The Teramac Custom Computer: Extending the Limits with Defect Tolerance

W. Bruce Culbertson, Rick Amerson, Richard J. Carter, Philip Kuekes, Greg Snider
Hewlett-Packard Laboratories
1501 Page Mill Road, Palo Alto CA 94304 USA

Abstract

Teramac is a reconfigurable custom computer, capable of running million-gate user designs at one megahertz and out-performing workstations a hundred-fold on highly parallel applications. It achieves these results in spite of thousands of defects. Teramac, composed of 1728 field programmable gate arrays and over a quarter million interconnections, was made possible by a very broad use of defect tolerance. Given a user design and a defect database, a compiler creates a Teramac configuration that implements the design and makes no use of defective resources in the system. System characterization software precisely locates defective resources so that the vast majority of good resources may be applied to user designs. Defect tolerance reduces the cost of Teramac systems by increasing the yields of usable parts and by permitting the use of low-cost components that would otherwise be prohibitively unreliable.

1 Introduction

Broad use of defect tolerance allows the Teramac custom computer [1] to exceed the limits that were believed possible of the materials and techniques used in its construction. Teramac's design goals dictate that it be large and highly interconnected to a degree that is almost unprecedented. Creating such a system with no defects would have required the development of new interconnect technologies and the imposition of extreme quality control measures, making the system prohibitively expensive. Defect tolerance permitted us to build a reliable system using relatively unreliable components that were inexpensive and readily available.

Custom computers [2, 3] are a new class of machines that facilitate the exploration of experimental computer architectures. They typically provide large numbers of programmable gates, wires, and memories that can be configured to implement user designs, which are expressed as netlists of gates. Custom computers execute experimental designs quickly, allowing architects to assess the correctness, quality, and usability of their designs without the expense of building physical prototypes. In addition, on some highly parallel applications, custom computers can achieve speeds rivaling supercomputers. Hence, they can be considered to be formidable computational engines all by themselves, rather than merely platforms on which to develop such machines.

Teramac's goals differ from those of most other custom computers in two important respects. These differences increase Teramac's interconnect requirements which, in turn, makes defect tolerance especially appropriate. First, most custom computers are designed with primarily nearest-neighbor interconnections, with the expectation that users will tailor their designs to fit the custom computer architecture.

Teramac was designed with the goal that its architecture would place minimal constraint on the users' designs. In fact, most Teramac users are not even aware of the Teramac architecture. Achieving this goal requires a far more elaborate interconnection network. A second goal is that user designs would be mapped onto Teramac via a completely automated process. Automatic mapping on many systems commonly fails when all routing resources are exhausted before routing is completed. This problem is minimized on Teramac by providing ample interconnection.

Automatic design mapping provides a natural way to introduce defect tolerance into a custom computer system. The mapping software reads a database of defects and then maps the user design onto resources in the system which are not defective. This process introduces no extra steps for Teramac users and, consequently, most users have been completely unaware that Teramac has defects. The defect database is created by system characterization tests.

Computers using *fault tolerance* continue to run correctly in spite of transient errors or permanent failures that occur at runtime. In contrast, the *defect tolerance* used in Teramac permits it to operate correctly in spite of defects introduced during manufacturing. In addition, it allows Teramac to overcome failures which occur after manufacturing, though normal operations on the system must halt until the new defects have been located and the user designs have been recompiled.

Few large defect tolerant systems have ever been built so, to a large extent, we had to invent ways to build them. Our vendors and even our own internal manufacturing processes were totally unprepared for our requests for defective parts. We learned a lot from our experiences. Some things we did well; others we will do differently in the future.

The programmable gates and wires in most custom computers, including Teramac, are implemented with field programmable gate arrays (FPGAs). The possibility of using defect tolerance with FPGAs and FPGA-based custom computers has been previously mentioned [4-9]. However, we believe Teramac is the first such system to make the use of defect tolerance a reality.

2 The Teramac system

Teramac is a scalable custom computer, with useful systems consisting of one to sixteen boards. A full sixteen-board system is capable of executing user circuits of one million gates at approximately one megahertz. A four-board system is shown in figure 1.

