INVITED PAPER *Special Section on Software Defined Radio Technology and Its Applications*

A Survey on Dynamically Reconfigurable Processors

Hideharu AMANO[†a)], ***Member***

SUMMARY Dynamically reconfigurable processors are consisting of an array of processing elements whose functions and interconnections can be dynamically changed. 9 commercial systems are picked up, and their array structures, processing elements and interconnection architectures are classified.

key words: *dynamically reconfigurable processors*

1. Introduction

SoC (System-on-a-Chip) which integrates an embedded CPU, standard I/O, and application specific hardware has been widely developed for various consumer electronics and mobile products including cellular phones, portable game machines, digital audio systems, DVD players, and network controllers. Devices implemented with SoC are well suited for intense applications and the custom design enables the reduction in die size and power consumption. Efforts to develop SoCs much faster than its current pace has led to the introduction of new design methodologies such as the C based description language and hardware/software co-synthesis models.

However, recent advances and introduction of new technologies in the areas such as signal processing, data communications, and network protocol handling have made the SoC a less attractive option. The higher development costs, diversification of the product line, necessity for swift and comprehensive response toward new standards, and low quantity of the devices shipped are some of the factors that discourage implementation onto an SoC. Moreover, floor-planning and chip layouts are becoming to be the new bottlenecks in design especially in advanced CMOS processes where wiring delay is critical. Unfortunately, high level design technologies for SoC design cannot contribute much in solving these problems.

A chip combining a CPU and a coarse grain reconfigurable fabric has received attention as a solution to this problem. Since the configuration of a coarse grain reconfigurable device is flexible, the same chip can be used for various applications. It can also be "refitted" after shipment by rewriting the configuration data. Because most applications do not need special types of computing units, fine grain reconfigurable architecture using LUT(Look Up Table)s is not always efficient in performance and cost. Although large scale FPGA(Field Programmable Gate Array)s with embedded CPUs (i.e. Xilinx's Virtex-II Pro, Virtex-IV/FX and Altera's Excalibur) are commercially available, their main target remains to be in prototyping due to high costs.

Recent coarse grain dynamic reconfigurable devices have been developed to achieve high performance and flexibility for a fraction of the cost of an FPGA. It incorporates the following properties: (1) A coarse grain cell consisting of an ALU (Arithmetic Logic Unit), a data manipulator, a register file and other functional modules is adopted as the primitive processing element; (2) In reducing the cost and die size, dynamically reconfiguration which enables time-multiplexed execution is introduced; and (3) High level design entry and functional synthesis techniques developed for SoC design can be adopted for designing these devices.

Here, recent commercially available dynamically reconfigurable processors are surveyed from the viewpoint of their architectural designs. First, we introduce some typical structures and how they work using some examples. Then, commercial systems are classified with their method of dynamic reconfiguration, structure of processing elements and interconnection networks.

2. Overview of Dynamically Reconfigurable Processors

2.1 Target Systems

Table 1 shows main target systems of this survey. Most of them are commercially available currently or near future except CS2112 which is picked up as a frontier commercial machine.

SONY's VME (Virtual Mobile Engine) [37] which is embedded in a portable game machine PSP cannot be included, since its detail has not been disclosed. NTT's dy-

Table 1 Target systems.

Name	Company	Reference
CS2112	Chameleon	[1]
DAPDNA-2	IPFlex	[2]
DRP-1	NEC electronics	[3]
FE-GA	Hitachi	[4]
Xpp-64	PACT	[5]
D-Fabrix	Elixent*	[6]
Kilocore KC256	Rapport	[8]
ADRES	IMEC	[9]
S5-engine	Stretch	[11]
Cluster machine	Fujitsu	[10]

Manuscript received July 18, 2006.
Manuscript revised August 19, 2006.

[†]The author is with the Department of Information and Computer Science, Keio University, Yokohama-shi, 223-8522 Japan.
a) E-mail: hunga@am.ics.keio.ac.jp

DOI: 10.1093/ietcom/e89–b.12.3179

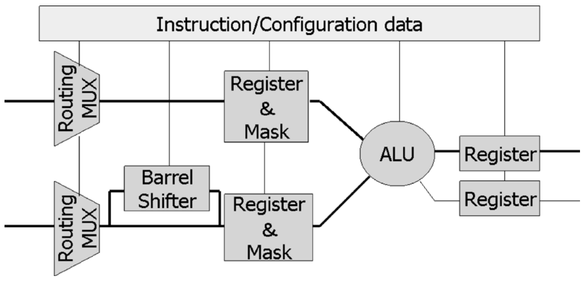


Fig. 1 An example of processing element (DPU of CS2112).

namically reconfigurable devices PCA [12] and PCA-2 [13] are also omitted with the following reasons: (1) Their architecture based on the asynchronous operation, serial data communication, and fine-grained logic elements is completely different from others and hard to be discussed together. (2) They are research prototypes rather than commercial ones.

2.2 PE Array Structure

Dynamically reconfigurable processors are consisting of an array of PEs (Processing Elements) whose functions and interconnection can be dynamically changed. A PE provides an ALU for numerical and logical calculations, logics for shift/mask operations, registers or register files and multiplexors for switching the data-flow between such components. Figure 1 shows a PE (Data Processing Unit: DPU) of CS2112 [1] which is consisting of typical components. Although the data bit-width of the DPU is 16bits, it is various from 4bits (D-Fabrix and S5-engine) to 32bits (DAPDNA-2) as shown later. The operation of ALU, shift/mask logic, and data paths between components are controlled with configuration data or instructions stored in configuration/instruction memory. Note that each PE does not have its own program counter nor instruction fetch unit. Although the element is called "Processing Element", it is not a common PE used in multiprocessors but used as a part of a large data-path by connecting with others.

