TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 30/49 pp169-174
Volume 12, Number S1, July 2007

Fault Tolerance Mechanism in Chip Many-Core Processors*

ZHANG Lei (张 磊)[1,2,**], HAN Yinhe (韩银和)[1], LI Huawei (李华伟)[1], LI Xiaowei (李晓维)[1]

1. Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China;
2. Graduate University of Chinese Academy of Sciences, Beijing 100080, China

Abstract: As semiconductor technology advances, there will be billions of transistors on a single chip. Chip many-core processors are emerging to take advantage of these greater transistor densities to deliver greater performance. Effective fault tolerance techniques are essential to improve the yield of such complex chips. In this paper, a core-level redundancy scheme called *N*+*M* is proposed to improve *N*-core processors' yield by providing *M* spare cores. In such architecture, topology is an important factor because it greatly affects the processors' performance. The concept of logical topology and a topology reconfiguration problem are introduced, which is able to transparently provide target topology with lowest performance degradation as the presence of faulty cores on-chip. A row rippling and column stealing (RRCS) algorithm is also proposed. Results show that PRCS can give solutions with average 13.8% degradation with negligible computing time.

Key words: chip many-core processors; yield; fault tolerance; reconfiguration; network-on-chip

Introduction

Since Moore's law is expected to continue delivering more transistors every process generation, and since platform power and energy budgets are increasingly limited, the trend is to deliver increased performance through parallel computing. Parallel execution in multi-core designs allows us to take advantage of these greater transistor densities to provide greater performance.

In the coming years, the number of cores on-chip will continue to grow. We will migrate from chip multi-core to many-core processors. Intel has deployed R&D on tera-scale computing[1], in which tens to hundreds of processor cores will be integrated on a single chip. An 80-core teraflop processor prototype was demonstrated at Intel Developer Forum 2006.

Chip many-core architectures present many challenges. Yield is one of the most serious problems[2]. Further increases in chip area and devices density result in a decreased yield ratio[3].The profitability of integrated circuits manufacturing depends heavily on the fabrication yield, thus fault tolerance techniques for yield enhancement should be used and will become more and more important[4].

Redundancy is a widely used mechanism to improve yield[4]. As the number of on-chip cores increases, a single core becomes small and inexpensive. Core-level redundancy will bring more benefits than micro-architecture-level. In this paper, we propose a core-level redundancy scheme called *N*+*M*.

For chip many-core processors, communication overhead greatly affects the performance of parallel

Received: 2007-02-01

* Supported by the National Natural Science Foundation of China (Nos. 60633060, 60606008, and 60576031), the National Key Basic Research and Development (973) Program of China (973) (Nos. 2005CB321604 and 2005CB321605) and the fund of Chinese Academy of Sciences (No. 20074010) due to the President Scholarship.

** To whom correspondence should be addressed.
E-mail: zlei@ict.ac.cn; Tel: 86-10-62600719

programs because these cores work cooperatively through on-chip networks. As a result, topology is an important aspect in such architecture. However, if faulty cores exist, on-chip topology may probably be changed. In this paper, we propose the concept of logical topology which is able to encapsulate various different physical topologies. A firmware can be added between operating system and hardware to transform different physical topologies to a unified topology. A row rippling and column stealing algorithm is introduced to configure a logical topology.

1 A Core-level Redundancy Scheme in Chip Many-core Processors

There have been previous attempts to improve yield through means mainly redundancy, other than process control, like memory built-in-self-repair (MBISR). However, the processor core has been left uncovered. Single fault in the processor kills the chip.

Tolerating faults in the processor is hard because it is not as highly-regular and redundant as memory. However, chip many-core processors have made it possible to begin considering core-level redundancy. As a single core becomes smaller and inexpensive, we don't have to pay too much effort to rescue a defective one at microarchitecture-level. Core-level redundancy will bring more benefits than microarchitecture-level. Figure 1a[5] shows that there is a crossover point, about 100 nm, with 6 cores on a chip from which yield adjusted throughput (YAT) of inter-processor is better than that of intra-processor redundancy. In Fig. 1b[6], YAT becomes lower and lower without redundancy as the slashed bars show. It is clear that the white part is longer than the grey part in the same bar, which means that microarchitecture-level redundancy actually brings YAT improvement, but smaller compared with core-level redundancy as technology advances.

