

## BOUNDARY SCAN NOR/NAND FLASH MEMORY PROGRAM GENERATION

### PRODUCT OVERVIEW

Several methods are used to program flash devices after they have been assembled onto a board. Each has advantages and disadvantages, but only on-board programming with JTAG is fast, convenient, and can be used throughout a product's life cycle. On-board programming with boundary scan enables you to use the same tool to load program data during design verification and debug, during manufacturing tests, and in the field for system upgrades. Programming times can approach theoretical times and file sizes are not the problem they are with other methods.

### FLASH MEMORY PROGRAMMING WITH JTAG OR BOUNDARY SCAN

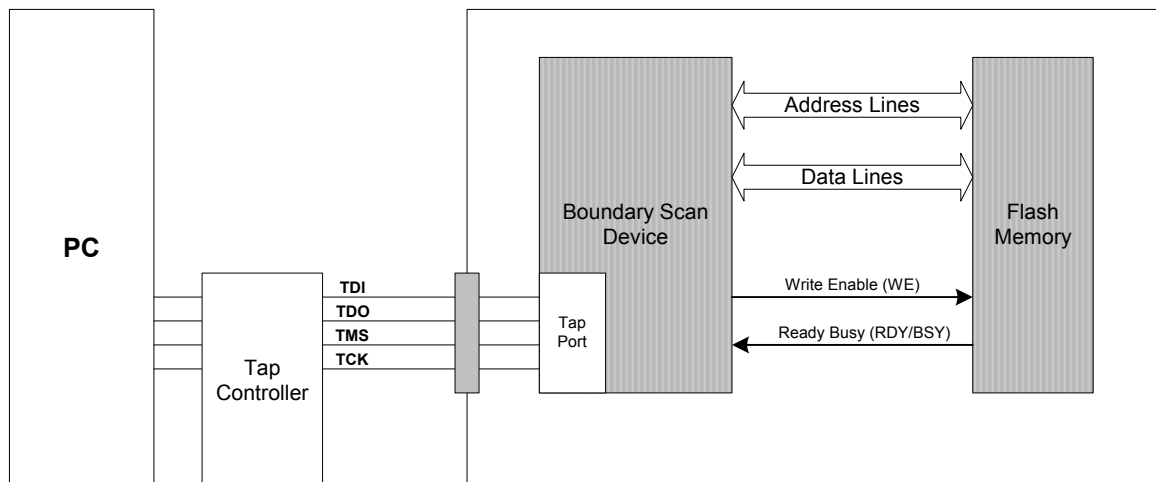
Flash memories have fairly simple algorithms for writing and reading data enabling the algorithm to be

emulated by a series of scan operations. These scan operations apply the address and program data to the appropriate pins and toggle the control signals at the appropriate time. This requires the data, address, and control signals to be accessible from a boundary-scan device, the device to be accessible to a boundary-scan tool, and the tool to know the sequence of operations needed to control the flash device.

Several considerations are required when estimating programming times using JTAG (boundary scan):

- Number of scan operations required
- Length of the boundary-scan path
- Maximum TCK supported by the design
- Amount of data to be loaded

The number of scans needed to write data to a flash device depends on the device type. Some require only



four scans while other require as many as eight. Controlling the write enable signal by a non-scan signal can reduce the number of scans. Reducing the number of scan required from four to two reduces programming time by approximately 50%. The length of the scan path can be minimized by the selection of the device used for access and by placing all other scan devices in BYPASS. Often there is only one option, allowing no improvements. The maximum TCK supported is the frequency supported by the slowest device in the path, or the maximum frequency supported by the board design. Careful device selection and board design minimizes programming times. The amount of data to be loaded is usually not optional, but by carefully planning the manufacturing flow, you can minimize the impact on manufacturing throughput.

