On In-System Programming of Non-volatile Memories

Anton Tsertov, Sergei Devadze, Artur Jutman, and Artjom Jasnetski

***Abstract*—With the continuous growth of capacity of non-volatile memories (NVM) in-system programming (ISP) has become the most time-consuming step in post-assembly phase of board manufacturing. This paper presents a method to assess ISP solutions for on-chip and on-board NVMs. The major contribution of the approach is the formal basis for comparison of state-of-the-art ISP solutions. The effective comparison pin-points the time losses, that can be eliminated by the use of multiple page buffers. The technique has proven to achieve exceptionally short programming time, which is close to the operational speed limit of modern NVMs. The method is based on the ubiquitous JTAG access bus which makes it applicable for the most board manufacturing strategies despite a slow nature of JTAG bus.**

***Index Terms*—in-system programming, processor-centric board, JTAG, non-volatile memory.**

I. Introduction

THE widely adopted DfT structures defined in IEEE 1149.1 standard [1] and complemented in [2] are heavily used for ISP and memory test, besides traditional post-assembly tests (Boundary-Scan tests). Despite of ubiquitous presence of boundary-scan (BS) [1] structures in modern electronic systems and components, the application of BS is often considered limited due to the low operation frequency. Typically BS clock (TCK) frequency is in range from 1 MHz to 50 MHz, whereas actual speed of data transfer is much lower due to the overhead data (JTAG instructions and UUT protocol) that accompanies each test pattern. Here and later the term BS will be used to stress that IEEE 1149.1 structures are used in isolation from the internal functionality of the chip. The term JTAG denotes the test infrastructures defined in IEEE 1149.1 standard.

Traditionally, the ISP is considered to be the final phase of BS test session and builds the foundation for the following functional tests [3]. The fundamental JTAG-based ISP solutions are proposed in [4], [5] and [6], that describe processor-controlled test and processor-centric board test (PCBT). These methods are based on the use of functionality of microprocessor (uP) or microcontroller (uC) to access peripheral components outside the uP or uC at normal operational speed of the board. The attractiveness of PCBT-based solutions are very high due to the usage of existing DfT structures without any modifications. Later in this paper, the formal characterization of ISP will be given in the terms from JTAG and PCBT methodology.

A. Tsertov is with the Department of Computer Engineering, Tallinn University of Technology, Tallinn, Estonia (e-mail: anton.tsertov@ttu.ee)

S. Devadze, A. Jutman, A. Jasnetski are with Testonica Lab OU, Tallinn, Estonia (e-mails: {sergey, artur, artjom}@testonica.com)

The ISP of NVMs with the use of PCBT methodology has proven to speed up the existing JTAG-based solutions [5] [7]. However, in the near future the size of the program images that are stored to the NVMs is expected to grow, mainly due to the grow of complexity of the available hardware functionality. Inevitably the speed up in ISP time introduced by PCBT methodology is not sufficient and the industry is looking for methodology that will run ISP at the operational speed limit of contemporary and future NVMs. State-of-the-art JTAG and UART-based solutions are limited by the speed of the data transfer link. In this paper we derive the technique that mitigates time-losses that are common to most ISP approaches.

A. *Problem statement and paper contribution*

After the bare PCB is populated with components it needs to be tested for manufacturing defects. Typically, the boundary scan tests are used to screen out boards with static structural defects. After the screening procedure the ”healthy” boards reside in the fixture for subsequent ISP and functional testing, e.g. using boot-loader image. In most cases, it is considered beneficial to program the NVMs with the same tester hardware that is used for verification and test of other components on the printed circuit board assembly (PCBA) under test. The PCBT methodology helps to reduce the programming time of flash memory from hours, as in case with BS, to minutes and even less. Nevertheless, the actual ISP time (in case of PCBT) heavily depends on the architecture of the debug interface of the uP, on the instruction set of the uP, on the performance of the flash memory controller inside the uP SoC and on the performance of the flash memory itself. The latter is discussed in details in the last section of this paper.

The PCBT-based ISP procedure of NVMs is formally characterized in Section II. Section III studies the drawbacks of the state-of-the-art ISP solution and proposes countermeasures to solve the named issues. The solution in Section IV allows in most cases to perform in-system programming of on-chip or on-board non-volatile memories at maximum speed. Whereas, maximum speed means that the bottleneck is not in the data transfer channel (JTAG), but in the capability of the memory itself to program the supplied data faster. The formal basis from Section II and III is used in the last section to asses and compare the experimental results of different test-cases (ISP of on-chip and on-board NVM).

II. Processor-centric Board Test

The processor-centric board test [6] is a collective term for the post-manufacturing tests for processor-centric boards. These tests target the defects related to the last stages of the board manufacturing process and also the first stages of the post-manufacturing product life (e.g. in-system programming (ISP), infant mortality diagnosis). The uP plays a role of on-board tester, that listens to commands from external test equipment and in response applies tests to other PCBA components.

