[12] B. Lin and A. Newton, "Synthesis of multiple-level logic from symbolic high-level description languages," in *Proc. VLSI 89*.
[13] J. Nestor, B. Soudan, and Z. Mayet, "MIES—A microarchitecture design tool," *Proc. 22nd Int. Workshop on Microprogramming and Microarchitecture*, pp. 217–222, 1989.
[14] G. Zimmermann, "The MIMOLA design system: A computer aided digital processor design method,", *Proc. 16th Design Automation Conf.*, pp. 65–72, June 1979.
[15] Apple Computer, Inc., *HyperCard Script Language Guide: The HyperTalk Language*. Reading, MA: Addison-Wesley, 1988.
[16] J. Hennessey and D. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufman, 1990.
[17] S. Stritter and N. Tredennick, "Microprogrammed implementation of a single chip microprocessor," in *Proc. 11th Ann. Microprogramming Workshop*, pp. 8–16, Nov. 1978.

On the Optimal Reconfiguration of Multipipeline Arrays in the Presence of Faulty Processing and Switching Elements

H. Lin, F. Lombardi, and M. Lu

Abstract—**This paper deals with the reconfiguration of multipipeline arrays in the presence of both faulty processing elements (PE's) and switching elements (SE's). Different fault models are used for the PE's and SE's: a PE can be either fault free or faulty; a SE is modeled using a novel functional approach which relates its switching capabilities to its status. This permits a PE to retain a partial functionality in the presence of a fault. An appropriate transformation of the multipipeline array reconfiguration problem to a maximum flow problem is then presented. The conditions under which this transformation is possible, are fully analyzed. A reconfiguration algorithm based on the Maximum Flow Algorithm, is presented; the proposed algorithm is optimal as the number of reconfigured pipelines is maximized.**

I. Introduction

Arrays of processing elements manufactured with technologies such as VLSI and WSI are widely used for establishing multipipelines to perform vector operations in supercomputers [1], [5], [13]. Also, the number of processing elements in each pipeline can be adjusted depending on the computational requirements by using programmable switches [4], [13]. As the number (and density) of processing elements (PE's) and switching elements (SE's) in these array chips is extremely high [1], the likelihood of faults must be considered. It is prohibitive to replace a VLSI chip or wafer because of the presence of faulty PE's and/or SE's [6]. Therefore, fault-tolerance by reconfiguration is necessary to circumvent faults and to maximize the functionality of the VLSI chip or wafer [2].

In [4], a heuristic algorithm for reconfiguring multipipelines in the presence of both faulty PE's and SE's has been proposed. The inclusion of faulty SE's distinguishes this work from that of earlier researchers [5], [7], [8]. The fault model proposed in [4] is strictly based on a physical characterization of the SE's in the multipipeline array and reconfiguration consists of a heuristic algorithm in which all SE's are visited from left to right on each row, starting with the first row. SE's configurations are based on the status of adjacent PE's and requests from switches on the preceding row. Harvesting of reconfigurable pipelines has been dealt in [7], [9] using a probabilistic model and a routing approach, respectively.

Manuscript received June 26, 1992; revised October 14, 1992. This work was supported in part by a grant from the National Science Foundation.

H. Lin and M. Lu are with the Department of Electrical Engineering, Texas A&M University, College Station TX 77843-3112.

F. Lombardi is with the Department of Computer Science, Texas A&M University, College Station, TX 77843-3112.

IEEE Log Number 9206237.

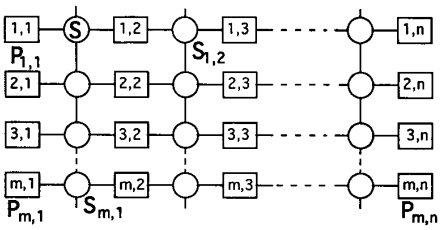

Fig. 1. Reconfigurable multipipeline array.

In this paper, the Multipipeline Array Reconfiguration (MAR) problem, is analyzed using a different switch fault model from [4]. In the proposed SE model, a functional characterization which relates the switching capability of the SE to its status, is used. As the status of a SE is based on its switching capabilities, the MAR problem is then transformed into a maximum flow problem [11]. The conditions by which the transformation is possible, are analyzed and a reconfiguration algorithm which generates the maximum number of pipelines, is presented.