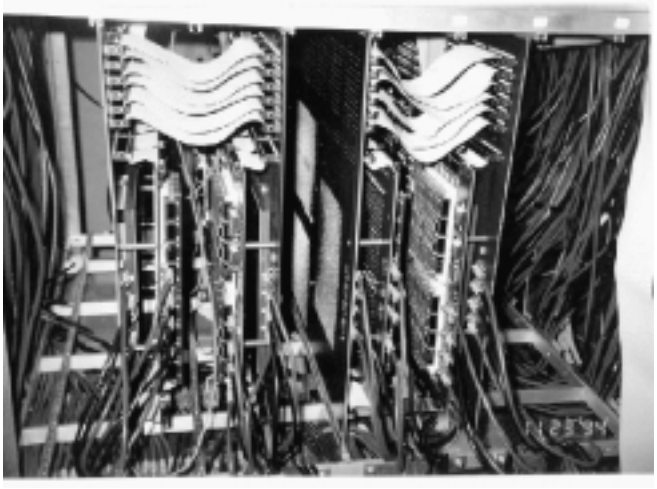

Figure 1. A four-board Teramac system.

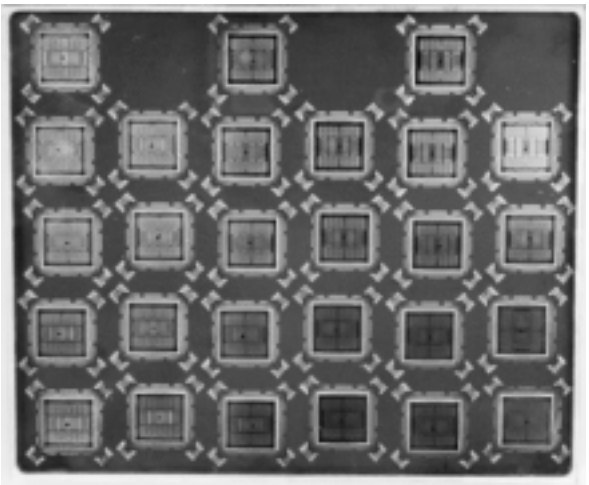

Figure 2. 6.1 inch by 7.4 inch MCM with 27 FPGAs.

Richard Rent [10] observed that, for typical digital designs, the number of signals crossing the boundary of a logic design partition was roughly proportional to the square root of the number of gates in the partition. Since we wanted Teramac to accommodate arbitrary user designs, it was designed with a hierarchical interconnect network satisfying Rent's rule at every level.

A custom field programmable gate array, called Plasma [11], supplies Teramac's programmable gates, registers, crossbars, and multiported register files. The bottom two levels of the interconnect hierarchy are implemented by the Plasma crossbars. Additional Plasma FPGAs provide the switching for the upper levels of the hierarchy; only the crossbars in these FPGAs are used. All FPGAs in the Teramac system are mounted on multichip modules (MCMs) [12]. Each MCM contains twenty-seven FPGAs, as shown in figure 2. The MCMs are mounted on printed circuit boards (PCBs), four per board. PCBs are interconnected via ribbon cables and insulation-displacement connectors. The MCMs, PCBs, and cables provide the wires for the upper levels of interconnect. Each Teramac logic board has associated with it a controller board. Each controller board includes four 8-megabyte RAMs and a microprocessor for transferring data and configurations between Teramac and a host workstation.

3 The Teramac defect tolerance scheme

Experience has shown that a significant number of defective system components (integrated circuits, MCMs, PCBs, cables, etc.) have thousands of perfectly good resources (gates, wires, crossbar lines, for example) for every defective resource. The cost of systems without defect tolerance is substantially increased by the necessity to discard such parts. The objective of using defect tolerance in Teramac is to make productive use of components with small numbers of defects and, in fact, to use nearly all the good resources on such components. This is achieved by first detecting and precisely locating the defective resources using characterization tests. Information about the defects is then stored in a defect database. Then, when user designs are mapped onto Teramac, the mapping software reads the database and maps the design only onto good resources. See figure 3.

