Non-Volatile Memory Programming on the Agilent 3070

Paul Montgomery
Agilent Technologies
Cleveland, Ohio
paul_montgomery@agilent.com

BACKGROUND

The trend in non-volatile memory, like the general trend in electronic products, is to increase capability, reduce size, reduce cost, and get the final product to market faster than the competition. With regard to non-volatile memory, the increased capability translates to increased capacity, and possibly increased speed. The increased capacity of the memory means greater code space to provide more accuracy or more flexibility in existing product features, or additional features in the final product. In order to reduce physical size of the product, that additional memory is attained through the use of higher capacity single devices, not multiple devices. Certainly reducing the required real estate on a circuit board will reduce the overall cost. A significant part of the cost of utilizing non-volatile memory, however, is contained both in the cost to program the device, and in the cost to ensure that the final assembly has the correct device containing the correct code. Finally, getting the product to market faster that the competition means quick turnaround on all phases of product and process development. Since test development is usually the last step in completing the production process, any time saved in test development will directly decrease the “time to market”.

OBJECTIVE

The purpose of this paper, then, is to investigate the tools available for programming and testing newer and larger non-volatile memories while reducing processing costs and reducing development time. The measurements made to substantiate these claims used the same test sources on two different testhead controllers. For consistency, the same testhead was used for all measurements; the testhead controllers were set up on an “A-B” switch. To represent the majority of Unix based controllers in the installed base a B180L testhead controller was used. The newer MS-Windows controller used was an IPC, or Industrial PC, which is the current controller shipped with a new testhead. Three non-volatile devices were chosen primarily for their capacities (512K x 16, 4M x 16, and 8M x 16) and their manufacturers (AMD, AMD, and Intel, respectively). Random data programming files were generated to require programming of every location within a device. The random data was screened for “FF”s (locations which do not require programming). When found, those locations were modified to random non-FF values. The resulting program files were compiled into “static” programming tests in all cases. Each measurement of compile and run time is the average of five iterations of the specific action.

FLASH PROGRAMMING SOLUTIONS for the AGILENT 3070

Agilent Technologies offers three methods for programming non-volatile memory. The first, and oldest, method uses standard VCL (Vector Control Language) source and object files to emulate the programming algorithms required by each non-volatile device. I will refer to this method as “Flash” from this point forward. Tests can be structured with static data, precompiled into the test object, or dynamic data, compiled at run time. The result is that static tests tend to take longer to compile, generate larger object files, and run faster at programming time than dynamic tests. The second method, known as “Flash70”, is primarily an improvement in the compilation process for non-volatile memory programming tests. In general, Flash70 tests tend to compile and run faster, but their objects can be considerably

larger than “Flash” objects, and their first run times are typically slower than Flash first-run times. Flash70 tests can also be structured with static or dynamic data. Flash ISP is the newest software product for the programming of non-volatile devices. It utilizes further changes to the compilation techniques of Flash70 along with hardware changes to the digital sequencer to provide faster compile times, much smaller object file sizes, and near “data-book” programming speeds. In order to achieve the highest programming speed possible, Flash ISP tests can only be structured as a static test. In other words, Flash ISP has been optimized for use on large blocks of fixed data, much the same as a standalone EPROM programmer. If a small block of dynamic data is required to supplement the fixed code, such as a unique serial number or test result codes, a small Flash or Flash70 program can be used.

RESULTS

Let’s begin this analysis by looking at older technology, both in memory devices and programming techniques. The smallest device used for this analysis was an AMD am29DL800, organized as 512K 16-bit words. According to the data sheet, it typically takes 5.9 seconds to program the entire device in word mode. Using a B180L controller, a Flash programming test took 3 minutes 49 seconds to compile, and programmed the device in just over 16 seconds. First run time (34.9 seconds) was just slightly more than double the mature run time. In order to program 512K words of data (or 1M byte), the Flash object was just over 2M bytes. Changing the test to Flash70 cuts the compile time in half, but doubles the size of the test object. The first run time for Flash70 increases to 43.6 seconds, but the mature run time decreases to 11.7 seconds, a 28% improvement in programming speed. Finally, promoting the test to Flash ISP decreases the compile time from 3 minutes 49 seconds down to 14.3 seconds. The resulting object file is just under 1.1M bytes, and the mature run time is decreased to 8.7 seconds, an improvement over Flash in programming speed of 46%. As good as these improvements sound, there are more speed improvements available. In general, upgrading to a PC testhead controller means compile times that are 5 to 25 times faster than the Unix based controller, and run times that are at least identical to and possibly up to 24% faster than the Unix based controller. In the extreme case, upgrading a Flash test on a B180L to a Flash ISP test on an IPC means cutting compile times from 3 minutes 49 seconds to 2.7 seconds, cutting mature run time from 16.2 seconds to 7.6 seconds (only 1.7 second slower than the “data-book” speed), and cutting disk usage for that test by about 1.1Mbyte of space. Additionally, the first run time is cut by about 75% using Flash ISP on an IPC. Similar gains are realized on the verify routine, which may or may not be used for production, but will definitely be used during debug. A typical debug session of 12 iterations of “compile the program test”, “erase the device”, “program the device”, and “examine the contents with the verify test” would save nearly an hour of time otherwise spent “waiting” for the 3070 to produce results.

