COVER STORY

Advanced Flash Memory Packages

Drive OBP Innovation

By Peter T. Larsen and George Mello

METHODS TO SIMPLIFY ON-BOARD PROGRAMMING OF NEW DEVICES USE ATE AND IEEE 1149.1 TOOLS TO EASE THEIR INTEGRATION INTO MANUFACTURING ENVIRONMENTS.

Vendors such as Intel recognize that consumer buying trends are driving the creation of smaller flash memory package sizes — significantly smaller even than the popular TSOP. For example, the flash memory micro-BGA package is a true CSP (chip size package) that gives system design engineers the smallest available flash memory footprint. In addition, micro-BGAs are supported with today's installed surface mount tool capability so that SMT lines featuring such equipment as pick-and-place and IR reflow can handle the devices with minimal-to-no cost or process changes. Finally, the package design permits trace routing underneath the package using standard PCB design rules, which often results in a smaller footprint than a chip-on-board solution.

Advantages and Limitations

Manufacturers incorporate on-board programming (OBP) to quickly and efficiently program components and to gain cost savings associated with this less expensive process. Nevertheless, any programming solution, including OBP, has advantages and limitations that must be considered in relation to the application.

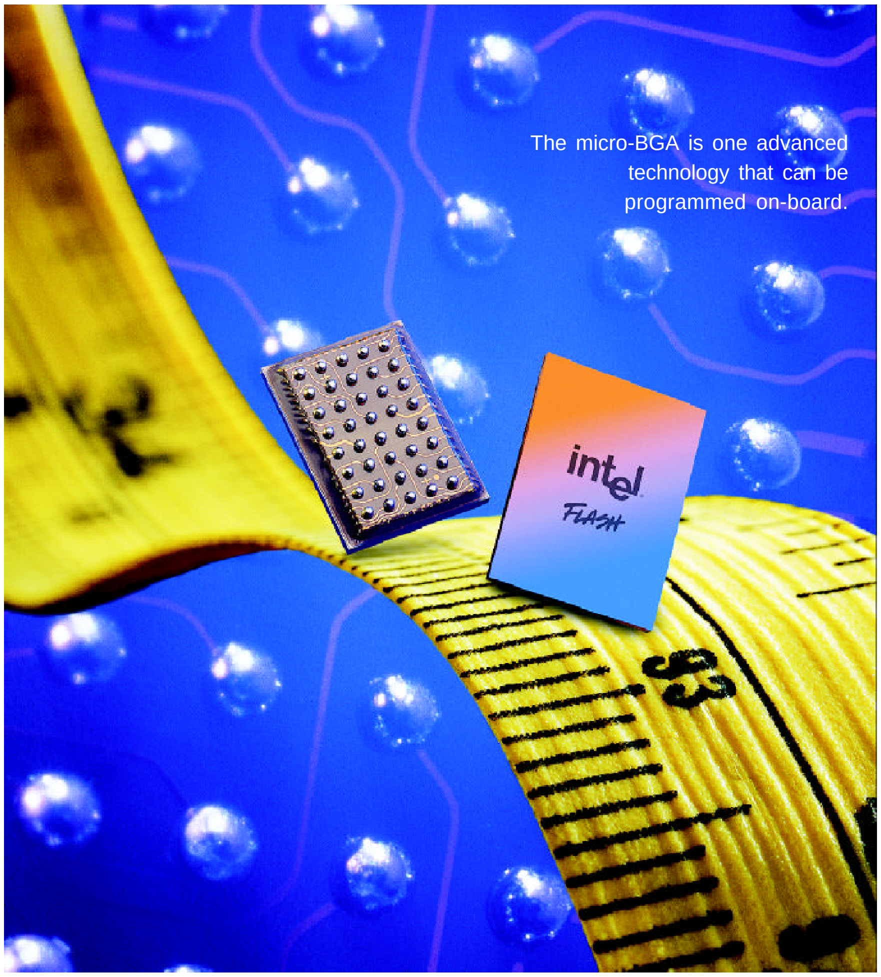
For example, OBP, by definition, eliminates off-line programming. The software is written while it is mounted on a PCB. OBP also reduces manual handling of components, eliminates individual device-labeling and enables use of economical tape-and-reel shipping media for flash memory components. The advantage here is that product assembly is quicker from tape-and-reel than from trays because, with the former, a mechanical search for a component is not required. In high volume manufacturing every second saved affects profit margins favorably. Lastly, with OBP, neither inventory storage nor additional handling equipment for fine-pitch packages are required. On the other hand, in some cases additional hardware for OBP is required to avoid bus contention. Designers may need to use connectors to link PCB applications to OBP equipment.