The next section presents a study on how ISP can benefit from the PCBT methodology.

A. Test application

Test path initialization and configuration belong to the **test access** functionality of the PCBT program. The rest of the PCBT program functionality is a part of the **test application**, which may be developed in accordance to online[1] or offline test application modes [8].

In case of the the offline (autonomous) mode, the complete test program (test vectors and expected values) is translated into the set of uP instructions and loaded as an ordinary program into memory inside the uP. The program execution inside the uP is started by the external tester. After test program execution is finished the result (PASS or FAIL for complete test) will be stored in the on-chip memory of the uP. It is retrieved through the debug interface and reported to the external tester for further evaluation and diagnosis.

The offline mode is fully independent and does not suppose continuous interaction with an external tester. This mode needs available on-chip memory to store test data. Obviously, this mode is not suitable for ISP of large images unless there is enough memory available (e.g. on-chip or on-board volatile memory) to hold the whole image to be programmed into on-board non-volatile memory.

The key difference between the online and offline modes is that in the online mode commands are executed under the control of external tester. In online mode the flash image is transferred through the test access path word by word directly to the non-volatile on-board memory. This implies also transferring the commands for NVM, which significantly reduces the test data throughput of test application.

The formula (I) is proposed for ISP time t_{ISP} calculation for the online mode of test application.

$$(I) : t_{ISP} = t_{WD} + dt_{ID} + t_B \frac{d}{b} + t_R \frac{d}{b}$$

- d - size in words of the flash image
- b - size in words of the flash page. Traditionally the data in NVMs is handled page-wise.
- t_{WD} - time to transfer over JTAG bus (shift in) the flash image
  $t_{WD} = d\frac{l}{h}$, where l is a length of the shift and h is a frequency of the test clock (TCK).
- t_{ID} - time to shift in the instructions that write a word to any memory location
  $t_{ID} = \sum_{i=1}^{k} \frac{l_i}{h}$, where k is the number of shifts to emulate the write instruction from external tester on the uP. k includes the JTAG TAP instructions, debug port instructions and uP instructions. l_i denotes that instruction of every type might be different in length.

[1]Online mode of test application is not the same as on-line testing

- t_B - programming time, required by Flash device to program one page
  t_B - is a constant specified in the datasheet of the memory
- t_R - time to read the word (to read status register after programming sequence)
  $t_R = \sum_{i=1}^{r} \frac{l_i}{h}$ where r is the number of shifts to emulate the read instruction and to shift out the data to the external tester. r includes the JTAG TAP instructions, debug port instructions and uP instructions. l_i denotes that instruction of every type might be different in length.

$\frac{d}{b}$ means the number of pages to program (in case of unaligned number of bytes the result is rounded up).

III. Hybrid Online ISP Mode

In order to speed up the ISP in online mode the industry has came up with a solution [5] [9] which we call in the following as hybrid-online mode of test application. In Figure 1 the uP-based SoC components that are active in hybrid mode are shown. The difference to pure online mode is in the software that is executed by the uP. This software is executed in accordance to the control commands from external tester and is called monitor software. Firstly, the external tester transfers part of the data image to the buffer, then the command to copy data from buffer to buffer inside NVM is send to the monitor software. The monitor software handles the data transfer from the buffer to the page buffer inside NVM by turn and sends write page commands to NVM. If the on-chip memory is not available the on-board volatile memory can be used to store the monitor software and host the intermediate buffer for data. The ISP time calculation for hybrid-online mode can be calculated using the proposed formula (II).

$$(II) : t_{ISP} = t_{WM} + t_{WD} + (m + d)t_{ID} + t_B \frac{d}{b} + t_{EX}$$

- m - size in words of the monitor software
- t_{WM} - time to shift in the monitor software
  $t_{WM} = m\frac{l}{h}$, where m is size of the monitor in words, l is a length of a word and h is a frequency of the test clock (TCK).
- t_{WD} - time to shift in the flash image
  $t_{WD} = d\frac{l}{h}$, where l is a length of a word and h is a frequency of the test clock (TCK).
- t_{ID} - time to shift in instructions that write a word to the register or location in volatile memory
  $t_{ID} = \sum_{i=1}^{k} \frac{l_i}{h}$, where k is the number of shifts to emulate the write instruction from external tester on the uP. k includes the JTAG TAP instructions, debug port instructions and uP instructions. l_i denotes that instruction of every type might be different in length.
- t_B - programming time per page
  t_B - is a constant specified in the datasheet of the memory
- t_{EX} - time taken by monitor to copy flash image from buffer to page buffer into flash memory. As the monitor is executed at the actual operating speed of the uP this time can be neglected (proven in Section V). Typically uP clock is at least one order of magnitude faster than TCK.