II. Preliminaries

The multipipeline array structure which is analyzed in this paper, is shown in Fig. 1. It has been shown that this arrangement provides low hardware overhead [6] and is amenable to high density technologies, such as WSI [8]. The multipipeline array M consists of n stages (or columns), each stage consists of m PE's. The PE in the leftmost jth stage (hereafter, referred to as the jth stage) and in the topmost ith row (hereafter, referred to as the ith row) is denoted by $P_{i,j}$. PE's in the same stage are identical, while PE's in different stages perform different functions (i.e., they are not identical). PE's in the jth stage are connected to PE's in the $(j+1)$th stage through programmable SE's. More specifically, processor $P_{i,j}$ is connected to processor $P_{i,j+1}$ through $S_{i,j}$, i.e., adjacent stages of PE's are separated by a stage of SE's. In the MAR problem, the maximum number of pipelines (each of length n) must be generated using a PE from each stage; an appropriate arrangement for switching must be provided. In this paper, switching is performed by an interconnection network made of SE's, each $S_{i,j}$ is connected through a link with two other SE's (given by $S_{i-1,j}$ and $S_{i+1,j}$).

The SE can be modeled as a four-terminal component (as corresponding to the most commonly used switch configuration [2]); for $S_{i,j}$ the terminals are given by E, S, N and W, respectively, and they are arranged to connect PE's as follows: W is adjacent to $P_{i,j}$, N is adjacent to S of $S_{i-1,j}$, E is adjacent to $P_{i,j+1}$ and S is adjacent to N of $S_{i+1,j}$. Each switch has four modes (as shown in Fig. 2); these modes are identified by an unique switching code. The codes for the modes (as shown in Fig. 2) are 00, 01, 10 and 11, respectively. The following assumptions are valid in this paper. (1) Faults occur in both SE's and PE's. (2) The reconfiguration process is controlled by a host. (3) The proposed reconfiguration approach is applicable to both run time and production time.

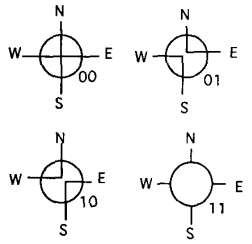

Fig. 2. Four modes of a fault-free switch.

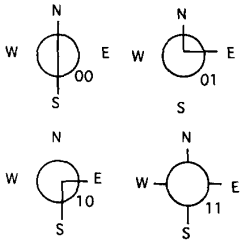

Fig. 3. Four modes of a faulty switch with bad terminal W.

III. The Fault Model

In the multipipeline array, both PE's and SE's can have faults. A functional fault model is used in the proposed analysis; 1) PE: A PE is either *faulty* or *fault-free*; a PE is referred to as *usable* only when it is fault free; 2) SE: A SE is either faulty or fault free; an external terminal of a switch is either *good* or *bad*; a SE is fault free only when all of four terminals are good. A bad terminal only damages the connections between itself and other terminals, without resulting in any effect on the connections between any other two terminals. Hence, a faulty SE in which, for instance, only terminal W is bad, is still able to realize the configurations shown in Fig. 3.

The fault model of the SE is different from the one used in [4]; the fault model of [4] consists of the following fault types: stuck-at or stuck-open faults on a data link, stuck-at faults on a control line and bridging faults between data links, faults in SE's and faults in PE's as long as each SE has a single fault. The proposed fault model consists of a functional characterization and is applicable to the widely used SE configuration of [6], [10]. This SE consists of a switch control register and a control circuit. Using a functional fault model, control stuck-at faults in SE's, stuck-at faults in data paths, faults on the switch control and faults in the PE bypass control, as well as link faults (of the type short and open) can be collapsed by testing only the function of the SE that leads to its adjacent PE or SE, thus ignoring any fault in a SE which does not manifest its malfunction, i.e., the proper behavior of a SE is directly related to its external terminals. Note that link faults are not directly addressed in this paper; however, by using the proposed functional model, they can be easily accommodated by collapsing them to the external terminals of the SE pair to which they connect.