A certain number of PEs (16-512) are connected to form an array structure. A typical structure is a square mesh, and as described later, both direct interconnection and switch connected bus structures are used. On the edge of the PE array, distributed memory modules are provided to hold streaming data. Input/Output data is directly transferred at the edge of the PE array directly to/from each PE or distributed memory modules. Figure 2 shows an array of PACT Xpp-64 [5]. 8×8 computational PE (ALU-PAE) are connected with 2 dimensional mesh, and at the both sides, memory modules (RAM-PAEs) for storing data are provided.

Like FPGAs, the paths between PEs are also decided with configuration data which is stored into the configuration memory provided in the switching modules. That is, the operation of PE and interconnection between them are fully programmable by the configuration data. The configuration data is often called "instructions" when the total PE

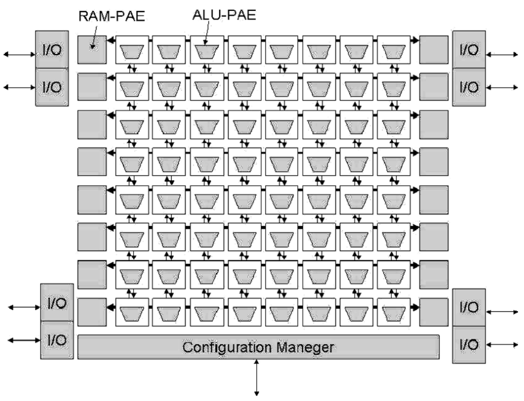


Fig. 2 An array structure of PACT-Xpp.

array is treated as a large data-path for computations.

Unlike FPGAs based on the fine-grain LUTs, a dynamically reconfigurable processor is a coarse grain programmable device. The coarse grain structure is less flexible than the fine grain structure, that is, it is not suitable to form state machines and complicated bit-wise random logics. However, it achieves high performance per cost for media processing required for most hardware accelerators of recent SoCs.

2.3 Dynamic Reconfiguration

Only by using the coarse grain structure, the performance per cost of programmable devices is far from that of dedicated hardware. So, by making the best use of reconfigurable property, dynamic reconfiguration is introduced to enhance the area efficiency by changing its structure dynamically. That is, by using a single PE array for multiple tasks, the semiconductor area can be utilized efficiently compared with dedicated hardware logic.

The simplest way for dynamic reconfiguration is providing a single or several on-chip memory modules in the chip, and storing multiple sets of configuration data. The array configuration can be changed by transferring new configuration data from the memory to each PE and switches through the configuration bus as shown in Fig. 3. Common media processing is consisting of multiple tasks which are executed sequentially. When a task executed on the PE array is finished, the configuration data corresponding to the next task is transferred and executed. Here, this method is called "configuration delivery", and a configuration data set corresponding to a task working on a PE array is called the "hardware context". Xpp, D-Fabrix and S5-engine fall into this type. It usually takes more than 10 micro-seconds to send the configuration, and during the configuration transfer, the computation on the array is, at least, partially suspended.

Another dynamic reconfiguration method is called "multicontext" reconfiguration. In this method, each PE provides a memory module that stores the configuration data sets for the corresponding PE and interconnection of

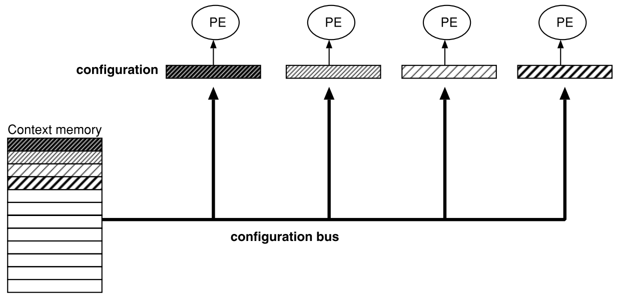


Fig. 3 Configuration delivery method.

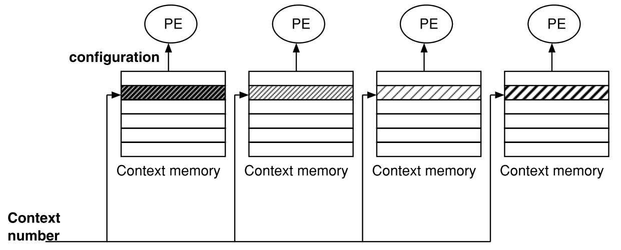


Fig. 4 Multicontext mechanism.

surrounding buses. The context number is broadcasted throughout the chip, and used as a pointer to the context memories. By changing the context number and reading the context memory simultaneously, the context is switched with a clock cycle (Fig. 4). This in turn means that the configuration data for a context is distributed to each PE, and a context is switched by configuration data read-out from each of the context memories.

With either method, the hardware context can be changed much faster than that of FPGAs which often requires milli-seconds to load the configuration data. One reason comes from that the total amount of configuration data for coarse-grain dynamically reconfigurable processor is much smaller (1/10–1/100) than those for fine-grained FPGAs [14]. Although context switching can be done with a clock cycle in multicontext devices, the area of each PE is increased with the distributed context memory. The area of configuration memory which provides 32 contexts is almost the same as that of a PE itself [9].

2.4 Interconnection with a Host Processor

A dynamically reconfigurable processor is used as an accelerator of a host embedded processor. Some of them are designed as an IP (Intellectual Property) which can be used in various SoCs.