One scheme of core-level redundancy is core sparing. A single core on the chip becomes useless if any fault resides in it. However, if enough of the remaining processors are functional, the chip can still be operational but in a reduced version because of performance degradation. For example, a quad-core processor can be degraded to a tri-core, dual-core or single-core processor. Core sparing is a good match of performance binning strategy that separates chips into bins of guaranteed performance levels rather than bins based solely on operating frequency as most manufacturers do today[5]. Core sparing has been adopted in Sun's UltraSPARC T1 processors[7,8].

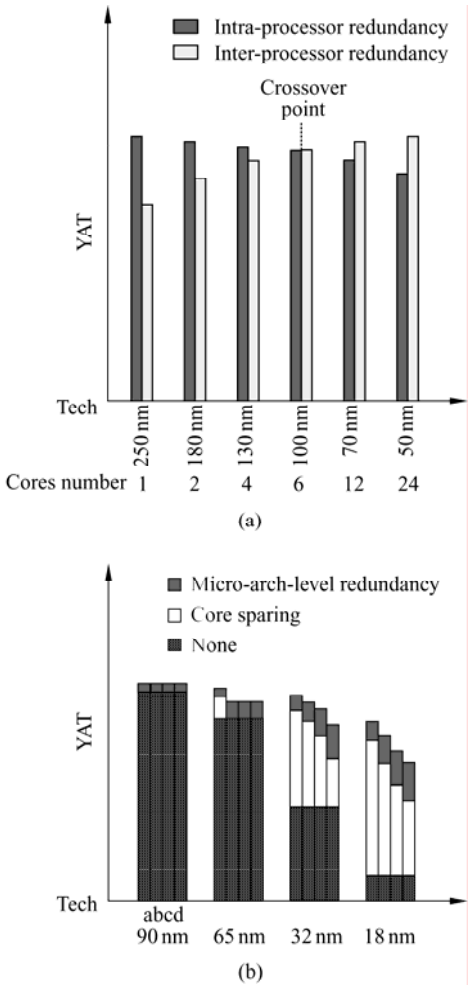

Fig. 1 Core-level redundancy vs. microarchitecture-level redundancy

In an *N*-core processor in which *N* is small, it is applicable to use as many as available operational cores to maximize the utilization of on-chip resources. However, as the number of cores increases we may get many less-than-*N*-core processors and the yield of the demanded *N*-core processor can not be promised. In addition, from a commercial point of view, as there are many degraded versions, it may cause some confusion in marketing.

As a result, we can make use of redundant cores in an alternative way. An *N*-core processor is fabricated with *M* redundant cores, if there are less than *M* faulty cores on a chip, it can be repaired. We always provide *N*-core processors to customers. There are possibly

fault-free cores left unused. As a single core becomes very small, simple and cheap compared with the entire chip, a little waste is acceptable. We call this *N*+*M* mechanism, in which we improve the demanded *N*-core processors' yield by providing *M* redundant cores. In this paper, we focus on *N*+*M* mechanism.

2 Related Research Works

Improving chip yield ratio through core-level redundancy with *N*+*M* scheme is a brand-new research problem. Similar problems exist in MBISR, reconfigurable VLSI array, because all of them have regular topologies as chip many-core processors. Previous research works mainly focused on reconfiguration of a faulty chip. We give some brief introduction in this section.

To avoid yield loss, redundant elements or spare elements are often added to memory design and most faulty cells can be repaired (i.e., replaced by spare cells). Memory is a high-density and highly-regular device, which eases the reconfiguration mechanisms[2,4]. Redundancy allocation has two types: 1-D and 2-D allocation. In 1-D allocation only spare rows or columns are provided while spare rows and columns are both provided in 2-D allocation. Reconfiguration in 2-D spare allocation is NP-complete[9]. Conventional algorithms contain two phases: must-repair phase as in 1-D allocation, and final-repair phase. Research efforts mainly focused on final repair phase. Many algorithms have been proposed. Repair-Most[10] is a greedy algorithm. It constructs a bitmap of faulty cells and reconfigures the device according row and column error counters. Essential spare pivoting (ESP)[10] maintains high repair rate without using a bitmap, thus has small area overhead.

A VLSI array is a large array of simple processing elements and functional units, which are integrated in a single chip to process recurrent algorithms such as image compression and large scale matrix manipulation. To improve the yield of VLSI arrays, fabrication time reconfiguration should be applied to repair faulty elements by spare ones. Rippling replacement[11] eliminates a faulty element by simply replacing it with its neighbor. The function of the replacing element is transferred to the next neighboring unit. Rippling algorithm can accommodate multiple faulty elements as long as no more than one faulty element in the same row if only one spare column is provided. Fault stealing[11] uses both a spare column and a spare row, which overcomes the problem of multiple faults in a row.