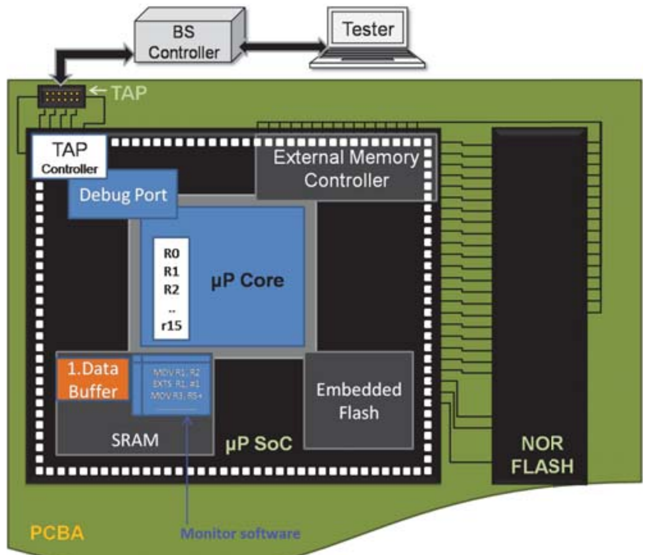


Fig. 1. Data path components in hybrid online mode of ISP

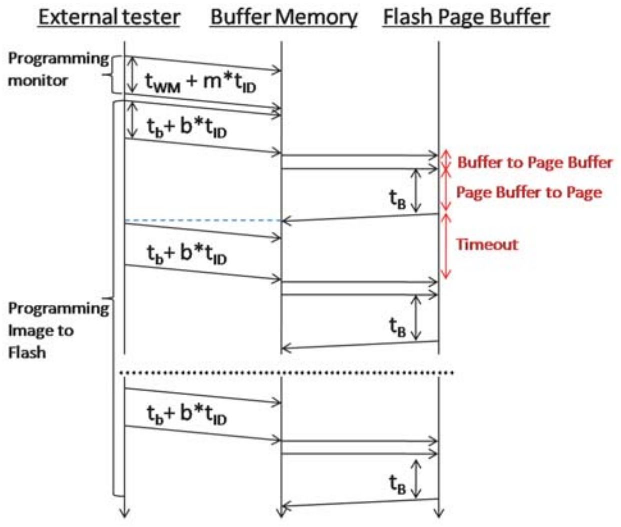


Fig. 2. Time flow in hybrid online mode of ISP

After substituting the notation of summands in formula II with the respective expressions, formula II takes the following form:

$$(III): t_{ISP} = m\left(\frac{l}{h} + \sum_{i=1}^{k}\frac{l_i}{h}\right) + d\left(\frac{l}{h} + \sum_{i=1}^{k}\frac{l_i}{h}\right) + t_B\frac{d}{b}$$

Obviously, in order to justify the effort spent on development of the monitor software the hybrid-online mode has to speed up the online mode. This consideration is expressed by the following inequality (I)>(II):

$$d\frac{l}{h}+d\sum_{i=1}^{k}\frac{l_i}{h}+t_B\frac{d}{b}+\frac{d}{b}\sum_{i=1}^{r}\frac{l_i}{h} > m\frac{l}{h}+d\frac{l}{h}+(m+d)\sum_{i=1}^{k}\frac{l_i}{h}+t_B\frac{d}{b}$$

that reduces to:

$$(IV): \frac{d}{b}\sum_{i=1}^{r}\frac{l_i}{h} > m\frac{l}{h} + m\sum_{i=1}^{k}\frac{l_i}{h}$$

Let us evaluate the obtained expression (IV). From (IV) it can be concluded that the time to load the monitor to volatile memory should be shorter than the time taken by reading the flash status after the page programming operations. In other words, the inequality (IV) never holds if the monitor size in words is bigger than the number of polls for flash status (assuming that the read operation and write operation takes the same number of shifts). Number of polls for flash status depends on the number of flash pages to program, which has direct relation to the size of the image to be programmed. Hence, the hybrid-online mode (in the form it is described here) will likely to be slower than the online mode for small images.

IV. Double Buffer Online ISP Mode

Despite the hybrid-online mode in general gives shorter ISP time, there is still a place for optimizations. In figure 2 is shown the flow of time components in hybrid-online mode of ISP. It should be stressed that in the ideal ISP solution the bottleneck is in the time taken by the NVM itself to program the page (t_B). In other words, in ideal solution ISP is as fast as the flash can allow. Hence, the timeout shown in Figure 2 needs to be minimized. The time that is used to program data from volatile buffer to page buffer inside flash ("*Buffer to Page Buffer*) and time t_B (*Page Buffer to Page*) are inevitable in any case.

In order to catch up with the time of the ideal solution let us consider adding another buffer to the hybrid-online mode of ISP. The idea is to use the time that flash needs to program the data in its page buffer to the actual page (t_B) for transferring data for the next page from external tester to the buffer in volatile memory. The data flow for double-buffered hybrid online method is shown in Figure 4. In this case the programming of the flash page and the data transfer via test access path are performed in parallel. The limitation of this approach is that the volatile memory has to be big enough to store the monitor and two pages (Buffer1 and Buffer2) of flash image (see Figure 3).