The host processor is often "configurable processors", and the dynamically reconfigurable processor can be tightly coupled by sharing registers. Figure 5(a) shows Toshiba's MeP configurable processor with Elixent's D-Fabrix as a reconfigurable extension [6]. In such systems, a task executed in the PE array is treated as a reconfigurable operation of the host processor, and the dynamically reconfigurable processor behaves like a kind of pipelined execution unit of the

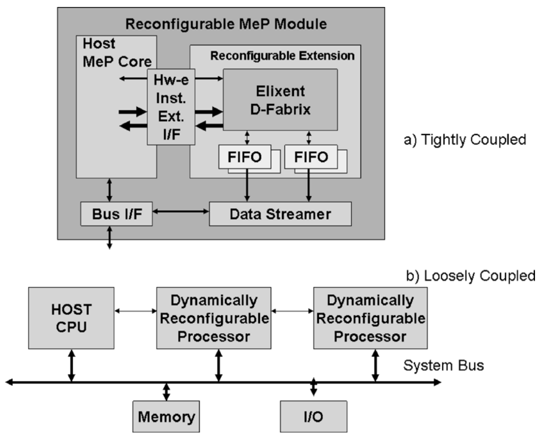


Fig. 5 Interconnection with the host processor.

host processor. Similarly, S5-engine is composed in a Tensillica's configurable processor Xtensa [11].

However, even in this case, the dynamically reconfigurable processor executes its task autonomously, that is, loop-level tasks are allocated and executed by using the direct data transfer path between the host memory and distributed memory modules in the PE array.

Some dynamically reconfigurable processors are connected with the host more loosely as shown in Fig. 5(b). In this case, the stream data is transferred between host memory and distributed memory modules in the PE array with the DMA controller similar to other accelerators in SoCs. Since a dynamically reconfigurable processor can execute a single task at a time, a multiple-core structure with multiple dynamically reconfigurable processors is efficient for performance enhancement. A multiprocessor with multiple FE-GA cores and multiple SH-4 CPUs is now under development by Hitachi [15], and Fujitsu's cluster machine is consisting of multiple clusters each of which is relatively small scale PE arrays [10].

2.5 Parallel Execution and C-Level Programming

By making the best use of their flexibility, various parallel algorithms can be executed on the PE array of the dynamically reconfigurable processors. The simplest way is generating data-flow graph from C language, and mapped into the array directly. Then, the streaming data is inserted and executed in the pipelined manner. Kilocore KC256 [8], which is a commercial version of PipeRench [17] has specialized structure for the pipelined execution. As shown in Fig. 6, it is consisting of "stripe"s each of which is corresponding to a stage of a pipeline. By the dynamic reconfiguration of "stripe"s, a pipeline with arbitrary number of stages can be virtually implemented on 16 stripes each of which has 16 PEs.

For media processing, a certain size of streaming data corresponding to a window or frame stored in distributed memory modules are processed simultaneously by PEs iteratively like the SIMD(Single Instruction stream Multiple

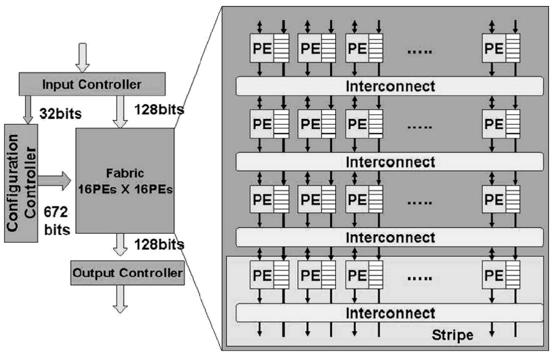


Fig. 6 KC256 for pipeline execution.

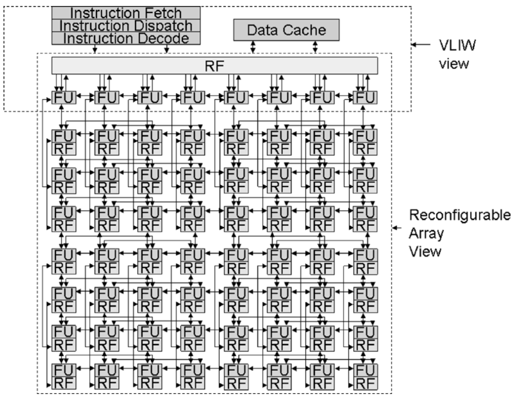


Fig. 7 The structure of IMEC ADRES.

Data Streams) manner. However, unlike common SIMD processing, operations of each PE and data transfer between PEs/memory modules can be various and flexible. In the multicontext dynamically reconfigurable processor, the context can be switched clock by clock, thus, the datapath for processing can be switched in every clock cycle. By using context switching, the ILP (Instruction Level Parallelism) from LLP (Loop-level parallelism) can be efficiently utilized.

C-language based programming is mainly used for dynamically reconfigurable processors, since the data-flow graph can be directly mapped into the PE array. For DRP-1 of NEC electronics, a sophisticated design tool Musketeer [16] divides the target task into an optimized number of contexts, and schedules them by using functional synthesis techniques. By using the tool, the program described in BDL (Behavior Description Language), a C-like hardware description language can be automatically translated into configuration data for each context. The programming environment for S5-engine analyzes the target C program, and detects the loop to be executed in the PE array. Other systems also prepare their own high level design tools.

2.6 How Different between Other Architectures?

Dynamically reconfigurable processors have been introduced mainly compared with fine-grained reconfigurable devices FPGAs. Here, they are compared with other architectures.