In this paper, it is assumed that a procedure has been executed to diagnose the status of all PE's and SE's (such that the faulty PE's and faulty SE's with bad terminals have been identified), then the MAR problem consists of choosing a mode for each SE in such a way that the number of generated pipelines (each of length n) is maximized; this reconfiguration scenario is related to a configuration of the multipipeline array and is referred to as *optimal* [8].

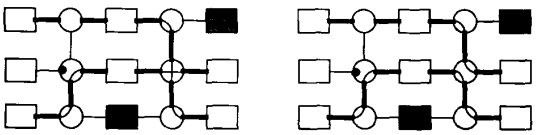

Fig. 4. An example of a crossover. (a) Contains a crossover. (b) Does not contain a crossover.

IV. Basic Properties

Several properties of the multipipeline array under the fault model of the previous section can be easily observed.

Property 1: If terminal W (or E) of $S_{i,j}$ is bad, then $P_{i,j}$ (or $P_{i,j+1}$) is unusable.

Property 2: If terminal N or S of $S_{i,j}$ is bad, then it is impossible to connect $P_{i_1,j}$ to $P_{i_2,j+1}$ where $i_1 < i < i_2$ or $i_2 < i < i_1$.

Property 3: There always exist an optimal configuration of the multipipeline array in which no two pipelines cross each other (this case is generally referred to as a crossover), that is, let $P_{q(i),j}$ and $P_{r(i),j+1}$ for $1 \leq i \leq p$, where $q(1) < q(2) < \ldots < q(p)$ and $r(1) < r(2) < \ldots < r(p)$, be the PE's in stage j and $j+1$ of configuration C_1 with p good pipelines. Then, there exits a configuration C_2 with p good pipelines such that $P_{q(i),j}$ and $P_{r(i),j+1}$ are in the same pipeline, for $1 \leq i \leq p$. An example of a crossover is shown in Fig. 4. A shaded box denote a faulty PE while a bad terminal in a SE is denoted by a shaded dot.

V. The Transformation

The proposed approach (and its optimality) is based on transforming the MAR problem into the problem of finding the maximum flow on a flow network, commonly referred to as the maximum flow problem [11]. A *flow network* $N = (s, t, V, A, c)$ consists of a digraph (V, A), together with a *source* $s \in V$ with an in-degree of 0, a *sink* $t \in V$ with an out-degree of 0, and with a *capacity* $c(u, v) \in Z^+$ for each $(u, v) \in A$. A *flow* f is a vector, such that $0 \leq f(u, v) \leq c(u, v)$ for all $(u, v) \in A$; $\sum_{(u,v)\in A} f(u, v) = \sum_{(v,u)\in A} f(v, u)$ for all $v \in V - \{s, t\}$.

The *value* of f, denoted by $|f|$, is defined as the sum of the flows provided by s, that is $|f| = \sum_{v\in V} f(s, v)$. The maximum flow problem is to find the flow f which results in a maximum $|f|$. Given a multipipeline array and all the diagnostic information about the status of the PE's and SE's, a flow network can be constructed in such a way that each node of the network, except the source and the sink, represents either a PE or a SE and the flow in an arc is either one or zero; therefore, a path from s to t with one unit flow on it represents a "good" pipeline and the flow of the network represents the reconfiguration of the multipipeline array, in particular, the maximum flow represents the optimal reconfiguration. In the rest of this section, the conditions and properties by which this transform is possible, will be analyzed in detail.

Formally, given a multipipeline array with m rows and n stages, (as shown in Fig. 1), each PE and SE is treated as a node in the network; for a node representing a SE, the term "terminal" is used in the context of the proposed graph model to identify the in- or out-edge to a node corresponding the SE terminal (as defined in the previous section). The terminals (and the arcs connected to them) are labeled in the graph as shown in Fig. 5. Two additional nodes s and t are added as source and sink, respectively. The notation $P_{i,j}$ and $S_{i,j}$ are used hereafter as the label assigned to the nodes corresponding to processor $P_{i,j}$ and switching element $S_{i,j}$. Nodes are connected by arcs as follows (see Fig. 5 for an example):

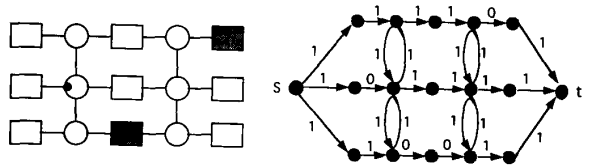

Fig. 5. A 3×3 multipipeline array with faults and its corresponding network. Terminal W of the switch in the first stage is bad. Faulty PE's denoted by grey boxes; the dot in the switch indicates the bad terminal W.

