A Flexible and General Solution for Reconfiguring Pipeline Computing Systems

Mircea Popa

Abstract: *The pipeline architectural concept ensures great speed for computing systems but its important disadvantages are the lack of flexibility and low fault tolerance. One way for minimizing these disadvantages which ensures optimum power consumption too for a specific application is its reconfiguration. The paper presents a flexible and general solution for reconfiguring pipeline computing systems which allows the reconfiguration at the device level or at the system level. After presenting the pipeline concept and its specific features, the new solution is described. It is based on introducing programmable hardware at every pipeline station which allows tailoring the optimal pipeline architecture for a specific application. The programmability of this hardware is simple and fast. Some examples are used which show how to reconfigure pipeline computing systems with four stations but they can be extended for any number of stations. The new solution is described in 2 variants: one with maximum flexibility but which is more complex too and one simpler but with a reduced flexibility.*

Key words: *pipeline computing systems, partial reconfiguration, total reconfiguration*

INTRODUCTION

The performances computing systems must satisfy are continuously growing. The speed and the power consumption are targets which must be permanently improved. A research and development direction for reaching the mentioned purposes is the pipeline architectural concept. This concept can be implemented at two levels: at the device level because the most programmable devices, such as microprocessors, microcontrollers, DSPs, have a pipeline structure; there are even devices which integrate more than one pipeline in the same chip, for example the Intel Pentium microprocessor family and at the system level, especially in supercomputers.

The great advantage of the pipeline concept is the speed it offers but it has important disadvantages too such as the lack of flexibility, the cost, low fault tolerance and inefficient power consumption. An efficient way for minimizing these disadvantages is the reconfiguration operation. It allows tailoring optimum pipeline architecture for a specific application. The reconfiguration can be done at the beginning of every run session or run time.

This paper presents a flexible and general solution for reconfiguring pipeline computing systems. It is based on introducing programmable hardware at every pipeline station. The programmability process is simple and fast. The new solution can be applied for both device level and system level implementation of the pipeline architectural concept and for both beginning and run time moments of the application. The supplementary hardware means a cost increase but this is minimum because cheap and ordinary hardware is used.

The second section shortly presents the pipeline architectural concept and its main features, the third section outlines the reconfiguration of the pipeline computing systems problem and some of its classical solutions, the fourth section describes the proposed solution and the last section summarizes the conclusions and future development directions.

THE PIPELINE ARCHITECTURAL CONCEPT AND ITS MAIN FEATURES

The pipeline architectural concept applied in computing systems consists of chaining together processing stations the outputs of one being inputs for the next one, [2]. Each station is specialized for an operation or a group of operations meaning that it can perform the operation in minimum time. The totality of these operations makes up the problem which must be performed by this structure. It is assumed that the problem can be divided in operations and every operation is performed by the adequate station. Maximum speed is obtained if at a given moment more problems are performed by the pipeline each one being in different phases of their execution.

For best performances, meaning maximum speed, it is desirable that the pipeline is made up by stations with narrow specialization that is for only one type of operations. Then the speed will be maximum if the problems of supplying inputs and taking over outputs are solved too. But this requires complex and expensive hardware. It also means that the pipeline will be able to solve only one type of problems.

The pipeline architectural concept for computing systems has some specific features, [2]. One of them is the speedup of a pipeline versus an equivalent no pipeline structure. We consider a pipeline with m stages which process n instructions or tasks. First it is necessary to fill up the pipeline with the operand of the first instruction and next the pipeline will be able to offer a result at every clock period. Thus the n instructions will be executed in (m + n – 1) clock periods: m represents the necessary time for filling up the pipeline, that is the time necessary for obtaining the first result and (n – 1) represents the necessary time for obtaining the results for the rest of the (n – 1) instructions, one result in every clock period. In a system not organized as a pipeline, with m processing elements too, the n instructions will be executed in n * m clock periods. The speedup is defined by the next relation, [2]:

$$v = \frac{n * m}{m + n - 1} \qquad (1)$$

The speedup will tend to m, the number of stations if the number of instructions or tasks is significantly greater than the number of stations. The relation (1) shows the maximum, theoretical, speedup the real one being always less because of several factors: the interdependencies between the instructions (tasks), the wait times determined by not rhythmically supplying the pipeline with inputs or not rhythmically taking over the outputs, the wait times determined by calls to the hardware resources, the jump or call instructions, the interrupts etc.