2.6.1 Tile Processors vs. Dynamically Reconfigurable Processors

Tile processors, a type of on-chip MIMD (Multiple Instruction stremas Multiple Data Streams) processors also consists of an array of processing elements. MIT's RAW [24], PicoChip [23] and Quicksilver's ACM [25] fall into this category. The most important difference is that a processing element of Tile processors is a powerful CPU with program counter and its own instruction memory, while one used in dynamically reconfigurable processors is just a part of datapath without any instruction fetch mechanism. So, the semiconductor area for a PE of dynamically reconfigurable processors is much smaller than that of Tile processors. This property is advantageous, since the main target of dynamically reconfigurable processors is embedded systems for consumer electronics and mobile systems.

2.6.2 VLIW vs. Dynamically Reconfigurable Processors

A multicontext device changes its structure with a clock cycle by loading new configuration data from context memory modules distributed to each PE and switch. If the context pointer is treated as a program counter, the total configuration data can be thought as a very long instruction. From this viewpoint, a dynamically reconfigurable processor is a VLIW (Very Large Instruction Word) computer that provides an extremely large datapath and a limited instruction fetch mechanism. From the opposite viewpoint, an instruction execution in a common stored programming computer is treated as a type of hardware context switching, since the interconnection and operations of the datapath are changed by executing an instruction.

However, in a common VLIW machine, source/destination operands are registers, and so an instruction is formed with relatively simple combination of operations. In dynamically reconfigurable processors, a certain set of data is stored in the distributed memory modules in the PE array, and processed iteratively within a context. That is, more parallelism can be easily utilized in dynamically reconfigurable processors. Instead, the number of available contexts is strictly limited in dynamically reconfigurable processors, and it is difficult to execute complicated programs. ADRES [9] has a VLIW part in the array of PEs. As shown in Fig. 7, the upper most 8 PEs (FUs) are directly connected with a shared register file, and work in the VLIW mode. For the task with more parallelism, the data is moved to the PE array, and processed in the dynamically reconfigurable processor mode.

2.6.3 SIMD vs. Dynamically Reconfigurable Processors

ALU arrays are also used in a special purpose SIMD (Single Instruction stream and Multiple Data streams) machines including ClearSpeed [26]. Since a single context pointer is

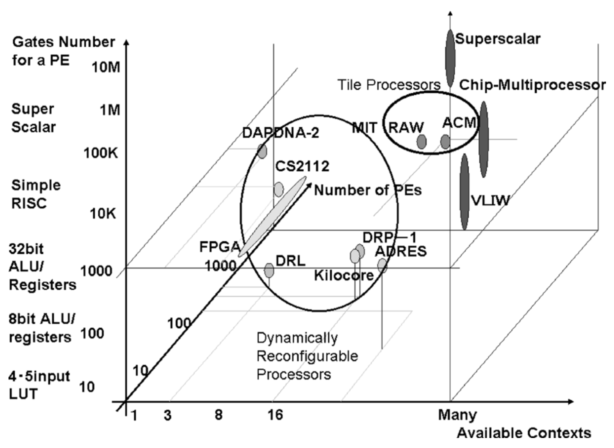


Fig. 8 The position of dynamically reconfigurable processors.

used in a dynamically reconfigurable processor, the control flow is simple as SIMD machines.

However, operations and interconnections of a dynamically reconfigurable processor are much flexible. Instead, the SIMD machine requires much smaller instruction code than that of dynamically reconfigurable processors and so a long and complicated program code can be executed with powerful instruction fetch mechanism.

Figure 8 shows the position of the dynamically reconfigurable processors. It classifies architectures with the complexity of each processing element, parallelism (number of used processing elements), and the number of available contexts. Chip-multiprocessors and VLIW machines based on the stored programming computers can execute enormous number of contexts by executing instructions but the number of PE is not many, so they are located on the right front upper side. Tile processors which have more number of processors with less complexity than typical chip-multiprocessors are located lower back position of them. In contrast, FPGA with numerous number of fine grain LUTs but cannot change the context occupies the left bottom back position. Dynamically reconfigurable processors are widely distributed between both ends of the diagram. Its position is lower than that of Tile processors because of less complexity of each PE, but of course, higher than that of FPGAs. From the viewpoint of parallelism (number of used processing elements), it is less than that of FPGAs but more than that of Tile processors. Since the space of dynamically reconfigurable processors is wide, the characteristics of systems are also various as shown in later.

3. A Survey of Detail Structures

3.1 Basic Classification

Table 2 shows fundamental features of the target systems. First, they can be classified by the method of dynamic reconfiguration: configuration delivery (D) or multicontext (M) introduced before. The maximum number of contexts which can be stored in a PE is attached for multicontext devices.

Table 2 Fundamental features.

Name	Conf.	PE array	Data	PEs
CS2112	M (8)	Hetero	16/32	108
DAPDNA-2	M (4)	Hetero	32	376
FE-GA	M (4)	Hetero	16	32
Cluster machine	M	Hetero	16	15/c
DRP-1	M (16)	Homo-Out	8	512
Kilocore KC256	M/D	Homo-All	8	256
ADRES	M (32)	Homo-All	16	64
Xpp-64	D	Homo-Non	24	64
D-Fabrix	D	Homo-Non	4	576
S5-engine	D	Homo-Out	4/8	-

Kilocore KC256 shown in Fig. 6 has a special configuration mechanism. In this architecture, a stripe is re-configured with a clock cycle to form a virtual pipeline with a powerful configuration bus. Since the configuration data can be transferred from outside the chip, the number of context is unlimited.

Then, they can be classified whether the PE array is homogeneous or heterogeneous. Homogeneous means that all PEs are the same structure, while more than two types of PEs are used in heterogeneous structure. The key design issue is how multipliers are implemented in a PE array. Needless to say, multipliers are essential for digital signal processing, but the semiconductor area of a multiplier is much larger than that for an adder or shift/mask logics. For some applications including encryption/decryption, it is often useless.

