New Area Management Method Based on "Pressure" for Plastic Cell Architecture

Taichi Nagamoto, Satoshi Yano, Mitsuru Uchida,
Yuichiro Shibata, and Kiyoshi Oguri

Department of Electrical Engineering and Computer Science,
Graduate School of Science and Technology, Nagasaki University,
1–14 Bunkyo-machi, Nagasaki 852-8521, Japan
{taichi, yano, uchida}@pca.cis.nagasaki-u.ac.jp
{shibata, oguri}@cis.nagasaki-u.ac.jp

Abstract. In the present paper, we propose a novel area management method based on the concept of "pressure". Plastic Cell Architecture (PCA) is a dynamically reconfigurable architecture that was proposed paying attention to the essentials of processing and the flexibility of Von Neumann architecture. Mechanisms in which area are managed require a function similar to "malloc" in the C language. However, for dynamically reconfigurable architectures such as PCA, uniform management and parallelism are incompatible. Therefore, it is necessary for any PCA circuit to determine its own vacant areas without assistance from an administrator. We therefore introduced a new area management method that obtains the vacant areas by moving the surrounding objects by "pressure". "Pressure" is realize by adding new command set to original commands of PCA. We describe the basic structure for the new command set, and consider herein the development of three new command sets. Finally, we evaluate these command sets with respect to execution area and rate of effective use.

1 Introduction

In the age of ubiquitous computing, the von Neumann type computer effects every aspect of life and has infiltrated every corner of our society. The key to the success of the von Neumann type computer is its generality. We believe that the architecture of the hardware and the software (i.e. program) drive the generality. Any function can be realized both in program logic and wired logic. Important difference between correct program logic and wired logic is that only program logic can allow dynamical constitution with an object instance during execution of a procedure. This heap management is thought to be the source of the pliability of the CPU + MEM structure. Therefore, heap management should also be considered in the domain of wired logic.

Plastic Cell Architecture (PCA) is a new computer architecture having a high degree of parallelism that can replaces the von Neumann type architecture. PCA was designed taking flexibility in wired logic domain [1], [2]. PCA has a structure in which LUTs (Look-Up Tables), which are small-scale truth value tables, are set into the routing network. PCA can perform “processing, transmission, and holding” of data by using the LUTs as memory or as a circuit and the routing network.

In order to use enable dynamic-reconfigurability, which is characteristic of PCA, a functions such as “malloc” and “new”, from the C/C++ language, are required. PCA is not yet provided with a domain management method for using these functions because of a problem concerning placement of a new instance. For example, there exists a method of using “domain administrator” that manages areas, but management and the parallel system are incompatible. Furthermore, access to the domain administrator may increase the processing overhead, causing deadlock.

In this paper, we propose a area management method that uses the concept of “pressure”. This method is based on a cell division mechanism, whereby existing cells are pushed out of the way when by the growth of a new cell. In this case, the “cell” is the new instance. This method can be performed independently in several places, so that domain administrator is not necessary. PCA achieves dynamic-reconfigurability using commands that are implemented in the built-in part (see Section 2). Therefore, this method is realized as a extension of commands in the built-in part.

In Section 2, we present the background and the related work. In Section 3, we propose a new area management scheme, and in Section 4, we describe the mechanism required to create a command set. In Section 5, three command sets are introduced, and these command sets are evaluated in Section 6. Finally, conclusions are presented in Section 7.

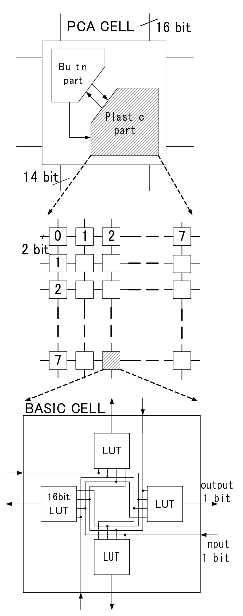
2 Background

2.1 PCA (Plastic Cell Architecture)

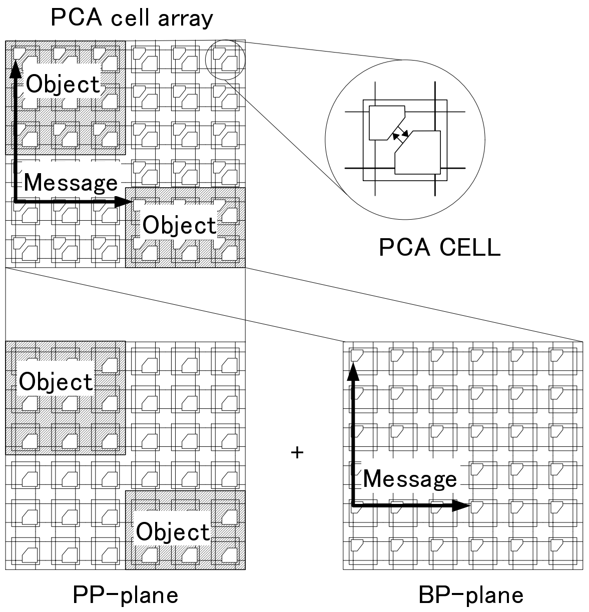
The composition of PCA is covered with LUTs that are in the routing network. In this composition, data processing, transmission, and memory can be performed directly and in parallel[1], [2].