The throughput is another important specific feature of a pipeline defined, [2], as the number of instructions (tasks) that can be processed by a pipeline per unit time. If we consider a pipeline with m stations, which process n instructions (tasks), T being the period of the clock, then the throughput is defined by the next relation:

$$r = \frac{n}{(m + n - 1) * T} \qquad (2)$$

The relation (2) shows that theoretically r tends to f = 1/ T meaning the throughput is equal to the clock frequency that is the pipeline offers a result at every clock period. But in the real case the factors mentioned earlier affect the throughput too, determining its decrease.

SOME SOLUTIONS FOR RECONFIGURING PIPELINE COMPUTING SYSTEMS

There are important reasons for reconfiguring a pipeline computing system:

- low flexibility: generally a pipeline is best suited for a certain type of applications and if another type must be processed which require a modification of the structure (for instance the modification of the number of the stations or of their chaining) only the reconfiguration will permit the processing of the new application;
- low fault tolerance because of the fact that the stations are chained one after the other and if one of them fails the entire pipeline will fail;
- important cost: a perform ant pipeline is expansive and can be used for a few application types; the reconfiguration ensures its use in more application types saving the costs of other pipelines;
- inefficient power consumption: the reconfiguration ensures the activation of only some stations, needed for the current application, minimizing the power consumption.

There are solutions for reconfiguring pipeline computing systems described in different works. In [2] a general solution is presented which is based on the existence in the same

system or device of more pipelines. Thus the increase of the flexibility and the fault tolerance are obtained by:
- the existence of more general identical pipelines, which can process independently; the free pipeline will be required to solve the current application; if a pipeline fails it will be replaced by another one;
- the existence of more specialized pipelines; the current application will be solved by that pipeline which best fits its requirements.

The disadvantage of the described solution is its cost which is increased by the multiplication of the pipelines.

Another solution is presented in [7]. The reconfiguration assumes the existence of FPGA circuits in the pipeline which are more expensive and more difficult to program than the circuits from the solution which will be described. In [3] a solution is described too but its purpose is only the low power dissipation. [1] is a technical report which treats the pipeline computing systems reconfigurability as part of the more general concept of reconfigurable computing, without presenting practical solutions. A general solution for reconfigurable system architecture is presented in [4] too which uses a complex programmable circuit for every processor of the system. The solution is more expansive than the one which will be described but offers more facilities.

A totally different solution is outlined in [6] based on optical channels. The solution from [8] is suited only at the device level and the solution from [5] is based only on FPGAs.

A NEW FLEXIBLE AND GENERAL SOLUTION FOR RECONFIGURING PIPELINE COMPUTING SYSTEMS

A new solution for reconfiguring pipeline computing systems is described. The solution is general because it can be used for both device level and system level implementation of the pipeline architectural concept and for both beginning and run time moments of the application. The solution is flexible because it allows both partial and total reconfiguration.

The reconfiguration assumes two operations: the change of the order of the stations and the elimination of some stations from the pipeline at certain moments.

This requires that different sources command the inputs of the stations that is the outputs of different stations are linked to the inputs of the stations. It means that a circuit which chooses the station whose output will be currently linked at the input of a station must exist at the respective station. This circuit can be implemented by a multiplexer that will choose the output of that station which will precede the respective station, in accordance to the requirements of the application.

Total or partial reconfigurability can be obtained. For total reconfigurability the following relationship must be carried out: $N_{MUX} = n$ where N_{MUX} is the number of data inputs of a multiplexer and n is the number of stations of the pipeline. The input of the entire pipeline will be connected at one of the n inputs of the multiplexer from a station and the outputs of the other $n - 1$ stations will be connected at the other $n - 1$ inputs of the same multiplexer. The output of the entire pipeline will also be generated by a multiplexer, which will select the station which will end the pipeline.