So, the array structure of dynamically reconfigurable processors can be classified as follows based on the multiplier implementation.

- Multipliers are implemented on some PEs (or some PEs are dedicated for multipliers), but not included in others. So, the array structure becomes heterogeneous (Hetero). In this structure, the number of PE with multipliers in the total PE array becomes a design choice. For example, FE-GA provides 8 multipliers (MLT) in 24 total PEs as shown in Fig. 9. The number is analyzed in Table 3[†], and the ratio becomes roughly 1 : 3.
- Every PE provides its own multiplier. In this case, the PE array becomes homogeneous (Homo-All).
- Every PE does not have any multiplier. The multiplier is structured with an array of multiple adders and shifters. The PE array also becomes homogeneous (Homo-None).
- Every PE does not have any multiplier, but the dedicated multipliers are provided outside the PE array. For example, DRP-1 provides eight multipliers outside the PE array. They can be connected with PEs in the array with some restrictions. S5-engine has dedicated array of multipliers other than the general purpose PE array. The PE array becomes homogeneous (Homo-Out), if outside multipliers are not taken into account.

The bit-width treated in a PE is another important factor, since it is related to the application target. 16bits-PE

[†]In Cluster machine, the PE is configurable [10].

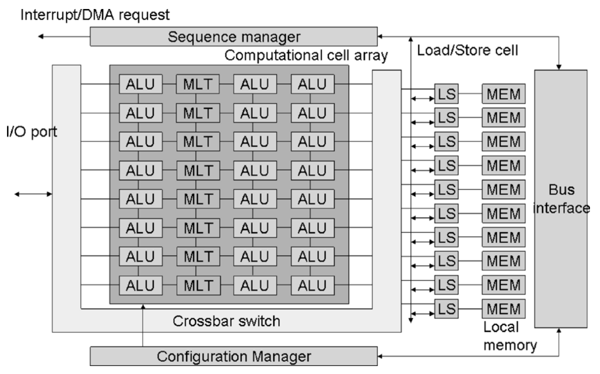


Fig. 9 Structure of FE-GA.

Table 3 Number of multipliers PE.

Name	Multipliers	Other PEs	Ratio
CS2112	24	84	1 : 3.5
DAPDNA-2	56	168	1 : 3.3
FE-GA	8	24	1 : 3

is popular but other bit-widths are also used. The number of PEs are not so important, since most of dynamically reconfigurable processors are scalable in a certain unit of PE array (The unit is often called a "Tile"). Table 3 also shows the size of the PE array[†] D-Fabrix and S-5 engine are tightly coupled with configurable processors, use the delivery configuration, and consist of a large size array of small PEs. In such machines, the time for configuration delivery can be hidden by the host CPU execution, and the high area-efficiency is required. On the other hand, DAPDNA-2 designed mainly for high-end image processing applications uses relatively large PEs and the multicontext structure.

3.2 PE Structure

Although the supported functions are almost similar, the PE structure is various. First, they can be classified by the input/output registers as shown in Fig. 10:

- Output only (O) : KC256, D-Fabrix, Cluster Machine and ADRES fall into this category. Some of them have a bypassing mechanism which allow the direct interconnection of the PE body.
- Input/Output (I/O): CS2112, DAPDNA-2, FE-GA and Xpp-64 are included. Most of them have the bypassing mechanism.
- Programmable (P): the registers used in DRP-1 can be connected both for input data and output data.

Practically, input registers are used only when the data transfer from the remote PEs or distributed memory module takes a long delay. For the viewpoint of the cost, output register only structure is advantageous. There are no PEs without register or with an input register only.

Next, a PE is characterized whether it provides register files (R) or not (N). The PE array which provides register files can store intermediate data in each PE, for iteratively

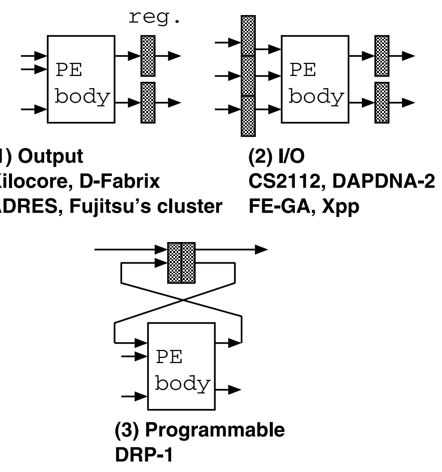
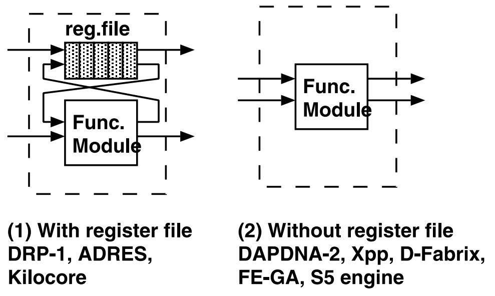


Fig. 10 Classification based on input/output registers.



reg.file
Func. Module
(1) With register file
DRP-1, ADRES, Kilocore
(2) Without register file
DAPDNA-2, Xpp, D-Fabrix, FE-GA, S5 engine

Fig. 11 Classification based on register files.

computing in SIMD/MIMD manner. Figure 11 shows the classification.

Finally, a PE can be categorized by the number of components and interconnection with them as shown in Fig. 12, that is, whether it is consisting of a single or multiple components. When a PE has only a component (S), it is a versatile ALU or Functional Units, and complex operations like shift-and-add can be executed. Such a component can have 2-input (S-2) or 3-input (S-3).