The above described related research works can be summarized to a common problem[12]. A physical array is an array after fabrication which may contain some faulty modules (memory cells or processing elements). A logic array is fault-free after reconfiguration. The reconfiguration problem can be formalized as: for a given $R\times C$ physical array and two positive integer r and c, find an $r\times c$ logic array under constraints of $r \geqslant i_1$ and $c \geqslant i_2$.

3 Logical Topology and Topology Reconfiguration Problem

3.1 Reference topology, physical topology, and logical topology

Now return to the *N*+*M* mechanism. The reconfiguration of chip many-core processor is very different from the above one.

In memory and reconfigurable VLSI array, the physical structures must be maintained. As in the above description, an $R\times C$ physical array degrades to an $r\times c$ logical array, both of which are 2-D mesh physically. This is determined by the uses of such VLSI arrays. The interconnections between adjacent processing elements are just hard wires with some MUXes and control lines.

However, in chip many-core processors, each core is an autonomous system which can send/receive packets to/from all other cores through on-chip communication infrastructures. Thus the physical topology can be arbitrary if faulty cores exist on-chip. Each packet will find the way itself.

Since the physical topology can be arbitrary if faulty cores exist, it seems as if the reconfiguration problem is not necessary in many-core processors. However, another problem arises. For example, we plan to fabricate 9-core chips with mesh topology. To improve the chip yield, another 3 spare cores are provided. This is a "9+3" configuration. If there are less than 3 faulty cores on-chip, it can be repaired. However, the target mesh topology is probably changed. As shown in Fig. 2, the topologies are changed except the first one.

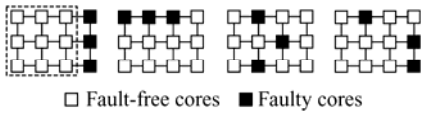

Fig. 2 Faulty cores change the target mesh topology

The performance of chip many-core processors is greatly affected by topology because on-chip cores work cooperatively through on-chip communication infrastructures. Operating system and parallel programmers have to optimize tasks and programs based on topology. For example, Microsoft Windows Server 2003 obtains a description of underlying physical topology through advanced configuration and power interface (ACPI) and provides this information to programmers by API functions[13].

The changed topologies become irregular and cause performance degradation because the average distance between processor cores becomes longer and the flow distribution across the on-chip network becomes unbalanced. The changed topologies also burden programmers because they have to optimize applications on different topologies and optimization on one topology may not work on another.

To deal with these problems, we propose a logical topology concept. In "9+3" chip many-core processors, the "3×3" mesh topology is the target topology or reference topology as shown in Fig. 3a.

If faulty cores exist on-chip, they will be removed from the system (NoC is supposed to be fault-free in this paper), which forms the physical topology. In the physical topology, as in Fig. 3b, the 6th node misses its upper and right neighbors. To transparently provide a target topology to programmers, we use 10th core to replace 9th and 4th core to replace 5th. Though the 10th core has 2 hops distance from 6th node, but they are logical neighbors, the same as 4th node. Logical topo logy as presented by the dotted line in Fig. 3c is a degraded version of reference topology. Programmers always see a unified target topology which greatly eases program optimization.

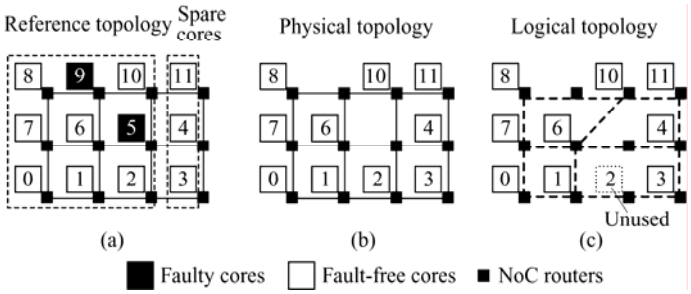

Fig. 3 Reference topology, physical topology, and logical topology

The *N*+*M* mechanism transparently provides demanded *N*-core chips to users no matter how many cores are underlying. Logical topology transparently provides the demanded topology to users no matter how underlying core are connected. They are different aspects of the same problem.

3.2 Topology reconfiguration problem

Logical topology is a degraded version of reference topology and there are many choices to configure a logical topology. We should choose the best one with the least degraded performance.

