Novel Techniques for Achieving High At-Speed Transition Fault Test Coverage for Motorola's Microprocessors Based on PowerPC^{TM} Instruction Set Architecture

Nandu Tendolkar, Rajesh Raina, Rick Woltenberg
Motorola, Inc.
7700 W. Parmer Lane, Austin, Texas 78727
Xijiang Lin, Bruce Swanson, Greg Aldrich
Mentor Graphics Corporation
8005 SW Boeckman Rd, Wilsonville, Oregon 97070
Contact e-mail: Nandu.Tendolkar@motorola.com, Tel.512-996-6093

Abstract

Scan based at-speed transition fault testing of Motorola's microprocessors based on the PowerPC^{TM} instruction set architecture requires broad-side transition fault test patterns that have a specific launch and capture clocking sequence. We describe the concepts we developed and incorporated in the ATPG tool to support efficient generation of such test patterns to achieve high transition fault test coverage and for analysis of undetected transition faults. Using the enhanced ATPG tool, we generated 15,000 transition fault test patterns and achieved 76% test coverage for the MPC7400 microprocessor based on the PowerPC^{TM} instruction set architecture that has 10.5 million transistors and runs at 540 MHz. Keywords: Microprocessor, Delay Testing.

1. Introduction

The presence of random defects can give rise to circuits with very slow-to-rise or very slow-to-fall switching transitions[1]. Such a defect can cause an increase in the delay of the paths going through the circuit. A microprocessor fails to operate at rated speed if it contains a delay defect that creates a path delay higher than the maximum allowed path delay[2]. The purpose of a delay test is to verify that a microprocessor operates correctly at the specified clock speed. Researchers have proposed two types of fault models for dealing with generating test patterns for delay defects detection. In the transition fault model[3], a gate output has a slow-to-rise and a slow-to-fall fault associated with it. In the other delay fault model, called the path delay fault model [4], a chip contains a path delay fault if it has a path whose delay exceeds a specified value. In this paper we use MPC74xx to refer to Motorola's microprocessors based on the PowerPC^{TM} instruction set architecture. We focus on the transition fault model test pattern generation, application and results for MPC74xx.

At-speed transition fault tests are used in the production testing of MPC7400 and MPC7450 [5,6]. MPC74xx is a LSSD design. Test pattern data is scanned into the latches and the test results are scanned out at slow tester speed (80 MHZ). The functional clocking required for at-speed (microprocessor speeds are 500 MHz or higher) testing is provided by an on-chip clock controller circuit [7].

Since functional clocks are used for launching the transition, the test is a broad side delay test. Broad side delay test is studied in [8]. When broad side delay test is used for LSSD designs, the first vector of the pair is applied using scan. Functional clock or clocks are applied to switch one or more latches to get the second vector of the pair. Thus the second vector is the combinational circuitry's response to the first vector. In at-speed test the response is captured by an at-speed clock in latches. Primary output pins are not used as observation points since the tester speed is slow. Further, because of slow tester speed, the primary input pins are held at a constant value and cannot launch a transition. These aspects of the design must be taken into consideration by the Automatic Test Patterns Generation (ATPG) tool.

In our previous work [7,9] we had addressed the problem of generating transition fault test patterns for the MPC7400. The ATPG tool available at that time could not generate transition fault test patterns for majority of faults. Excessive run time for test pattern generation had resulted in low test coverage. For one MPC74xx chip, the number of undetected faults were close to a million and our estimation showed that the ATPG run time would be a year for these undetected faults. We also had no debugging tools to determine why the ATPG tool was unable to generate a test pattern for a specific fault. This made it very difficult for us to identify how to improve the test coverage. Since we generated test patterns using ad hoc methods, the test pattern volume was very high. Some test patterns generated by the ATPG tool did not have the correct clocking sequence and had to be discarded. Although we had proved the concept of at-speed transition fault testing by using the tests for screening bad chips in manufacturing,

Proceedings of the 20 th IEEE VLSI Test Symposium (VTS'02)
1093-0167/02 $17.00 © 2002 IEEE

we were not meeting our test coverage and test pattern volume goals. It was clear that we needed an enhanced ATPG tool so that we could do the following for transition faults:

- ATPG tool must be able to generate transition fault test patterns in a reasonable amount of time on a fault list containing 7 million faults and the test patterns generated must have clocking sequences supported by the at-speed clock controller.
- It must be able to achieve test coverage of 80% or higher.
- ATPG tool must generate a minimal set of test patterns. Tester memory is limited and it was desirable to keep the test pattern volume under 15000.
- The ATPG tool must provide a mechanism for us to determine the cause of why it is unable to generate a test pattern for a fault.
- Transition fault ATPG is a sequential test pattern generation process and is inherently slow. We needed to invent techniques to speed up the process.