When a PE is consisting of multiple components, they are specialized modules like an ALU, a shift/mask logic, or a multiplier. In FE-GA and Xpp-64, they cannot be connected with each other inside the PE and works independently (M-I). That is, these components only share the input/output of the PE. On the other hand, in DRP-1 and DAPDNA-2, these components can be connected inside the PEs (M-C). The interconnection is changeable with some limitation.

Table 4 shows the summary of PE structure. This table shows that the combination of each feature is various and independent from the fundamental features shown in Table 2. Some of features may come from their main target applications, but it is difficult to find a certain tendency.

[†]The number of D-Fabrix is one which is embedded in ET1D (MeP). The number of PEs in S5-engine has not been disclosed.

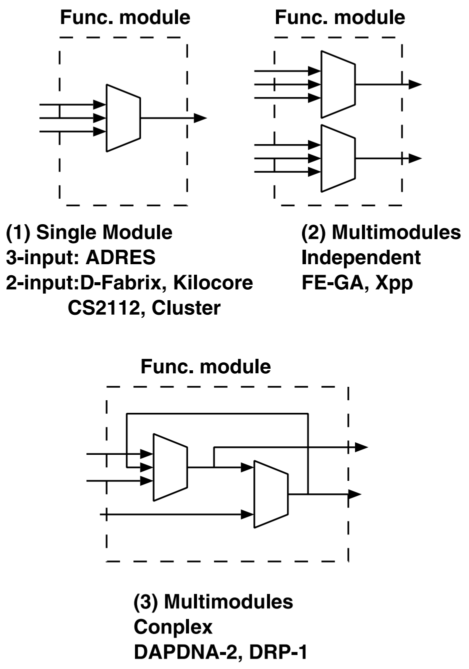


Fig. 12 Component structure.

Table 4 PE structure.

Name	Register	Reg.File	PE body
CS2112	I/O	N	S-2
DAPDNA-2	I/O	N	M-C
FE-GA	O	N	M-I
Cluster machine	O	N	S-2
DRP-1	P	R	M-C
Kilocore KC256	O	R	S-2
ADRES	O	R	S-3
Xpp-64	I/O	N	M-I
D-Fabrix	O	N	S-2

3.3 Interconnection Structure

Interconnection networks used in the PE array is also various as shown in Table 5. The popular interconnection method is a square bus structure (2D-bus) providing a switch matrix at intersection of buses like island-style FPGAs (Fig. 13(1)). Like FPGAs, a switch matrix is statically set with the configuration data. Connection blocks to connect PEs and buses are also required. However, unlike FPGAs, most of buses are uni-directional, that is, the direction of data transfer is fixed. The double, quad and long length wires are not used, and all wires are single length which connect neighboring switches. This comes from the fact that compared with fine grain FPGA, the number of PE is small but its area is large. So, various length wires are not needed and fixed directional data transfer is preferred. Finally, the most important difference is that the setting of switching matrix can be changed by dynamic reconfiguration, especially, every clock cycle in multicontext devices. That is, in such devices, additional wiring resource is available by switching the context.

Table 5 Interconnection structure.

Name	Interconnect
CS2112	Tile base, 2D-bus
DAPDNA-2	Segment base, 2D-bus
FE-GA	2D-mesh direct, Crossbar for memories
Cluster machine	3-stage switch
DRP-1	Tile base, 2D-bus
Kilocore KC256	Crossbar for row direction
ADRES	2D-mesh direct with extra links
Xpp-64	2D-bus and direct
D-Fabrix	Chess-board like switch connection

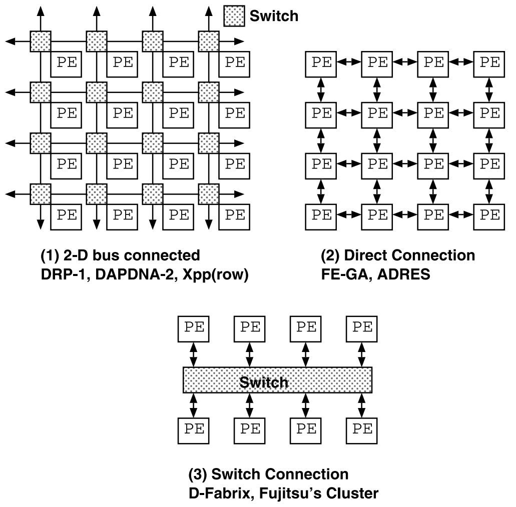


Fig. 13 Interconnection structure.

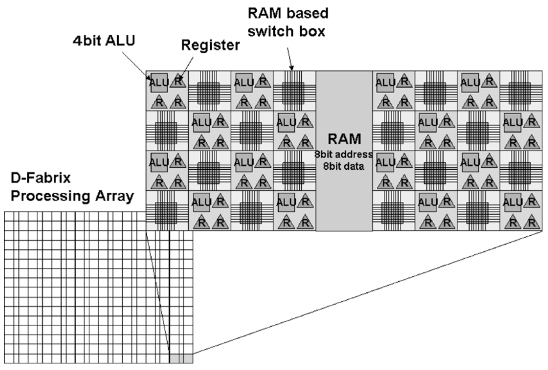


Fig. 14 Interconnection structure of D-Fabrix.

Another popular method is based on direct interconnection (Direct) between PEs like Tile processors (Fig. 13(2)). In FE-GA (Fig. 9), complete nearest neighbor connection is provided between computational PEs, while long additional links are used in ADRES (Fig. 7). In the direct interconnection, the delay for switch matrices and connection blocks of the square bus structure can be omitted. Thus, the low latency data transfer can be done between direct connected PEs. On the other hand, a long delay is needed to transfer data between distant PEs.