Figure 1(a) shows the basic structure of PCA, which is referred to as the PCA cell. PCA cells are arranged in a two-dimensional mesh. Each PCA cell consists of a plastic part and a built-in part. Plastic parts are reconfigurable circuits that can perform memory and operation tasks, and built-in parts performs control and communication tasks for each of the plastic parts. Plastic parts have 8×8 basic cells. The basic cell consists of four 16-bit LUTs. An LUT has a memory of 4 bits for address input and 1 bit for data output.

Figure 1(b) shows an example of communication between objects. An aggregate of PCA cells that performs processing is referred to as an object. Objects that become elements (e.g. add, sub, conditional branch) of a process are arranged to correspond to a Data Flow Diagram. Communication between objects is performed by a built-in part. When two or more objects send messages to each other, the built-in part operates in parallel. Furthermore, an object can be dynamically created by sending a composition message to the built-in part.



(a) PCA Cell



(b) Object and Message

Fig. 1. Plastic Cell Architecture

In PCA, an asynchronous circuit system with easy timing control at the time of connecting a new object is adopted. PCA adopts a bundled protocol with four-cycle signaling.

2.2 Bit Serial PCA

An architecture called Bit Serial PCA, based on PCA has been proposed in a previous study [3]. The characteristics of Bit Serial PCA are described below.

There are two types of data-processing system: the bit parallel system and the bit serial system. Although the bit parallel system is most commonly used at present, if the skew between signals and the rate of wiring delay increase, then the bit serial system may provide better high-speed performance than the bit parallel system. In addition, since PCA hardware can be reconfigured, a switch for reconfiguration exists in the wiring, further increasing the overhead. PCA also arranges several operation circuits along the data flow and performs arbitrary processing. Therefore, the bit serial system with small circuits for each function is suitable for PCA. Therefore, Bit Serial PCA adopts the bit serial system. As a result, circuits can be constituted only with state machines and shift registers, allowing the circuit to be compact while offering high-speed performance.

Each plastic part operates as a state machine and a shift register, and communication of plastic parts is performed through a built-in part.

2.3 Related Work

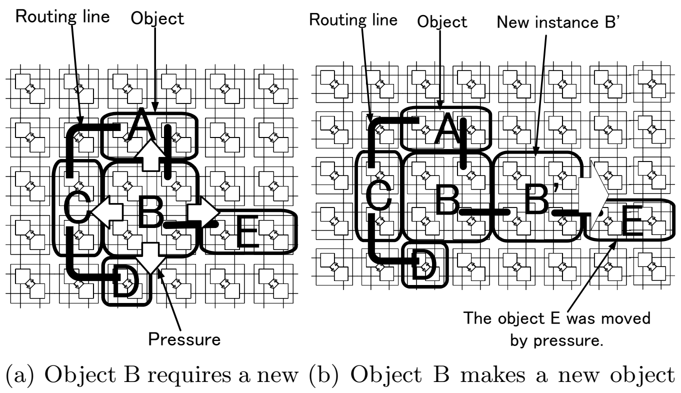
Interest in dynamic reconfigurable technology has been increasing in recent years. Replacing hardware having pliability with exclusive hardware is expected to enable various applicable fields on one chip [4].

As methods that realize a dynamically reconfigurable system, the method of changing functions and connection of a composition element is examined. The multi-context system uses a method that changes two or more pieces of composition information (commands) stored in the internal memory if needed. Multi-context type reconfigurable devices were initially proposed for reconfigurable devices to examine the degree of granulation. For example, at Keio University, a performance improvement was achieved by using a variable clock [5]. The DRP, developed by NEC, is another example of a multi-context type reconfigurable device [6].

NTT also proposed a PCA system [7], [8], [9]. Furthermore, a prototype PCA-2, which combines a high degree of integration and high speed, has been developed [10]. PCA-2 has a function whereby its composition can be partially changed, based on the data generated by the PCA objects themselves. In PCA, the composition of circuits fluctuates somewhat. In order to employ the reconfigurable function of PCA efficiently, the problem of how the domain of circuit should be constituted is encountered. At the University of Aizu, a method of expressing the entire circuit by a unit that includes duplicates or deletions, called a stage, was proposed [11].

