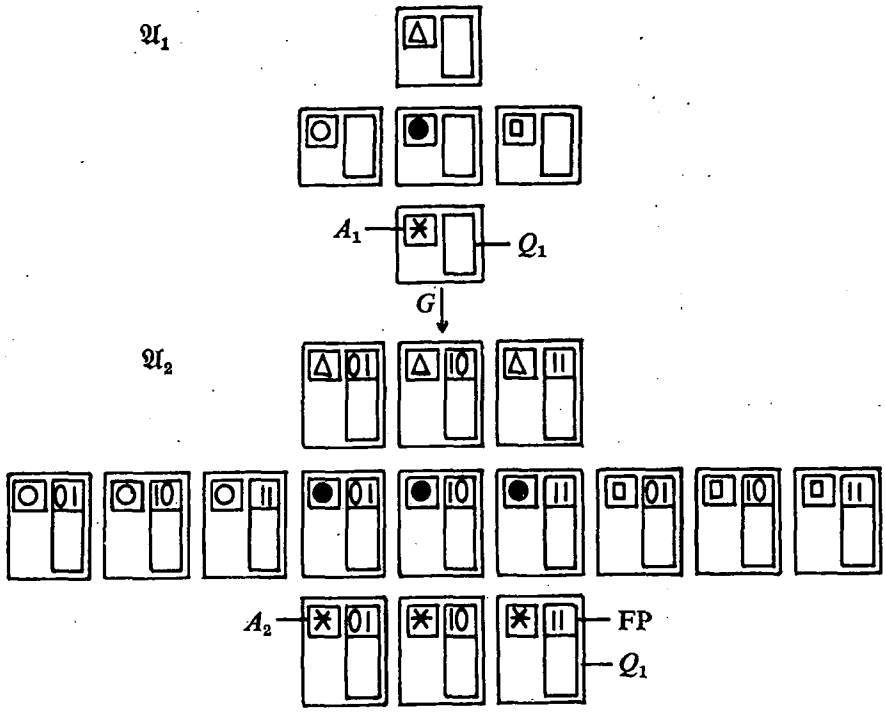

Fig. 2
Function G

Each cell $\mathbf{x}\in\mathbf{Z}^2$ of $\mathfrak{A}_1$ is mapped into the cell $\mathbf{x}\cdot T$ by the transformation matrix T. Now we add the coding unit to it. It is easy to see that we get for each cell in $\mathfrak{A}_1$ exactly three cells in $\mathfrak{A}_2$ in this way. In the following we denote $\mathbf{x}\cdot T+M$ as a block. Each cell of $\mathbf{x}\cdot T+M$ contains a copy of $\mathbf{x}$.

Since we want to carry out the error correction with the majority function, each cell in $\mathfrak{A}_2$ has to know to which block it belongs. This information is stored in the qualifier of the cell by G. The set of states and the qualifier of $\mathfrak{A}_2$ are defined by $A_2=A_1$ and $Q_2=Q_1\times\mathrm{FP}$, where $\mathrm{FP}=\{1,2,3\}$ is a so-called fingerprint. The qualifier of a cell $\mathbf{x}\in\mathbf{Z}^2$ in $\mathfrak{A}_2$ is delivered by $\tilde{q}(\mathbf{x})=(q_1(\mathbf{x}),\mathrm{fp})$.

Now we can define the functions G and H.

1) For any $(c,q)\in\bar{C}_1$ and for any $\mathbf{x}\in\mathbf{Z}^2$

$$G(c(\mathbf{y}),q(\mathbf{y}))=(c(\mathbf{x}),(q(\mathbf{x}),\mathrm{fp})),$$

where $\mathbf{y}\in(\mathbf{x}\cdot T+M)$ and $\mathrm{fp}=1+(y_1 \text{ modulo } 3)$.

2) Function H is locally defined by the triple (M, T, m), where m denotes the majority function. If for any total configuration $(c, \tilde{q})$ of $\mathfrak{A}_2$ there exists a total configuration (c', q') of $\mathfrak{A}_1$ with $c'(\mathbf{x})=m(c(\mathbf{x}\cdot T+M))$ and $q'(\mathbf{x})=m(q_1(\mathbf{x}\cdot T+M))$ for all $\mathbf{x}\in\mathbf{Z}^2$ then $H(c'(\mathbf{y}), q'(\mathbf{y}))=(c'(\mathbf{x}), (q'(\mathbf{x}), \mathrm{fp}))$, where $\mathbf{y}\in(\mathbf{x}\cdot T+M)$ and $\mathrm{fp}=1+(y_1 \text{ modulo } 3)$.