Several things complicate the problem of locating the defects. For example, it is not possible to energize and sense arbitrary points in an integrated circuit. Consequently, resources cannot be tested individually. Instead, we combine many resources into test circuits that include some resources that can be probed. A further complication is that no resources are known to be good at the outset of testing. Hence,

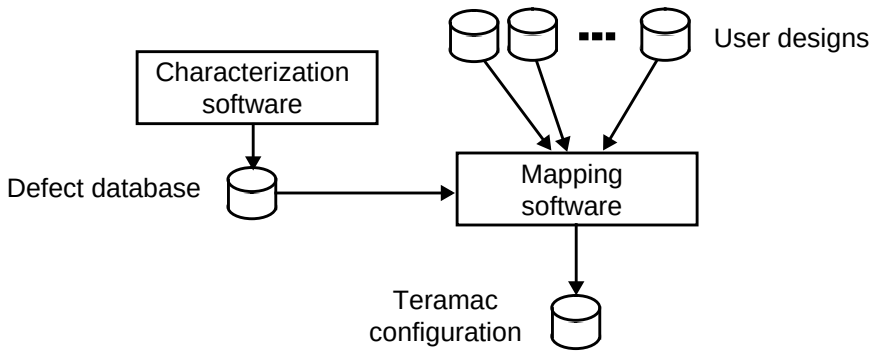

Figure 3. The Teramac defect tolerance scheme. Characterization software precisely locates the defects in the system. Mapping software implements users designs in such a way that defective resources are not used.

it is not possible to construct test circuits with many known-good resources and just a single resource-under-test; this makes it hard to find the bad resources that cause tests to fail.

Most of our characterization tests combine numerous resources into signature generators, circuits that produce long, non-repeating sequences of pseudo-random numbers. On each clock cycle, a new number is produced as a function of its predecessor. A correctly operating signature generator, initialized with a specific value and clocked a given number of cycles, will compute a unique final number, called a *signature.* Any error computed by the circuit will be propagated through the sequence, resulting in an incorrect signature. Thus, signature generation is a sensitive test of the resources constituting the generator. Our signature generator circuits store the current value of the sequence on each clock cycle in FPGA registers. The FPGA scan chain permits the registers to be read and written, which in turn allows the sequence to be initialized and the final signature to be checked.

If a test passes, i.e. a signature generator circuit computes the correct signature, all the resources in the circuit can safely be assumed to be good. On the other hand, if a test fails, the circuit must contain a defective resource. Unfortunately, a failing circuit typically also contains many good resources and an incorrect signature does not allow us to identify which resources in the circuit are good and bad. Consequently, we build the defect database entirely from information obtained from passing tests.

In order to find the good resources despite failing test circuits, we use redundant testing. Specifically, each resource is tested many times, each time grouped with different other resources. This makes it very likely that a good resource will eventually be tested in a group composed entirely of good resources and, hence, will be shown to be good. When all the redundant tests are complete, resources that have not been shown to be good are declared to be bad and are noted in the defect database. Redundant testing is illustrated in figure 4.

The algorithms for mapping user designs onto custom computers are similar to those used for VLSI design. A gate or wire in the user design is assigned a physical gate or wire in the target system. The physical resource is then removed from the pool of resources available for assignment. This continues until all gates and wires in the user design are assigned. Adapting this algorithm for defect tolerance is easy—defective resources are simply removed from the pool of available resources before assignment begins.

Mapping user designs is difficult and, as a result, mapping tools are never able to use 100% of the available resources. Consequently, custom computer designers always add an extra margin of gates and wires beyond the target capacity. We did not substantially increase the margin to support defect tolerance on Teramac.