3 New Method of Area Management

The concept of PCA is shown in Fig. 2. The squares in the figure indicate objects, and the thick lines indicate routing lines. Each object consists of one or more PCA cell. Each object works as a circuit. Each object performs an



(a) Object B requires a new object (b) Object B makes a new object B'

Fig. 2. Concept of PCA

operation (e.g. add, sub, conditional branch). Communication between objects is performed through the built-in parts. A new method of area management is shown below.

When object B needs a new object B', it sends a "pressure command" to the surrounding objects (Fig. 2(a)). The surrounding objects receive the pressure command and move to vacant areas. (In the figure, object E moved (Fig. 2(b)).) If the area is vacant, object B makes a new object (B'). This mechanism is realized as an extension of the command in the built-in part.

In clarifying the characteristics and the effect of area management caused by the pressure, we assumed the following in order to simplify the problems.

1. Routing lines between objects are not considered.
2. An object consists of one PCA cell.

There are numerous command sets that would allow us to realize this method. However, creating the correct command set is difficult. In this paper, we present three types of correct command sets having different characteristics. These command sets were obtained by trial and error.

4 Mechanism Required in Order to Create a Command Set

In order to create a command set, we implemented a simulator on a PC and performed trial and error repeatedly. As a result, the mechanism shown in Fig. 3 was found to be necessary. Cells are arrayed onto 2D meshes, as in PCA, to process pressure commands (Fig. 3 left). Each cell consists of a state machine to identify the condition (e.g. vacant state, circuit state, or receiving a pressure state). Cells have five input-output ports (north, south, east, west and feedback), and a switch to limit the direction of entered commands (Fig. 3 right).

Communication Path for Commands. In communication between cells, deadlock is a very important problem. The structure buffer and e-cube routing, for example, are reported to be communicative methods that do not incur deadlock. However, these method are expensive to implement on PCA hardware. When deadlock occurs, the sender must wait until a receiver becomes

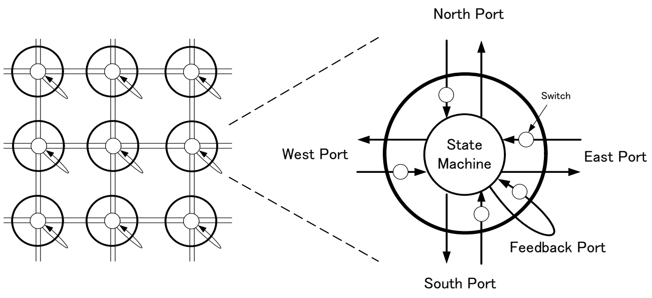


Fig. 3. Cells

available. Therefore, we adopted a communication system that does not wait for responses. The system is based on wormhole routing and can output commands without waiting for a response from the receiver side, even if a cell that receives a command is processing. Although this system prevents deadlock, the possibility exists that commands will be overwritten.

Function to Limit the Input Port. When a cell moved by pressure duplicates, a contradiction occurs if this information is not correctly communicated. Therefore, in order to transmit the information correctly, we added a function whereby the cells do not process commands from cells that are not companion cells. In other words, the cell can limit the direction from which commands are accepted.

Feedback Port. Even if a cell outputs a pressure command once, the neighborhood is not always vacant. Therefore, the cell must output the pressure command several times. As such, we added one input-output port for feedback so that the cell can operate continuously with outputting a command to itself.

5 Explanation of Command Sets

We present three kinds of command sets that were obtained by trial and error. These command sets can increase an object to 1,024 (1, 2, 4, ..., 512, 1,024) cells. The correctness of the increase is checked by an ID that is distributed to objects.

5.1 Command Set 1

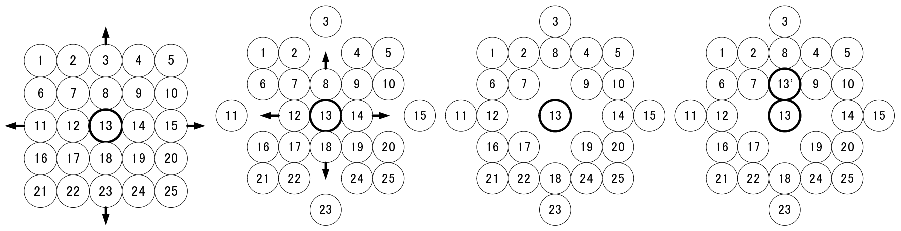
Command Set 1 indicates that an object that is going to duplicate itself places pressure only on the objects that exist in the vertical and horizontal directions, as shown in the figure. Therefore, it is not possible to increase the number of cells when the cells in the vertical and horizontal directions are all objects.