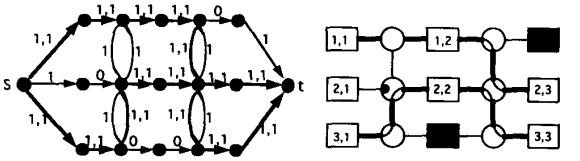

Fig. 6. Network with maximum flow and the corrsponding optimal configuration.

1) s is connected to the m nodes $P_{i,1}$, $1 \leq i \leq m$, by m arcs starting from s and ending at $P_{1,j}$'s; the m nodes $P_{i,n}$, for $1 \leq n \leq m$, are connected to the sink t by m arcs starting from the nodes and ending at t;
2) For $1 \leq i \leq m$ and $1 \leq j \leq n-1$, $P_{i,j}$ is connected to $S_{i,j}$ by arc $(P_{i,j}, S_{i,j})$ and node $S_{i,j}$ is connected to $P_{i,j+1}$ by arc $(S_{i,j}, P_{i,j+1})$;
3) For $1 \leq i \leq m-1$ and $1 \leq j \leq n-1$, there are two arcs between $S_{i,j}$ and $S_{i,j+1}$: one is from $S_{i,j}$ to $S_{i+1,j}$, while the other from $S_{i+1,j}$ to $S_{i,j}$;

The capacity of an arc depends on whether the corresponding PE or SE is faulty or fault-free. The capacity of arc $(P_{i,j}, S_{i,j})$ (denoted as $c(P_{i,j}, S_{i,j})$) is designed as one unit if both $P_{i,j}$ is fault free and the terminal W of $S_{i,j}$ is good; otherwise, it is designed as zero. The capacity of arc $(S_{i,j}, P_{i,j+1})$ is designed as one unit if both the terminal E of $S_{i,j}$ is good and $P_{i,j+1}$ is fault-free; otherwise, it is designed as zero. The capacities of the two arcs between $S_{i,j}$ and $S_{i+1,j}$ are either both one or zero; in particular, they are one if the terminal S of $S_{i,j}$ and the terminal N of $S_{i+1,j}$ are both good. The arcs which start from s or end at t, have capacity of 1.

Note that the set of arcs in the digraph (except those arcs which are connected to s and t) correspond to the possible connections as allowed by the provided interconnection network and the status of the PE's and SE's. The relationship between the multipipeline array and its corresponding flow network is first examined by proving two lemmas (all proofs are omitted due to lack of space; the interested reader should refer to [3]).

Lemma 1: Let C be the configuration of the multipipeline array M which has p good pipelines and no two pipelines are crossing (by Property 3), and let F be the corresponding network described above, then there exit a flow f on F, with $|f| = p$.

Lemma 2: Let f be the flow in F corresponding to M, with $|f| = p$, then there exits a configuration C with p good pipelines with no crossover.

Fig. 6 shows an example of a reconfigurable multipipeline array. The main result of this paper follows directly from Lemma 1 and Lemma 2.

Theorem 1: The optimal configuration C of a multipipeline array M contains p good pipelines if and only if the corresponding network F has a maximum flow of value p.

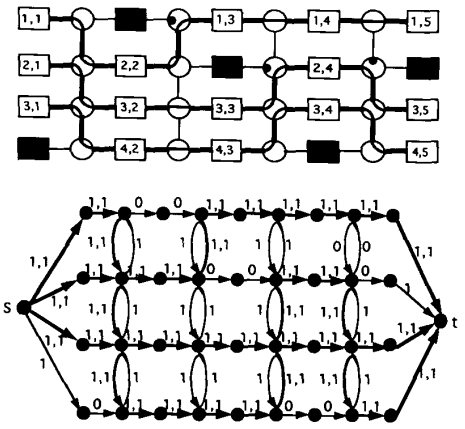

Fig. 7. An example.

VI. The Reconfiguration Algorithm

The algorithm for multipipeline array reconfiguration can be described by three steps. *Input* is the multipipeline array M with the status of all PE's and SE's. *Output* is the switching modes for every SE which result in the optimal configuration of the multipipeline array.