Large switching matrix can be directly used to connect a number of PEs as shown in Fig. 13(3). Since the number of PEs connected with a single switching matrix is limited, various types indirect interconnection is used. In D-Fabrix, chess-board like interconnection between PEs and switch matrix is used as shown in Fig. 14. Fujitsu's cluster machine uses three-stage indirect switching network to connect all PEs in a cluster. In FE-GA, although computational PEs are connected directly with wires, they are connected with distributed memory modules with Load/Store units through a powerful switch matrix (Fig. 14). Using this switch matrix, the data stored in memory modules can be transferred to any PEs located at the edge of the array.

4. Application to Wireless Communication

Although dynamically reconfigurable processors have not been utilized for software radio directly, researches on wireless communication using such devices have been reported recently.

Fujitsu's cluster architecture is designed mainly for wireless communication. By using customized clustered structure, it achieves better performance than DAPDNA-2 in several tasks for wireless LAN. A RAKE receiver [38] and OFDM receiver [39] have been implemented using coarse grain dynamically reconfigurable processors. An adaptive Viterbi-decoder [40] was implemented on DRP-1 that can change its structure depending on the S/N (Signal/Noise) ratio to optimize the power consumption. Encryption/Decryption used in wireless communication is one of main target applications of dynamically reconfigurable processors and so various types of implementation have been tried [41].

Dynamically reconfigurable processors have a various benefits for the software radio, but further studies are required especially for decreasing the power consumption.

5. Conclusion

Tables 2, 4 and 5 show that the structure of dynamically reconfigurable processors are various for their target application and the usage in the SoC. That is, there is no architecture which is suitable every target application field. Since the practical use of them has just started, the architectural trade-off has not been well analyzed qualitatively. However, some practical analysis results has been reported recently [7], [18], and in the near future, the optimal structure will be selected automatically when the target application and SoC are fixed.

In this survey, since the targets are focused on systems from companies, important research activities [19]–[22] and early contributions [27], [30]–[36] for establishing dynamically reconfigurable processors are omitted. Some of them can be followed by the references.

References

[1] X. Tang, M. Aalsma, and R. Jou, "A compiler directed approach to hiding configuration latency in chameleon processors," Proc. FPL, (LNCS 1896), pp.29–38, 2000.

[2] T. Sugawara, K. Ide, and T. Sato, "Dynamically reconfigurable processor implemented with IPFlex's DAPDNA technology," IEICE Trans. Inf. & Syst., vol.E87-D, no.8, pp.1997–2003, Aug. 2004.

[3] M. Motomura, "A dynamically reconfigurable processor architecture," Microprocessor Forum, Oct. 2002.

[4] T. Kodama, T. Tsunoda, M. Takada, H. Tanaka, Y. Akita, M. Sato, and M. Ito, "Flexible engine: A dynamic reconfigurable accelerator with high performance and low power consumption," Proc. COOL Chips IX, pp.393–408, April 2006.

[5] M. Petrov, T. Murgan, F. May, M. Vorbach, P. Zipf, and M. Glesner, "The XPP architecture and its co-simulation within the simulink environment," Proc. FPL, pp.761–770, 2004.

[6] T. Stansfield, "Using multiplexers for control and data in D-fabrix," Proc. FPL, pp.416–425, Sept. 2003.

[7] T. Matsumoto, K. Kimura, H. Takano, T. Amatsubo, K. Mori, K. Senda, S. Inoue, and M. Matsui, "Performance evaluation of reconfigurable processing array in area efficiency and operating frequency," Proc. COOL Chips IX, pp.423–434, April 2006.

[8] B. Levine, "Kilocore: Scalable, high-performance, and power efficient coarse-grained reconfigurable fabrics," Proc. Int. Symp. on Advaned Reconfigurable Systems, pp.129–158, Dec. 2005.

[9] F.-J. Veredas, M. Scheppler, W. Moffat, and B. Mei, "Custom implementtion of the coarse-gained reconfiguarble ADRES architecture for multimedia purposes," Proc. FPL, pp.106–111, Sept. 2005.

[10] M. Saito, H. Fujisawa, N. Ujiie, and H. Yoshizawa, "Cluster architecture for reconfigurable singal processing engine for wireless communication," Proc. FPL, pp.353–359, Sept. 2005.

[11] J.M. Arnord, "S5: The architecture and development flow of a software configurable proecssor," Proc. ICFPT, pp.121–128, Dec. 2005.

[12] T. Shiozawa, K. Oguri, K. Nagami, H. Ito, and R. Konishi, "A hardware implementation of constraint satisfaction problem based on new reconfigurable LSI architecture," Proc. FPL 1998 (LNCS 1482), pp.426–430, Aug. 1998.

[13] N. Imlig, T. Shiozawa, K. Nagami, Y. Nakane, R. Konishi, H. Ito, and A. Nagoya, "Scalable space/time-shared stream-processing on the run-time reconfigurable PCA architecture," Proc. RAW 2001, pp.1441–1449, April 2001.

[14] T. Kitaoka, H. Amano, and K. Anjo, "Reducing the configuration loading time of a coarse grain multicontext reconfigurable device," Proc. FPL (LNCS2778), pp.171–180, 2003.

[15] H. Shikano, Y. Suzuki, Y. Wada, J. Shirako, K. Kimura, and H. Kasahara, "Performance evaluation of heterogeneous chip multiprocessor with MP3 audio encoder," Proc. COOL Chips IX, pp.349–363, April 2006.

[16] T. Awashima, "Dynamically reconfigurable processor and its C-level design tool," Proc. Int. Symp. on Advaned Reconfigurable Systems, pp.159–172, Dec. 2005.