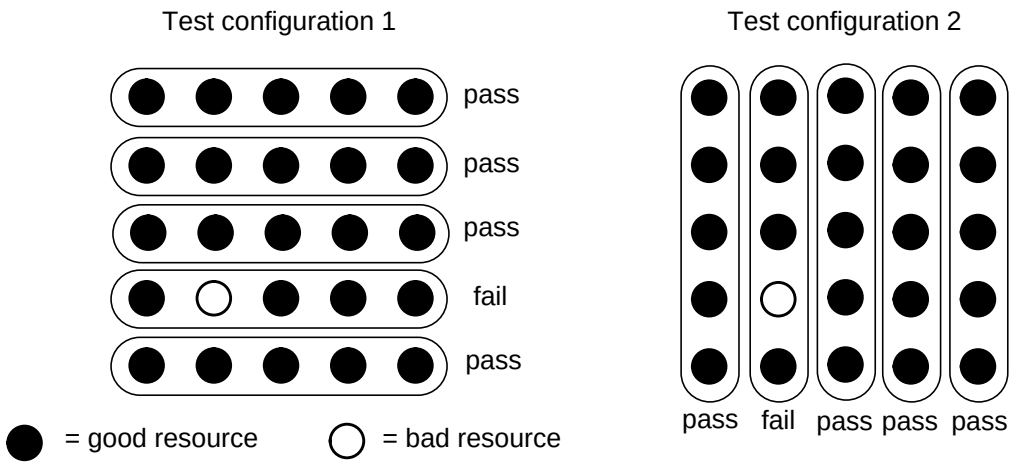

Figure 4. Resources are grouped into test circuits and are declared good after they have participated in a passing test. Thus, after test configuration 1 is run, all the resources except those in row 4 are declared good; the status of those in row 4 remains unknown. In configuration 2, the resources are grouped differently and tested again. After both configurations are run, all the good resources are identified and, indirectly, the defect is located.

4 Design and construction for defect tolerance

Defect tolerance had a large impact on the design and construction of Teramac. Many features were added to the design to minimize the chance that defects would substantially decrease the capacity of a system. Several steps in the assembly of Teramac systems differ from systems without defect tolerance.

The Teramac FPGAs, MCMs, and PCBs each contain critical areas, areas in which a defect would cause a substantial loss of capacity. To maximize yields of these components, a considerable effort was made to minimize their critical areas. The critical areas of components are thoroughly tested before the components are assembled into systems and failing components are rejected. The noncritical areas of components are not comprehensively tested and the testing must detect many noncritical defects for a part to be rejected.

The FPGAs include state machines for reading and writing configurations and scan chains. A defect in a state machine renders the entire chip useless so the state machines are considered critical. Careful design limited the critical area of the FPGA to 7%. The critical area of the FPGAs is tested during wafer testing. The MCMs and PCBs carry many wires which are part of the interconnect hierarchy. These wires are not regarded as critical. Other wires carry clock and control signals to the FPGAs. Defects in these wires make FPGAs unusable; hence, they are considered critical. There are also MCM wires which carry signals from FPGAs to memories; these are also regarded as critical. MCMs contain 6030 wires, of which 4.6% are critical. PCBs contain 8198 wires, of which 7.8% are critical. Although the critical wires are tested before assembly, some redundancy was nevertheless included in their design. For example, four copies of each FPGA control signal are supplied to each MCM, each copy connecting to a quarter of the chips. If one of these control lines is shorted to ground during assembly of the MCM, for example, only a quarter of the FPGAs would become unusable. This would be unfortunate but not fatal. There is also redundancy in the memory signals. For example, more than one FPGA can drive each memory signal.

The FPGAs on an MCM are connected in a cylindrical mesh network for the purpose of reading and writing configurations and scan chains. Messages from a controller board to a specific FPGA are first routed to an FPGA at the top end of the cylinder. The messages contain routing headers, with a sequence of "go-down" and "go-right" instructions, to direct them to their destination FPGAs. As a result, there are multiple paths between each FPGA and the controller board. If one path to an FPGA is blocked by another FPGA which has failed, an alternate path can be found.

The noncritical area of the FPGAs is spot-checked during wafer testing; wafer testers are too limited to permit thorough testing. After considerable trial-and-error, we established a criterion for scoring these tests such that chips that passed had a good likelihood of having high capacity in the system. Testing was complicated by the fact that wafer testers normally reject chips and stop testing after the first test fails, a policy that is incompatible with defect tolerance. It was not easy to make our tester implement our system of scoring multiple tests.