The formula (V) is proposed to calculate the ISP time for the double-buffer approach. The formula (V) is derived from (III) with assumption (VI). In (VI) is assumed that buffer programming time is shorter than a flash page programming time (t_B), which is given in the datasheet. As it is shown later the assumption (VI) holds in case of most on-chip and on-board flashes unless the flash has extraordinary small page programming time. In addition, formula (VI) bounds the speed of communication with external tester (h) from the lower side, when the equation is satisfied.

$$(V): t_{ISP} = m\left(\frac{l}{h} + \sum_{i=1}^{k}\frac{l_i}{h}\right) + b\left(\frac{l}{h} + \sum_{i=1}^{k}\frac{l_i}{h}\right) + t_B\frac{d}{b}$$

$$(VI): b\left(\frac{l}{h} + \sum_{i=1}^{k}\frac{l_i}{h}\right) \leq t_B$$

In formula (V) is shown that ISP time (t_{ISP}) is formed by

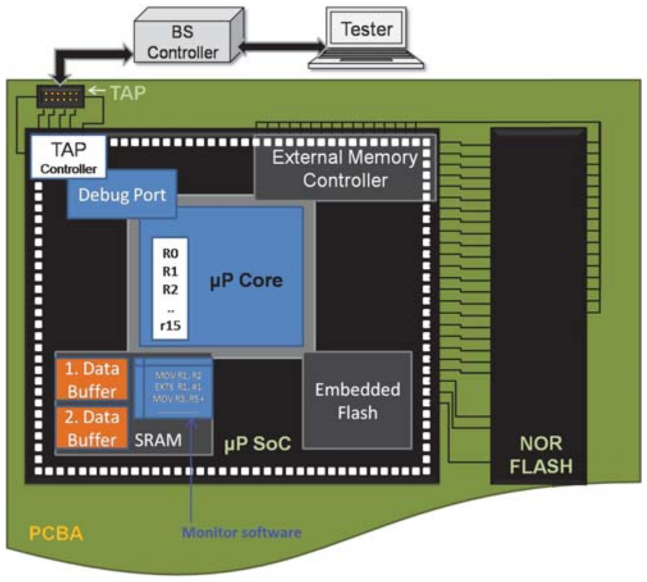


Fig. 3. Data path components in double-buffer hybrid online mode of ISP

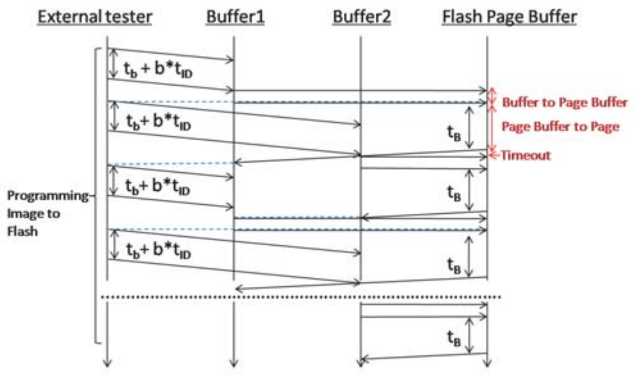


Fig. 4. Time flow in double-buffer hybrid online mode of ISP

the monitor programming to volatile memory, transferring the first page to buffer inside volatile memory (the rest of the pages are transferred in parallel to the programming of the previous page to flash) and the page programming time multiplied by the number of pages to program.

The applicability of the double-buffer hybrid-online mode of ISP in the particular test case can be evaluated using inequality (VI). When the inequality is satisfied the overall ISP time is limited by the flash performance, otherwise the double-buffer based approach does not give a significant speed up and the ISP is limited by the throughput of the test access path.

V. From Theory to Practice

The efficiency of proposed methodology is studied for two use-cases. The first use-case considers programming of the embedded (on-chip) flash of the uC. In the second use-case the external (on-board) flash is programmed.

For the first use-case the uC XC2361E from Infineon [10] is used. The embedded flash memory of XC2361E is built from pages of size 128 bytes. The typical programming time for single page is 3ms [10]. Thus, the theoretical programming time of 256KB is 6144ms. The experimental results presented

TABLE I
Infineon XC2300E uP on-chip flash ISP of 256KB

Method	Program Time (s)	Throughput KB/s
Keil (UART) [11]	151,38	6,76
Keil (JTAG) [11]	14,72	17,12
PCBT [12]	12,43	20,27
PCBT (2-buff.)	7,81	32,25
Theoretical	6,144	35,714

in Table I show the difference in programming 256KB of data into embedded non-volatile memory using toolchain from Keil [11], PCBT approach and modified PCBT approach that is based on the proposed double buffer technique. PCBT approaches are executed on the toolchain from Goepel Electronic [12]. The results from Keil-based toolchain show the state-of-the-art approach time. It should be mentioned that the actual communication frequency is not available for that approach, while for the PCBT the 20MHz test clock (TCK) frequency was used.