For example, consider the case of the cells shown in Fig. 4(a). Suppose Cell 13 wants to make a duplicate of itself. In this case, the pressure command is first sent in the vertical and horizontal directions. Then, Cells 3, 11, 15, and 23, which are at the outermost location in these direction are pushed out outward (Fig. 4(b)). Since locations next to Cell 13 are not vacant yet, the press command is resent, and Cells 8, 12, 14, and 18 are pushed outward, leaving a vacancy next to Cell 13 (Fig. 4(c)). Since the locations next to Cell 13 are now vacant, a clone can be generated (Fig. 4(d)).

5.2 Command Set 2

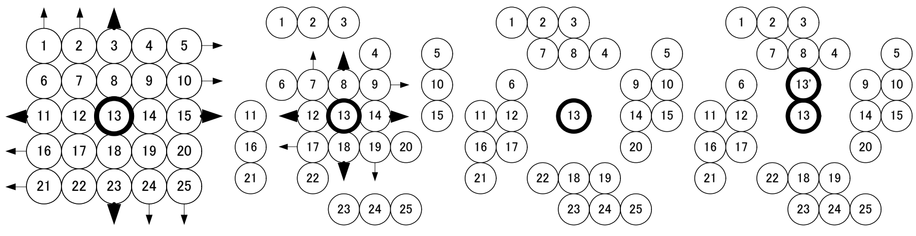
This command set has the ability to apply pressure in all directions.

In Fig. 5(a), when cell No. 13 increases, it sends "main pressure commands" to the surrounding cells (Nos. 8, 14, 18 and 12). Main pressure commands are

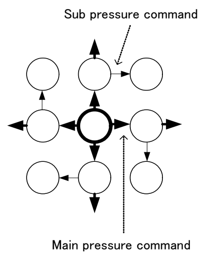


(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4

Fig. 4. Example operation in command set 1



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4



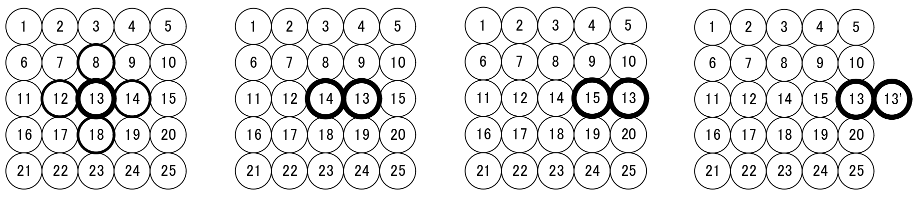
(e) Pressure command

Fig. 5. Example operation in command set 2

transmitted directly and generate a "sub-pressure command" on the way (Fig. 5(e)). Cell Nos. 3, 15, 23 and 11, which received the main pressure command, are extruded. In addition, cell Nos. 1, 2, 5, 10, 25, 24, 16 and 21 are extruded by the sub-pressure command (Fig. 5(b)). Since the cells surrounding cell No. 13 are not vacant, cell No. 13 sends main pressure commands. As a result, cell Nos. 8, 14, 18 and 12) are extruded by this command, and cell Nos. 7, 9, 19 and 17) are extruded by sub-pressure commands (Fig. 5(c)). Since the cells surrounding cell No. 13 are vacant, a copy of cell No. 13 is made (No. 13') (Fig. 5(d)).

5.3 Command Set 3

This command set has a different method of applying pressure. A cell that is going to duplicate itself moves until a vacancy is found. The movement of the cell is realized by exchanging locations with the next cell until a vacancy is found.



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4

Fig. 6. Example operation in command set 3

In Fig. 6, Cell 13 gives increase demand to Cells 8, 12, 14, and 18 (Fig. 6(a)). Cell 13 exchanges locations with Cell 14, which gives the fastest acknowledgement (Fig. 6(b)). Cell 13 exchanges locations with Cell 15, in the previous direction (Fig. 6(c)). Cell 13 duplicates Cell 13' in the vacancy (Fig. 6(d)).

6 Evaluation of Command

6.1 Area of Implementation

In order to evaluate the mounting area, we compared the number of commands and the number of states among the command sets. The results are shown in Table 1.

This table shows the number of commands and the number of states for each command set. These values (number of states and number of command) are related to the mounting area. As a result, "command set 1" can be implemented in the smallest area and "command set 3" can be implemented in the largest area.