	Flash		Flash70		Flash ISP	
	B180L	IPC	B180L	IPC	B180L	IPC
Avg. Compile	229.33	26.81	122.83	17.92	14.28	2.72
Avg. First Run	34.90	30.53	43.65	41.11	10.78	8.79
Avg. Mature Run	16.22	12.28	11.69	9.63	8.69	7.63
Object Size	2,277,853	2,277,787	4,776,913	4,776,847	33645	18091
Image Size	na	na	na	na	1,060,956	1,060,956
Segment Size	4096	4096	8192	8192	512	512

AM29DL800 (512K x 16)

Table 1

Similar speed improvements (in terms of percentages) are realized with larger devices as well. Using an AMD AM29DL640D arranged as 4M x 16bit words, the same comparisons were made using Flash, Flash70, and Flash ISP on both a B180L and an IPC. According to the data sheet, this part typically takes 30 seconds to program the entire device in word mode. Using a B180L controller, a Flash programming test took 14 minutes 23 seconds to compile, and programmed the device in 96 seconds. First run time (3 minutes 42 seconds) was considerably more than double the mature run time. In order to program 4M words of data (or 8M bytes), the Flash object was about 14M bytes. Changing the test to Flash70 cuts the compile time by two-thirds, but increases the size of the test object by a factor of 2.5. The first run time for Flash70 increases to 5 minutes 30 seconds, but the mature run time is cut roughly in half. Finally, promoting the test to Flash ISP decreases the compile time from 14 minutes 23 seconds down to 1 minute 20 seconds. The resulting object file is just over 8Mbytes, and the mature run time is decreased to 38 seconds, an improvement in programming speed of 60%. Again, there are more speed improvements available by using an IPC for a testhead controller. On this larger device, improvements in compilation speed over the Unix controller were as expected. The mature run time, however, showed no improvement for the IPC over the B180L controller. Once again, though, the extreme case of upgrading a Flash test on a B180L to a Flash ISP test on an IPC means cutting compile times from 14 minutes 23 seconds down to 13 seconds, cutting mature run time from 3 minutes 42 seconds down to 38 seconds (only 8 seconds slower than the “data-book” speed), and cutting disk usage for that test by about 26Mbytes of space. As on the first device, the first run time is cut by about 75% using Flash ISP on an IPC. And again, similar gains are realized on the verify routine. For this device, a typical debug session of 10 iterations of “compile the program test”, “erase the device”, “program the device”, and “examine the contents with the verify test” would take 4 to 6 hours for a Flash test on a B180L. The Flash ISP test debugged on an IPC testhead controller would shorten that debug session by more than three hours of time otherwise spent “waiting” for the 3070 to produce results.

	Flash		Flash70		Flash ISP	
	B180L	IPC	B180L	IPC	B180L	IPC
Avg. Compile	862.64	45.29	256.84	33.89	79.77	13.07
Avg. First Run	221.73	213.18	331.58	257.15	46.07	45.65
Avg. Mature Run	95.68	94.69	47.28	46.79	37.89	37.97
Object Size	13,919,181	13,919,181	34,578,427	34,578,427	19245	19245
Image Size	na	na	na	na	8,487,004	8,487,004
Segment Size	4096	4096	32768	32768	512	512

AM29DL640 (4M x 16)