The double buffer PCBT approach outperformed the rest of the approaches and showed the time close to the theoretical one. The time for experiments listed in Table I include not only the ISP time, but also the board and uP initialization time. Hence, the difference between theoretical estimation and the double buffer PCBT approach represents the time for initializing/booting the board and uP and also the time for programming monitor software and the first page into intermediate buffer: $7810 - 6144 = 1666(ms.)$. The pure programming time for hybrid online PCBT is $12430 - 1666 = 10764(ms)$. Hence, the time for transferring 256KB of data to intermediate buffer is $10764 - 6144 = 4620(ms)$ and time to transfer 1 page to that buffer is $4620/(2048-1) = 2.26(ms)$. This shows that inequality VI holds, whereas the left part is equal to $2.26ms.$ and the $t_B = 3ms$ as was stated previously.

For the second experiment the board with Emerald-P microprocessor and NOR flash from Numonix was selected. Emerald-P is based on ARM Cortex-A9 core that is paired with ADIv5 debug interface from ARM. The experimental results for ISP of 1 MB image into on-board NOR flash and the theoretical expectations both for hybrid online-hybrid PCBT (ISP_{OH}) and two-buffer online-hybrid PCBT (ISP_{2BOH}) approaches are shown in Table II. The theoretical result is based on the data from respective Numonyx manual [13] for selected NOR flash, where the programming time is defined as a range ($min < t_B < max$) for selected 512 byte long page buffer. From Table II it is seen that the experimental result for double buffer PCBT does not fit into the theoretical range. Which leads us to the conclusion that for the given case the bottleneck is in the throughput of test access path. However, there is a noticeable time improvement achieved in PCBT conditioned by implementation of the second buffer.

In this experiment the size of the flash page and the buffer size is 512 bytes. From data shown in Table II the overall time to program one flash page in PCBT is $t_{ISP_{OH}} = (4434 - 100)/2048 = 2.12(ms)$, where 2048 is the number of buffers in 1 MB. The same operation for the double buffer

PCBT takes: $t_{ISP_{2BOH}} = (2930 - 100)/2048 = 1.38(ms)$. The difference between these two approaches conforms to $t_{OH} - t_{2BOH} = 4434 - 2930 = 1504(ms)$. The same in the formal view:

$$(VII) : t_{OH} - t_{2BOH} =$$

$$d\left(\frac{l}{h} + \sum_{i=1}^{k} \frac{l_i}{h}\right) + t_B\frac{d}{b} - b\left(\frac{l}{h} + \sum_{i=1}^{k} \frac{l_i}{h}\right) - t_B\frac{d}{b} =$$

$$(d-b)\left(\frac{l}{h} + \sum_{i=1}^{k} \frac{l_i}{h}\right)$$

If

$$(VI) : b\left(\frac{l}{h} + \sum_{i=1}^{k} \frac{l_i}{h}\right) \le t_B$$

Otherwise:

$$(VIII) : t_{OH} - t_{2BOH} =$$

$$d\left(\frac{l}{h} + \sum_{i=1}^{k} \frac{l_i}{h}\right) + t_B\frac{d}{b} - d\left(\frac{l}{h} + \sum_{i=1}^{k} \frac{l_i}{h}\right) - t_B =$$

$$t_B\left(\frac{d}{b} - 1\right)$$

In order to state whether inequality VI holds or not it is needed to calculate the exact t_B value. Let us assume that inequality VI does not hold,

then from VIII $t_B = 1504/(d/b - 1) = 1504/2047 = 0.73(ms)$.

The experiments are executed using 20MHz clock for TCK signal ($h = 20000(ms)$). Thus, the inequality VI takes the following form after substituting parameters with their values:

$$512\left(\frac{32}{20000} + \sum_{i=1}^{k} \frac{l_i}{h}\right) \le 0.73$$

after simplifying:

$$0.82 + 512\sum_{i=1}^{k} \frac{l_i}{h} \le 0.73$$

Which indeed leads to contradiction in inequality VI and this was an assumption we made. Let us assume the opposite (inequality VIII holds) to verify correctness of our previous assumption, then:

$$(d-b)\left(\frac{l}{h} + \sum_{i=1}^{k} \frac{l_i}{h}\right) = 1504$$

$$(1048064)\left(\frac{32}{20000} + \sum_{i=1}^{k} \frac{l_i}{h}\right) = 1504$$

$$1677 + 1048064\sum_{i=1}^{k} \frac{l_i}{h} = 1504$$

From where it is seen that last statement is incorrect, because the left part is greater than the right part. Hence, the first assumption that inequality VI does not hold is correct and $t_B = 0.73(ms)$. Finally, time to program 1MB to the flash equals $0.73 * 2048 = 1495.04(ms)$, which conforms to the range given in Table II.