6.2 Rate of Effective Utilization

In order to evaluate the rate of effective use, we measured the rate of use at the time of increase until it became impossible for the number of cells to increase. The experiment was conducted for 12×12 cells. This is the number of cells of PCA-2 [10]. The rate of utilization was deduced from the number of objects when it becomes impossible to increase. A rate of utilization of 100% indicates that all of the cells can be used. The increase command is given randomly. The rate of utilization was obtained by averaging ten measurements.

The results are shown in Table 2. The rate of utilization for "command set 2" is 100%, indicating that all cells can be used ($12 \times 12 = 144$). Among the three

Table 1. Number of commands and number of states

Command set	Number of commands	Number of states
Command set 1	6	5
Command set 2	8	5
Command set 3	11	4

Table 2. Rate of utilization

Command set	Rate of utilization(%)
Command set 1	87.7
Command set 2	100.0
Command set 3	97.4

command sets, "command set 1" had the lowest rate of use of 87.7%. These results indicate that "command set 2" can use cells efficiently.

6.3 Result

The obtained results indicate that among the command sets, "command set 2" has the best performance. Since "command set 2" can use cells efficiently, it can be implemented in small areas.

7 Conclusion

In this paper, we proposed a new method of area management based on the concept of "pressure". In order to realize this method, we presented three command sets. Evaluation of the characteristics of these command sets revealed that "command set 2" gave the best performance. In this paper, the basic operation for a completely new area management method that requires no administrator was clarified.

References

1. K. Oguri, N. Imlig, H. Ito, K.Nagami, R. Konishi and T.Shiozawa, "General purpose computer architecture based on fully programmable logic," Proc. International Conf.Evolvable System (ICES'98), pp.323-334, 1998.
2. K.Nagami, K. OGURI, T. Shiozawa, H. Ito, and R.Konishi, "Plastic cell architecture: A scalable device architecture for general-purpose reconfigurable computing," IEICE Trans.Electron., vol.E81-C, no.9, pp1431-1437, Sept.1998.
3. Kiyoshi OGURI, Yuichiro SHIBATA, Nobuyoshi MURAKAMI, Akira KAWANO and Akira NAGOYA, "Asynchronous Bit-Serial Datapath for Object-oriented Rconfigurable Architecture PCA", IEICE TRANS.INF.&SYST., VOL.E82, No.1, JANUARY 2005
4. J.M.P.Cardoso and M.P. Vestias, "Architectures and Compilers to Support Reconfigurable Computing", ACM Crossroads, Spring 1999.
5. Hideharu AMANO, Yoshinori ADACHI, Satoshi TSUTSUMI and Kenfichiro ISHIKAWA, "A context dependent clock control mechanism for dynamically reconfigurable processors", IPSJ SIG Technical Reports, vol.2005 No.8, pp.13-22, 2005
6. Takayuki SUGAWARA, Keisuke IDE, Tomoyoshi SATO, "Dynamically Reconfigurable Processor Implementated with IPFlex's DAPDNA Technology," IEICE Transaction, vol.E87-D, no.8, pp.2004-2010, August 2004.

7. R. Konishi, H. Ito, H. Nakada, A. Nagoya, K. OGURI, N. Imlig, T. Shiozawa, M. Inamori, and K. Nagami, "PCA-1: A fully asynchronous, self-reconfigurable LSI," Proc. 7th International Symposium on Asynchronous Circuit and Systems (ASYNC 2001),pp.54-61, March 2001.
8. H. Ito, R. Konishi, H. Nakada, A. Nagoya, K. Oguri, A. Nagoya, N. Imling, K. Nagami, T. Shiozawa, and M. Inamori, "Dynamically reconfigurable logic LSI-PCA-1,"Proc.2001 Symposium on VLSI Circuits,pp103-106,June 2001.
9. M. Inamori, H. Nakada, R.Konishi, A. Nagoya, and K. Oguri, "A method of mapping finite state machine into PCA plastic parts," IEICE Trans.Fundamentals, vol.E00-A,no4,pp.804-810,April 2002.
10. H. Ito, R. Konishi, H. Nakada, H. Tsuboi and A. Nagoya, "Dynamically Reconfigurable Logic LSI designed as Fully Asynchronous System - PCA-2" , Proc. of COOL Chips VI, pp. 84, Apr. 2003
11. Tomoki KAMIYAMA, Keigo KURATA, Yousuke IKEHATA, Junji KITAMICHI and Kenichi KURODA, "Proposal and Implementation of Framework for Self-Reproductive applications on Dynamically Reconfigurable Device PCA", IPSJ SIG Technical Reports, vol.2005 No.8, pp.141- 146, 2005