5 Results

Two Teramac systems have been built, one with 8 boards and one with a single board. They have received heavy use [13] and many of the users have been experts in neither defect tolerance nor custom computers. Teramac has met both its performance and capacity goals in spite of thousands of defects in the systems. Table 1 lists the quantities of various kinds of defects found in an actual system.

5.1 Cost reduction due to defect tolerance

Defect tolerance significantly reduced the cost of most of the Teramac components. The FPGAs, MCMs, and inter-board cables are good examples. The eight-board Teramac system contains 864 FPGAs. Two hundred and seventeen of these are free of defects. Thus, defect tolerance increased the yield (and, hence, decreased the cost) of usable FPGAs by a factor of four. When first approached, the MCM vendor had no idea how to price an MCM with defective wires. A contract was finally negotiated in which Hewlett-Packard paid the standard price for perfect MCMs and, for every perfect MCM, also received, free of charge, another MCM with only critical wires guaranteed to be perfect. This, in effect, cuts the MCM price in half. We verified that all the free MCMs we received did, in fact, have defects. Thus, defect tolerance at least doubles the MCM yield.

Resource type	Number	Number defective	Percent defective
Programmable logic cell	221,000	23,000	10.4%
Interchip signal	145,000	13,800	9.54%
Crossbar line	4,880,000	146,000	2.97%
Crossbar buffer	2,420,000	37,000	1.52%
Total	7,670,000	220,000	2.87%

Table 1. Actual defects in the prototype eight-board Teramac system. Interchip defects can be caused by defects in chip I/O pads, chip wire bonds, MCM wires, PCB wires, and/or ribbon cable wires.

Component	Size	Complexity
FPGA	16.2mm by 16.2mm	0.8 micron CMOS
MCM	15.57cm by 18.80cm	39 layers
PCB	44.45cm by 73.66cm	14 layers

Table 2. Defect tolerance permitted high yields to be obtained on Teramac components in spite of their large size and high complexity.

Defect tolerance allows us to interconnect PCBs with one of the least expensive interconnect technologies: ribbon cables and insulation-displacement connectors. Neither this technology, nor any others we are aware of, would be capable of carrying the 38,400 signals with reliability sufficient for a system lacking defect tolerance.

5.2 Extending the limits with defect tolerance

The Teramac design uses materials and technologies in ways that would not have been possible without defect tolerance. The Teramac FPGAs, MCMs, and PCBs are large to an extent that would have been economically impractical, or even infeasible, in a system without defect tolerance. This is because large size significantly reduces yields in conventional systems. The large size of these components allows more gates to be interconnected with fewer chip, MCM, and board crossings. This, in turn, helps Teramac meet its performance goal. Table 2 gives the dimensions of several large Teramac components.

The MCM vendor initially told us that our MCM yield would be too low if our design exceeded twenty layers. With defect tolerance, we achieve an acceptable yield with 39 MCM layers. The Teramac MCM is believed to be the most complex MCM ever made.

5.3 Costs and problems due to defect tolerance

In the process of designing and building Teramac, we encountered some extra costs and problems due to our use of defect tolerance. Some of these are intrinsic to defect tolerant systems while others can be avoided with hindsight.

Ideally, a defect affecting a fraction of the physical area of a part will cause at most a loss of a similar fraction of the functional capacity of the part. We experience two kinds situations in which Teramac loses functional capacity out of proportion to the defect area. The first kind is caused by single physical structures that affect many logical structures. For example, the Plasma configuration memory is composed of 128-bit words and the programmable gates have approximately a hundred configuration bits. Unfortunately, there are configuration words that control the programming of 32 different gates. As a result, a defective configuration word, which can be caused by a very small physical defect, can render 32 gates unusable. This could have been avoided by dedicating a single configuration word for the programming of each gate.

Rent's Rule is the second reason we lost disproportionately large amounts of functional capacity. Rent's Rule says that a partition of N gates will be connected to the rest of the circuit by a number of wires that is proportional to the square root of N. The automatic mapping of designs depends on Rent's Rule being satisfied. Hence, if some fraction F of wires in the L'th hierarchy level are defective, then

only $((1 - F)^2)^L$ gates may be used in the part of the hierarchy below the defective wires. For example, if 1% of the wires in the third hierarchy level are defective, 5.9% of the gates cannot be used.

