problems that can be debugged. Test cells can perform internal scan as well as boundary scan operations. Internal scan can help chip designers debug their logic, because all of the flip-flops in a chip can be incorporated into a scan chain. The scan chain can be loaded with a specific test state, the chip can be clocked once, and then the new state can be shifted out. This is akin to single-stepping software. While not speedy, JTAG provides excellent visibility into a chip's inner workings with little added cost.

Boundary scan is the more interesting side of JTAG for board-level debugging, because it allows the scan chain to drive and sense I/O pins independently of the logic that normally controls them. This means that the connections between two JTAG-equipped ICs can be fully verified electronically. One device can drive a wire, and the other can capture that driven state and report it back to the JTAG test program. JTAG is an excellent complement to BGA technology, because it can rapidly determine the correctness of otherwise hidden connections. Most connections can be verified only when the ICs at each end both contain JTAG logic. Connections from a JTAG-enabled IC to a memory IC often can be tested by writing data to the memory through the boundary scan chain and then reading it back in the same manner. JTAG logic is very common in BGA-packaged ICs, but not as common with other packages. If you are designing with BGA or very fine-pitch packages, JTAG support is a criterion that should be considered when selecting components. In some cases, there is a choice between vendors who do and do not support JTAG. In other cases, you may have to select a non-JTAG device and accept the reduction in testability.

The JTAG interface is designed to be easy to implement so that it does not place undue demands on already complex boards. Four primary signals compose the interface: test mode select (TMS), test clock (TCK), test data input (TDI), and test data output (TDO). A fifth signal, test reset (TRST*), is supported by some vendors but is not strictly necessary for JTAG operation. Devices are daisy chained by repeatedly connecting TDO to TDI until a chain has been formed with a set of TDI and TDO signals as shown in Fig. 19.9. TMS, TCK, and TRST* are bussed to all devices. One of the first questions that comes up in synchronous design is a circuit's tolerance for clock skew. JTAG can tolerate almost arbitrary clock skew, because inputs are sampled on the rising edges of TCK, and outputs are driven on the falling edges. Almost half a clock period of skew is permissible. If JTAG is run at several megahertz, the skew is unconstrained for all practical purposes, because hundreds of nanoseconds corresponds to wire lengths well in excess of what can fit onto a normal circuit board. JTAG's skew tolerance means that normally restrictive rules for clock distribution such as length

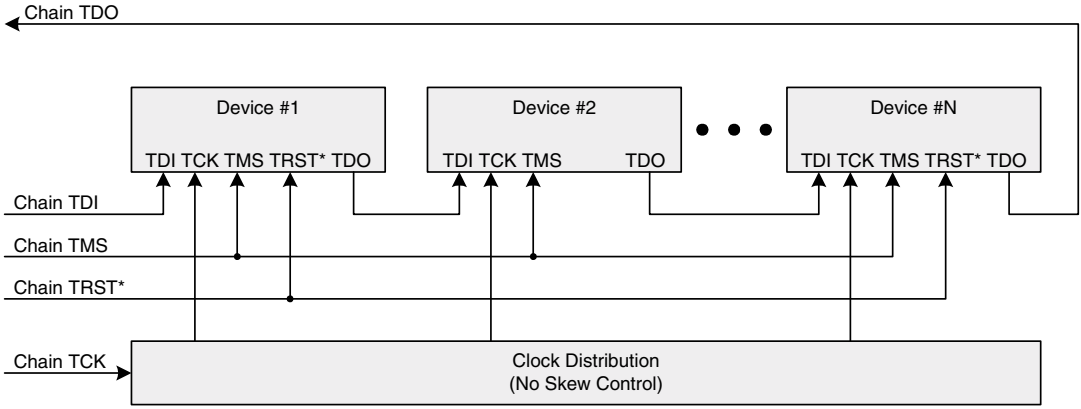

FIGURE 19.9 JTAG chain.

matching and low-skew buffers can be largely ignored. Of course, signal integrity considerations still apply so that TCK is delivered without excessive distortion to each IC. TRST* is active low and allows supporting devices to have their JTAG logic restarted. TMS is active high and places an IC into test mode by activating its JTAG controller.

JTAG scan chains are normally operated with special software designed to apply and check user-defined test patterns, or vectors. IC manufacturers who support JTAG provide boundary scan description language (BSDL) files that tell JTAG software how to interact with an IC. A standard procedure is to combine a PCB netlist (a file that lists each connection on the board in detail) with multiple BSDL files in a JTAG software package to come up with a set of test vectors. These vectors are then run through each board after it is assembled to verify connectivity between JTAG-equipped devices. Vendors of JTAG testing packages include Asset InterTech, Corelis, and JTAG Technologies. Verifying an assembled PCB with JTAG can save days of manual debugging when there may be problems in the assembly of BGA and fine-pitch components.

19.7 DIAGNOSTIC SOFTWARE

Software can help the debugging process by implementing scope loops, but the potential exists for much higher-level assistance. Special-purpose diagnostic programs are extremely helpful in detecting problems that could otherwise take much more time to isolate. The basic idea behind a diagnostic is for the software to thoroughly test elements of the hardware one at a time. A complete memory diagnostic, for example, would test every bit in every memory location. If the test fails, the nature of the failure provides valuable clues as to what is wrong. If there is a pattern to the failure, an improperly connected address or data bit may be the culprit. If the pattern is random, there may be a timing problem that causes marginal behavior over time or the device may be bad. Diagnostics are useful in several phases of development, including initial debug, extended reliability testing, troubleshooting damaged systems, and screening newly fabricated systems in either laboratory or manufacturing environments.

During initial debug, when little has already been shown to work, diagnostics can serve as both scope loops and sources of stimulation that exercise a wide range of functionality so that all features can be tested at least once. Memory tests are one of the most common classes of diagnostics, because a reliable memory interface is critical to any microprocessor-based system's operation. It is impractical to manually test millions of memory locations, but a program can easily perform this task in a quick, automated fashion. The specific patterns that a memory diagnostic uses should be chosen with knowledge of potential problems that may be uncovered. These patterns are written to memory in a complete write pass and then verified on a subsequent read pass. It is undesirable to test one memory location at a time, because certain problems can lie hidden from this approach. If the data bus is not connected, simple capacitance on the data wires might allow the microprocessor to "read" back a data value that was just written.

Aside from timing problems that can affect all transactions, circuit board connections are prime candidates for trouble because of potential errors during assembly or subsequent damage from excessive handling. A good diagnostic will pass only if there are no shorts or opens in the address, control, and data bus wiring. A shorted or open control signal will generally be easiest to find, because many, if not all, memory accesses are affected by these faults. Data bus wiring problems manifest themselves as a failure to read or write certain values to individual bit positions. An open data bit wire would result in reading a constant or random value, depending on the specific circuit. If a data bit is shorted to ground or power, the read value would be stuck at that logic level. If two data bits are shorted together, they will always assume the same value and will not be independently written.