We define a metric coupling degree (CD) which is used to evaluate the average distance among processor cores. Firstly, we define the coupling degree between two nodes ($CD_{nn'}$) as the reciprocal of the hops between nodes *n* and *n*'. Greater hops result in less coupling of the two nodes, and vice versa.

$$CD_{nn'} = \frac{1}{Hops_{nn'}};$$

The coupling degree of node *n* (CD_n) is defined as the average CD between node *n* and all its logical neighbors.

$$CD_{n'} = \frac{\sum_{\text{all } n'} CD_{nn'}}{k};$$

(Node *n* has *k* logical neighbors)

Lastly, the coupling degree of a logical topology (CD) is defined as the average CD of all nodes in the logical topology.

$$CD = \frac{\sum_{\text{all nodes}} CD_n}{N};$$

(There are *N* nodes in the logical topology)

A larger CD of a topology means more coupling of

the topology, or much closer between nodes, and thus less degraded performance.

As a result, we can conclude a topology reconfiguration problem: find a logical topology with the largest CD for a given physical topology. The ideal CD is 1 as in reference topology. This is a combinational optimization problem. We give a preliminary algorithm to solve this problem in the next section.

4 Row Rippling and Column Stealing Algorithm

Row rippling and column stealing in chip many-core processors (RRCS) is a deterministic algorithm. It is based on the analysis that performance degradation is mainly caused by the physical irregularity of a logical topology compared with reference topology. As a result, RRCS tries to maintain the regularity of logical topologies in row and column unit. Cores which are in the same row in the physical topology are configured in the same row in logical topology as much as possible. Column configuration is the same. It can be used in mesh or torus topologies.

Firstly, scan each row and record defective cores and spare cores of each row as D_r and S_r, respectively. Configure each row according to descending D_r. If $D_r \leqslant S_r$, only row rippling is enough, else both row rippling and column stealing is needed. Cores in a same column are organized as a cyclic queue.

To apply *Row Rippling* on a row, scan the row from left to right. When a faulty core is detected, replace it by the first "*unreplacing*" && "*healthy*" core on its right hand and mark the replacing healthy core as "*replacing*".

In *Column Stealing*, firstly from $(D_r–S_r+1)^{th}$ to D_r^{th} faulty cores, use *Row Rippling*. For 1^{st} to $(D_r–S_r)^{th}$ faulty cores, scan the same column and replace the faulty one by the first "*unreplacing*" && "*healthy*" core. Mark the replacing core as "*replacing*" && "*ill*".

An example of RRCS is shown in Fig. 4. For "16+4" processors with 4×4 mesh reference topology, the physical topology in Fig. 4a only uses *Row Rippling* because there is only one faulty core each row. Fig. 4b uses both rippling and stealing.

RRCS is a simple and fast algorithm, with the time complexity of O(R×C) in mesh or torus topology.

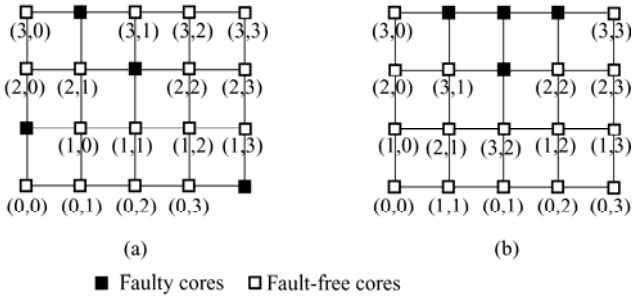

Fig. 4 An example of RRCS algorithm

Table 1 gives the results of RRCS on various different physical topologies. RRCS is able to achieve better CD.

Table 1 CDs of RRCS on various physical topologies

Row	Column	Spare	Faulty	CD
6	6	2	2	0.8259
6	6	4	2	0.8283
6	6	5	2	0.8440
6	6	6	1	0.9528
6	6	6	2	0.9177
6	6	6	4	0.8624
6	6	6	6	0.8216
7	7	7	7	0.8317
8	8	8	7	0.8329
10	10	10	6	0.8870
10	10	10	8	0.8697
10	10	10	10	0.8472
12	12	12	8	0.8846
12	12	12	10	0.8698
12	12	12	12	0.8520

5 Reconfigurable Network-on-Chip Router Architecture

Network-on-chip is the communication backbone of many-core processors as shown in Fig. 5. The network routing logics are distributed to four sides of a processor core. Each side comprises an input and an output controller. For example, NO stands for north output controller and LI for local input controller.