In this paper we present several ideas that we have used to enhance our ATPG tool to address the goals mentioned above. We also present examples of how the new functions added to the ATPG tool are used to attack the problem of transition fault test pattern generation and test coverage improvement. The paper is organized as follows. Section 2 presents the clocking sequences that need to be supported by the ATPG tool. In Section 3, we describe the new functionality added to our ATPG tool to specifically support at-speed transition fault ATPG and how it relates to the goals stated above. Experimental results are included in Section 4. Section 5 contains conclusions. Future challenges are described in Section 6.

2. At-speed Clocking Sequences

The clock controller supports the generation of specific clocking sequences. The first clock generated by the clock controller is a slow clock, which is followed by a sequence of at-speed clocks and the last clock generated is a slow clock. Since at least two consecutive at-speed clocks are required for at-speed transition fault test, the clock sequence in each test pattern must include no less than four clocks.

There are two kinds of latches in the LSSD design used in MPC74xx, master latch and slave latch. Functional data input to a master latch is clocked into it by a C1 clock and the functional data input to a slave latch is clocked into it by a C2 clock. We first consider the clocking sequence used for testing of logic between a master latch and a slave latch.

Combinational logic between a master latch and a slave latch is tested by launching a transition from one or more master latches and propagating the fault effect to one or more slave latches. Obviously, this can be accomplished by applying a C1 clock followed a C2 clock provided that both clocks are at-speed clocks. The strategy used in MPC74xx is to program the clock controller so that it will generate a clocking sequence that includes the desired at-speed clocking subsequence. For example, the clock controller can be programmed to generate the clocking sequence {C2, <C1, C2>, C1}, where the clocks enclosed in the angle brackets are at-speed clocks. This clock sequence provides the at-speed launch and capture clocks needed to detect the faults in the combinational logic between a master latch and a slave latch.

The reasons why the first clock is slow and the last clock is slow are explained as follows.

Transition fault test begins with a scan operation with the chip in scan mode. At the end of the scan operation, the tester switches the chip under test to functional mode. The first functional clock appears as soon as the chip switches from the scan mode to the functional mode. The tester then asserts a pin (called *ACSTART*) to inform the controller to generate the programmed at-speed clocking sequence. The clocks start switching at-speed some time later. Once the clock starts switching, all subsequent clocks are at-speed until the final clock of the clock sequence becomes active. Once the final clock turns on, it remains on until the chip under test is switched from functional mode to scan mode by the tester. The switch from scan mode to the functional mode and vice versa is controlled by the tester and can be at any slow speed, which is a design requirement.

In Figure 1, we show a clock control waveform used to test a transition fault in the logic circuits between master latches and slave latches. The clocking sequence consists of four clocks {C2, <C1, C2>, C1}. The pins, *SCAN_EN*, *L1TST/A, L1TST/B* and *ACSTART* are chip pins controlled by the tester. *L1TST/A* and *L1TST/B* are shift-A and shift-B clocks for scan shifting. The switch from scan mode to the functional mode occurs when the signal *SCAN_EN* becomes low. This turns the clock C2 on. The first at-speed clock C1 occurs some time after the *ACSTART* pin is asserted by the tester and it is followed by the second at-speed clock C2. The last clock C1 remains on until *SCAN_EN* becomes high.