Next we must define the neighbourhood template N_2 of $\mathfrak{A}_2$. Since each cell of $\mathfrak{A}_2$ must have sufficient state information within its neighbourhood to calculate its next state correctly we choose N_2 as the 21-element template sketched in Fig. 3. The hatched cell is the origin of the neighbourhood template.

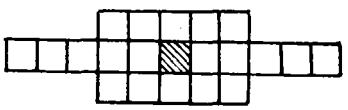

Fig. 3
Neighbourhood template $N_2=(H_1\cdot T+M)-M$

It is easy to see that every cell of the central block of $\mathfrak{A}_2$ in Fig. 2 disposes of the whole information of the 'extended von Neumann neighbourhood template' $H_1\cdot T+M$. We choose the set $K=N_2$, because it must be guaranted that the majority function has at least two correct values. The real L-processor $\mathfrak{A}_2$ may be constructed in such a manner that it has exactly the same number of local transition functions as the L-processor $\mathfrak{A}_1$, i.e. $l:=k$. For any $\mathbf{x}\in\mathbf{Z}^2$ the local transition functions $f_2^{(j)}$ $(j=1, \ldots, k)$ are defined by:

$$(c'(\mathbf{x}), (q_1'(\mathbf{x}), \mathrm{fp})) = f_2^{(j)}(c(\mathbf{x}+N_2), (q_1(\mathbf{x}), \mathrm{fp})),$$

where $c'(\mathbf{x})$ and $q_1'(\mathbf{x})$ are defined by:

$$\begin{aligned}(c'(\mathbf{x}), q_1'(\mathbf{x})) = f_1^{(j)}\big(&m(c(\mathbf{x}-(\mathrm{fp}-1, 0)), c(\mathbf{x}-(\mathrm{fp}-2, 0)), c(\mathbf{x}-(\mathrm{fp}-3, 0)))\\ &m(c(\mathbf{x}-(\mathrm{fp}, 0)), c(\mathbf{x}-(\mathrm{fp}+1, 0)), c(\mathbf{x}-(\mathrm{fp}+2, 0)))\\ &m(c(\mathbf{x}-(\mathrm{fp}-1, -1)), c(\mathbf{x}-(\mathrm{fp}-2, -1)), c(\mathbf{x}-(\mathrm{fp}-3, -1)))\\ &m(c(\mathbf{x}-(\mathrm{fp}-4, 0)), c(\mathbf{x}-(\mathrm{fp}-5, 0)), c(\mathbf{x}-(\mathrm{fp}-6, 0)))\\ &m(c(\mathbf{x}-(\mathrm{fp}-1, 1)), c(\mathbf{x}-(\mathrm{fp}-2, 1)), c(\mathbf{x}-(\mathrm{fp}-3, 1))), q_1(\mathbf{x})\big).\end{aligned}$$

It is possible to change from $f_2^{(j)}$ to $f_1^{(j)}$, because the template $\mathbf{x}+N_2=\mathbf{x}+K$ contains enough information. The above constructed real L-processor $\mathfrak{A}_2$ simulates the L-processor $\mathfrak{A}_1$ in realtime.

Theorem 1. If we choose $\mathfrak{A}_2$ and K as defined above, then $\mathfrak{A}_2$ corrects any K-separated error.

Proof. The proof is delivered by considering all possibilities of single errors.

1) The state of a cell is incorrect.

In consequence of the K-se condition the states of all other cells of the extended von Neumann neighbourhood template (see Fig. 2) are correct, i.e. the majority function is able to correct the error.

2) The qualifier of a cell is incorrect.

Now the cell works with an incorrect state and an incorrect local transition function in the next step in general. Subsequently the cell always announces an

incorrect state in general, i.e. we may consider the cell as permanently broken down. In consequence of the K-se condition all other cells of the extended von Neumann neighbourhood template must work correctly so that they announce correct states in the next step, i.e. the error may be corrected (see 1).