There is another requirement which must be carried out too for ensuring maximum reconfigurability namely the possibility to command with different configurations the selection inputs of the multiplexers. Close to that is the need to obtain a general rule for selecting a certain data input of a multiplexer. This rule must consider the way in which the connections between the outputs of the stations and the data inputs of the multiplexers were done.

Two different solutions can be used for commanding the selection inputs of the multiplexers namely:
- separate lines for the selection inputs of every multiplexer; the command block will place on the selection inputs the configuration needed by each multiplexer ensuring also it's storing until the next reconfiguration;

- common lines for the selection inputs of all the multiplexers: the command block will place the configuration needed by a multiplexer, accompanied by its address; the storing of this configuration must be locally achieved by the multiplexer.

The proposed general rule for choosing a certain data input of a multiplexer is: the stations are numbered from left to right with 0, 1, 2, … and the data inputs of the multiplexers are also numbered with 0, 1, 2, . . . from bottom to top. The output of the station i is connected to the i^{th} data input of every multiplexer, excepting its own multiplexer at which the input of the entire pipeline will be connected. The output of the pipeline is generated by a separated multiplexer at which the i^{th} data input is connected to the output of the station i. This solution is original and simply solves the problem of commanding the selection inputs of the multiplexers. The general rule is: the selection configuration for each multiplexer is the address of the currently previously situated station in the pipeline. With this rule the chaining of the stations can be modified. The selection configuration for the multiplexer situated at the end of the pipeline will be the address of the station which is the end of the pipeline. When one establishes the chaining of the stations in a pipeline, in accordance to the requirements of an application, it immediately results the selection configurations for the multiplexers of the stations as being the addresses of the previously situated stations.

A total reconfigurable pipeline with 4 stations is presented in fig. 1. The stations were noted with numbers and the multiplexers were noted with the M capital. Because there are only 4 stations multiplexers with 4 data inputs and 2 selection inputs will be necessary. S_{0i} and S_{1i} are the selection inputs of the multiplexers, i = 0, 1, 2, 3, f. For instance the chaining 0 ? 1? 2 ? 3 will impose the next selection configurations which must be supplied and stored:

$S_{00} = 0$ $S_{01} = 0$ $S_{02} = 1$ $S_{03} = 0$ $S_{0f} = 1$
$S_{10} = 0$ $S_{11} = 0$ $S_{12} = 0$ $S_{13} = 1$ $S_{1f} = 1$.