Throughput (beat-rate) will decrease by the time required for programming. And PCBs with limited space may not accommodate test land pads. (In this case, alternative OBP methods such as using the JTAG access port can be used.)

Device Requirements to Perform OBP

To perform on-board programming, the target device must be receptive even if other devices are connected to its pins. Compatible devices include flash memories, EEPROMs, EEPROM-cell-based PLDs for in-circuit programming, EEPROM-cell FPGAs for in-circuit programming and microcontrollers containing internal EEPROM. Devices not designed for OBP include those with voltage (V_{cc}) on logic pins during programming, where V_{cc} is raised during programming, devices that require ultraviolet light to erase the chip and chips that are one-time-programmable (OTP) or those programmed once and never erased.

Test engineers use automatic test equipment (ATE) to perform in-circuit testing of assembled PCBs. ATE validates PCB


The micro-BGA is one advanced technology that can be programmed on-board.
intel FLASH

construction by seeking assembly faults, open traces, shorted traces and misaligned and missing components. Test engineers can obtain additional value from ATE by integrating programming routines in the test flow. For example, a bed-of-nails, or multiple spring-loaded pins that contact test driver circuits, can link ATE with the PCB. When the board is secured over the test points, some will contact the target device's pins directly and others the test land pads (which then connect to the target device). ATE software defines which test points are active and permits the ATE to drive signals on test points that contact appropriate device pins (while disabling noncontacting test points).

Test engineers who use ATE for OBP must have hardware, software and target device product awareness to create, erase and program routines, which are integrated into the test flow and used to program the target device. If necessary, ATE vendors will assist in developing hardware and software to fit an application.

ATE typically performs programming operations faster than other OBP methods. To gain the best programming performance, engineers must optimize ATE programming code. Only *necessary* programming operations are performed. A thorough understanding of target-device programming requirements, as outlined in its datasheet and other technical documentation, will help reduce cycle and overall programming times. If an application contains a microprocessor, its outputs must be disabled before attempting to program the target device. This can be accomplished by using the microprocessor's RESET, HOLD or ONCE modes, which place its control signals and local bus in a high impedance state to avoid bus contention between the microprocessor and target device signals. Lastly, forward biasing of pins on peripheral components must be avoided by maintaining stringent voltage tolerances supplied by the ATE (Figure 1). Forward biasing occurs when input voltages exceed V_{cc} of the peripheral component.

Programming via ATE systems must provide for bed-of-nails probe access to all component pins with adequately sized test land pads leading to the component for a specific test fixture. Finally, design engineers, knowing that it is poor practice to leave unused inputs floating, will connect them to V_{cc} or GND through a 500 Ω series resistor. This permits pin control in the ATE environment.

Programming with the IEEE 1149.1 (JTAG) Access Port

The ability to add extra pins to packages without significantly increasing component size provides greater device functionality while maintaining (or even decreasing) required PCB space. An example of increased functionality in some components is the addition of a joint test action group (JTAG) access port. JTAG-compatible pins support boundary-scan test, in-circuit reconfiguration and on-board programming functions.

The JTAG Test Access Port (TAP) is an emerging OBP method. By communicating serially (one bit at a time) with the PCB application, JTAG is a viable programming alternative in a manufacturing environment, provided that the application contains a JTAG-compliant component and manufacturing can tolerate somewhat longer programming times than those of ATE programming. Alternatively, manufacturers can use JTAG to program boot code into the flash memory, with the remainder of the device being programmed via in-system write. By utilizing JTAG communication equipment that inserts into a PC add-in card slot and connects to the compliant application, engineers can send commands and data through the TAP to program the target device, in this case a flash memory component (Figure 2).