3. Architecture of New Features for Transition Fault ATPG

In this section, we present three new techniques used for at-speed transition fault test generation. The proposed techniques include user defined clocking sequence for supporting embedded PLL, fault classification for speeding up test generation process, and analysis of undetected faults for debugging purposes. We also give examples of

Proceedings of the 20 th IEEE VLSI Test Symposium (VTS'02)
1093-0167/02 $17.00 © 2002 IEEE

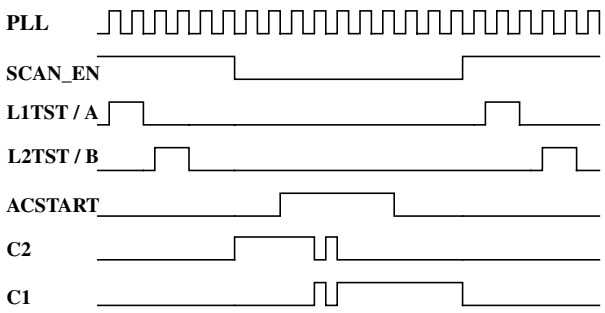

Figure 1: Clock Control Waveforms

how we use them in MPC74xx transition fault ATPG.

3.1. User Defined Clocking Sequences

In general, to generate a test pattern that detects a targeted fault, deterministic ATPG is not concerned about what type of clocking sequence constitutes the generated test pattern as long as the test pattern detects the targeted fault at an observation point. Only the clock sequences supported by the clock controller can be used to test MPC74xx. Therefore, the ATPG tool was enhanced to allow the user to define the clocking sequence information to generate test patterns correctly.

For the MPC74xx design, the clock controller circuit can produce clocking sequences that start with a slow clock and end with a slow clock, with at-speed clocks in between. To provide these clocking sequences to ATPG we use four procedures named launch procedure, capture procedure, post capture procedure, and clock order procedure. These procedures describe each clocking sequence and will be loaded by the ATPG tool to guide the test generation process. The purpose of describing each clocking sequence by four procedures is to distinguish the at-speed clocks from the slow clocks. When testing transition faults with a test pattern, only the at-speed clocking cycles are able to activate the fault sites. The slow clocking cycles can only be used for propagating the fault effect to observation points if they follow the at-speed clocking cycles. We will describe each procedure in detail next.

1. The launch procedure is used to describe the clocking sequence right before the second at-speed clock in the complete clocking sequence, i.e., all clocks except the one applied last are slow clocks. The fault site cannot be activated in any cycle of this procedure.
2. The capture procedure is used to describe the clocking sequence between the second at-speed clock and the last at-speed clock in the complete clocking sequence. The fault site is able to be activated at any cycle in this procedure when the fault activation condition is satisfied.
3. The post capture procedure is used to describe the slow clocking sequence between the last at-speed clock and the scan unloading. The clocking cycles defined in this procedure are used for propagating the fault effect only. The fault site cannot be activated in any cycle of this procedure because all clocks in this procedure are slow clocks.
4. The clock order procedure is used to ensure that the ATPG tool uses clocking sequences that are alternating C1/C2 clocks. We do not permit consecutive sequence of C1 clocks or C2 clocks (e.g. C1-C1 or C2-C2 is not allowed) in the test patterns.

After loading a user defined clock procedure into the ATPG tool, the test generation process creates an ATPG window including as many clock cycles as specified in the user defined clocking sequence. Then it applies the logic values declared in the four procedures to corresponding cycles in the ATPG window. All implied values are fixed during test generation. To generate a test pattern for a target fault, the fault is injected at a clock cycle defined in the capture procedure and ATPG tries to generate a test pattern that can detect the fault at an observation point in the last clock cycle defined in the post capture procedure.

We illustrate the declaration of a clocking sequence with an example. Suppose we want to generate transition fault test patterns that have the following clocking sequence: {C1, <C2, C1>, C2}.The following statements specify the clocking sequence to the ATPG tool:

```
CLOCK LAUNCH =
  PULSE /C1  1;
  PULSE /C2  2;
END;
CLOCK CAPTURE =
  PULSE /C1  1;
END;
CLOCK POST_CAPTURE /C1 =
  PULSE /C2  1;
END;
CLOCK ORDER =
  /C1 /C2 ;
  /C2 /C1 ;
END;
```

The last C2 clock is not at-speed and the values captured by the previous at-speed C1 clock are used as the final results of the test. At the end of the above clocking sequence, the first scan clock would be a shift-B clock to move data from the master latches to the slave latches.

In general, using this feature we can specify clocking sequences that begin with a slow functional clock, apply two or more at-speed clocks and end with a slow clock. The ATPG tool generates test patterns that match the

defined clocking sequence.