The results in Table II and the computations show that in case with a fast flash memory the ISP is limited by the throughput of test access path. However, the addition of the second buffer to the hybrid-online ISP mode even in case of the fast memory allows to achieve significant speed up. According to Figure 2 and 4 the possible improvement in ISP time is limited. The range of possible improvement is from 0 to T_1, where the upper limit is the time when flash is idle in the hybrid-online mode (T_1). The time when flash is idle is called in this paper as timeout. The flash idle period in the double buffer mode is denoted here as T_2. It should be stressed that the flash is not idle during the process of copying data from buffer memory to the page buffer inside Flash. For the rest of the paper the time taken by this process is defined as C. In Figure 2 and 4 this time period is outlined as *Buffer to Page Buffer*.

Time for programming one page of data from external tester to the buffer memory (see Figure 2) equals to: $t_B + T_1 + C$.

The timeout in the hybrid online PCBT (T_1) is caused solely by the time needed to transfer next page data to the buffer from external tester and this time is equal in both experiments (ISP of on-board NOR Flash using hybrid online and double-buffered hybrid online methods).

$$t_B + b\left(\frac{l}{h} + \sum_{i=1}^{k} \frac{l_i}{h}\right) + C = 2.12(ms)$$

$$b\left(\frac{l}{h} + \sum_{i=1}^{k} \frac{l_i}{h}\right) + C = 1.38(ms)$$

Time for programming one page of data from external tester to the buffer memory in double buffer approach (see Figure 4) equals to: $t_B + T_2 + C = 2830/2048 = 1.38(ms)$, where:

$$t_B + T_2 = b\left(\frac{l}{h} + \sum_{i=1}^{k} \frac{l_i}{h}\right)$$

And $T_2 = 1.38 - t_B - C = 0.65 - C(ms)$. In order to calculate the $C = 0.65 - T_2$ let us subtract time to program one buffer into flash for these two approaches:

$$(t_B + T_1 + C) - (t_B + T_2 + C) = 2.12 - 1.38 = 0.74$$

$$T_1 = 0.74 + T_2 = 0.74 + (0.65 - C) = 1.39 - C$$

After substituting T_1 and T_2 values to $T_1 - T_2 = 0.74(ms)$ we get: $(1.39 - C) - (0.65 - C) = 0.74$ we get $C = 0(ms)$. This means that we can neglect C value in computations due to much less magnitude of measurement (not ms, but ns). It is obvious that ARM Cortex-A9 based uP is capable of copying 512 bytes of data from one memory to another in nanoseconds.

The next unresolved issue questions the possibility to overcome the bottleneck in the face of test access path. From inequality VI it is seen that the only parameter that is not fixed in any test case is the frequency of the TCK signal. In presented experiments (Table II) the theoretical expectations are not reached with the TCK frequency fixed at 20 MHz.

TABLE II
ISP TIME FOR ON-BOARD FLASH DEVICES

	PCBT (ms)	2-buff. PCBT (ms)
Theoretical limit	768 - 2333	768 - 2333
Experimental data	4434	2930
Initialization	100	100

Figure 5 presents the chart that depicts dependency of the ISP programming time of one flash page on the TCK signal frequency. In Figure 5 in blue is shown the t_B which is constant for the given test case and the red curve is the test access path throughput, which is a function of TCK signal frequency (horizontal axis). In other words, Figure 5 is a representation of inequality VI. At TCK frequency of 38 MHz the page transfer time becomes equal to page program time (t_B), thus inequality (VI) becomes satisfied. At higher frequency values the test access path throughput is not a bottleneck any more as the time is shorter than the t_B. Thus, execution of ISP at e.g. 40 MHz of TCK signal is excessive to achieve the shortest possible time for the given test case. Hence, the ISP time will hit the theoretically calculated limit at 38 MHz.

A. *Feasibility study*

In order to see the feasibility of the hybrid online ISP solution improved by doubling the buffer we conducted experiments with various Flash devices. The results are presented in Table III. These devices were selected to study the influence of different technologies (NAND and NOR), interfaces (serial and parallel) and internal memory organisation (e.g. page size) on the overall ISP time. To make the comparison fare the test access path is similar in all cases. The test access path cannot be identical in all cases due to the different implementation of memory controllers for NOR, SPI NOR and NAND Flash devices. However, the rest of the test access path includes uPs with identical debug ports and core architecture. These experiments are executed on the uP that implement ARMv7 [14] architecture (ARM Cortex-A9 cores) and ARM ADIv5 [15] debug interface.