JTAG equipment permits communication with any JTAG-compliant device using

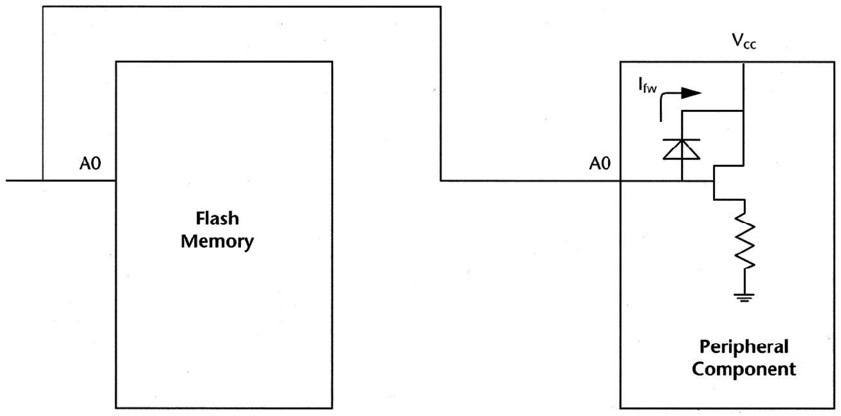


Figure 1. Forward biasing occurs when input voltages exceed V_{cc}. In the schematic, the peripheral component ESD protection diode sinks current when forward biased.

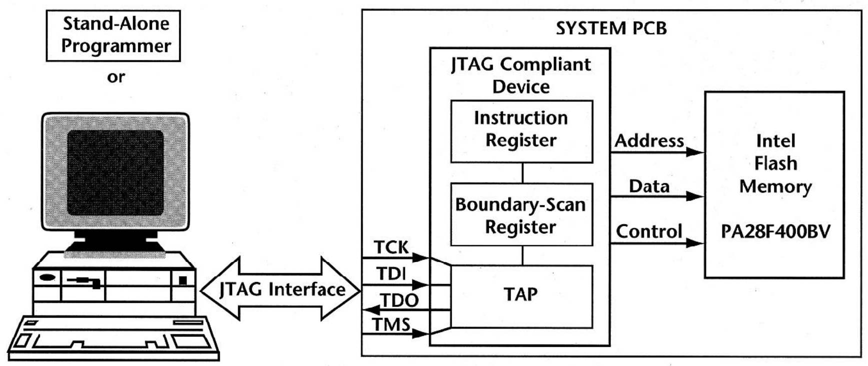


Figure 2. The CPU's JTAG test-access port provides an interface to program a flash memory device.

engineer created software to program the target device. Because programming at high speeds requires optimized code, the software engineer must have a clear understanding of the target device programming algorithm and JTAG communication equipment. A JTAG-compliant device contains the following four pins:

TCK — Test Clock Input, a clock separate from the system clock.

TDI — Test Data In, wherein data are shifted into the JTAG-compliant device.

TDO — Test Data Out. Data are shifted out of the compliant device.

TMS — Test Mode Select commands select test modes as defined in the JTAG specification.

Programming speed depends on several variables: TCK frequency, device speed, number of Boundary-Scan Cell (BSC) bits and flash memory density. This example assumes the following variables:

A. Calculate time for one boundary-scan register (BSR) shift:

- TCK = 80 ns cycle time (12.5 MHz TCK)
- 224 BSC bits x 80 ns (TCK) = 17.92 µs

B. Calculate time to program one byte/word:

- 4 signal transitions x 17.92 µs = 71.68 µs
- 4 signal transitions (BSR shifts) to drive "high" and "low" levels on address, control and data signals

C. Calculate programming time for a 4 Mbit device:

- 262,144 words (flash memory density) x 71.68 µs = 18.79 seconds

D. Calculate programming time for one 8K word boot block:

- 8,192 words (boot block size) x 71.68 µs = 0.59 seconds

E. The formula used to calculate JTAG programming performance:

- (TCK frequency) x (BSC bits) x (BSR shifts) x (flash memory density) = programming time

Algorithm Optimizations

The programming flowchart provides a good understanding of test-flow optimizations (Figure 3). To reduce programming time and increase manufacturing beat-rate, modifying the flow to perform only necessary operations may be considered. For instance, since manufacturers ship blank memory components from the factory, engineers can reduce programming time by eliminating erase check on new devices.