At-speed transition fault simulation is used to find the transition faults detected by an at-speed transition fault test pattern. When fault simulating the test patterns generated by the deterministic test generator, the fault simulator must be enhanced to take the at-speed clocking information into account in order to obtain accurate fault coverage. First, fault activation conditions are checked at the at-speed clocking cycles only and all slow clocking cycles are treated as fault-free. Next, a fault effect can only be measured at the observation points in the last clocking cycle of the test pattern being simulated.

3.2. Fault Classification

ATPG may require more than one clocking sequence to detect all the detectable faults in a design. A gate is said to be in a clock domain *C* if the gate reaches only to the data inputs of the latches/DFFs controlled by the clock *C*. A fault that is in the clock domain of clock C2 cannot be detected by a clocking sequence that uses clock C1 as the capture clock. If the deterministic test generation targets a fault by applying a user defined clocking sequence that cannot detect the fault, the test generation effort is wasted. Fault classification allows us to select faults that are appropriate for a particular clocking sequence. Based on the above observation, we use several criteria described below to classify a fault set into different categories:

- *Data cone and clock cone*: Under this criterion, the fault set is classified into four categories: faults reaching only to the data inputs of the latches/DFFs, faults reaching only to the clock inputs of the latches/DFFs, faults reaching only to both the data inputs and the clock inputs of the latches/DFFs, and the leftover faults.
- *Clock domain*: Under this criterion, the fault set is classified into *N*+1 categories, where *N* is equal to the number of clocks in the design. All faults in a clock domain are placed in the same category and the leftover faults are placed in an unclassified category.
- *Sequential depth*: Under this criterion, the fault set is classified into two categories, faults whose observation depth is less than *D* and the faults whose observation depth is greater than *D*, where *D* is a user specified number. Here, the observation depth is defined as the minimum number of sequential elements between the fault site and any observation point. When the clocking sequence length is equal to *D*, all faults in the second category are untestable faults.
- *RAM*: Under this criterion, the fault set is classified into five categories: faults reaching only to RAM control lines, faults reaching only to RAM address lines, faults reaching only to RAM data lines, faults reaching only to RAM inputs, and the leftover faults.
- *Primary input*: Under this criterion, the fault set is classified into *N*+3 categories, where *N* is equal to the number of clocks in the design. We place faults reachable only from non-clock primary input through combinational logic in one category (if the non-clock primary inputs must be held during testing, all faults in this category are untestable faults) and faults reachable from more than one clock through combinational logic in another category. For the leftover faults, faults reachable from each clock through combinational logic are placed in *N* different categories and all unclassified faults are placed in the last category.
- *Primary output*: Under this criterion, the fault set is classified into two categories, faults observable only at primary outputs and the leftover faults. When all primary outputs are masked during testing, faults in the first category are untestable faults.

Different classification criteria can be used iteratively to classify each fault category further. For example, we classify the fault set based on the criterion data cone and clock cone first. The faults in the data cone are classified further by using the criterion clock domain. For the faults in each clock domain, we apply different user defined clocking sequences to generate test patterns. By selecting appropriate user clocking sequences for each fault category, the test generation process is sped up significantly.

An example of the use of fault classification to reduce the ATPG time for one MPC4xxx chip is presented next. The fault list for this chip had 3,527,560 faults. We wanted to determine how many faults propagate to the data port of the master latch (MLD) and how many faults propagate to the data port of the slave latch (SLD) so that we can use the {C1, <C2, C1>, C2} clocking sequence on the former and the {C2, <C1, C2>, C1} clocking sequence on the latter. Using fault classification (which runs much faster than test pattern generation) we found the faults in MLD class and the SLD class. The MLD class contained 2,345,478 faults and the SLD class contained 472,702 faults.

After the faults are classified, we start two sessions of the ATPG, one for the faults in the MLD class and the other for the faults in SLD class. Test patterns are generated in parallel. This speeds up the time required to generate the test patterns. Without fault classification, ATPG has to be done serially, i.e., using one clock sequence on the entire fault list first, and then using the other clocking sequence on the remaining undetected faults. The reduction in run time is obvious here. Assuming that the test coverage for MLD faults is 80%, it can be easily seen that total number of faults processed using fault classification is 17% lower than the total number of faults processed without using fault classification.