Table 2

Next, let’s examine the programming of a device that is just “one size larger” than the AM29DL640D. The device is Intel’s 28F128J3A, organized for these tests as 8M of 16-bit words. Due to the capacity of this device, the timing comparisons for this device cannot parallel the comparisons for the two devices above. While I was able to compile Flash, Flash70, and Flash ISP versions of the test on the IPC, only the Flash ISP test would compile on the B180L. After doubling the HP-UX kernel parameters “maxdsize” and “maxssize” on the B180L, I was able to compile the Flash70 test, but the Flash test still would not compile. Certainly with the “proper adjustments”, the Flash test could be coerced into compiling. Based on the Flash70 mature run time, however, it would be an agonizingly long test to use in a production test scenario. Therefore, the comparisons for this part will use the Flash70 test as the baseline. According to the data sheet, it typically takes 1 minute 56 seconds to program the entire device in word mode. Using a B180L controller, a Flash70

programming test took 15 minutes 16 seconds to compile, and programmed the device, on average, in 14 minutes 27 seconds. First run time of 41 minutes 11 seconds was slightly less than triple the mature run time. In order to program 8M words of data (or 16M byte), the Flash70 object was just over 171M bytes. Changing the test to Flash ISP decreases the compile time to 2 minutes 34 seconds. The resulting object file is just under 17M bytes, and the mature run time is decreased to 2 minutes 15 seconds, an improvement in programming speed of 84%. Once again, the primary speed advantages to employing an IPC are in the compile times. The extreme case here, upgrading a Flash70 test on a B180L to a Flash ISP test on an IPC means cutting compile times from 15 minutes 16 seconds down to 27 seconds, cutting mature run time from 14 minutes 27 seconds down to 2 minutes 15 seconds (only 19 seconds slower than the “data-book” speed), and cutting disk usage for that test by about 154Mbyte of space. Additionally, the first run time is cut by about 94% using Flash ISP on an IPC. This test has one additional advantage if created using Flash ISP. The object for the Flash70 test (171Mb) is so large that it will not fit in the testhead memory with all of the other device tests for a specific board test. The result is that the testhead will “swap” this object and the “other” test objects in and out of testhead memory for each board tested. Since this object must be loaded into memory for each test cycle, it will **always** run at the first-run rate of 41 minutes 11 seconds. The much smaller Flash ISP object has a much greater chance of fitting into testhead memory with all of the other objects that comprise the board test. Here again, large time saving can be realized in the test debug process. For this device, a typical debug session of 6 iterations of “compile the program test”, “erase the device”, “program the device”, and “examine the contents with the verify test” would take 8 to 10 hours for a Flash70 test on a B180L. The Flash ISP test debugged on an IPC testhead controller would shorten that debug session by more than six hours of time otherwise spent “waiting” for the 3070 to produce
results.

	Flash		Flash70		Flash ISP	
	B180L	IPC	B180L	IPC	B180L	IPC
Avg. Compile		82.65	915.77	55.21	154.60	27.16
Avg. First Run			2471.32	2470.06	166.90	151.20
Avg. Mature Run			867.20	865.92	135.27	134.56
Object Size		34,712,207	171,225,589	171,225,589	124957	124957
Image Size		na	na	na	16,851,036	16,851,036
Segment Size		2048	2048	2048	512	512

28F128J3A (8M x 16)

Table 3

CONCLUSION

The trend that I have observed in product design is that large non-volatile memory devices are becoming more prevalent, and larger devices are on the way. The wisdom accepted in the manufacturing community is that In-System programming of non-volatile devices is most cost effective from parts inventory, material handling, and product accuracy standpoints. The fact that you can confirm with Agilent Technologies is that HP-UX as an operating system is rapidly approaching the end of its service life. Armed with these realities and the data from this investigation, then, I draw several conclusions. First is that regardless of the testhead controller platform you choose, Flash ISP provides the ability to program larger devices in less time than Flash or Flash70. As we have seen, devices that would be impractical to program using either Flash or Flash70 can be programmed in just 124% of the “data-book” rate. Second is that tests will certainly compile much faster, and sometimes even run faster on an IPC testhead controller than on a B180L controller. Further, tests that

will not compile on the B180L as factory configured will compile on the IPC. Test objects that consume a significant portion of the 9Gbyte disk on the B180L make less of an impact on the 80GByte disk in the IPC controller. Third is that at some point in time, performance or capabilities enhancements will be made for the IPC platform only. Version 05.30p is the last release planned for the Unix based controllers. Any enhancements beyond that will be available only on the MS-Windows based platform. Considering the “re-configurable” nature of the Control-XTP card, the range of possible enhancement requests is limitless. Finally, conversion from Unix test programs to MS-Windows test programs continues to get easier. For software revisions prior to 05.20p, there is an automated conversion utility to translate “UX to PC” or “PC to UX”. In software revisions 05.20p and above, you can make you test programs “interoperable” between the Unix based and the MS-Windows based platforms, allowing you to merely copy test files instead of “converting” the files when you want to move them between Unix based test systems and MS-Windows based test systems. With all of the speed enhancements demonstrated here as well as the ease of conversion to the MS-Windows based controller, now is a great time to start saving time in test development and in manufacturing by upgrading to the latest in hardware and software tools available for the Agilent 3070.