The characterization tests have costs associated with them. First, it takes time to write them and they are far more complicated than tests that merely detect defects, rather than locate them. Second, development of an effective suite of tests requires experimentation. Early versions of our suite missed certain types of defects so additional tests had to be written. Third, it takes time to run the tests. Currently, it takes a week to run the suite on a single board. There are obvious ways to make our tests run faster but we have not had time to implement them. For example, we currently test one FPGA at a time, although the hardware supports testing hundreds simultaneously.

Although the critical areas on all the FPGAs are tested during wafer test, the critical areas on a few FPGAs nevertheless have failed to operate in the MCMs. Increased attention to static discharge during MCM assembly has improved, but not eliminated, this problem. To date, we have not developed a process to replace these chips, which number about 2% of the total in our eight-board system. Although disappointing, this failure rate has not prevented Teramac from attaining its capacity goal.

6 Conclusion

Using defect tolerance in the Teramac custom computer, we have been able to attain far more ambitious performance and capacity than otherwise would have been possible. Defect tolerance has substantially improved component yields, thereby decreasing the cost of the system, and it has allowed us to use inexpensive, readily available technologies which would otherwise have been prohibitively unreliable. We have developed new strategies for designing, building, and testing defect-tolerant systems. In the end, we have built a robust system that runs applications a hundred times faster than a workstation, yet three-quarters of the integrated circuits from which it was constructed contain defects.

References

[1] R. Amerson, R. Carter, B. Culbertson, P. Kuekes, G. Snider, *Teramac—Configurable Custom Computing*, Proceedings of the 1995 IEEE Symposium on FPGA's for Custom Computing Machines.

[2] P. Bertin, D. Roncin, and J. Vuillemin, *Introduction to programmable active memories*, Systolic Array Processors, Prentice-Hall, 1989, pages 301-309.

[3] J. M. Arnold, D. A. Buell, and E. G. Davis, *Splash 2*, Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures, 1992, pages 316-322.

[4] K. T. Johnson et al, *General Purpose Systolic Arrays*, IEEE Computer, November 1993, pages 20-31.

[5] Neil J. Howard, Andrew M. Tyrrell, Nigel M. Allinson, *The Yield Enhancement of Field-Programmable Gate Arrays*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol 2, No. 1, March 1994.

[6] Jason L. Kelly, Peter A. Ivey, *Defect Tolerant SRAM based FPGAs*, Proceedings of the IEEE International Conference on Computer Design, 1994, pages 479-482.

[7] Nobuo Tsuda, Tsutomu Ishikawa, Yukihiro Nakamura, *Totally Defect-Tolerant Arrays Capable of Quick Broadcasting*, Proceedings of the IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems, November 1995, pages 117-125.

[8] Adit D. Singh, *ADTS: An Array Defect-Tolerance Scheme for Wafer Scale Gate Arrays*, Proceedings of the IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems, November 1995, pages 126-133.

[9] S. Goldberg, S. J. Upadhyaya, *Utilizing Spares in Multichip Modules for the Dual Function of Fault Coverage and Fault Diagnosis*, Proceedings of the IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems, November 1995, pages 234-242.

[10] B. Landman and R. Russo, *On a Pin vs. Block Relationship for Partitions of Logic Graphs*, IEEE Transactions on Computers, December 1971, pages 1469-1479.

[11] R. Amerson, R. Carter, W. Culbertson, P. Kuekes, G. Snider, *Plasma: An FPGA for Million Gate Systems*, Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, February 1996, pages 10-16.

[12] R. Amerson, P. Kuekes, *The Design of an Extremely Large MCM-C—A Case Study*, The International Journal of Microcircuits and Electronic Packaging, Volume 17, Number 4, Fourth Quarter 1994, pages 337-382.

[13] B. Culbertson, R. Amerson, R. Carter, P. Kuekes, G. Snider, *Exploring Architectures for Volume Visualization on the Teramac Custom Computer*, Proceedings of the 1996 IEEE Symposium on FPGA's for Custom Computing Machines.