3.3. Analysis of Undetected Faults

When the at-speed test coverage achieved by ATPG is unacceptably low, it is important to have debugging tools to help the designers find the reasons that cause the test generation problem. Based on reasons found, designers may modify the design to improve the test coverage.

For the MPC74xx designs, **fault analysis and test stimulus analysis** procedures are implemented for debugging aborted/untestable faults.

Fault analysis consists of the following steps:

- *Observability analysis*: Find all potential observation points based on the sequential depth specified in the user defined clocking sequence. If the sequential depth is too small, the minimum sequential depth required to control and observe the fault is reported.
- *Controllability analysis*: To activate a transition fault site, it must be possible for the fault site to take logic values 0 and 1. This analysis is used to determine if the fault site can be set to the logic values 0 and 1.
- *Launchability analysis*: This analysis is used to determine if a transition can be launched at the fault site when applying the user defined clocking sequence.
- *Stuck-at fault analysis*: A transition fault is untestable if its corresponding stuck-at fault is untestable. This analysis focuses on applying the user defined clocking sequence to test the stuck-at fault at the transition fault site.
- *Observation point analysis*: When fault analysis fails to propagate the fault effect to an existing observation point, the observation point analysis allows the user to insert a new observation point in the fault propagation path and ATPG tries to propagate the fault effect to the new observation point by applying the user defined clocking sequence.

Test stimulus analysis focuses on analyzing the controllability of a set of gate tuples simultaneously. A gate tuple is defined as $\{g, v, c\}$ and is used to specify control of the gate g to take logic value v at cycle c in the user defined clocking sequence. One application of the test stimulus analysis is to analyze an uncontrollable gate. We may trace back from the uncontrollable gate and analyze the controllability of the gates in its support cone iteratively until the desired information is collected. Another application of the test stimulus analysis is for sensitization analysis, i.e., we specify a set of gate tuples which are used to sensitize a propagation path partially and check if these gates can be justified simultaneously.

Next, we give a brief description of how these procedures are used for improving MPC74xx test coverage.

We run the fault analysis procedure on an undetected fault and find whether a transition can be launched at the fault site. If the transition cannot be launched at the fault site, we use the test stimulus procedure to find the sources that cause the problem. When the transition can be launched successfully at the fault site, there must be no path from the fault site to an observation point which can be sensitized. We then use the fault analysis procedure again by inserting an artificial observation point at a gate located in the path between the fault site and a real observation point. If the fault can be detected at the inserted observation point, we will pick another gate in the same path. This step is repeated several times until the fault cannot be detected at the inserted observation point. Next, we use the test stimulus analysis procedure to find the cause of the blockage.

The usage of the test stimulus procedure is explained here by using an AND gate with two inputs a and b, and an output c. To create a rising transition on c, there are two choices:

- b=(0,1) and a=(X,1), where the numbers in parenthesis are logic values assigned at the first and second test cycles.
- b=(X,1) and a=(0,1).

The test stimulus analysis procedure allows us to specify the desired logic values of a and b at different test cycles, and answer the question about if a test pattern exists that can create the specified transition at the gate's inputs. This allows us to find whether input a or input b is the source of the problem. Let's suppose that the test stimulus analysis procedure can create the transition a=(0,1). Then we can conclude the input b is the source of the problem. By using the test stimulus procedure recursively, we can identify which latches or primary input pins make the fault undetectable.

In one of our netlists, we used a combination of the fault analysis procedure and the test stimulus analysis procedure to find that a large number of undetected faults were caused by the scan latches whose data inputs are tied to 0. This makes the latch's output be 0 after the latch's clock is turned on. Detection of these faults required that this latch should be at 1 for two clock cycles. By changing the design as follows, we make these faults detectable: The output of the latch was fed to its data input. Thus the latch output stayed at 1 if a 1 was scanned into it.

4. Experimental Results

We have tested the enhanced ATPG tool on two MPC74xx chips. We compared the performance of the original ATPG tool to that of the enhanced ATPG tool by running both on the MPC7400 chip. On MPC7400 chip, we spent several months to generate 23,000 test patterns to achieve 73.8% test coverage by using the original ATPG tool and manually generated test patterns. Using the

enhanced ATPG tool, in two weeks, we were able to generate 15,000 test patterns that achieve 76% test coverage.