3) The majority function of a cell works incorrectly.

Since the local transition function $f_1^{(j)}$ works with incorrect arguments, in general the cell announces an incorrect state and works with incorrect local transition functions in future. In consequence of the K-se condition all other cells of the extended von Neumann neighbourhood template must work correctly so that 1) or 2) hold in the next step. □

4. Error correction in 5-slow with von Neumann neighbourhood template

Now we present a real L-processor $\mathfrak{A}_2=(A_2\times Q_2, 2, H_1, \{f_2^{(1)}, \ldots, f_2^{(l)}\})$ which simulates the L-processor $\mathfrak{A}_1=(A_1\times Q_1, 2, H_1, \{f_1^{(1)}, \ldots, f_1^{(k)}\})$ in 5-slow, i.e. $k_2=5\cdot k_1$ (see Definition 2.9). We choose the transformation matrix T and the coding unit M as above. In consequence of the von Neumann neighbourhood template the L-processor $\mathfrak{A}_2$ works in two different steps. First the information of the extended von Neumann neighbourhood template must be compressed into the H_1-neighbourhood template of each cell of $\mathfrak{A}_2$ and second the cells have to change their state. Since every cell of the central block of $\mathfrak{A}_2$ in Fig. 2 must have access to the whole information of the extended von Neumann neighbourhood template to change its state, it is necessary to compress four times. Therefore the set of states of $\mathfrak{A}_2$ is given by $A_2=A_1^9$ and the set of qualifiers by $Q_2=Q_1\times\mathrm{FP}\times\mathrm{MZ}$ where $\mathrm{FP}=\{1, 2, 3\}$ is a fingerprint as above and MZ a modulo-5-counter. Fig. 4 shows the state- and qualifier-register of a cell.

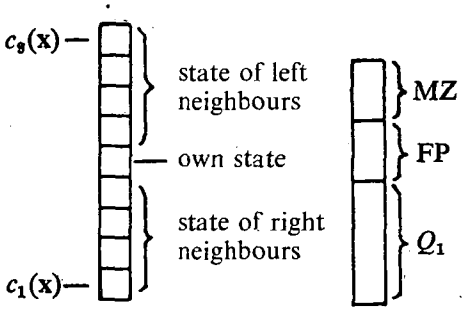

Fig. 4
State- and qualifier-register of a cell

The state of a cell $\mathbf{x}\in\mathbf{Z}^2$ in $\mathfrak{A}_2$ is delivered by $\tilde{c}(\mathbf{x})=(c_9(\mathbf{x}), \ldots, c_1(\mathbf{x}))$ and the qualifier by $\tilde{q}(\mathbf{x})=(q_1(\mathbf{x}), \mathrm{fp}, \mathrm{mz})$.

Now we can define the functions G and H.

1) For any $(c, q)\in\bar{C}_1$ and for any $\mathbf{x}\in\mathbf{Z}^2$

$$G(c(\mathbf{y}), q(\mathbf{y})) = ((-, -, -, -, c(\mathbf{x}), -, -, -, -), (q(\mathbf{x}), \mathrm{fp}, \mathrm{mz})),$$

where $\mathbf{y}\in(\mathbf{x}\cdot T+M)$, $\mathrm{fp}=1+(y_1 \text{ modulo } 3)$ and '$-$' denotes any state. Besides mz will be chosen such that it specifies that no information has arrived.

2) Again function H is locally defined by the triple (M, T, m). If for any total configuration $(\tilde{c}, \tilde{q})$ of $\mathfrak{A}_2$ there exists a total configuration (c', q') of $\mathfrak{A}_1$ with $c'(\mathbf{x})=m(c_5(\mathbf{x}\cdot T+M))$ and $q'(\mathbf{x})=m(q_1(\mathbf{x}\cdot T+M))$ for all $\mathbf{x}\in\mathbf{Z}^2$ then

$$H(c'(\mathbf{y}), q'(\mathbf{y})) = ((-, -, -, -, c'(\mathbf{x}), -, -, -, -), (q'(\mathbf{x}), \mathrm{fp}, \mathrm{mz})),$$

where $\mathbf{y}\in(\mathbf{x}\cdot T+M)$, $\mathrm{fp}=1+(y_1 \text{ modulo } 3)$, '$-$' denotes an arbitrary state and mz will be chosen such that it specifies that no information has arrived.