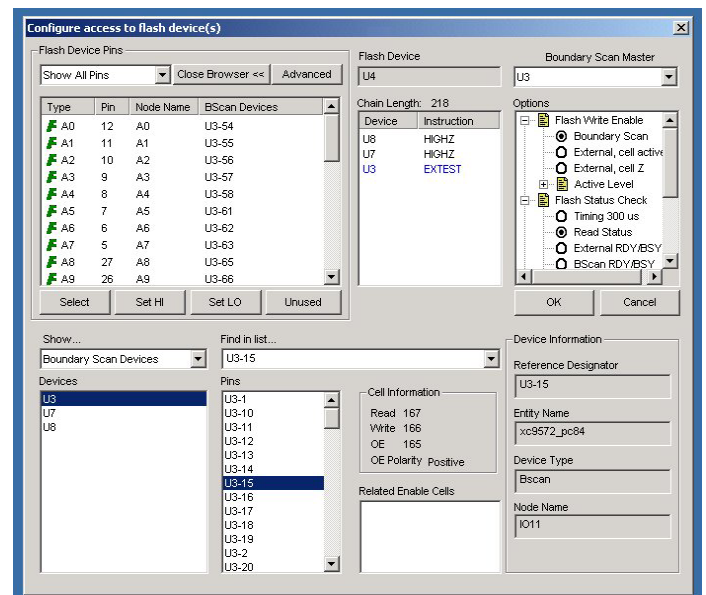
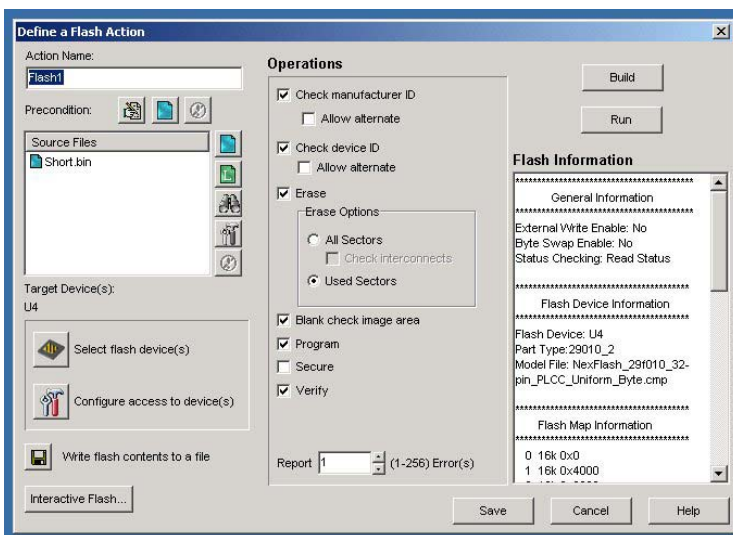
## FLASH MEMORY PROGRAM GENERATION WITH SCANWORKS®

ScanWorks supports on-board programming of flash memories as part of a complete test and programming solution or as an independent station for flash programming only. The flash memory program generation option can be added to a Test

Development Station, Diagnostics & Repair Station, or a Programming Station. Any flash programming application created on these stations can be applied on any station, including a Manufacturing Station.

Flash memory program generation is implemented as a ScanWorks action, just like all other ScanWorks tests and programming applications. The action is associated with a design that is described with the standard ScanWorks design manager tools from BSDL files and netlists. The flash memory generation tool provides all the features needed to easily create and verify a flash programming action. These features include:

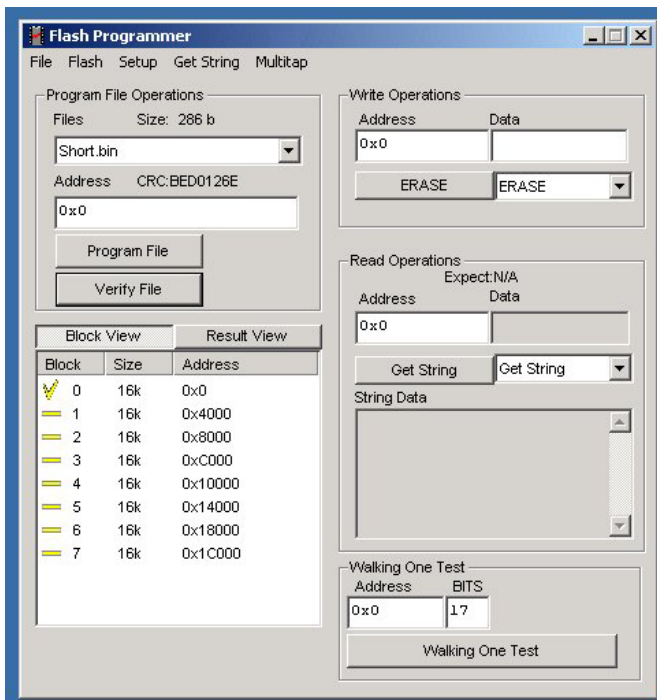
- Library of popular flash memory models provided
- Automated selection of the boundary-scan device(s) used for access
- Easily resolve access issues and assign constraints for 'safe' programming (Figure 2)
- Data image file in Binary, S-record, and Hex supported
- Image files can be accessed from a networked location



- Multiple image files can be loaded during a programming operation
- User-defined flash device models supported
- Non-scan control of Write Enable and RDY/BSY signals
- All standard flash memory operations supported
- Boundary-scan device-to-flash device interconnect test applied before programming (optional)
- Manufacturers' ID and Device ID checked before programming
- Programming time estimates provided before application, progress indication during programming, and actual programming time written to output log
- Interactive read, write and verify operations to verify and debug flash access (Figure 3)
- User controlled write times (until status indicates write complete, until maximum time has elapsed or for user specified time)

## FLASH PROGRAMMING IN MANUFACTURING

Once the flash programming action has been created and verified, it can be applied from the flash action definition UI or added to a ScanWorks sequence for use in manufacturing. It can also be called from the ScanWorks API and included in a custom UI created with LabVIEW, Lab Windows/CVI, Visual Basic, or in C++. These are the same methods used for test applications, making the integration of flash programming with test a seamless operation.



### Highlights

- Library of popular flash devices provided
- Automated selection of boundary-scan access devices
- Fast programming time using non-scan control of write enable and RDY/BSY signals
- Interactive read, write, and verify operations for easy access verification
- Estimated and actual programming times reported
- Binary, Hex, and S-record input files supported