[17] S.C. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R.R. Taylor, and R. Laufer, "PipeRench: A coprocessor for streaming multimedia acceleration," Proc. 26th Annual International Symposium on Computer Architecture, pp.28–39, 1999.

[18] Y. Hasegawa, S. Abe, S. Kurotaki, V. Tuan, N. Katsura, T. Nakamura, T. Nishimura, and H. Amano, "Performance and power analysis of time-multiplexed execution on dynamically reconfigurable processor," Proc. Reconfigurable Architecture Workshop (RAW2006), April 2006.

[19] T. Miyamori and K. Olukotun, "A quantitative analysis of reconfigurable coprocessors for multimedia applications," Proc. FCCM, pp.2–11, 1998.

[20] H. Singh, M-H. Lee, G. Lu, F.J. Kurdahi, N. Bagherzadeh, and E.M. Chaves, "MorphoSys: An intergrated reconfigurable system for data-parallel and computation-intensize applications," IEEE Trans. Comput., vol.49, no.5, pp.465–480, 2000.

[21] C. Wolinski, M. Gokhale, and K. McCabe, "A polymorphous computing fabric," IEEE Micro, vol.22, no.4, pp.56–68, Sept./Oct. 2002.

[22] K. Tanigawa, T. Kawasaki, and T. Hironaka, "A coarse-grained reconfigurable architecture with low cost configuration data compres-

sion mechanism," Proc. ICFPT, pp.311–314, 2003.

[23] http://www.picochip.com/

[24] M.B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "The RAW microprocessor: A computational fabric for software circuits and general purpose programs," IEEE Micro, vol.22, no.2, pp.25–35, March/April 2002.

[25] F. Furtek, E. Hogenauer, and J. Scheuermann, "Interconnecting heterogeneous nodes in an adaptive computing machine," Proc. FPL, pp.125–134, 2004.

[26] http://www.clearspeed.com/

[27] A. DeHon, "Dynamically programmable gate arrays: A step toward increased computational density," Proc. Canadian Workshop on Field Programmable Devices, pp.47–54, 1996.

[28] K. Tanigawa, T. Hironaka, A. Kojima, and N. Yoshida, "PARS architecture: A reconfigurable architecture with generalized execution model — Design and implementation of its prototype processor," IEICE Trans. Inf. & Syst., vol.E86-D, no.5, pp.830–840, May 2003.

[29] F. Furtek, E. Hogenauer, and J. Scheuermann, "Interconnecting heterogeneous nodes in an adaptive computing machine," Proc. FPL, pp.125–134, 2004.

[30] S. Trimberger, D. Carberry, A. Johnson, and J. Wong, "A time-multiplexed FPGA," Proc. FCCM, pp.22–28, 1997.

[31] T. Fujii, K. Furuta, M. Motomura, M. Nomura, M. Mizuno, K. Anjo, K. Wakabayashi, Y. Hirota, Y. Nakazawa, H. Ito, and M. Yamashina, "A dynamically reconfigurable logic engine with a multi-context multi-mode unified-cell architecture," Proc. Intl. Solid-State Circuits Conf., pp.360–361, 1999.

[32] M. Wirthlin and B. Hutchings, "A dynamic instruction set computer," Proc. FCCM, pp.99–107, 1995.

[33] R. Witting and P. Chow, "OneChip: An FPGA processor with reconfigurable logic," Proc. FCCM, pp.126–135, 1996.

[34] J. Hauser and J. Wawrzynek, "Garp: A MIPS processor with a reconfigurable coprocessor," Proc. IEEE Symposium on FPGAs for Custom Computing Machines, pp.12–21, 1997.

[35] S. Hauck, T.W. Fry, M.M. Hosler, and J.P. Kao, "The chimaera reconfigurable functional unit," Proc. FCCM, pp.87–96, 1997.

[36] X.-P. Ling and H. Amano, "WASMII: A data driven computer on a virtual hardware," Proc. FCCM, pp.33–42, 1993.

[37] Y. Kurose, I. Kumata, M. Okabe, H. Hanaki, K. Seno, K. Hasegawa, H. Ozawa, S. Horiike, T. Wada, S. Arima, K. Taniguchi, K. Ono, H. Hokazono, T. Hiroi, T. Hirano, and S. Takashima, "A 90 nm embedded DRAM single chip LSI with a 3D graphics, H.264 codec engine, and a reconfigurable processor," Hot Chips 16, 2004.

[38] G.J.M. Smit, P.J.M. Havinga, L.T. Smit, P.M. Heysters, and M.A.J. Rosein, "Dynamic reconfiguration in mobile systems," Proc. FPL, pp.171–181, 2002.

[39] C. Ebeling, C. Fisher, G. Xing, M. Shen, and H. Liu, "Implementing an OFDM receiver on the RaPiD reconfigurable architecture," Proc. FPL, pp.21–30, 2003.

[40] S. Abe, Y. Hasegawa, T. Toi, T. Inuo, and H. Amano, "Adaptive computing on the dynamically reconfigurable processor," Proc. COOL Chips IX, pp.412–421, April 2006.

[41] Y. Hasegawa, S. Abe, H. Matsutani, H. Amano, K. Anjo, and T. Awashima, "An adaptive cyptographic accelerator for IPsec on dynamically reconfigurable processor," Proc. FPT, pp.163–172, Dec. 2006.

Hideharu Amano received the Ph.D. degree from Keio University, Japan in 1986. He is now a Professor in the Department of Information and Computer Science, Keio University. His research insterests include the area of parallel architectures and reconfigurable computing.