Before a cell $\mathbf{x}\in\mathbf{Z}^2$ in $\mathfrak{A}_2$ can change its state, the information of the extended von Neumann neighbourhood template must be compressed. Fig. 5 demonstrates the cells from which x collects its information during the compression.

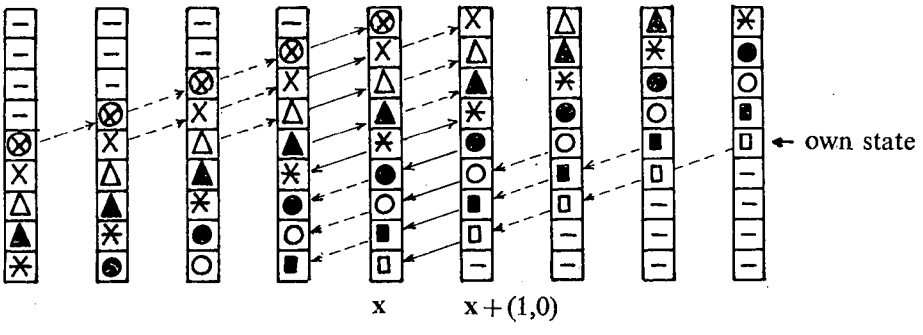

Fig. 5
Compression of information. '$-$' denotes an arbitrary state

Flow of compression. Each cell sends its own state to its left and right neighbours, the states of its left neighbours only to its right one and the states of its right neighbours only to its left one (see cell x) as shown by the arrows in Fig. 5. At the beginning each cell disposes only of its own state. After one cycle a cell disposes of its own state, the state of its left neighbour and the state of its right neighbor in its state register. The remaining six components of the state register contain arbitrary states. After four cycles each cell disposes in its state register of its own state, the states of its four left neighbours and its four right neighbours in this way.

This is the time where each cell in a block of the extended von Neumann neighbourhood template is provided with sufficient information to compute its next state. The compression is realized by hardware or by the following function.

Let $\mathbf{x}, \mathbf{y}, \mathbf{z}\in\mathbf{Z}^2$, $\mathbf{y}=\mathbf{x}-(1, 0)$ and $\mathbf{z}=\mathbf{x}+(1, 0)$. Then a function $h: A_2^5\times Q_2'\to A_2\times Q_2'$ is defined by

$$h(c(\mathbf{x}+H_1), (q, \mathrm{fp}, \mathrm{mz})) = \begin{cases} ((c_8(\mathbf{y}), \ldots, c_5(\mathbf{y}), c_5(\mathbf{x}), c_5(\mathbf{z}), \ldots, c_2(\mathbf{z})), \\ (q, \mathrm{fp}, \mathrm{mz}\oplus 1)), \quad \text{if} \quad 0 < \mathrm{mz}\oplus 1 < 4 \\ ((c_8(\mathbf{y}), \ldots, c_5(\mathbf{y}), c_5(\mathbf{x}), c_5(\mathbf{z}), \ldots, c_2(\mathbf{z})), \\ (q', \mathrm{fp}, \mathrm{mz}\oplus 1)), \quad \text{if} \quad \mathrm{mz}\oplus 1 = 4 \end{cases}$$

$\oplus$ denotes the add-operation modulo 5.

$Q_2'=(Q_1\times\{0,1\})\times\mathrm{FP}\times\mathrm{MZ}$, i.e. the qualifier of $\mathfrak{A}_1$ is extended by one bit indicating whether $\mathfrak{A}_2$ is in a phase of compression or transformation, i.e. q' differs from q only in this bit. If $\mathrm{mz}\oplus 1=0$ holds, $\mathfrak{A}_2$ is in a phase of transformation. If we use the function h for compression the functions G and H must be modified adequate to Q_2'. In this case we define $f_2^{(k+1)}:=h$, i.e. $l:=k+1$. In the following we consider only cells of the central block of the extended von Neumann neighbourhood template and their direct neighbours to define the local transition functions $f_2^{(j)}$ $(j=1,\ldots,k)$. After compression these cells contain the following relevant states, see Fig. 6.