On a new MPC74xx core which has 3,173,295 faults, we generated 38,894 test patterns in150 hours to achieve 88.07% test coverage. On this netlist, we were also able to quickly find why a set of 87,855 faults could not be detected. After analyzing these faults by using fault analysis and test stimulus analysis procedures, we quickly identified the sources that caused the problem. An easy design change made these faults detectable and the test coverage increased by 3%.

5. Conclusions

In this paper, we proposed several new techniques to target at-speed transition fault testing. The experimental results demonstrated the effectiveness and efficiency of the proposed technologies in reducing the ATPG run time to an acceptable level for netlists containing over three million faults and achieving over 85% test coverage in a reasonable amount of time. By using the developed debugging tools, we are able to identify the design changes required for meeting our test coverage goals. However, test pattern volume is still high and we are working on new methods to reduce the test pattern volume.

6. Future Challenges

Continued technology enhancements will make even larger, more complex, and faster chips available each year. These designs are expected to be more sensitive to delay defects. With wide acceptance of at-speed scan based testing for delay defects, there is a need to reach the same high level of test coverage for delay faults that is typically achieved for stuck-at faults. This means that test coverage for delay faults will need to be in excess of 90%. Continued improvement and innovation in ATPG algorithms and performance as well as at-speed debugging capabilities, will be required to achieve these high coverage targets while still keeping total pattern counts in check. With these high coverage requirements, design teams will also need to focus on design for at-speed testability early in the design phase. Focusing only on stuck-at coverage during the design phase may not be sufficient to yield adequate delay fault coverage.

As the number of circuits on a chip continues to increase, the number of latches in it also increases. Because of the limited number of pins available for scan, this results in higher numbers of latches per scan chain. This requires us to test larger designs with fewer test patterns and still get higher test coverage due to the tester memory limitations. To contain test pattern volume, significant innovations in compression technology will also be required.

Acknowledgment

The authors would like to acknowledge the contributions of Dawit Belete, Ashu Razdan, Carol Pyron, Bruce Long, Bob Bailey and George Joos of Motorola, Inc. and of Rob Thompson of Mentor Graphics.

References

[1] P. Nigh, D. Vallett, A. Patel, J. Wright, F. Motika, D. Forlenza, R. Kurtulik, W. Chong, “Failure Analysis of Timing and IDDq-only Failures from SEMATECH Test Methods Experiment,” *Proc. International Test Conference*, 1998, pp. 43-52.

[2] N. N. Tendolkar, "Analysis of timing failures due to random AC defects in VLSI modules," 22*nd Design Automation Conference Proceedings*, June, 1985, Las Vegas, pp. 709-714.

[3] John A. Waicukauski et. al, “Transition Fault Simulation by Parallel Pattern Single Fault Propagation,” *Proc. International Test Conference*, October 1986, pp. 542-549.

[4] G. L. Smith, “Model for Delay Faults Based on Paths,” *Proc. International Test Conference*, 1985, pp. 342-349.

[5] C. Pyron, M. Alexander, J. Golab, G. Joos, B. Long, R. Molyneaux, R. Raina, and N. Tendolkar, “DFT Advances in the Motorola’s MPC7400, a PowerPC^{TM1} Microprocessor,” *Proc. International Test Conference*, 1999, pp. 137-146.

[6] R. Raina, R. Bailey, D. Belete, V. Khosa, R. Molyneaux, J. Prado and A. Razdan, “DFT advances in Motorola’s Next-Generation 74xx PowerPC^{TM} microprocessor,” *Proc. International Test Conference*, 2000, pp. 131-140.

[7] N. Tendolkar, R. Molyneaux, C. Pyron and R. Raina, “At-speed Testing of Delay Faults for Motorola’s MPC7400, a PowerPC^{TM} Microprocessor,” *Proceedings of the 18th IEEE VLSI Test Symposium*, April 2000, pp.3-8.

[8] Jacob Savir and Srinivas Patil, “Broad Side Delay Test,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol 13, No. 8, August 1994, pp. 1057-1064.

[9] Nandu Tendolkar and Rajesh Raina, “A Study of FastScan^{TM} Transition Fault ATPG Capability for PowerPC^{TM} Microprocessors,” *Mentor Graphics Users Group International Conference*, 1998

1. MOTOROLA and the Stylized M Logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners. © Motorola, Inc. 2001.