Step 1: construct the network F;
Step 2: use the maximum flow algorithm to find f;
Step 3: assess the switching mode for every SE, according to f.

The procedure for Step 1 is trivial. As for Step 2, there have been many papers [11] dealing with efficient algorithms for the maximum flow problem. It is worth mentioning that the network F, obtained from M according to the proposed transformation, is planar; therefore, a faster algorithm than for the general case is available [12].

The principles for assessing the switching modes and implementing the reconfiguration of the multipipeline array can be derived from Lemma 2, the following is the procedure without proof.

Procedure_step_3:

(1) Let $P_{q(i),j}$ and $P_{r(i),j+1}$ for $1 \leq i \leq p$ and $1 \leq j \leq n-1$ be the nodes with one unit out-flow. For $1 \leq i \leq p$ and $1 \leq j \leq n-1$. 1) If $q(i) = r(i)$, then set switch $S_{r(i),j}$ to mode 00; 2) If $q(i) < r(i)$, then set switch $S_{q(i),j}$, $S_{r(i),j}$ to mode 01 and set switches $S_{k,j}$ to 00, for $q(i)+1 \leq k \leq r(i)-1$; 3) If $q(i) > r(i)$, then set switch $S_{q(i),j}$, $S_{r(i),j}$ to mode 10 and set switches $S_{k,j}$'s to 00, for $r(i)+1 \leq k \leq q(i)-1$;

(2) All remaining SE's are set to mode 11.

The time complexity of the proposed reconfiguration algorithm can be established as follows. Since there are mn PE's in M, the number of nodes and arcs in F is bounded by $O(mn)$. The assignment of a capacity to an arc corresponds to check whether the corresponding PE (or the terminal of a SE) is faulty (or bad) or not. Therefore, Step 1 can be done in $O(mn)$. As the network obtained by the proposed transformation has a unit capacity on an arc, Step 2 takes $O((mn)^{5/3})$ time [11] ,[12]. It is clear that Procedure_step_3 can be done in $O(mn)$ time. Thus it is possible to conclude that the MAR problem can be solved in $O((mn)^{\frac{5}{3}})$ time. Fig. 7 shows an example of a reconfigured multipipeline array (made of 20 PE's) in the presence of 4 faulty PE's, and 3 bad SE's and the flow network as generated using the proposed approach.

TABLE I

r	Y_{pe}	Y_{se}	P_p	P_c	z_0
5	0.90	0.98	6.984	6.148	7.374
5	0.80	0.96	5.276	3.765	5.911
5	0.70	0.94	4.274	2.319	4.674
5	0.60	0.92	3.396	1.446	3.563
10	0.90	0.99	7.139	6.453	7.367
10	0.80	0.98	5.388	4.130	5.926
10	0.70	0.97	4.493	2.651	4.700
50	0.90	0.998	7.133	6.670	7.364
50	0.80	0.996	5.541	4.479	5.930
50	0.70	0.994	4.483	3.021	4.718
infinity	0.90	1.00	7.184	6.731	7.365
infinity	0.80	1.00	5.677	4.548	5.925
infinity	0.70	1.00	4.482	4.548	4.683
infinity	0.60	1.00	3.485	2.129	3.602
infinity	0.50	1.00	2.436	1.562	2.622

VII. Simulation Results

The proposed approach has been simulated extensively and compared with the approach of [4]. A Monte Carlo simulation has been performed on a 10×10 array (i.e., with 100 PE's); hence, the length of each reconfigured pipeline must be 10. Let $r = \frac{1-Y_{pe}}{1-Y_{se}}$, where Y_{se} (Y_{pe}) is the yield of SE's (PE's). Table I shows the simulation results for different values of r; in this table, P_p (P_c) denotes the average number of pipelines reconfigured using the proposed approach (the approach of [4]) using the same proposed functional fault model and z_0 denotes the least number of fault free PE's in any column of the multipipeline array. Note that in Table I $P_p < z_0$: z_0 is the *theoretical upper bound* on the number of reconfigured pipelines *provided all switches were fault free* and SE's between two adjacent stages of PE's *allow every possible connection* between all fault free PE's [4]. This is not always possible using only one column of SE's as in Fig. 1: as an example, consider a 4×5 multipipeline array: let $PE_{3,1}$, $PE_{4,1}$ $PE_{1,2}$ and $PE_{2,2}$ be faulty and all SE's be fault free; hence $z_0 = 2$. However, only one pipeline can be reconfigured because $PE_{2,1}$ can not be connected to $PE_{4,2}$ if $PE_{1,1}$ is also connected to $PE_{3,2}$.