Now we can define the local transition functions $f_2^{(j)}$ $(j=1,\ldots,k)$. Let $\mathbf{x}\in\mathbf{Z}^2$

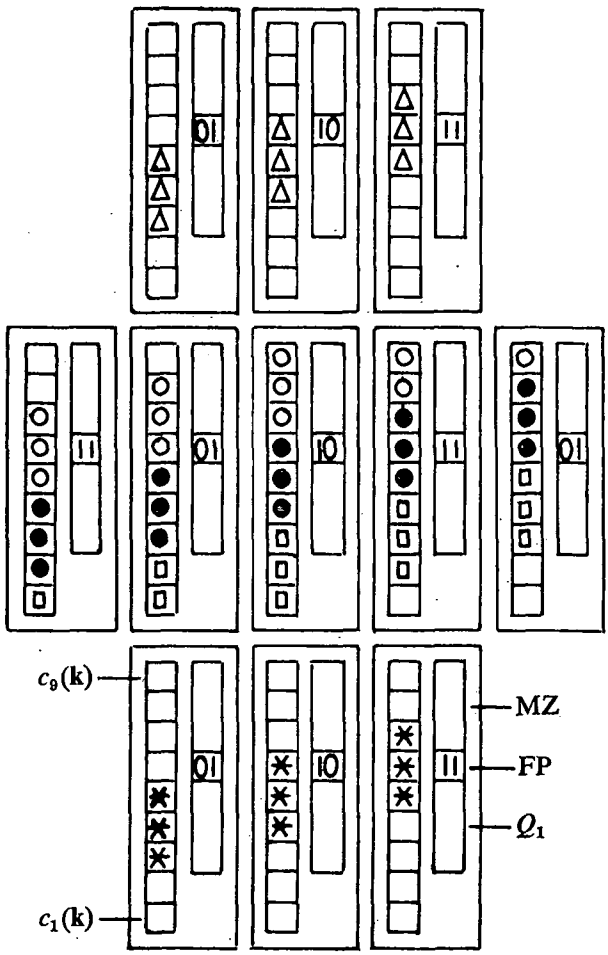

Fig. 6
Relevant states for the central block of the extended von Neumann neighbourhood template after compression

and $\mathbf{a}_1=(0,0)$, $\mathbf{a}_2=(-1,0)$, $\mathbf{a}_3=(0,1)$, $\mathbf{a}_4=(1,0)$, $\mathbf{a}_5=(0,-1)$ the five elements of the von Neumann neighbourhood template.

$$\begin{aligned}(c'(\mathbf{x}), q'(\mathbf{x})) &= f_2^{(j)}(c(\mathbf{x}+H_1), (q_1(\mathbf{x}), \mathrm{fp}, \mathrm{mz})) \quad \text{or}\\ &\quad f_2^{(j)}(c(\mathbf{x}+H_1), (\{q_1(\mathbf{x})\}\times\{0,1\}, \mathrm{fp}, \mathrm{mz}))\\ &= (-,-,-,-, c_5'(\mathbf{x}), -,-,-,-), (q_1'(\mathbf{x}), \mathrm{fp}, \mathrm{mz}\oplus 1)) \quad \text{or}\\ &\quad (-,-,-,-, c_5'(\mathbf{x}), -,-,-,-), (\{q_1'(\mathbf{x})\}\times\overline{\{0,1\}}, \mathrm{fp}, \mathrm{mz}\oplus 1)),\end{aligned}$$

where '$-$' denotes any state and '$\oplus$' the add-operation modulo 5 again. $\overline{\{0,1\}}$ means that $f_2^{(j)}$ forms the complement of $\{0,1\}$, if the compression is realized by software. $c_5'(\mathbf{x})$ and $q_1'(\mathbf{x})$ are defined by