Eliminating redundant program verification checks is another option. Memory components with internal write-state machines automatically verify data written to the memory array. Hence, program verification operations initiated by the programmer are redundant. To gain performance improvements test engineers should continually look for any possible optimizations to reduce overall programming times.

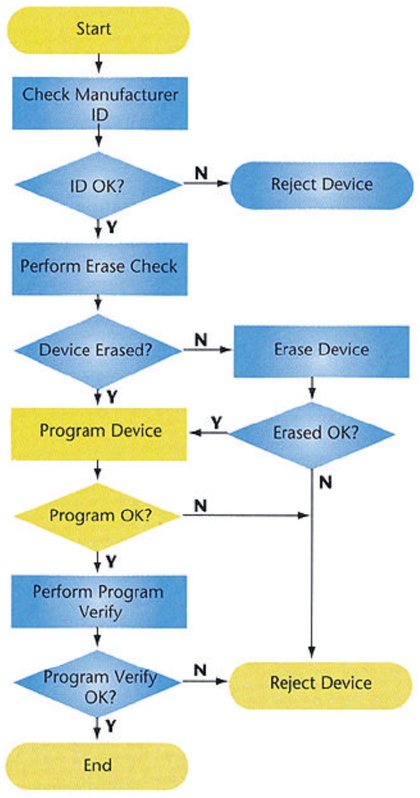


Figure 3. The programming flowchart. Darkened areas can be omitted for optimized programming times.

Engineers also should optimize programming code to reduce software overhead. Each signal transition requires time to execute. For example, if a particular signal is frequently switched additional system overhead is consumed for each transition. To reduce system overhead, opportunities to reduce the number of signal transitions performed should be sought.

Conclusion

Designers using fine-pitch packages in embedded applications are discovering new programming methods for these devices. Tools are now more plentiful and offer the manufacturing solutions that best fit specific applications. The micro-BGA provides an example of an emerging package technology that takes advantage of these tools. The micro-BGA package is also an ideal candidate for use with automated SMT equipment since the tools are already in place.

ATE and JTAG programming methods are two OBP examples of tools that streamline the process. OBP, in turn, is well suited for the manufacturing test flow on such devices as nonvolatile memories, microcontrollers and PLDs. The foregoing offers some suggestions for OBP but ultimately the project design and software engineers must analyze individual OBP environments to implement the best solution for a particular project. SMT

WORKS CONSULTED

1. Kenneth P. Parker, *The Boundary-Scan Handbook*, Kluwer Academic Publishers.
2. Harry Bleeker, Peter van den Eijnden, Frans de Jong, *Boundary-Scan Test*, Kluwer Academic Publishers.
3. "In-Circuit Programming of Flash Memory," AP-Note JTAG-102, Corelis Application Note, (310) 926-6727.
4. "AP-624: Introduction to On-Board Programming with Intel Flash Memories (1996)," Intel Order Number 292179; (800) 548-4725.
5. "AP-629: Simplify Manufacturing by Using Automatic-Test-Equipment for On-Board Programming (1996)," Intel Order Number 292185; (800) 548-4725.
6. "AP-630: Designing for On-Board Programming Using the IEEE 1149.1 (JTAG) Access Port (1996)," Intel Order Number 292186; (800) 548-4725.
7. *IEEE Std 1149.1 Standard Test Access Port and Boundary-Scan Architecture*, IEEE Inc., 345 East 47th St., New York, N. Y. 10017.

PETER T. LARSEN and GEORGE MELLO may be contacted at Intel Corp., Memory Components Div., FM3-83, 1900 Prairie City Road, Folsom, Calif. 95630; (916) 356-2269; Fax: (916) 356-2803.