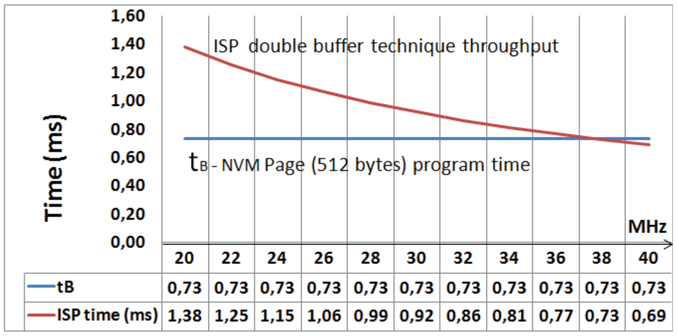
The experiment with NAND Flash was conducted on the PCBA equipped with MT29T Micron [16] device and Zynq uP from Xilinx [17]. As the representative of the serial NOR Flash the device from Numonyx (N25Q128 [18]) is selected, which is paired again with Zynq uP. For the NOR Flash the data from previous experiment (see Table II) is reused.

Table III outlines the differences between various Flash devices by comparing page sizes and page programming time (t_B). In the leftmost two columns of Table III we present the measurement results of running hybrid online (t_{OH}) and improved hybrid online (t_{2BOH}) ISP solutions on the physical boards using test equipment from Goepel Electronic GMBH. The obtained values are used to calculate the actual time for programming a page of data, which is shown in the third column (t_B). The computation algorithm was explained in the previous example. The actual t_B is not given in the datasheets and it varies in accordance to the number of factors (e.g. temperature, supply voltage). However, it is essential in calculations of theoretically minimal ISP time under given conditions. This theoretical time value is used to assess the test access path throughput value of the ISP solution obtained in the same conditions.

Graphically the feasibility of ISP solution is presented in Figure 6. Figure 6 shows the improvement in test access path throughput that can be achieved by increasing the TCK signal frequency. In the same figure is also plotted the programming time of 1 MB data for various flash devices. When the throughput curve crosses the t_B line for 1MB it shows that the memory itself becomes a bottleneck of ISP solution. The latter shows that further increase of TCK frequency will not speed up the ISP time for particular memory. However, this figure also shows that for NAND memory the reasonable increase in TCK frequency is incapable to sufficiently increase the test access path throughput to reach the theoretical minimal ISP time.

VI. CONCLUSION

The contribution of this paper is twofold. The first one is the formal characterization of in-system programming of non-volatile on-chip or on-board memories. This formal basis allows to point out time losses in ISP solution and to assess the efficiency of various ISP approaches. The second contribution is the novel ISP technique that has proven to hasten programming operation to the limit that is set by the NVM performance. The proposed technique is an extension of



	20	22	24	26	28	30	32	34	36	38	40
tB	0,73	0,73	0,73	0,73	0,73	0,73	0,73	0,73	0,73	0,73	0,73
ISP time (ms)	1,38	1,25	1,15	1,06	0,99	0,92	0,86	0,81	0,77	0,73	0,69

Fig. 5. ISP Time as a function of TCK frequency

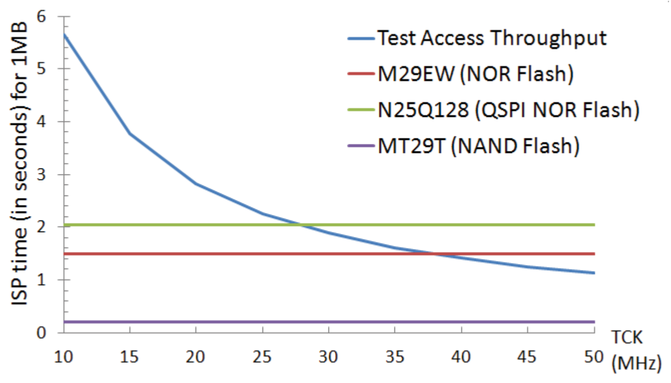


Fig. 6. Test access path throughput comparison with programming time of various Flash devices

TABLE III
ISP TIME FOR VARIOUS ON-BOARD FLASH DEVICES

FLASH Device	**Page Size (bytes)**	t_B **(ms.)**	t_B **(ms.) for 1 MB**	t_{OH} **(ms.) for 1 MB**	t_{2BOH} **(ms.) for 1 MB**
NOR - M29EW Numonyx	512	0,73	1505	4434	2930
QSPI NOR - N25Q128 Numonyx	256	0,50	2050	5022	2973
NAND - MT29T Micron	2048 + OOB	0,39	200	3097	2897

the hybrid-online ISP mode with the double buffer concept. Even in case of exceptionally fast NVM device the proposed technique is capable to guarantee the significant speed up in comparison to the state-of-the-art solutions.

It is foreseen that the tendency of enlargement of the storage capacity of NVMs continues, thus the significance of proposed ISP technique is remarkable.

ACKNOWLEDGMENT

This work has been partially supported by Estonian Science Foundation (grant 9423), Archimedes funded ICT program project FUSETEST, as well as by Enterprise Estonia project ELIKO and Estonian IT Foundation (EITSA).