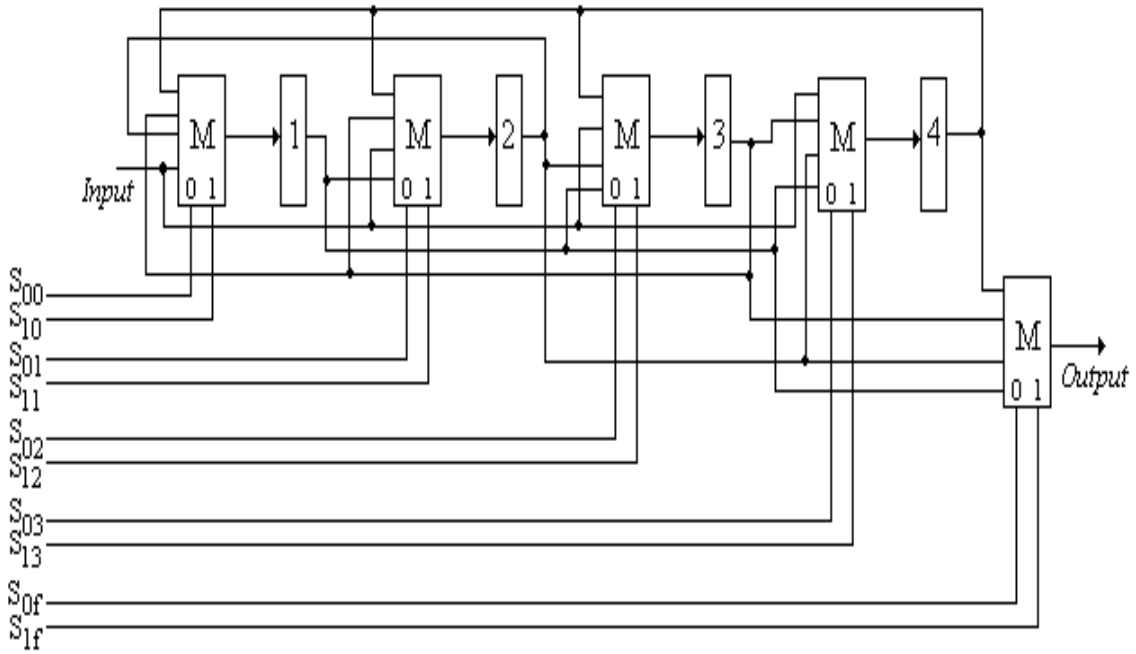

Fig. 1 A total reconfigurable pipeline with 4 stations

Changing the configurations at the S_{0i} and S_{1i} inputs will lead to another chaining of the station. Maximum flexibility is ensured by the proposed solution: 2^n chainings are possible using only n + 1 multiplexers for a pipeline with n stations.

If the command of the selection lines is simplified the flexibility degree will be reduced leading to a partial reconfigurable pipeline. The maximum simplification is obtained if the selection lines for all the multiplexers will be common. Considering a pipeline with n stations,

$n = 2^k$, the number of necessary selection lines will be $\log_2 n$ and the maximum number of possible chainings will be $2^{\log n} = n$. If $n ? 2^k$ then the number of necessary selection lines will be $a = [\log_2 n] + 1$ and the maximum number of possible chainings will be 2^a.

Another connection rule is necessary because the rule described for the total reconfigurable pipeline shows that for any configuration brought to the common selection lines the pipeline will be composed by a single station. The new connection rule is based on the close connection between the chainings of the stations in the pipeline, the configurations brought to the selection lines of the multiplexers and the connections at the data lines of the multiplexers. If we start from the chainings we can obtain the connections but we can use the inverse way too that is we start from certain imposed connections and we obtain the possibly chainings in the pipeline.

For example the partial reconfigurable pipeline from fig. 2 is able to perform the next chainings:

0 ? 1 ? 2 ? 3 for the configuration $S_0 = 0$, $S_1 = 0$ or
0 ? 3 ? 1 ? 2 for the configuration $S_0 = 1$, $S_1 = 0$ or
1 ? 3 ? 2 ? 0 for the configuration $S_0 = 0$, $S_1 = 1$ or
2 ? 0 ? 1 ? 3 for the configuration $S_0 = 1$, $S_1 = 1$.