$$\begin{aligned}(c_5'(\mathbf{x}), q_1'(\mathbf{x})) = f_1^{(j)}(&m(c_{\mathrm{fp}+2}(\mathbf{x}+\mathbf{a}_1), c_{\mathrm{fp}+3}(\mathbf{x}+\mathbf{a}_1), c_{\mathrm{fp}+4}(\mathbf{x}+\mathbf{a}_1)),\\ &m(c_{\mathrm{fp}+4}(\mathbf{x}+\mathbf{a}_2), c_{\mathrm{fp}+5}(\mathbf{x}+\mathbf{a}_2), c_{\mathrm{fp}+6}(\mathbf{x}+\mathbf{a}_2)),\\ &m(c_{\mathrm{fp}+2}(\mathbf{x}+\mathbf{a}_3), c_{\mathrm{fp}+3}(\mathbf{x}+\mathbf{a}_3), c_{\mathrm{fp}+4}(\mathbf{x}+\mathbf{a}_3)),\\ &m(c_{\mathrm{fp}}(\mathbf{x}+\mathbf{a}_4), c_{\mathrm{fp}+1}(\mathbf{x}+\mathbf{a}_4), c_{\mathrm{fp}+2}(\mathbf{x}+\mathbf{a}_4)),\\ &m(c_{\mathrm{fp}+2}(\mathbf{x}+\mathbf{a}_5), c_{\mathrm{fp}+3}(\mathbf{x}+\mathbf{a}_5), c_{\mathrm{fp}+4}(\mathbf{x}+\mathbf{a}_5)), q_1(\mathbf{x})).\end{aligned}$$

The definition is similar to that in section 3. First we apply the majority function to each component block. These results and the first part of the qualifier of the cell are the values for the local transition function $f_1^{(j)}$, which delivers the new state of component c_5 and the new first part of the qualifier. Again it is possible to change from $f_2^{(j)}$ to $f_1^{(j)}$, because the template $\mathbf{x}+H_1$ contains enough information.

Up to now we did not say anything about the K-se condition which must be modified, if the above real L-processor should correct errors. The modification is necessary because additionally to the error correction there are two further problems.

1) We have a phase of compression.
2) An error correction is not possible concerning the qualifier of a cell.

In the first solution we have permitted that single cells may break down totally. Now such cells will interrupt the information-flow during the compression. Because of that the four right and left neighbours of these cells would hold up to four successive incorrect states in their components, i.e. an error correction is not possible.

Fig. 7 shows the information-flow through one cell during compression. '|__|' denotes a block. The local transition functions of cells marked by 'X' deliver an incorrect state, because they work with at least one faulty argument. 1) shows the faulty cells if the left cell of a block is broken down, 2) shows these cells if the central cell of a block is broken down, and 3) shows these cells if the right cell of a block is broken down.

Therefore the first modification of the K-se condition consists in that only single components of a cell may be incorrect. It is easy to see that the errors must have a distance of three, i.e. if c_1 is incorrect, c_2 and c_3 must be correct. Since the errors of a cell will be carried off during the compression, the condition that only one cell of each area $\mathbf{x}+K$ ($\mathbf{x}\in\mathbf{Z}^2$) may be incorrect, cannot maintained. The second modification of the K-se condition runs as follows. If a cell $\mathbf{y}\in(\mathbf{x}+K)$ is incorrect all other cells of that region have to work correctly, i.e. they are only

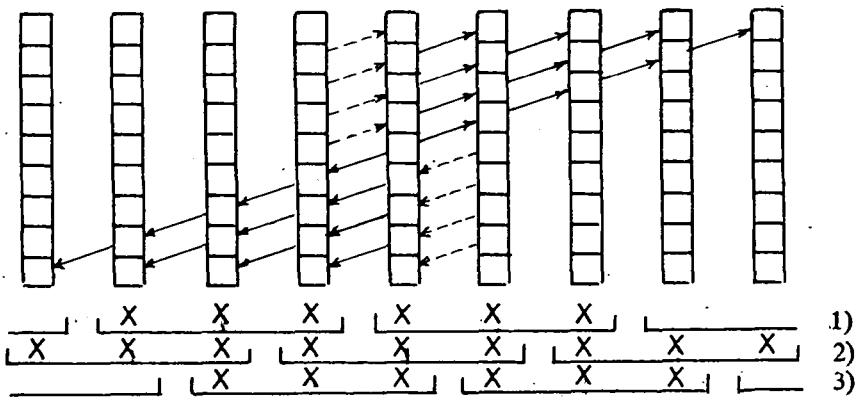

Fig. 7
Information-flow through one cell during compression

allowed to take over the errors of cell y. If the qualifier of a cell is incorrect, in general that cell must be considered as totally broken down. Therefore we must demand that no errors occur in the qualifier of a cell. Only under this new stronger *K*-se condition the above *L*-processor may correct errors.