VIII. Conclusions

This paper has presented a reconfiguration algorithm for multipipeline arrays in the presence of both faulty PE's and SE's. This is based on transforming the MAR problem into a maximum flow problem; hence, the proposed algorithm is optimal, i.e., it generates the maximum number of pipelines. It differs from the approach of [4] due to the functional fault model of the switches (the approach of [4] uses a physical stuck-at fault model) and the nature of the algorithm (the proposed approach is optimal, while the approach of [4] utilizes a heuristic condition at a lower execution complexity, thus in Table I $P_p > P_c$).

The following additional conclusions can be drawn. (1) Using the proposed approach, minimum delay reconfiguration can also be accomplished. (2) The MAR problem analyzed in this paper deals with multipipelines made of heterogeneous stages of PE's; this condition is amenable to a transformation into the maximum flow problem. Future research should address optimality also with regard to homogeneous multipipeline arrays of length longer than the number of stages. (3) An optimal algorithm of linear execution complexity is possible; however, this is applicable only to multipipeline arrays with column(s) of switches between two PE stages, i.e., it is not applicable to the general case of rows of PE's separated by rows of SE's [3].

Acknowledgment

The authors wish to acknowledge the constructive comments of the reviewers and, in particular, Reviewer 2 for pointing the existence of an optimal algorithm of linear execution complexity (see comment (3) in the previous section).

References

[1] H. T. Kung, "Why systolic architectures?" *Computer,* vol. 15, no. 1, pp. 37–46, 1982.
[2] M. Chean and J. A. B. Fortes, "A taxonomy of reconfiguration techniques for fault-tolerant processor arrays," *IEEE Computer,* vol. 23, no. 1, pp. 55–69, 1990.
[3] H. Lin, F. Lombardi, and M. Lu, "On the optimal reconfiguration of multipipeline arrays," Internal Rep., Dep. Computer Science, Texas A&M Univ., 1992.
[4] Y. H. Choi, "Reconfigurable multipipelines," in *Int. Conf. on Parallel Processing,* pp. 566–570, 1991.
[5] R. Negrini, M. G. Sami, and R. Stefanelli, "Restructuring and reconfiguring DSP multi-pipeline arrays," in *Proc. MSTS,* Phoenix, AZ, 1987.
[6] R. Negrini, R. Stefanelli, and M. G. Sami, "Fault tolerance technique for array structures used in supercomputing," *IEEE Computer,* vol. 19, no. 2, pp. 78–87, 1986.
[7] F. Lombardi, P. Koo, and D. Sciuto, "A routing algorithm for harvesting multipipeline arrays with small intercell and pipeline delays," in *Proc. ICCAD,* pp. 2–5, 1990.
[8] R. Gupta, A. Zorat, and I. V. Ramakrishna, "A fault tolerant multipipeline architecture," in *Proc. FTCS 16,* pp. 350–355, 1986.
[9] W Shi. and K. Fuchs, "Harvest rate of reconfigurable pipelines," in *Proc. IEEE Int. Workshop on DFT in VLSI Systems,* pp. 93–102, 1991.
[10] Y. H. Choi, D. Fussell, and M. Malek, "Fault diagnosis of switches in wafer scale arrays," in *Proc IEEE ICCAD,* pp. 292–294, 1986.
[11] R. E. Tarjan, "Data structures and network algorithms," in *Proc. CBMS-NSF Reg. Conf. Series Appl. Math.,* vol. 44, 1983.
[12] A. Itai and Y. Shiloach, "Maximum flow in planar networks," *J. SIAM Comp.,* vol. 8, no. 2, pp. 125–150, 1979.
[13] S. Sahni, "Scheduling multipipeline and multiprocessor computers," *IEEE Trans. Computers,* vol. C-33, pp. 637–645, 1984.