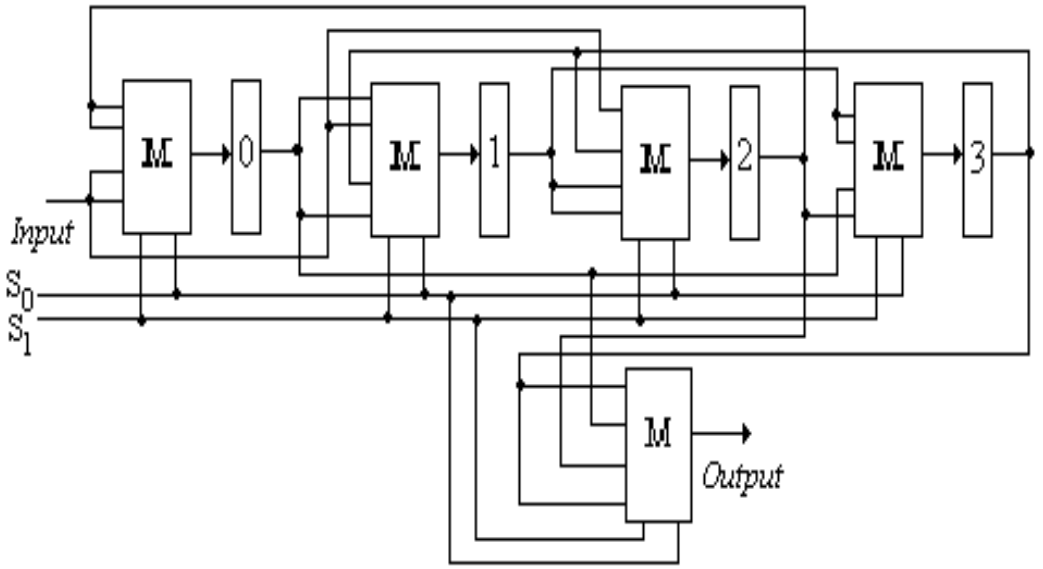

Fig. 2 A partial reconfigurable pipeline with 4 stations

Only the mentioned connections are allowed. If another chaining is needed it must replace an existing one because the pipeline can be implemented by only four chainings and other connections must be done.

CONCLUSIONS

The paper treats the reconfiguration of pipeline computing systems. After a short description of the pipeline architectural concept and its main features the necessity of the reconfiguration is emphasized. The low flexibility, the low fault tolerance, the cost and the inefficient power consumption of the pipelines are disadvantages which can be minimized by the reconfiguration operation. Some solutions are described for the reconfiguration of pipeline computing systems.

A new flexible and general solution for total and partial reconfigurability is described. The solution imposes supplementary hardware at the station level but this is not complex, fast and easy to program. The rule for connecting the outputs of the stations to the data

inputs of the multiplexers ensures a simple command of the selection lines of the multiplexers, which will determine a certain chaining of the pipeline that is the desired position for every station in the pipeline. Two examples are described which show the advantages of the proposed solution.

Future development directions for the presented solution are:

- a fast and simple command block for the selection lines of the multiplexers; the overhead the command introduces must be minimized and the flexibility of the command must be maximized;
- fast and cheap solutions for the multiplexers; the time needed for reconfiguring the system must be much less than the delay of the pipeline.

REFERENCES

[1] Enzler, R. The Current Status of Reconfigurable Computing, Technical Report, Electronics Lab., Swiss Federal Institute of Technology (ETH) Zurich, July 1999

[2] Hwang, K., F. A. Briggs. Computer Architecture and Parallel Processing, McGraw – Hill Book Company, New York, 1987

[3] Kim, S., C. H. Ziesler, M. C. Papaefthymiou. Fine – grain real – time reconfigurable pipelining, IBM Journal of Research and Development, Vol. 47, No. 5/ 6 September/ November 2003

[4] Ogrenci – Memik, S., E. Bozorgzadeh, R. Kastner, M. Sarrafzadeh. SPS: A Strategically Programmable System, Proceedings of the 8th Reconfigurable Architectures Workshop (RAW), April 2001, San Francisco, CA

[5] Rubini, S., D. Lavenier. Les architectures reconfigurables, Calculateurs Paralleles, 9 (1), 1997

[6] Sakr, M. F., S. P. Levitan, C. L. Giles, D. M. Chiarulli. Reconfigurable Processor Employing Optical Channels, Proceedings of the 1998 International Meeting on Optics in Computing, OC'98, Brugge, Belgium

[7] Schmit, H. Incremental Reconfiguration for Pipelined Applications, Proceedings of the 5th Annual IEEE Symposium on FPGAs for Custom Computing Machines, April 1997, pp. 16 - 18

[8] Siemers, C., S. Wenneckers. The Universal Configurable Block/ Machine System – An approach for a Homogeneous Configurable SoC Architecture, Proceedings of the IEEE Workshop: Heterogeneous Reconfigurable Systems on Chip (SoC), Hamburg, Germany, April 2002

ABOUT THE AUTHOR

Prof. Mircea Popa, PhD, Computer and Software Engineering Department, Faculty of Automation and Computers, University "Politehnica" Timisoara, ROMANIA, Phone: +40 256 403263, E-mail: mpopa@cs.utt.ro, http://www.cs.utt.ro/~mpopa