Theorem 2. Let K be the 21-element template of Fig. 3. Then the above *L*-processor $\mathfrak{A}_2$ corrects any errors if the stronger *K*-se condition holds.

Since the proof is similar to that of Theorem 1, we do not carry it out here again.

5. Conclusions and outviews

The primary objective of this paper has been to present suggestions for fault tolerant *L*-processors. The first solution is characterized by the following features:

- $A_2=A_1$ and $Q_2=Q_1\times\mathrm{FP}$, i.e. the set of external states is not increased,
- the real *L*-processor $\mathfrak{A}_2$ has exactly the same number of local transition functions as the simulated *L*-processor $\mathfrak{A}_1$,
- $\mathfrak{A}_2$ simulates $\mathfrak{A}_1$ in realtime,
- $\mathfrak{A}_2$ has a relatively large neighbourhood template of 21 elements.

The features of the second solution are:

- $A_2=A_1^9$ and $Q_2=Q_1\times\mathrm{FP}\times\mathrm{MZ}$, i.e. the set of external states of $\mathfrak{A}_2$ is enlarged, too,
- the real *L*-processor $\mathfrak{A}_2$ has either the same number of local transition functions as the simulated *L*-processor $\mathfrak{A}_1$ (compression per hardware) or at most one local transition function more than $\mathfrak{A}_1$ (compression per software),
- $\mathfrak{A}_2$ simulates $\mathfrak{A}_1$ in 5-slow,
- $N_2=H_1$,
- very strong *K*-se condition.

As shown above the second solution is not practicable for error correction, because the simulating *L*-processor must have nearly no errors. Therefore a fault

tolerant L-processor with von Neumann neighbourhood template has to use some different coding.

It is easy to modify the above solutions so that the resulting L-processors may detect errors.

Acknowledgements. I am deeply grateful to my dear colleague Mr. W. O. Höllerer for some suggestions in formal describing L-processors. I would also like to thank Prof. R. Vollmar for many stimulating discussions.

Abstract

This paper treats the problem of designing L-processors with error correction capabilities. First such notions as configuration, real L-processor, K-separated error, and simulation are defined. Then a fault tolerant real L-processor is introduced which simulates a given L-processor in real time. Next a fault tolerant real L-processor with von Neumann neighbourhood index is presented which simulates a given L-processor in 5-slow. In both cases the original L-processors have a von Neumann template. Since an L-processor may be understood as a cellular space with Mealy-type cells, this approach may be considered as a generalization of the work of Nishio and Kobuchi.

LEHRSTUHL D FÜR INFORMATIK
TECHNISCHE UNIVERSITÄT BRAUNSCHWEIG
GAUSSTR. 11
D—3300 BRAUNSCHWEIG

References

[GROS 79] GROSS, S. R., Fehlererkennung und -korrektur in zellularen Automaten, Diplomarbeit, Braunschweig, 1979.

[HANO 75] HARAO, M., S. NOGUCHI, Fault tolerant cellular automata, *J. Comput. System Sci,*. v. 11, 1975, pp. 171—185.

[LEGE 76] LEGENDI, T., Cellprocessors in computer architecture, *Computational Linguistics and Computer Languages,* v. 11, 1976, pp. 147—167.

[LEGE 77] LEGENDI, T., Programming of cellular processors, *Informatik-Berichte*, Nr. 7703, Braunschweig, 1977, pp. 53—66.

[NIKO 75] NISHIO, H., Y. KOBUCHI, Fault tolerant cellular spaces, *J. Comput. System Sci.*, v. 11, 1975, pp. 150—170.

[VOLL 79] VOLLMAR, R., *Algorithmen in Zellularautomaten,* Teubner Verlag, Stuttgart, Bd. 48, 1979.

[WRIG 76] WRIGHT, L. E., Cellular automata with nonworking cells, Ph. D. diss., Rensselaer Polytechnic Institute, Troy, New York, 1976.

(Received April 10, 1981)