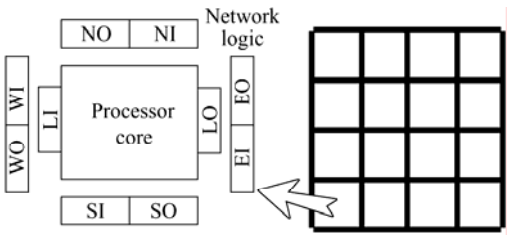

Fig. 5 Network-on-chip in chip many-core processors

Reconfigurable NoC should be adopted to support the *N*+*M* mechanism. We give a design using fuse disconnecting switch as shown in Fig. 6. An array of fuses is inserted before local input controller and after local output controller. If a core with a coordinate (row, col) is faulty, the row and col select lines and programming voltage V_p will open the AND gate. Thus V_{dd} will be applied and disconnect the fuse array. As a result, the faulty core will be isolated from the chip. All packets will pass through the core and no packets will be delivered to it.

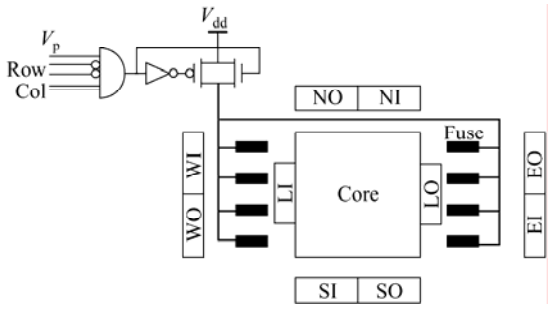

Fig. 6 Reconfigurable NoC router architecture

6 Conclusions

In this paper, we propose an *N*+*M* mechanism to improve the yield of *N*-core processors by providing *M* redundant cores. We also propose the logical topology concept, which transparently provides unified topology to programmers. *N*+*M* mechanism and topology reconfiguration are different aspects of a same problem. They both transparently provide target design goals to users no matter what the underlying actually is. The choice of logical topology is a combinational optimization problem. We propose a preliminary RRCS algorithm which can give better results. We also give a reconfigurable NoC router design to support the *N*+*M* mechanism.

References

[1] Intel White Paper. From a few cores to many: A tera-scale computing research overview. ftp://download.intel.com/rese-arch/platform/terascale/terascale_overview_paper.pdf, 2006.

[2] Koren I, Pradhan D K. Yield and performance enhancement through redundancy in VLSI and WSI multiprocessor systems. *Proceedings of the IEEE*, 1986, **74**(5): 699-711.

[3] Koren I. Should yield be a design objective? In: Proceedings of International Symposium on Quality Electronic Design. San Jose, USA, 2000: 115-120.

[4] Koren I, Koren Z. Defect tolerance in VLSI circuits: Techniques and yield analysis. *Proceedings of the IEEE*, 1998, **86**(9): 1819-1838.

[5] Shivakumar P, Keckler S W, et al. Exploiting microarchitectural redundancy for defect tolerance. In: Proceedings of International Conference on Computer Design. San Jose, USA, 2003: 481-488.

[6] Schuchman E, Vijaykumar T N. Rescue: A microarchitecture for testability and defect tolerance. In: Proceedings of Annual International Symposium on Computer Architecture. Wisconsin, USA, 2005: 160-171.

[7] Parulkar I, Ziaja T, Pendurkar R, et al. A scalable, low cost design-for-test architecture for UltraSPARC^{TM} chip multi-processors. In: Proceedings of IEEE International Test Conference. Baltimore, USA, 2002: 726-735.

[8] Tan P J, Le T, et al. Testing of UltraSPARC T1 microprocessor and its challenges. In: Proceedings of IEEE International Test Conference. Santa Clara, USA, 2006: 1-10.

[9] Kuo S Y, Fuchs W K. Efficient spare allocation in reconfigurable arrays. *IEEE Design and Test of Computers*, 1987, **4**(1): 24-31.

[10] Wang L T, Wu C W, Wen X Q. VLSI Test Principles and Architectures. San Francisco, USA: Morgan Kaufmann, 2006.

[11] Negrini R, Sami M, Stefanell R. Fault tolerance techniques for array structures used in supercomputing. *IEEE Transactions on Computers*, 1986, **19**(2): 78-87.

[12] Fukushi M, Fukushima Y, Horiguchi S. A genetic approach for the reconfiguration of degradable processor arrays. In: Proceedings of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems. Monterey, USA, 2005: 63-71.

[13] Application software considerations for NUMA-based systems. http://www.microsoft.com/whdc/system/platform/server/datacenter/numa_isv.mspx., 2002.