REFERENCES

[1] *IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE Std. 1149.1-2001, 2001.

[2] *IEEE Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture*, IEEE Std. 1149.7-2009, 2010.

[3] P. Maxwell, I. Hartanto, and L. Bentz, "Comparing functional and structural tests," in *Proc. of International Test Conference*, Atlantic City, NJ , USA, 2000, pp. 403 – 407.

[4] J. Webster, B. Fenton, D. Stringer, and B. Bennetts, "On the synergy of boundary scan and emulation board test: a case study," in *Proc. of Board Test Workshop*, Charlotte, USA, 2003, p. 10.

[5] (2010) High speed programming of non-volatile memories using the xjtag development system. White Paper. XJTAG. [Online]. Available: http://www.xjtag.com/

[6] A. Tsertov, R. Ubar, A. Jutman, and S. Devadze, "Automatic soc level test path synthesis based on partial functional models," in *Proc. of 20th Asian Test Symposium (ATS)*, New Delhi, India, 2011, pp. 532 – 538.

[7] (2009) Combining boundary scan and jtag emulation for advanced structural test and diagnostics. White Paper. T. Wenzel and H. Ehrenberg. [Online]. Available: http://tmworld.resourcecenteronline.com

[8] S.Devadze, A.Jutman, A.Tsertov, M.Instenberg, and R.Ubar, "Microprocessor-based system test using debug interface," in *Proc. of 26th IEEE NORCHIP Conference*, Tallinn, Estonia, Nov. 2008, pp. 98–101.

[9] T.Wenzel and H.Ehrenberg. (2009) Combining boundary scan and JTAG emulation for advanced structural test and diagnostics: White paper. Goepel Electronic GmbH. Jena, Germany. [Online]. Available: http://tmworld.resourcecenteronline.com

[10] *XC236xE Data Sheet*, V1.2 2012-06, Infineon Technologies AG, 2012.

[11] (2009) Getting started. creating applications with uvision4. Manual. Keil, Tools by ARM, and ARM Ltd. [Online]. Available: http://www.keil.com/product/brochures/uv4.pdf

[12] (2012) Jtag/boundary scan is probably the most ingenious test process. Web Article. Goepel Electronic Ltd. [Online]. Available: http://www.goepel.com/en/jtag-boundary-scan/boundary-scan-instruments.html

[13] *M29EW Datasheet*, 208045-10, Numonyx Axcell, 2010.

[14] *ARM Architecture Reference Manual - ARMv7-A and ARMv7-R edition*, ARM DDI 0406C, ARM Limited, 2011.

[15] *ARM Debug Interface v5 Architecture Specification*, ARM IHI 0031A, ARM Limited, 2006.

[16] *MT29F2 Datasheet*, m79m 2gb nand, Micron, 2010.

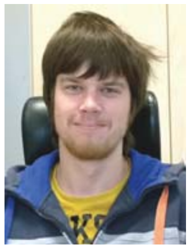
[17] *Zynq-7000 EPP Technical Reference Manual*, UG585 (v1.1), Xilinx, 2012.

[18] *N25Q128 1.8V Datasheet*, Rev 1.0, Numonyx Axcell, 2010.

Anton Tsertov received his M.Sc. and Ph.D. degrees in computer engineering from Tallinn University of Technology, Estonia in 2007 and 2012 respectively and currently holds the position of researcher in Tallinn University of Technology. His research interests include such topics as system and board level test, high-level system modelling, microprocessor functional and structural test.

Sergei Devadze has received his Ph.D. degree in computer engineering from Tallinn University of Technology, Estonia in 2009 and currently holds the position of researcher in this university. His research interests embrace such topics as usage of chip-embedded instrumentation for system test and ISP, fault tolerance and fault management architectures of digital systems, extended structural board test. He is a co-author of over 50 scientific papers in the field of digital design and test published in international journals and refereed conference proceedings.

Artur Jutman received his M.Sc. and Ph.D. degrees in computer engineering from TU Tallinn, Estonia in 1999 and 2004 respectively. His research interests include: embedded instrumentation for board and system test, system modeling, DFT and self-test (adding up to over 120 scientific publications). Artur Jutman has been a visiting researcher and invited lecturer in several European universities in Germany, Sweden, Poland, and Portugal. Dr. Jutman is a council member of the European Association for Education in Electrical and Information Engineering (EAEEIE) and a technology development center ELIKO. He is also a member of the executive committee of the Nordic Test Forum (NTF) society. He is a managing director of Testonica Lab company, which main focus lies in the field of system test instrumentation. Dr. Jutman has been actively involved in numerous FP5, FP6, FP7 R&D projects serving as a coordinator in two of them.



Artjom Jasnetski received his M.Sc.degree in computer engineering from Tallinn University of Technology, Estonia in 2013 and currently he is Ph.D. student at Tallinn University of Technology. His research interests include such topics as SiP test, digital system modelling, ISP and HW driver implementation.